Customizing Pattern-Based Tessellation for NURBS Surface Reconstruction with Irregular Boundary Conditions

DISSERTATION

Submitted for consideration in partial fulfillment of the requirements

for the degree of Doctor of Philosophy in Computational Design

by

Tsung-Hsien Wang
School of Architecture
Carnegie Mellon University

THESIS COMMITTEE

Professor Ramesh Krishnamurti
School of Architecture
Carnegie Mellon University

Professor Kenji Shimada
Department of Mechanical Engineering
Carnegie Mellon University

Professor John Folan
School of Architecture
Carnegie Mellon University

Professor Robert Woodbury
School of Interactive Arts and Technology
Simon Fraser University

April, 2012

I hereby declare that I am the author of this dissertation

I authorize Carnegie Mellon University to lend this dissertation to other institutions or individuals for the purpose of scholarly research.

I further authorize Carnegie Mellon University to reproduce this dissertation by photocopy or other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

_____

Tsung-Hsien Wang

Carnegie Mellon University

College of Fine Arts
School of Architecture

Dissertation

Submitted in Partial Fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Title: Customizing Pattern-Based Tessellation for
NURBS Surface Reconstruction with Irregular Boundary Conditions
Presented By: Tsung-Hsien Wang

Accepted By:

Dan J. Martin	Dean	Date
Stephen R. Lee	Head	Date
Ramesh Krishnamurti	Advisor	Date
Kenji Shimada	Co-Advisor	Date

Carnegie Mellon University

College of Fine Arts
School of Architecture

Dissertation

Submitted in Partial Fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Title: Customizing Pattern-Based Tessellation for
NURBS Surface Reconstruction with Irregular Boundary Conditions
Presented By: Tsung-Hsien Wang

Accepted By:

Ramesh Krishnamurti	Advisor	Date
Kenji Shimada	Advisor	Date
John Folan	Advisor	Date
Robert Woodbury	Advisor	Date

ABSTRACT

A growing trend in contemporary architectural practice, pioneered by such avant-garde architects as Frank Gehry, Zaha Hadid and others, exploits NURBS (**N**on-**U**niform **R**ational **B**asis **S**pline) surfaces to design and model intricate geometries for projects which otherwise would be impossible to realize. In doing so, they have liberally borrowed digital fabrication techniques developed in the automobile and aerospace industries (Kolevaric 2005a, 2008; Pottmann 2008). A NURBS surface is a mathematical model for freeform shapes. To manifest a NURBS surface, a discrete model, namely, mesh, is utilized. Transforming a NURBS surface into a mesh appropriate for application is computationally intensive, and generally, it is not an easy task for architects or designers who have no formal geometry training.

In order to design, model, and, subsequently, fabricate intriguing, sometimes intricate, freeform shapes, this research looks at the surface tessellation problem, which is an extension of the problem of meshing a NURBS surface, with an added consideration of incorporating constructible building components. There are close relationships and analogies between the elements of a mesh and the components of a freeform design, e.g., face to panel, edge to structural frame, etc.

Initially, features of a NURBS surface and contemporary tessellation methods are examined. Mathematically, a NURBS surface is regulated by a set of control points and edges. The control points are used mainly to interpolate a continuous shape using a higher order equation, in most cases, usually cubic. The edges delineate the appearance of the freeform shape. For a surface, edges (also called boundaries) indicate where the surface analysis starts and where it ends, and thus, plays a significant role in the meshing process.

Two kinds of boundaries are examined in this research. The first are global boundaries, which form the overall appearance, e.g. exterior edges, or interior trimming edges. The second kind is a

local boundary, which specifies how a discrete element is formed—namely, the pattern of a face, e.g. triangle or quadrilateral. By looking at given surface boundary conditions and tessellation patterns, this research presents an algorithmic approach to pattern-based surface tessellation and develops strategies to resolve issues that stem from the juxtaposition of computational geometry and freeform architectural design.

The contributions includes the technical implementation of boundary-driven mesh generation, which demonstrates the potential of utilizing featured boundaries for customizable polygon-based tessellation in comparison to conventional iso-parametric subdivision. This is described through examples by extending the optimized mesh network for various pattern generations. In addition, pedagogical implications are exemplified by solving the geometric constraints for surface tessellation within the parametric modeling paradigm. These contributions are expected to support future sustainable development in the field of freeform architectural design.

Keywords: Pattern-based surface tessellations, irregular boundary conditions, meshing

ACKNOWLEDGEMENTS

I would like to thank all my committee members for their immeasurable support, mentoring and patience over these years at Carnegie Mellon University. Professor Krishnamurti, my thesis advisor, for his unconditional support throughout the work on this thesis and my study at Carnegie Mellon University. Professor Shimada, my thesis co-advisor, for his encouragement and supervision especially on meshing. Professor Folan and Professor Woodbury, for their insightful comments and discussions. I am truly grateful to all of your support and help that made this work possible.

In addition, I want to thank all my friends; special thanks to Brubaker family, Brian, Yahtyng, Kieth and Ellen, for giving me a helping hand while staying in Pittsburgh. I would also like to thank my colleagues, Tony for his encouragement; Tajin and her family for her enormous help. Thank you all for enriching my stay with joyful memories at Carnegie Mellon University.

Most of all, I would like to thank all my family for their love, encouragement and support. My hearty gratitude goes to my grand-mom, parent, brother, Eddie, and sister, Winnie, who have been there for me throughout this journey.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

Introduction

Geometry is at the very core of modern architectural design, particularly with the growing interest in being able to compute and construct non-simple, intricate, geometric forms (Pottmann et al., 2007b; 2008b; Pottmann, 2010). For architectural application, approaches to designing and modeling freeform geometry require a heavy dose of computation, which is reflected, mainly, in the way parametric control is exercised over the whole generative process. In trying to realize such geometries, the prevalent approach is to approximate curvilinear freeform surfaces by discrete fabrication-friendly building components—in essence, a surface tessellation process. This is computation-intensive and poses serious challenges to architects and designers who wish to explore fluid freeform designs, especially for those who have had little formal geometry training.

In particular, NURBS (**N**on-**U**niform **R**ational **B**asis **S**pline) surfaces are considered in this research. NURBS based freeform surfaces, featuring easy-manipulation and high accuracy of shape, have gained increasing popularity in large-scale, sometimes geometrically intricate, architectural projects in recent years (Duesing, 2007). Moreover, NURBS are useful for a number of reasons. They: i) are invariant under common transformations e.g., affine, perspective and other transformations; ii) offer a common mathematical form for both standard analytical shapes and freeform shapes; iii) provide flexibility for designing a large variety of shapes; iv) reduce memory consumption when storing shapes; and v) can be evaluated reasonably quickly by accurate numerically stable algorithms. Hereinafter, the term *surface* refers to a *NURBS surface*.

1.1 Motivation

When considering manifesting freeform surfaces, how geometrical elements are configured indicates the development of underlying geometry, supporting structure and corresponding fabrication machinery and material. In general, the entire process can be treated as the combination of three sub-processes: (1) geometry construction; (2) structure development; (3) fabrication (Figure 1-1). For instance, when developing structural systems, the geometry configuration shows how forces affect structural members (Veltkamp, 2007). Additionally, designated geometric elements indicate potential fabrication techniques and applicable materials. For example, planar elements are constructible by utilizing two-dimensional fabricating techniques with planar sheet material. Moreover, structural performance is subject to the chosen production method and material properties (Kim et al., 2008). These three sub-processes, in a sense, are closely related to each other, and play significant roles in the process of realizing complex surfaces.

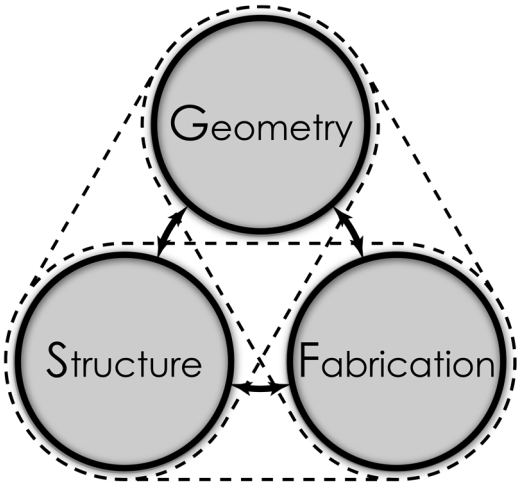


Figure 1-1 The entire process when manifesting freeform surfaces

In this dissertation, the major research scope focuses on a general process of tessellating a surface into its discrete counterparts, which are expected to be utilized for corresponding structural and fabrication development. The subject surface is double-curved NURBS surface with potential irregular boundary conditions. The following section discusses number of distinct areas in computational geometry that this dissertation draws upon.

1.1.1 Meshing Surfaces

Meshing is the process of transforming a continuous model, such as a surface, into collection of discrete parts—namely, its mesh elements. From a computational geometry perspective, the *mesh*—which can be treated as a discrete model of a surface—is a structured network consisting of vertices, edges and faces. The finer the mesh the more closely it resembles the original surface. The art of good meshing is to find the right level of granularity for the mesh to adequately model the surface.

In principle, meshing a surface is the process underlying the creation of a surface tessellation. Such tessellations are useful in architectural applications; for instance, any face of a triangulated surface can be considered as the geometry for a panel component of a freeform façade; edges of a mesh can represent structural frames underneath a freeform skin; vertices of a mesh can designated as the joints where these frame components meet (Figure 1-2). Architects and other designers often employ mesh-like representations to explore aggregations of constructible structures. In using the term, *meshing*, this research includes the added consideration of potentially incorporating constructible components as the part of the process.

Figure 1-2 (*Left*) Facade Panels (*Middle*) Surface Structural Frames (*Right*) Structural Joints
Images after Pottmann et. al. (2007b)

Meshing a surface is a discretization process. It involves computation with respect to (i) the base shape(s) of the mesh; and (ii) the organization of these base shapes to form the given surface. The base shapes are user (designer)-specified. The organization represents a mesh layout. The goal of meshing is to approximate a continuous freeform shape by discrete elements, which can be further embodied to form constructible components for application, e.g., from face to panel, from edge to

structural frame, and so on. For architectural applications, these constructible components correspond to building components.

1.1.2 NURBS Representation

NURBS curves and surfaces behave similarly except that curves are simpler to explain. The following material is standard (Peigl, 1991; Peigl & Tiller, 1996; Rogers, 2001).

A NURBS curve is, normally, defined by *order*, a set of *weighted control points*, and a *knot vector*. NURBS curves are generalizations of B-splines or Bezier curves. Curves evolve along a single parametric direction, commonly referred to as u. The control points are used to interpolate a continuous shape by a higher order equation.

$$C(u) = \frac{\sum_{i=0}^{n} w_i P_i N_{i,k}(u)}{\sum_{i=0}^{n} w_i N_{i,k}(u)} \tag{1-1}$$

Here w_i are weights associated with P_i, the control points. $N_{i,k}$ are B-spline basis functions of degree k. Order (= k+1), specifies the number of control points that influences any point on the curve. Typically, the NURBS curves used in architecture are cubic, that is, k = 3, and there should be at least four (k+1) control points for the curve interpolation. The B-spline basis functions are recursively defined:

$$N_{i,k}(u) = \frac{u - t_i}{t_{i+k} - t_i} N_{i,k-1}(u) + \frac{t_{i+k+1} - u}{t_{i+k+1} - t_{i+1}} N_{i+1,k-1}(u) \quad , \quad N_{i,0}(u) = \begin{cases} 1 & \text{if } t_i \le u < t_{i+1} \\ 0 & \text{else} \end{cases} \tag{1-2}$$

The t_i's are knots forming a non-decreasing sequence $U = (t_0, \ldots, t_{m=n+k+1})$ where $t_i \leqslant t_{i+1}$. U is the knot vector. Each successive pair of knots represents an interval $[t_i, t_{i+1})$ for the parameter values to calculate a segment of a shape. Since knot spacing could be non-uniform, the B-splines are not the same for each interval $[t_i, t_{i+1})$. The number of knots for each segment interpolation can also vary in

relation to the degree of B-spline basis functions. Over the whole range of parameter values represented by the knot vector, the different B-splines build up continuous (overlapping) blending functions $N_{i,k}(u)$ as defined above.

Equation (1-1) can be rewritten using rational basis functions:

$$C(u)=\sum_{i=0}^{n} P_i R_{i,k}(u) \quad \text{and} \quad R_{i,k}(u)=\frac{w_i N_{i,k}(u)}{\sum_{j=0}^{n} w_j N_{j,k}(u)} \tag{1-3}$$

A NURBS surface S is similarly defined except that surfaces evolve over two parametric directions commonly referred to as u and v.

$$S(u,v)=\sum_{i=0}^{n}\sum_{j=0}^{m} P_{i,j} R_{(i,k)(j,l)}(u,v) \quad \text{and} \quad R_{(i,k)(j,l)}(u,v)=\frac{w_{i,j} N_{i,k}(u) N_{j,l}(v)}{\sum_{r=0}^{n}\sum_{s=0}^{m} w_{r,s} N_{r,k}(u) N_{s,l}(v)} \tag{1-4}$$

The rational basis functions R have the same properties as the blending functions. The advantage of equations (1-1), (1-3) and (1-4) are that knots are absorbed into the expressions and need not be the concern of the user, which is usually an advantage in practice. As before, $P_{i,j}$ and $w_{i,j}$ respectively denote the control points and weights. $N_{i,k}$ and $N_{j,l}$ are the B-spline basis functions of degree k and l in the u and v parametric directions respectively. u and v are also known as the *isoparameters* of the surface. Figure 1-3 illustrates a NURBS surface and its control points, which as a whole specify the *control polygons* of the surface. Edges in a surface where an analysis starts and ends are referred to as surface *boundaries*.

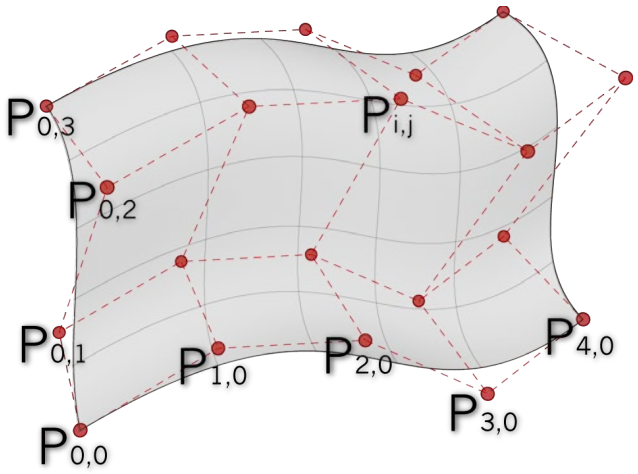


Figure 1-3 NURBS surface construction with control points

The isoparameters u and v represent a mapping from a uniform two-dimensional grid to a three-dimensional manifold. See Figure 1-4. A surface is specified by a pair of ranges of parameter values. Each point on the surface satisfies equation (1-4). Intervals $[u_0\ u_1)$ and $[v_0\ v_1)$ define a sub-surface quadrilateral, where u_0 and u_1 are the starting and ending parameter values of the interval along the U direction, and likewise, v_0 and v_1, along the V direction. This approach is flexible and offers precision for surface analysis. Contemporary approaches to modeling and analyzing NURBS surfaces rely heavily on the two regional isoparameters (Pottmann, Asperl, et al., 2007a).

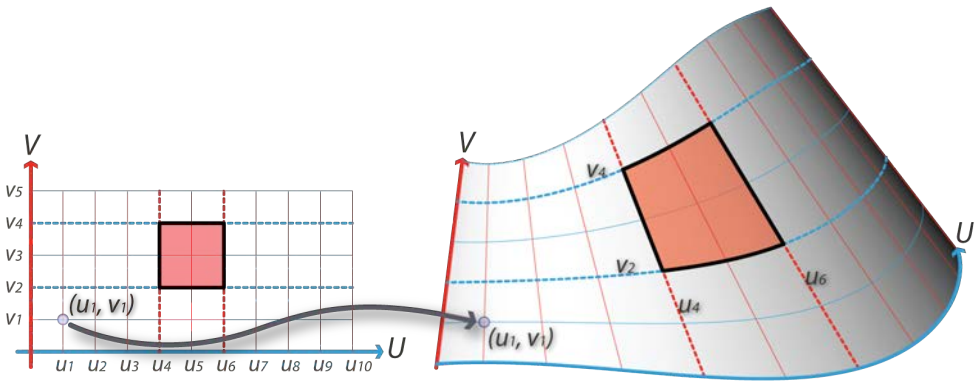


Figure 1-4 Mapping from the two-dimensional UV coordinate system onto a NURBS surface
Sub-surface patches in the intervals $[u_4, v_6)$ and $[v_2, v_4)$ are shown colored red

1.1.3 Non-Regular Boundary Conditions

In practice, in a design context, not every "originally designed" surface remains intact—that is, as a surface without having been trimmed, cut or otherwise modified. In other words, in the course of designing, a surface is typically altered, that is, has parts removed and new boundaries introduced. The new entities usually have a close relationship to the original untrimmed, uncut or unmodified surface so that they can be properly placed on the original surface. Figure 1-5 shows a trimmed surface (shown shaded red) overlaid with the original untrimmed surface. The new boundary edges are composed of two trimming edges, E_1 and E_3, and two partial original boundary edges, E_0 and E_2. As E_3 is on the surface, every point $p(u, v)$ on E_3 satisfies the same definition, namely, equation (3).

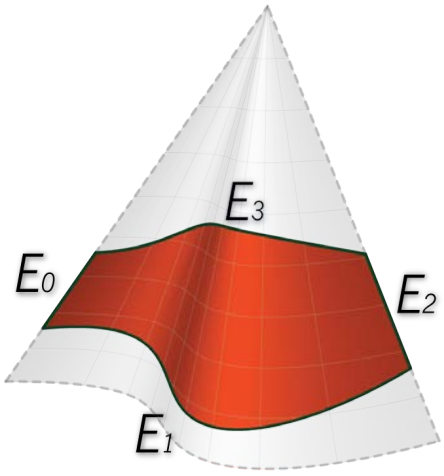


Figure 1-5 New trimmed boundaries E_0, E_1, E_2, E_3

A surface with non-regular boundary conditions – trimmed edges or cut holes – can cause potential problems when the surface is meshed, especially for customized elements. For instance, in a quadrilateral mesh—a mesh consisting of only quadrilateral faces, irregular faces such as triangles could be generated around trim edges. These irregular faces could be aesthetically unsightly; moreover, when faces are brought into fabrication, additional costs might be incurred, e.g., standard quadrilateral panels may have to be specially customized to fit trimmed edges and holes. In other words, new boundaries generated by additional surface operations, such as trim, might engender renewed attention, or require new strategies when segmenting the surface.

1.1.4 Pattern-Based Tessellation

A *surface tessellation* is a pattern of figures that fills the surface without overlaps and gaps. The most popular pattern seen in surface tessellation is the triangle. The major reason for this is that a triangular panel makes for easy fabrication. Moreover, it is always feasible to pack any arbitrary surface with triangles without being limited by the boundaries. However, triangles are not always the appropriate solution for a design or architectural application. Other polygonal patterns offer alternative creative design solutions with various aesthetic, or, sometimes, constructional considerations. In fact, architects and designers often actively seek potential alternatives to customize patterns, for instance, using quadrilaterals or hexagons, in order to control and to further design the tessellation pattern. However, as is often the case, as boundary conditions grow more complex, the resulting irregular boundaries no longer provide an easy handle for regular polygonal segmentation. Special treatment is required to fit a desired pattern on a given surface with the intended considerations given to direction, dimensions and featured boundary conditions. Moreover, these constraints may conflict with each other, thus over-constraining the tessellation problem. Inspired by the need to explore and provide alternative ways of tessellating a surface with given configurable patterns, in this research a pattern-based scheme is explored for the surface tessellation problem with added consideration given to the underlying surface boundary conditions.

1.2 Problem Statement

The following problem arises from meshing a surface for post-modeling fabrication-oriented application:

> *How can a surface be decomposed into smaller 'easy-to-construct' modules for a given pattern-based parametric tessellation scheme with irregular boundary conditions?*

This dissertation addresses the problem by describing *a general algorithmic approach to generating and optimizing a polygon-based surface tessellation with minimal irregularity*.

Leaving to one side the notion of 'easy-to-construct,' there are two key issues to consider in regard to this research question. The first relates to the development of parametric tessellation schemes for surface panelization; the second relates to boundary-driven optimization of the surface discretization by selected polygonal patterns.

1.2.1 Parametric Tessellation Schemes for Surface Panelization

Panelization is the process of realizing a freeform surface by a collection of constructible components, in particular, face-based panels and supporting structures. For architectural applications, this tessellating process describes how panels (building components) are utilized to construct the designated freeform shapes. From a parametric modeling perspective, each panel can be procedurally built on a given base shape, namely, the pattern of the local boundary representation. For instance, underlying a quadrilateral panel are four vertices that define the local boundary, a quadrilateral face. Contemporary approaches to modeling these panels are primarily based on the surface isoparameters, u and v. While the approaches are simple and efficient, they are, at the same time, limited. For instance, isoparameters do not distinguish between the curvatures of adjacent surface patches, which might affect both aesthetic appearance and final manifestation. Nor do the parameters control the size of sub-surface generation(s). Commonly, a uniform parametric interval is employed; this usually results in non-uniform sub-surface generation. The size of each panel is in fact closely related to the initial control polygons. The control polygons govern the control points, which are used to interpolate the ultimate surface presentation. If the vertices of the control polygons are uniformly distributed, an equi-dimensional patch is more likely to be generated. However, given the freedom with which control points can be modified in the modeling environment, they rarely remain uniformly distributed once designers start manipulating, often arbitrarily.

Unlike curve decomposition wherein a curve is divided into equidistant segments by a prescribed circle, there is no general way of dividing a surface into uniform sub-surfaces. Figure 1-6 illustrates three possible segmentation schemes that generate vastly different sub-surface patches with variations in size. The figure shows three distinct surface segmentations with the same number of sub-surface patches. The left-most figure illustrates initial surface subdivision by uniform isoparameters, with equal intervals along both the U and V domains. The middle figure illustrates an attempt to equalize the size of the sub-surface patch. The right-most figure illustrates a customized surface segmentation, where patch size is inversely proportional to the rate of change of surface curvature, the higher the rate of change the smaller the patch. Using a different tessellation scheme, another different exclusive mesh will be created.

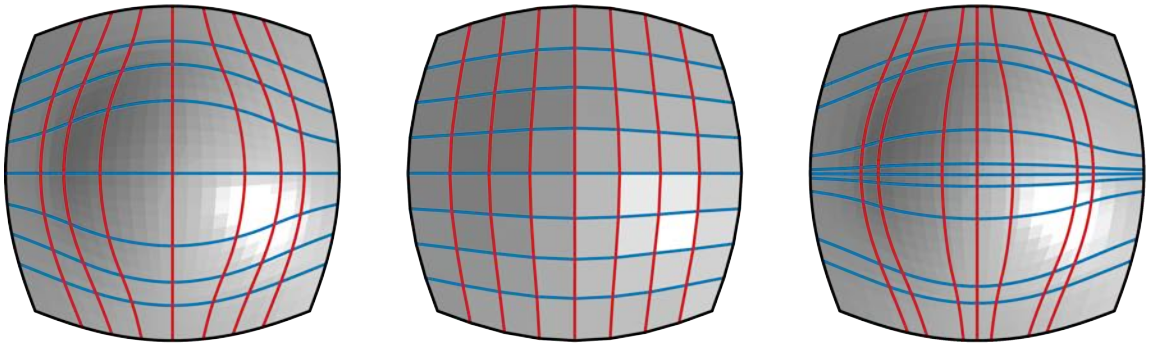


Figure 1-6 Three possible types of UV-based segmentation
Division by (*left*) uniform isoparametric intervals; (*middle*) optimizing face size via equi-dimensional intervals; and (*right*) customized iso-parametric intervals

Current meshing applications for architectural design rely mainly on uniform isoparametric control. That is, patches are generated according to rules similar to those that produce the leftmost illustration in Figure 1-6. For visualization purposes, a surface is rendered by a subdivision scheme that is optimized for surface curvature. Yet, for architectural applications, if one takes into consideration the machining of parts, equi-dimensional surface patches are then more practical. This is quite challenging to achieve.

If one has a well-defined constructive or generative procedure, intricate as well as performative surface tessellations are realizable. Figure 1-7 illustrates an example based on a procedural modeling approach (Wang, 2009). In the example shown, each panel is procedurally generated for the given local boundary, namely, a pattern of quadrilaterals. The aperture of each panel is parametrically constructed and controlled by examining light gains over a period of time using the Solar Position Algorithm[1] (Reda and Andreas, 2008). The reconstructive procedure includes the following four steps: (1) retrieving surface boundaries for reconstruction; (2) developing generative principles for surface panels; (3) post-design variations via performative simulations (lighting simulation is employed in this case); and (4) surface component analysis for manufacture. The underlying pattern still dominates any manifest appearance—that is, the pattern, ultimately, determines the shape of the form that the tessellated surface takes.

[1] Solar Position Algorithm for Solar Radiation Applications from National Renewable Energy Laboratory, http://rredc.nrel.gov/solar/codesandalgorithms/spa/. Last accessed on Apr 8th, 2010.

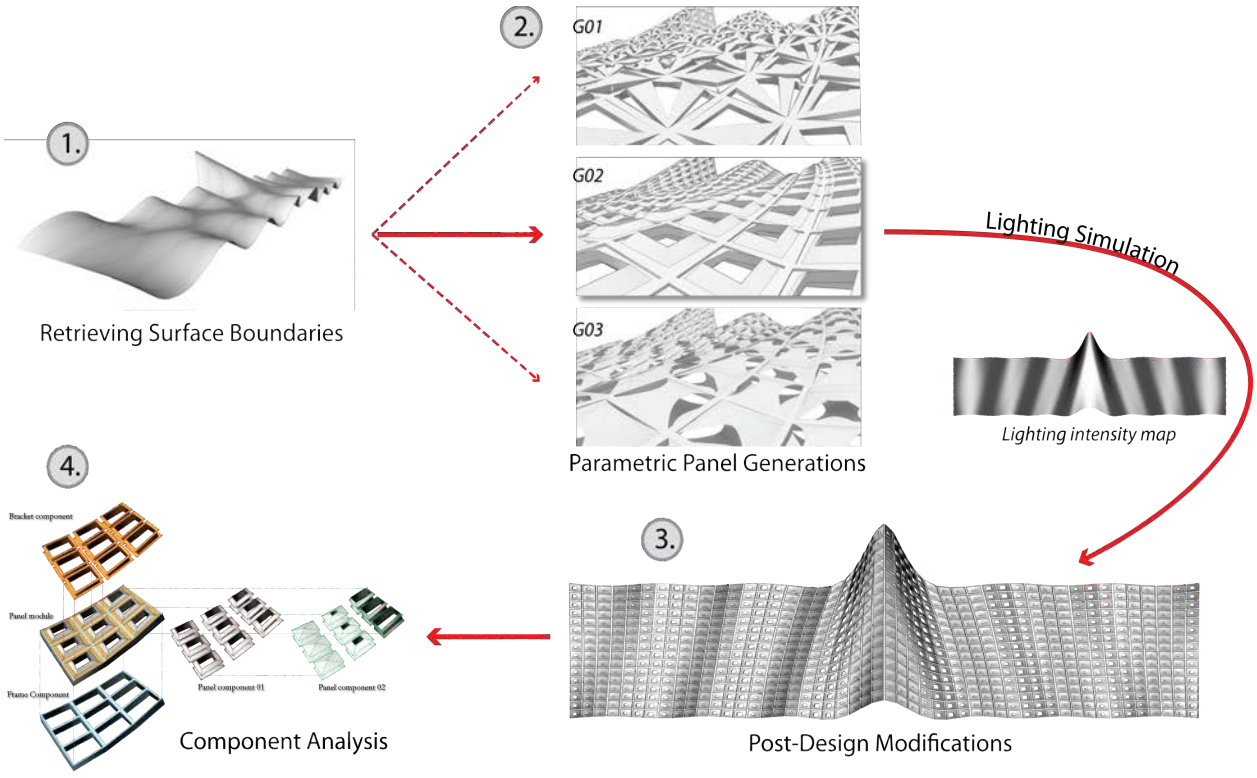


Figure 1-7 Procedural modeling for surface reconstruction (Wang, 2009)

The first research task can be summarized thus: to examine parametric strategies and processes involved in pattern generation, for example, exploring quadrilateral, hexagonal and other potential polygonal shapes that are treated as local boundaries in a surface tessellation process. The objective is to mesh the surface with customized boundaries—that is, a pattern—with respect to a global boundary condition, which correspond to the primary edges that define the overall appearance of the freeform surface. Furthermore, how such a parametric framework can be used procedurally to construct building components within the governing boundaries will be considered. From a constructive geometric perspective, the relationship between polygon-based patterns and pattern propagation will be examined and discussed.

1.2.2 Boundary-Driven Optimization

When tessellating NURBS-based surfaces, the form and formation of target shapes and types of discrete elements (panels) are preferably identified prior to the tessellation process. These considerations require a computational scheme for resolving issues that stem from local boundary

conditions—mesh patterns—and from global boundary conditions—exterior boundary edges and interior trimming holes.

It helps when a surface has been defined mathematically, because the surface can then be precisely determined and analyzed. Initial boundaries are calculated directly from the control polygons; these, in turn, consist of primary control points, which are used to formulate the exterior boundary edges. However, during design, the surface is not always guaranteed to remain intact—that is, without being trimmed or having holes cut out. Whenever part of a surface is altered from its original, new boundaries, trim edges or cuts, may be created. The newly formed surface is referred to as a *trimmed surface*. Any new boundaries introduced by the trimming process are usually untreated, resulting in a number of irregular mesh faces; in other words, irregularly shaped panels around the trimmed edges. This irregularity should be addressed and reduced so that any potential additional cost, say, for fabrication or aesthetic appearance, can be minimized. From a geometric perspective, the new boundaries share identical isoparameters because points on the curve should also be on the surface on which the curves are located. Thus, the same parameters u and v can be used to describe the new boundaries. Assuming that we possess the techniques to describe new curves on a surface, how these boundaries can be examined is briefly discussed next.

From a design perspective, trims are introduced to meet specific design intentions, for example, openings for lighting, viewing or circulation, etc. Figure 1-8 illustrates the west elevation of a surface, which has been trimmed for skylight, natural view and entrance. Problems immediately occur when these new boundaries are introduced to the original untrimmed surface. The trimmed edges, for example, cut through the uniform shapes of certain surface panels. For this particular design, the panels are quad-dominant. Irregular panels surrounding the trim edges are generated and these, in turn, affect the overall aesthetic appearance of the surface manifestation as well as the final fabrication. To address the issues stemming from trimmed edges, a second research task is considered. This second task explores how global boundary conditions, which primarily determine the final freeform surface, can be used to affect or tune the tessellation process, particularly, to optimize the layout of the mesh elements towards a more balanced solution. For example, directions of panelizing could be modified (or better instructed) to avoid, or reduce, the irregular panels as the boundary conditions evolve. Since boundaries define the ultimate appearance and given that panelization can take all boundaries into considerations parametrically and algorithmically, the hypothesis is that a coherent surface tessellation can be achieved.

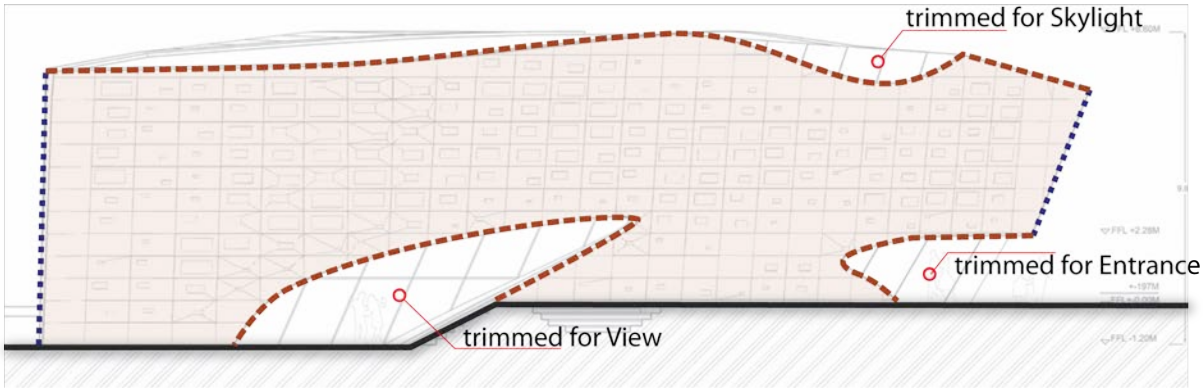


Figure 1-8 West elevation for trimmed boundaries
Image adapted and modified by the author from an image of the Next-Gene Museum by Zaha Hadid (2008)

The west elevation shown in Figure 1-8 is utilized in Chapter 5 as the example to demonstrate how boundary conditions are used to regulate the surface tessellation layout for pattern-based exploration.

1.3 Research Objectives

There are limitations to current surface tessellation algorithms, especially, when boundary conditions become complex. In this dissertation the surface tessellation problem is looked at from the perspective of the boundary conditions. First, the important ingredients and processes involved in reconstructing surfaces with pattern-based elements are identified. Second, a meshing algorithm is presented, which will ease the mesh generation process by solving the constraints inherited from the featured surface properties and boundary conditions. Then, an integrated process of pattern-based construction and exploration are given as an example to demonstrate the usage of the proposed work within the context of freeform surface design.

1.3.1 A parametric framework for tessellating surfaces with polygonal patterns

A parametric framework for polygon-based surface tessellation is investigated to identify the essential ingredients for tessellating surfaces. Two categories of patterns are examined. The first category is the Archimedean tiling pattern, and the second is the interwoven pattern.

The Archimedean tiling is a two-dimensional vertex transitive edge-to-edge pattern (Grünbaum and Shepherd, 1987), which provides a distinct aspect on how to combine various regular polygons in tiling the plane. How these polygons are configured for plane tilings are discussed with details in Chapter 3. Overall, owing to the planarity of 2 dimensional Euclidean operations, the sum of the angles at any vertex equals 360° and only a limited number of variations are possible. In total, there are three regular patterns and eight semi-regular patterns. In the first pattern investigations, three major operators are presented to derive all the possible variations for the Archimedean tiling and other parametric mutations based on the same operative principles.

The interwoven pattern is an extended version of the polygonal patterns, and is inspired by Erwin Hauer's works on screen wall design. In addition to the symmetry or transitivity properties of configurations of polygonal shapes, each shape is treated as a constructive reference to develop an interweaving module. Examples of interwoven patterns are provided, some are recreations from Erwin Hauer's own designs and others are variations.

By taking into account both categories of patterns, in which one category can be derived from the other, the studies involved in generating pattern-based tessellation are discussed and analyzed to articulate the parametric process for surface tessellation.

1.3.2 Automated mesh generation with irregular boundary conditions

The second objective is to automate the mesh generation by examining the surface boundary conditions. Three major mesh components are introduced in the meshing process; these are the Boundary-Driven Tensor, Boundary-Driven Curve and Boundary-Driven Mesh.

1.3.3 An integrated constructive process for design exploration- User-Controllable manipulation

After setting up the computing environment for surface tessellation, various key parameters are extracted for explorative purposes. By formalizing the parametric control, it can improve the ease and possibilities for exploring various design alternatives. In addition, the integrated process will demonstrate the complex geometric constraint solving from a parametric modeling perspective.

In summary, this dissertation explores the surface tessellation problem, slanted towards contemporary freeform architectural design where pattern-based panelization is essential for physical construction and fabrication. This research focuses on the boundary conditions of the input surface

and takes into consideration other design inputs such as, panel patterns, panel size, or panelization direction. In particular, how surface boundaries affect panel layout, and the propagation of customized pattern-based components is looked at. The main objective is to render designers an algorithmic approach, with constructive strategies, to achieve a more aesthetic, attractive, yet fabrication-friendly structure, when developing their freeform architectural designs.

Surface tessellation is identical to a discretization process of meshing the surface from a given underlying geometry perspective, yet, requiring further detailed constructive principles on how the tessellated component should be procedurally constructed. The transformation of a freeform surface into a mesh is regarded as essential for the final tessellation. This research aims to go beyond current limitations embedded in isoparametric analyses, by developing a parametric scheme in which constructive procedures for segmenting a freeform surface with discrete constructible components can be encapsulated. The proposed approach affords designers a flexible manner of exploring surface tessellations.

1.4 Structure of the dissertation

Chapter 1 starts with the introduction to the major problem statement, customizing pattern-based tessellation with irregular boundary condition. The major steps and objectives are presented in relation to the components of the thesis.

Chapter 2 provides the background review in relation to pattern-based surface tessellation. First, the background review of the parametric modeling process is discussed and following by the current active meshing techniques from both computational geometry and engineering perspectives. Then, contemporary fabrication techniques are discussed in relation to freeform surface construction.

Chapter 3 illustrates the preliminary studies on the pattern generation. Two categories of patterns are examined: one is the Archimedean pattern and the other is the interwoven pattern inspired by Erwin Hauer (2007). The constructive procedures among various polygonal and interwoven patterns are investigated and formalized as generative rules, which would be later applied for surface tessellation applications.

Chapter 4 describes the implementation of the boundary-driven (BD) meshing algorithm. The underlying characteristics of the given surface and surface boundaries are first analyzed and extracted. Then, three major meshing components, including BDTensor, BDCurve and BDMesh are illustrated

by drawing the relationships from the featured boundary conditions. Specifically, the quadrilateral mesh is presented to address the formation of pattern-based freeform discretization with irregular boundary conditions.

Chapter 5 illustrates examples of tessellations by examining the influences as the boundary conditions grow. For practical purposes, the west façade of Zaha Hadid's Next Gene Museum is remodeled and utilized as the example. By procedurally constructing the target surface, a series of meshing results are illustrated along the changes from the evolving boundaries. Lastly, given the well-structured mesh, pattern-based applications are explored in this chapter as examples to demonstrate the generative process from surface decomposition to pattern-based tessellation.

Chapter 6 discusses the results and findings from the investigations and addresses the main contributions and limitations. This section will project potential future applications of the algorithmic and parametric approach described in this dissertation.

Chapter 2

Research Background: Parametric Design Process, Meshing, and Digital Fabrication

---

An important aspect of parametric modeling in contemporary design practice is the emphasis placed on the use of parameters, in particular, customized procedures, by which designers can process associated information directly albeit points, lines, surfaces or building-related data, such as orientation, site, weather, etc. (Woodbury et al., 2007; Woodbury, 2010; Meredith et al., 2008; Schumacher, 2009). Avant-garde architects like Frank Gehry (Linsey, 2001) and Zaha Hadid (Jodidio and Hadid, 2009) utilize similar techniques while experimenting with intricate geometries for designs, which they manage to transform into real buildings based on underlying constructive principles. An essential feature of this approach is that the procedures allow changes parametrically over a controlled design space—this in turn makes the design solution more manageable from conceptual exploration to final manifestation. From a computational geometry point of view, most of such geometrically complex projects have curvilinear surfaces. To realize such a surface depends on two

main aspects: (1) a discretized model of the target surface; and (2) applicable fabrication processes. In both cases, a parametric modeling approach is integral to resolving design issues.

Discretization is, essentially, meshing. Recall that meshing is the process of transforming a continuous model, such as a surface, into collection of discrete parts—namely, its mesh elements. A *mesh* is a structured network consisting of vertices, edges and faces. The finer the mesh the more closely it resembles the original surface. The art of good meshing is to find the right level of granularity for the mesh to adequately model the surface. The quality of a discretized model—mesh—can be established by the geometric closeness to the target surface and the layout of designated mesh elements in relation to surface properties such as curvature and boundary (Eigensatz, Kilian, et al., 2010). Certain features of mesh elements such as shape, planarity, dimension, and direction of mesh elements, are commonly sought out for subsequent physical manifestation.

For physical construction, factors such as material, fabrication techniques, costs and so forth are usually considered. Manufacturing a surface—that is discretized as thousands of distinct parts—by a digital fabrication process relies heavily on contemporary computer numerical control (CNC) machinery for mass customization (Kieran and Timberlake, 2004; Schodek et al., 2005; Corser, 2010). Owing to the potential for intricate geometrical configurations, mass customization becomes indispensable. However, given that cost is a design constraint for constructions, a discrete model can be optimized toward less expensive configurations; for instance, minimizing the differences between parts. Details and examples of this are discussed in Section 2.3. In a sense, the parametric modeling approach has made possible the incorporation of heterogeneous information from computational geometry and fabrication; this, in turn, allows designers to explore designs computationally in a coherent manner.

To summarize, the techniques employed by contemporary complex-geometry projects are rooted in three main disciplines or subject areas, which this dissertation builds upon and relates to. These are the parametric design process, surface fitting (meshing), and digital fabrication.

2.1 Parametric Design Process

Parametric modeling is the process wherein designers utilize relational constraints to construct and manipulate geometrical entities (Madjdoub, 1999; Maleki and Woodbury, 2008; Woodbury, 2010). There is considerable computational design research that investigates the use of constraints for design exploration (Sutherlan 1963; Gleicher 1991; Medjdoub 1999) and also a number of commercial

constraint-based modeling tools that are available to designers (Autodesk, 2010; McNeel, 2010; Bentley, 2010; Graphisoft, 2010).

2.1.1 Parameterizing constraints for design computation

A parametric design process involves abstracting design concepts as collections of computable procedures, in which sequences of constrained operations are used to generate geometric objects. These relational constraints, imposed on the geometric objects, can assist design exploration in the different phases of design. Parameters are representative controllers for propagating design alternatives. The objective of parametric design is to enable design generation and to assist design exploration through computable handles in an efficient, generative, and, occasionally, algorithmic fashion.

The very first parametric modeling system was Sketchpad developed by Sutherland (1963). In Sketchpad, drawings were captured directly by user input from a light pen. Among some of its most influential features are automated design repetition, duplication, and change propagation though a Graphical User Interface (GUI). See Figure 2-1.

Figure 2-1 Sketchpad: A man-machine graphical communication system (Sutherland, 1963)
Image source: http://www.cadazz.com/cad-software-Sketchpad.htm

With the advent of Sketchpad came a concerted focus on computer-aided design and accompanying research into computer systems that could assist designers and into design automation resulting (now) in several generations of commercial CAD software. Parameters in these conventional CAD systems are often used statically, that is, only one-to-one relations could be specified with a default prescriptive constraint. The parameters served more like property placeholders storing, mainly, numerical values and sometimes, material properties of the constrained object. Utilizing parameters was also static, thus, making it difficult for designers to accommodate the dynamically changing nature of a design. Most designers had no choice but to use conventional CAD tools as a post-design process, in which the design was nearly complete. In other words, conventional CAD tools were used for drafting, merely replacing manual work.

To fulfill the need for efficiently making design changes and generating alternatives, sometimes algorithmically, the focus of contemporary CAD development has been geared toward a dynamically controllable environment, in which parameters can be defined by users and used as drivers for future alternative explorations. Information and operations between parameters are dynamically regulated and changes propagated in real time. In this approach, parameters are constructed and utilized differently. For instance, parameters that encapsulate design constraints between geometric objects vary in a number of ways such as: one-to-one, one-to-many, many-to-one, etc. Moreover, a parameter can be associated with a single numerical value or with compound information, which may include both data and operations. In a sense, the parameter becomes an additional design instrument in the generative design process.

The application of computing with design constraints 3provides the computational media with an active role in the design process enabling designers to accommodate potential future changes in real time. For instance, Kolarevic (1993) presents a computational environment— ReDRAW, in which geometric relations are employed for design conceptualization and exploration. In his study, a relations-based framework is proposed as a computational vehicle to restructure the underlying configuration and thus enable design exploration. Figure 2-2 illustrates an example of alternative configurations for Mario Botta's Casa Rotunda by manipulating the constructed relation.

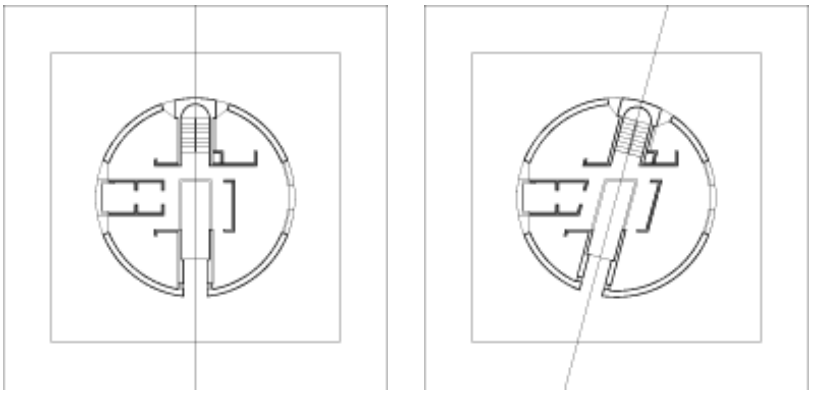


Figure 2-2 (*Left*) Original composition (*Right*) Restructured configuration by relational constraint
Image after Kolarevic (1997)

Likewise, Moustapha (2004, 2006) presents a formal approach to represent constructs of spatial and geometric relations, and describes a computational framework—ICE (Interactive Configuration Exploration) for architectural explorations. Kilian (2006) explored various heterogeneous constraints such as quantitative, geometric, topological and functional constraints, and demonstrated the potential of supporting form finding and manifestation through multi-directional constraint modeling. In short, all these research endeavors emphasize the importance of constructing computable relational constraints for design generation, propagation and exploration. Parameters are no longer used as holders for static information, but more as relational constructs to computationally regulate geometry. By employing constraints and through support from the computational media, design exploration can be expanded—the objective here is to support designers in investigating more possibilities.

To describe constraint-based modeling, *p-graphs* are introduced. A p-graph structure is a convenient way of representing and manipulating relationships between regulated objects. Such graphs feature acyclic directed structures, and depict how changes are propagated through the entire network.

2.1.2 P-graphs – An acyclic directed graph structure

Definitions: A *graph* is a structure comprising *nodes* and *edges*. Edges connect nodes pairwise. Edges can be *directed.* For any directed edge, $a \rightarrow b$, between nodes *a* and *b*, *a* is a *predecessor* of *b* and *b* is a *successor* of *a*. For any two nodes, *a* and *b*, *a* is an *ancestor* of *b* if there is a sequence of successor nodes from *a* and its successors to *b*. *b* is then referred to as a *descendent* of *a*; likewise, *a* is an ancestor of *b*. If *b* is a descendent of *a,* we say that there is a *path* from *a* to *b*. The length of a

path is equal to one less than the number of its nodes inclusive of the first and last node. A path has at least length one. If a node is a descendent of itself then the path is a *cycle*. A graph without a cycle is *acyclic*.

Graphs are used to represent parametric models. Figure 2-3 shows two graphs. The graph on the left is an acyclic directed graph in which every predecessor node can be 'processed' before its successor nodes in a left-to-right manner (Grey→Orange→Blue). The graph on the right, on the other hand, is not acyclic because nodes B, D and F, and likewise, nodes B, C and F form a cycle. Here, we cannot establish any deterministic order by which these nodes can be processed. A valid graph for parametric modeling is acyclic, and depicts how geometric objects can be hierarchically and parametrically constructed. Node order is important when traversing the tree for a solution. The nodes of a valid graph representation for parametric modeling can always be topologically sorted. We refer to an acyclic graph for parametric modeling as a *p-graph*.

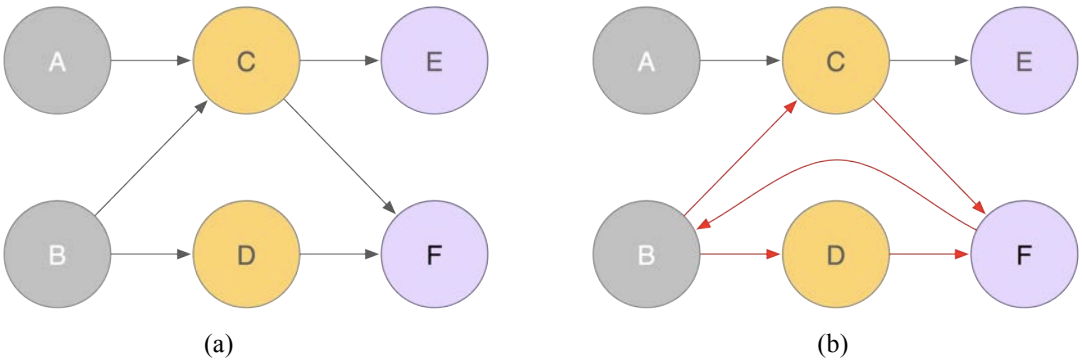


Figure 2-3 (*a*) Valid and (*b*) Invalid p-graphs

In a p-graph, information is processed at a predecessor level prior to being passed on to a consequent successor level in a strict hierarchical order, e.g., top-down as in Bentley Systems Generative Component (GC), which is compatible with Bentley MicroStation®, or left-to-right as in Grasshopper® (GH), which is tightly integrated with Rhino-3D (McNeel, 2010). Each connection between nodes has a direction, which is irreversible, and represents either a flow of data communication, or a collection of computing operations. In this sense, there is an essential difference between GC and GH in that GC provides a scripting environment with symbolic representation for graphs; on the other hand, GH relies on direct manipulation of the visual symbols, both geometrical and functional. In addition, nodes are also used differently. In GC, the graph merely visualizes the computed result, in which a node presents the data calculated from expressions in the specified

procedures. In GH, a node can be manipulated directly to store, execute mathematical calculations, and also output results. Links between nodes simply indicate types of data that are transmitted. In comparison, nodes of an acyclic directed graph are used more actively and directly in GH than in GC.

The word *parametric* here stands for the usage of parameters in both simple and compound fashions. A *simple* parameter is associated with single value only, such as a number or a string. A *compound* parameter, on the other hand, may be a geometric entity, composed of multiple values, for example, a point is a compound object consisting of a triple of numbers, namely, the x, y, z coordinates. A compound parameter may also be a function, which executes commands and outputs results. For instance, an energy node may consist of functional calls to trigger the simulation and retrieve the simulation results as the output.

The results of modifications on associated parameters and changes are propagated through successor levels of the p-graph unless otherwise explicitly paused or blocked. This manner of modeling gives designers greater flexibility for exploring variations in real time; at the same time, it makes modeling become somewhat more abstract without creating geometric objects directly. Typically, difficulties arise when converting vague design concepts into concrete computable components; this translation is neither straightforward nor easy for designers without formal training in programming. On the other hand, once a geometric object can be parametrically regulated, the resulting system can be adapted and extended for other propagations.

2.1.3 Computer-aided design tools for parametric modeling

There is a number of commercially available software to create geometric models by constructing relationships. These include the previously mentioned Bentley Systems Generative Component (GC), and Grasshopper® (GH). Other notable modeling software are Gehry Technologies' Digital Project™ (DP), compatible with CATIA®; Autodesk® Revit® Architecture; and GraphiSoft® ArchiCAD®. Of these, GC/Bentley, GH/Rhino and DP/CATIA are among the more commonly used for architectural applications. Amongst the Building Information Modeling (BIM) community, Revit and ArchiCAD are preferred.

Overall, such tools provide specific approaches for constraint modeling through their graphical user interface, programming language interface, or a combination of both. GC and GH are good representative exemplar tools. GC and GH share certain similarity of representation insofar as parametric design is concerned. Both employ an acyclic directed graph structure; in GC for symbolic representation, and in GH as a directly manipulable medium. A tree, the definition of which is made

precise in Section 2.1.3, is an acyclic graph structure in which nodes are connected by directed edges. Nodes represent geometric entities. Direct geometric constraints are constructed as edges representing relations between tree nodes. Certain functional constraints that may involve complex computations such as energy simulation, fluid dynamics, or structural analysis, etc., can be introduced as a special node, and may be potentially built from various functional objects among the disparate computational platforms. Owing to the potential complexity of constraint construction, various levels of encapsulation occur at distinct interfaces in the above computational environments.

2.1.4 Parametric modules

Abstraction and encapsulation are two essential concepts for parametric modeling. Abstraction enables conversion from design concepts to a set of computable parameters together with corresponding operations. Encapsulation groups sequences of operations and maintains all required data within a single entity for subsequent use. These two constructs make the parametric process both applicable and accessible within a computational media. During the course of designing, one often performs repetitive and other similar operations—it makes more sense to formulate such repetitive procedures as reusable modules. The key to successful module making is the flexibility to accommodate as many scenarios as possible. This same concept of making something reusable, as a modular component for repetition and duplication, is commonly seen in day-to-day architectural practice. For example, a surface panel is a physical module, which can be used to assemble a building façade. It can be varied in its dimensions and/or material properties. Within the parametric modeling approach, this surface panel is a virtual component, defined and constructed by parameters that govern its dimension as well as any relationship to the underlying surface properties. Ultimate manifestation of a surface is fashioned by propagating panel components and constraints through the entire surface domain. For any given well-structured parametric model, the physical performance of the surface manifestation can be analyzed. The analytical results can then further be utilized to optimize surface design.

A parametric module may contain one to multiple operations and data entities. In a p-graph, a module can be treated either as a single node or as a sub-graph consisting of multiple data and operational nodes. This adoption of module making makes for efficient “divide-and-conquer” design problem solving, in which a complex design problem is divided into smaller solvable sub-problems. Furthermore, owing to the continuous changing nature of designs, constructive modules offer flexibility for adapting to potential changes during the design process. Woodbury (2007, 2010) has addressed the importance of parametric design patterns using GC, and illustrated underlying

constructive principles for pedagogical purposes. These same design patterns were implemented in GH (Wang, 2010).

Figure 2-4 illustrates an example of a parametric module, which procedurally transforms a given polygon into a smaller inscribed polygon. The figure consists of two columns: left and right.

The left column illustrates the transformation rule and the right shows two corresponding parametric operations, rotation and insertion. The rotation angle, *r*, specifies the degree through which the given polygon is rotated about the centroid. The insertion parameter, t_n, is computed by the rotation angle, *r*, and the edge count of the input polygon. This is later utilized to interpolate the inscribed polygon as output. The parameter, *n,* is specified for the number of edges of the input polygon.

The bottom row illustrates the procedure in GH consisting of various geometrical components and numerical controllers. Overall, the entire constructive process can be divided into three groups: (1) input geometry and control, (2) parameterization and computation, and (3) output geometry. Figure 2-6 shows the representative p-graph of a resulting module.

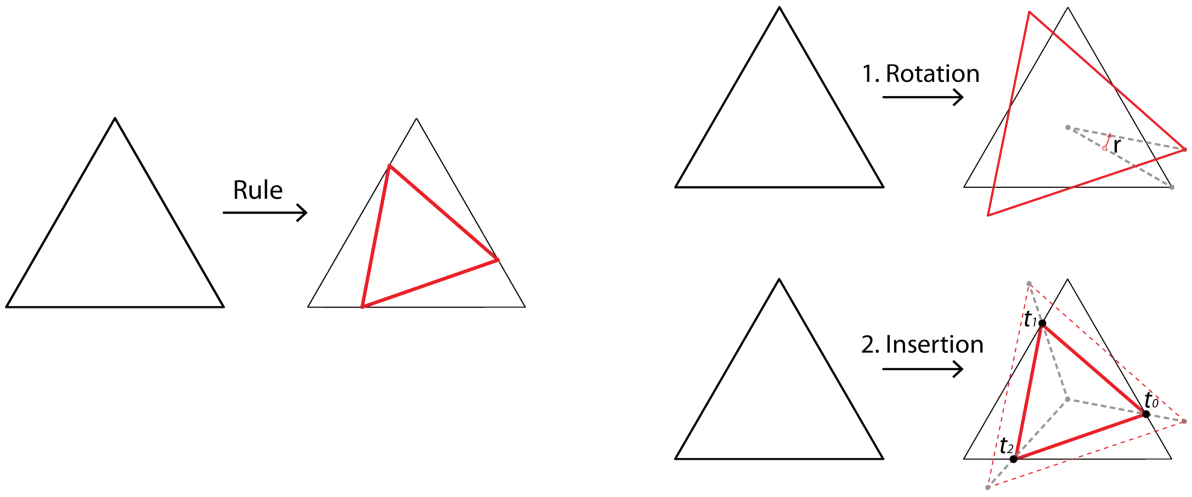


Figure 2-4 A parametric module for constructing an inscribed polygon
(*Left*) Transformation rule
(*Right*) Breakdown of required operations: rotation and insertion

Figure 2-5 shows an exemplar implementation of the parametric module shown in Figure 2-4 using Grasshopper/Rhino. The implementation consists of three segments in a left-right order:

(1) input geometry; (2) parameterization and computation; (3) output. Figure 2-6 illustrates the representative p-graph of the implemented module shown in Figure 2-5.

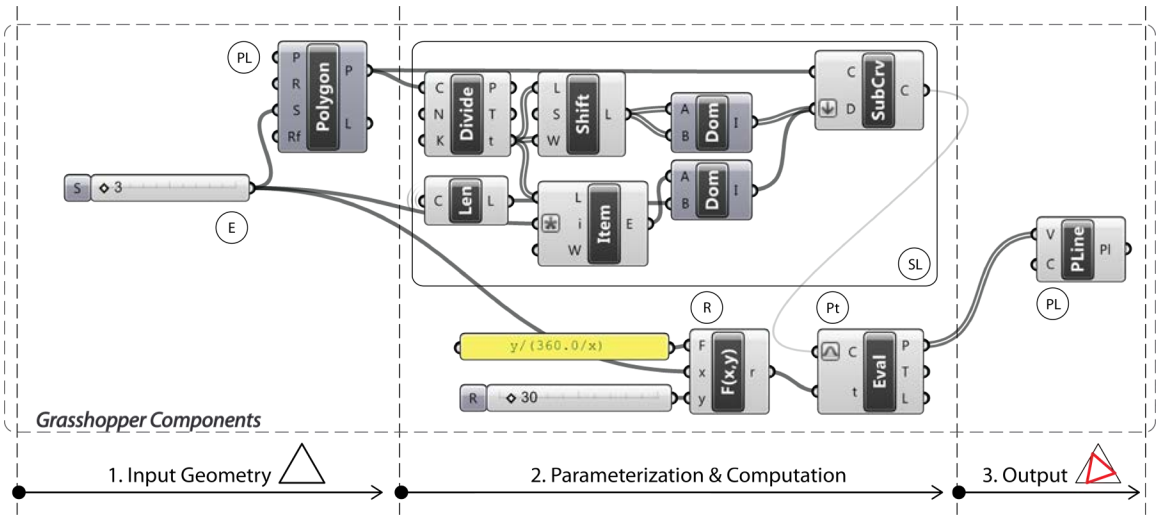


Figure 2-5 Implemented GH components for Figure 2-4

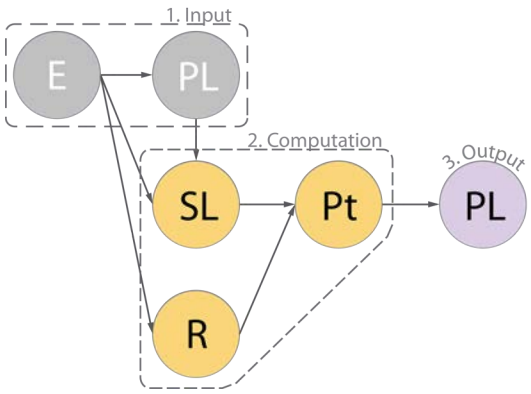
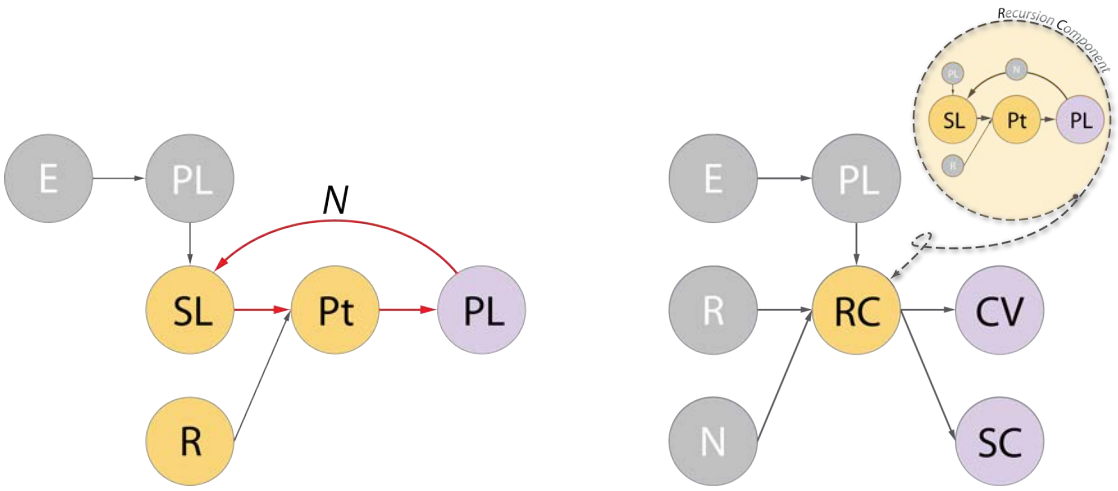


Figure 2-6 Representative p-graph of the parametric module in Figure 2-4

Suppose the objective is to apply the above procedure recursively—that is, the output polygon is employed as input for a number of iterations. We can consider this as a graph in two ways. See Figure 2-7. The top-left image illustrates the scenario of recursively applying the transformation procedure. This results in transforming a valid p-graph into a graph containing cycles. The path from nodes SL, Pt, PL, forms a cycle; in turn, this makes the graph an invalid p-graph. To address this

cycle problem, we consider the p-graph shown in the top-right, which demonstrates a solution obtained by encapsulating the recursion as a tree node, RC. This new now node, RC, maintains the recursion locally and thus keeps the overall p-graph structure intact. The bottom row shows the procedure implemented in GH. Note that a new parametric controller, *N*, has been added so as to define the number of iterations for the recursion.



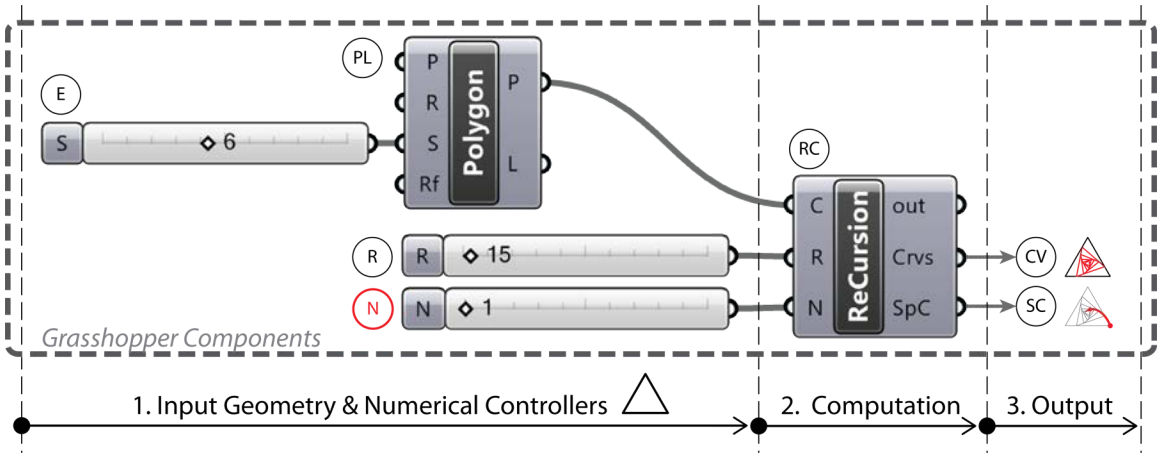


Figure 2-7 (*Top-Left*) Invalid p-graph (*Top-Right*) Valid p-graph (*Bottom*) GH components for the recursion module

In a p-graph, every node basically contains a runtime procedure, which specifies the computation for the given inputs, including geometric objects, numeric data, etc., and outputs the computed results.

In the above example, recursion occurs at tree node, *RC*, in which a *while-loop* is implemented for iterative operations. Figure 2-8 shows the programming interface provided by GH and the corresponding program code segments.

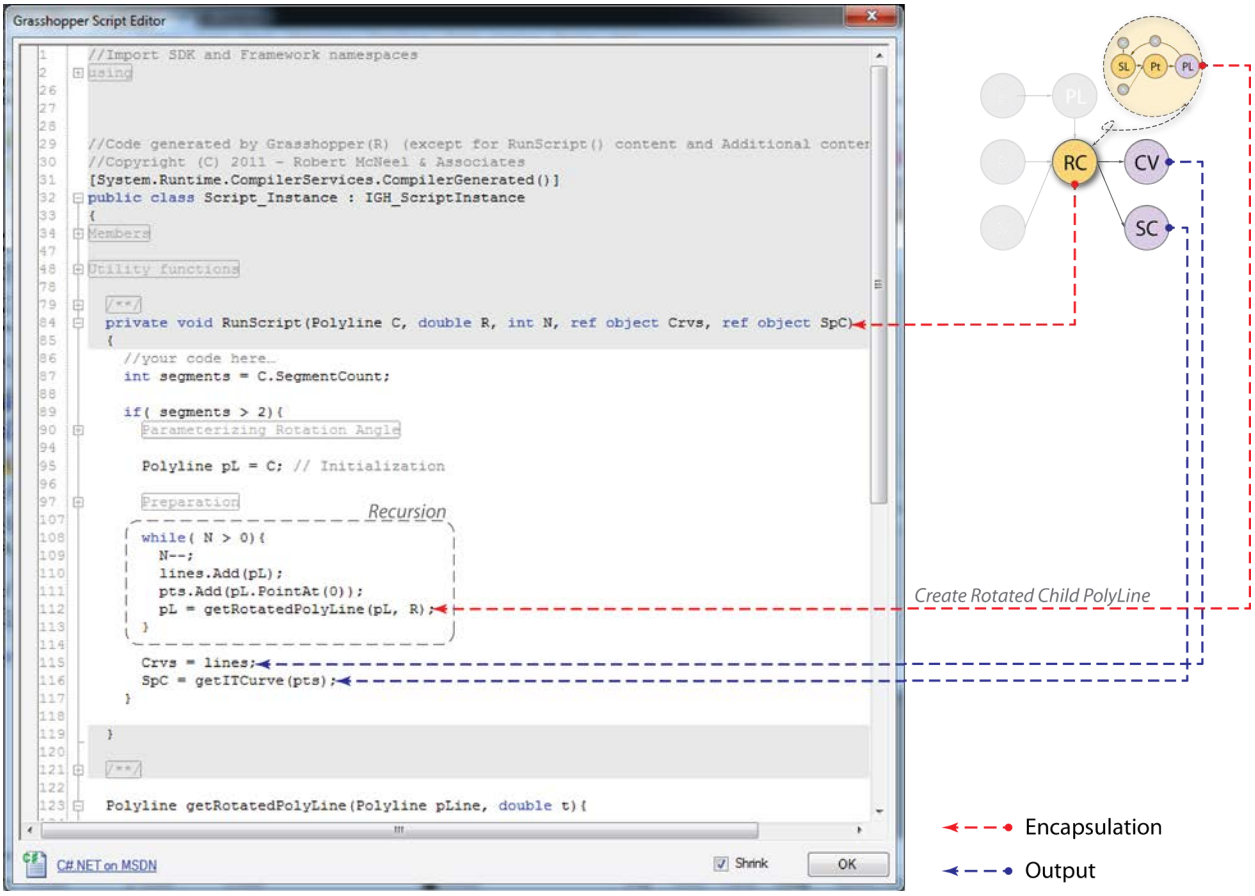


Figure 2-8 The programming editor in GH

Figure 2-9 illustrates a collection of propagated results for the following two-dimensional constraints: (1) the number of polygon edges, *E*, and (2) the number of iterations, *N*. The rotation angle, *R*, in this exercise is fixed and can be further extended to explore alternative generation in another dimension.

The main difference in the two p-graphs shown respectively in Figure 2-6 and Figure 2-7 lies at the level of abstraction and encapsulation. A well-structured module with a high level of abstraction and encapsulation may enable efficiency and flexibility for parameterization. However, from a

pedagogical point of view, it may also hinder understanding and learning for users with limited knowledge in geometry construction and, perhaps, programming.

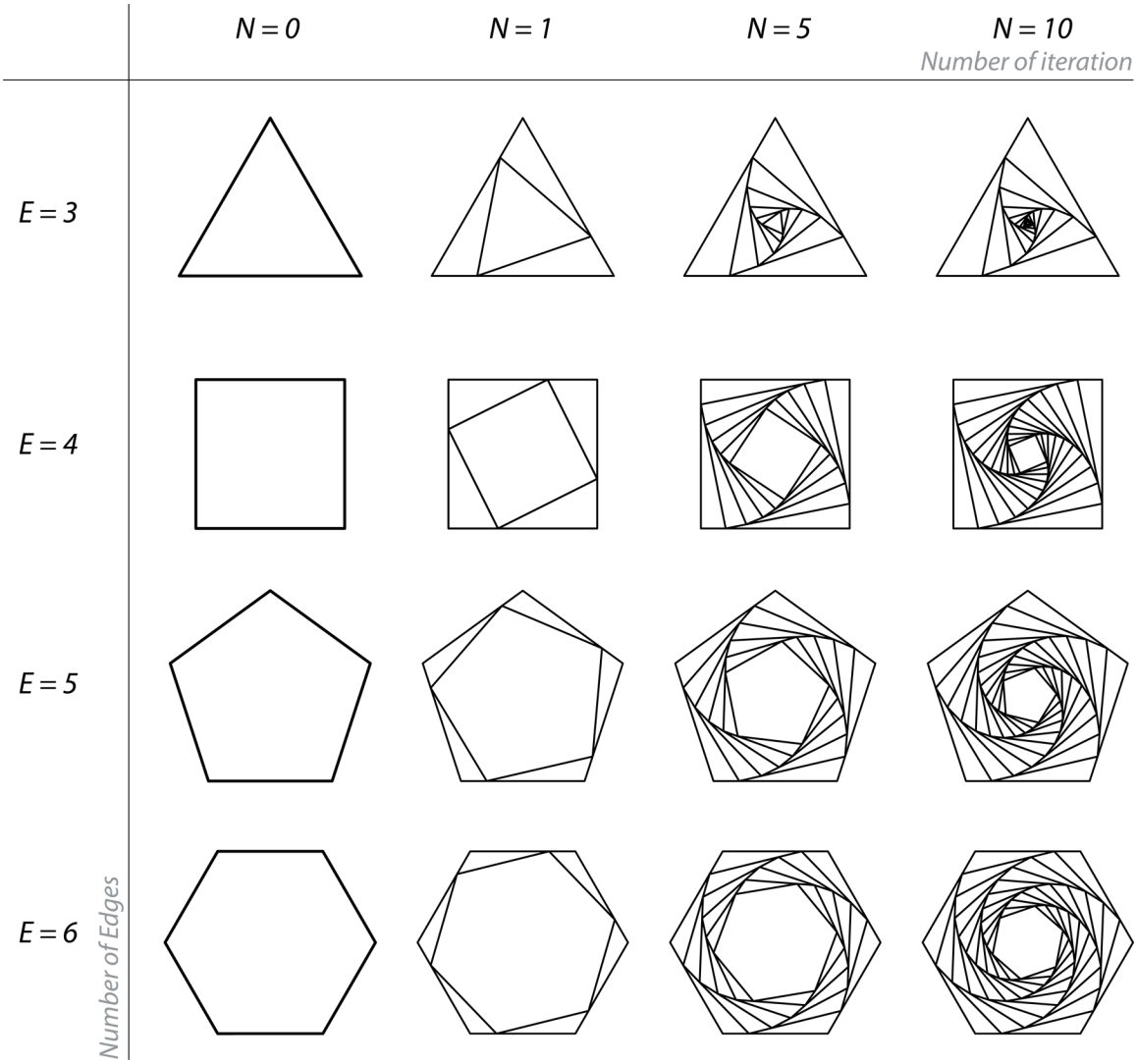


Figure 2-9 Propagating results from the recursion module

Applying the concept of dividing architectural design into a collection of modular components facilitates the parametric process becoming more amenable to iterative design. A module performs like an operation container, which can be procedurally defined and altered as the design progresses. For instance, consider a panel module, which in the design conceptualization phase can be as simple as a surface tessellation procedure. In this phase, the panel defines a polygonal face by its corner vertices in relation to a target surface. The module can be later expanded to include added concerns

relating to constructible building elements by recycling the relational information that can be gathered from the constructed geometry such as face to panel, edge to structure frame, etc. In order to fulfill the needs of building a reusable module, an understanding of constructive geometry principles is necessary; moreover, with the increasing interest in building complex geometries, it is essential that designers possess the programming skills needed for the parametric modeling process. In this dissertation, components of the research are presented as parametric design modules towards solving the surface tessellation problem. In doing so, the steps in relation to the problem definition—namely tessellating surfaces with irregular boundary conditions, and corresponding problem solving strategies are described and demonstrated in Chapters 4 and 5.

2.2 Mesh-Surface Reconstruction

When dealing with complex geometries it is common in contemporary architectural practice to subdivide a surface into components that are easily fabricated. When approximating a target surface, subdivision typically involves a single kind of polygonal face. Planar faces are always preferable for physical fabrication and from the perspective of manufacturing techniques, production time, production cost, constructability, and so on, are more feasible and efficient in comparison to making non-planar panels. As an example, since triangles always guarantee planarity, it is not surprising that the more commonly seen built freeform projects employ mainly triangulated parts. Nonetheless, there are digital fabrication techniques, especially in the automobile and aerospace industries that use more complex shaped fabrications. To draw the connection from surface discretization to physical constructability, in Section 2.2 surface subdivision techniques are examined, mainly meshing, in relation to the types of triangulation and constructible properties inherently associated with an underlying surface.

2.2.1 Triangulation with planar faces

Although a three-dimensional triangle can be viewed as a flat planar face, easily cut from or manufactured out of sheet material, there are, however, certain triangular configurations, which are better than others in terms of constructability. For instance, a configuration with skinny triangles and widely diverse panel dimensions may not be suitable for fabrication due to the physical constraints embedded in material and manufacturing machinery. Only properly constrained triangulations lead to well-structured configurations; these, in turn, ensure successful manifestation.

Among the more widely adopted tessellating algorithms is the *Delaunay triangulation* (Delaunay 1934; Lee and Schachter, 1980), which is used to solve a variety of computational geometry problems (Berg et al., 2008), for instance, automatic mesh generation with optimized angles. We consider a set of points, P, where every point in P is a vertex in the triangulation. A Delaunay triangulation, DT(P), is a network in which no point in P is inside the circumcircle of any triangle in DT(P) not containing the point. Figure 2-10 illustrates a violation of the Delaunay criterion. In the left image, which highlights the edge, P_0-P_2, point P_3 is inside the circumcircle of the triangular face formed by points P_0, P_1, and P_2. Likewise, P_1 is inside the circumcircle of the triangular face formed by points P_0, P_3, and P_2. By removing the violating edge, P_0-P_2, and inserting new edge, P_1-P_3, we can produce a valid Delaunay triangulation, shown in the right image of Figure 2-10. A notable property of such triangulations is that all minimum angles are optimized such that skinny angles are removed. In a sense, by conforming to the triangulation constraints as stated above, skewed triangles always violate the constraints and thus are removed. When extended to three-dimensional space, a circumsphere rather than a circumcircle is employed to verify the Delaunay criterion.

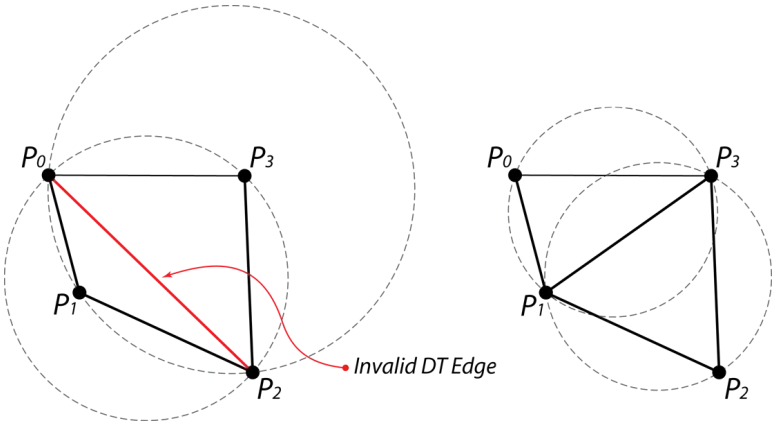


Figure 2-10 Criterion for Delaunay triangulation (Delaunay, 1943)
(*Left*) Invalid Delaunay triangulation edge (colored shaded red solid line)
(*Right*) Valid Delaunay triangulation

For mesh automation, a constrained Delaunay triangulation (CDT) is often used (Seidel, 1988; Fleischmann, 1999). CDT may potentially contain non-Delaunay edges, in particular, at the boundary edges. In CDT, the boundary constraint dominates the Delaunay criterion to ensure integrity of the mesh boundary. Figure 2-11 illustrates a non-Delaunay edge, e, with a highlighted half circumcircle. The edge point, P, which is within the smallest circumcircle, will not affect the edge, e.

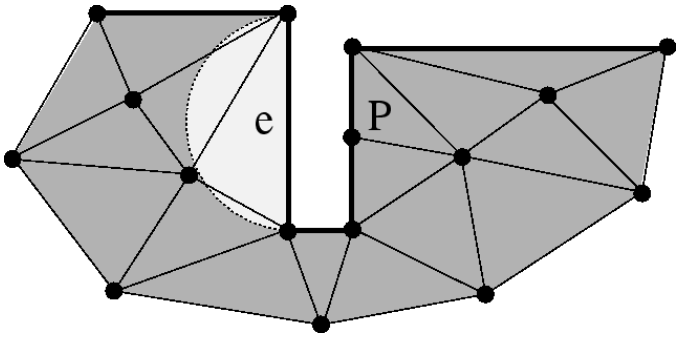
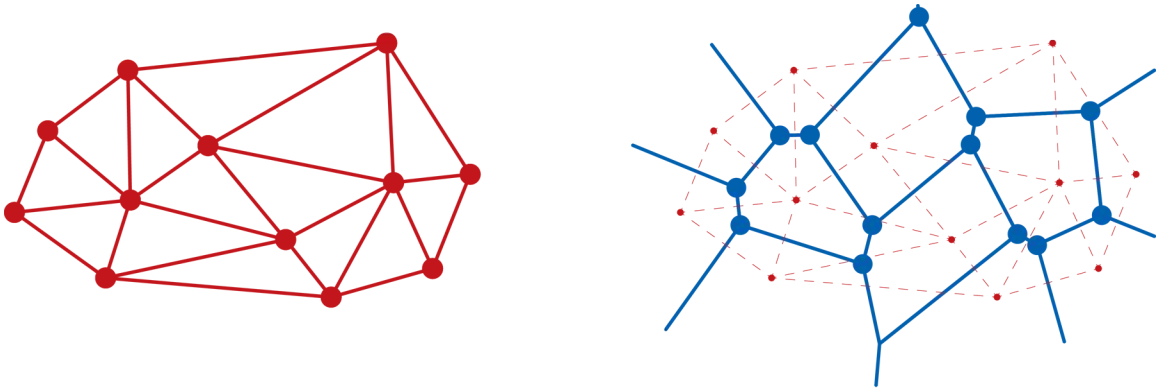


Figure 2-11 A constrained Delaunay Triangulation with a non-Delaunay edge, e
Image after Fleischmann (1999: http://www.iue.tuwien.ac.at/phd/fleischmann/node52.html)

The dual of a Delaunay triangulation is a Voronoi diagram. A *Voronoi diagram*, or *Voronoi tessellation*, (Voronoi, 1908; Dirichlet, 1850; Aurenhammer, 1991) divides space into cells called *Voronoi cells* or *site*. For any given set of points, P, each point in P is associated with a Voronoi cell and all points in the cell are closer to the associated point than all other points in P. The left side image in Figure 2-12 shows a Delaunay triangulation, DT(P), with twelve points, or, sites; the right image illustrates the corresponding Voronoi diagram overlaid on top of the underlying DT(P). Figure 2-13 shows the steps in the construction; by intersecting the perpendicular bisectors of connecting Delaunay edges at common points of interest. The newly created intersection points are Voronoi vertices; they are also circumcenters of the triangles in the DT.



Delaunay Triangulation *Voronoi Diagram*

Figure 2-12 (*Left*) DT-Delaunay triangulation (*Right*) Voronoi Diagram

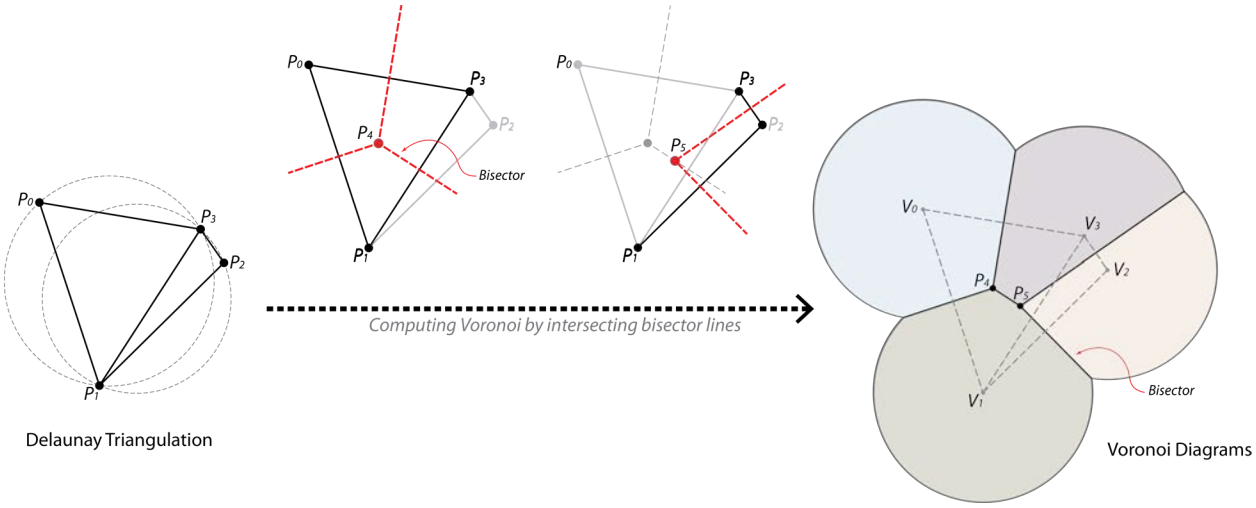


Figure 2-13 Constructing Voronoi diagram by intersecting bisector lines of Delaunay edges

2.2.2 Quadrangulations

A quadrangulation is a tessellation formed by quadrilateral (*quad*) faces. Consider the surface shown on the left in Figure 2-14. The right image illustrates a subdivision of the surface using only quad faces. In three-dimensional space, a quad face is not necessarily planar. One simple way to verify flatness of a quad face is by examining its face warping.

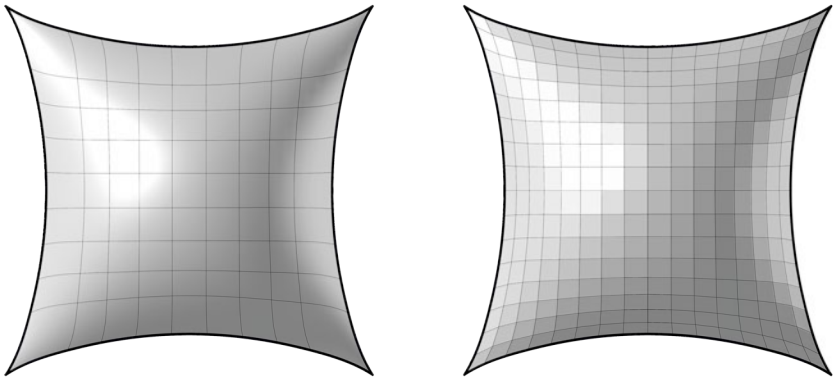


Figure 2-14 (*Left*) Target surface; and (*Right*) Surface quadrangulation

Warping is a surface property, which indicates the distortion of the face. Figure 2-15 illustrates warping of a quadrilateral face. We consider a parameter, *d*, which is the distance between mid point,

$MP_{\_BD}$, on the diagonal line formed by P_B and P_D, and mid point, $MD_{\_AC}$, on the other diagonal line formed by P_A and P_C. When d is zero, the quad face is called flat; otherwise, the face is warped.

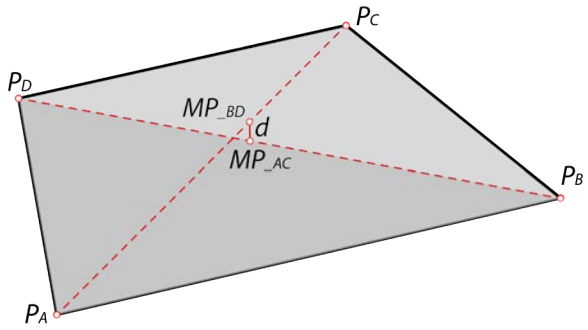


Figure 2-15 Warping—distortion of the quad face

In Figure 2-16 and Figure 2-17, the quadrangulated (*quad*) mesh is analyzed for various surface properties, namely, *Gaussian curvature*, *mean curvature*, *maximum principal curvature* and *face warping*. For any point on a surface, there are two principal curvatures, one is *maximum* principal curvature and the other is *minimum* principal curvature. The maximum principal curvature is the curvature of the curve that has the maximum curvature value from all the intersecting curves by the frenet frames at the point of the interest. The other principal curvature is the minimum principal curvature direction, which has the minimum curvature values. Gaussian curvature is the product of two principal curvatures. Mean curvature is the average of these two principal curvatures.

In the following images mesh faces are colored by sampling the curvature and warping information from four corner vertices. For instance in the face warping analysis, minimally warped faces should exist at surface areas, in which the quadrilateral face distortion is minimal. Figure 2-18 illustrates filtering of mesh faces for varying face flatness or warping thresholds. Here, in the illustration, a threshold parameter, w, is introduced and determined by calculating the ratio of the distance between the two mid points of the diagonal lines to the dimensions of the target quad face. By gradually increasing the parametric threshold, w, flat faces appears from the surface center to the boundary edges and finally to corner locations, in which quad faces are mostly distorted. Overall, the central areas have minimal curvature discrepancy and the four corner areas have maximal curvature discrepancy. In short, the difference in curvatures provides more practical information or constraints for optimizing planar quadrilateral meshes.

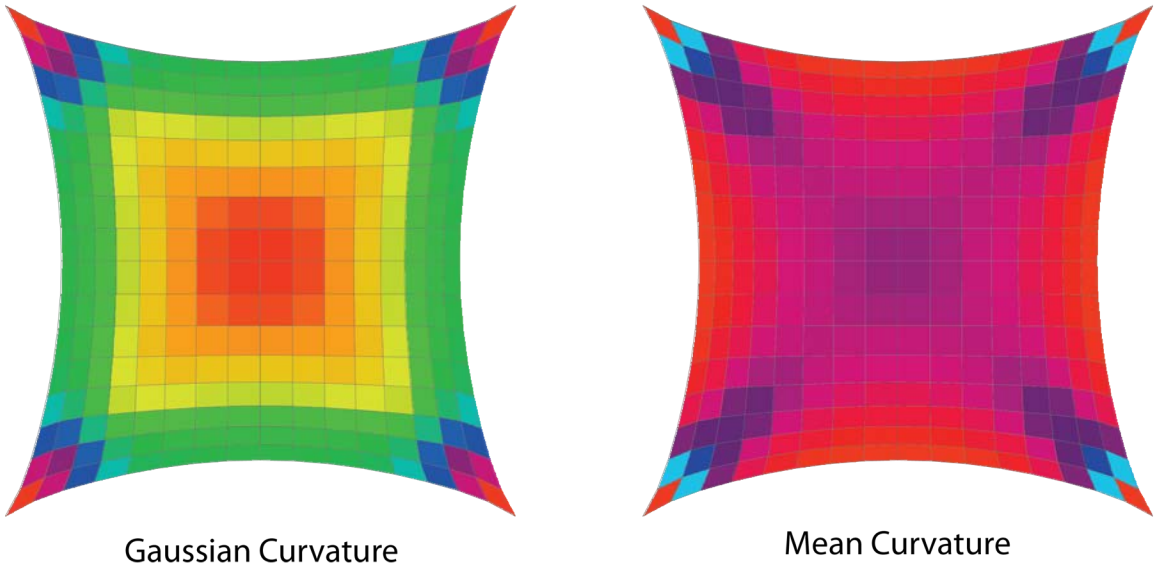


Figure 2-16 Mesh analysis by Gaussian curvature and mean curvature

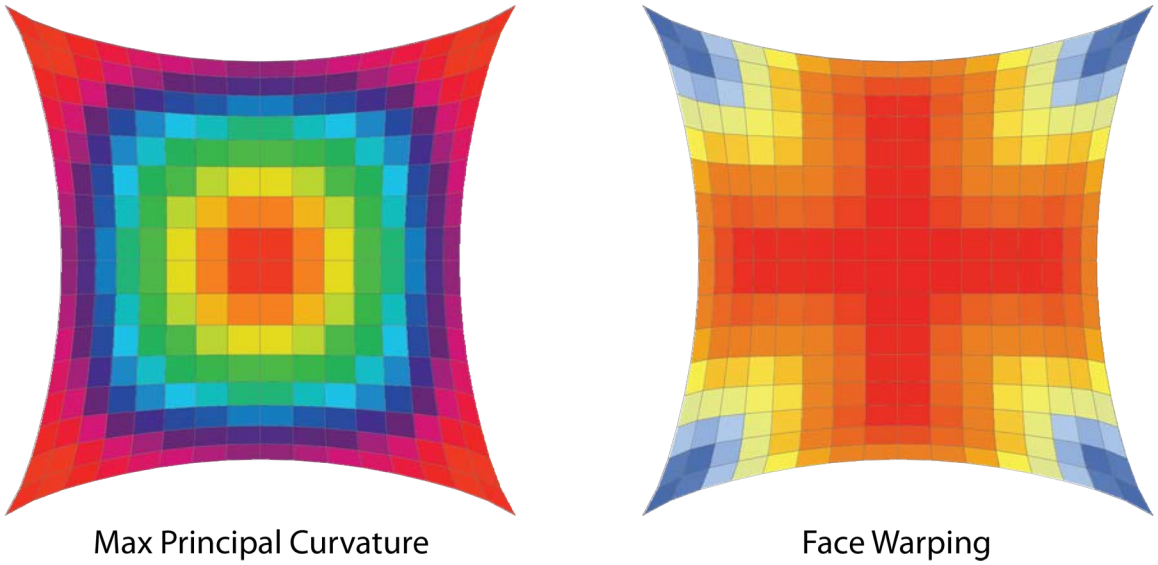


Figure 2-17 Mesh analysis by max principal curvature and face warping

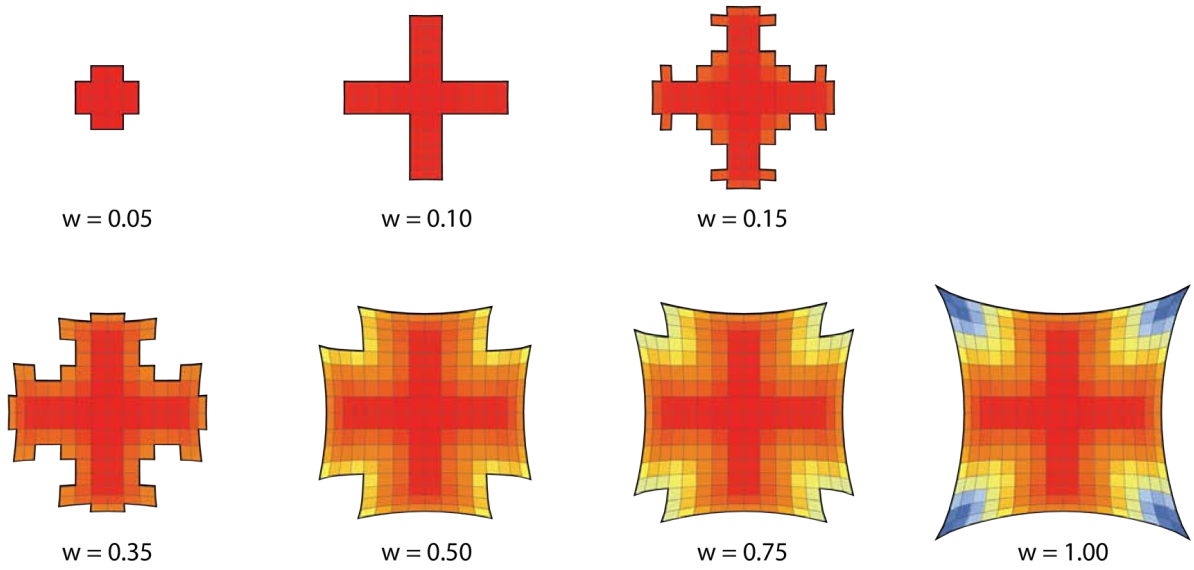


Figure 2-18 Mesh flatness analysis

By increasing the threshold of face warping, various numbers of "flat" faces are filtered

2.2.3 Subdivision

Once a surface has been tessellated using triangular or quadrilateral faces, subdivision schemes can be further employed to convert a coarse mesh into a smoother mesh with smaller face elements. We consider two schemes: Loop and Catmull-Clark subdivisions.

Loop subdivision (Loop, 1987) uses triangular subdivision in which each triangular face is replaced by *four* smaller triangular faces, including three *corner faces* and one *central face*. The subdivision process initiates from existing vertex modulations and new edge midpoint insertions. For each triangular face, corner vertices are combined with two connected mesh-edge midpoints to form three corner faces. One central face is constructed by three mesh-edge midpoints. At the end of each iteration only the mesh-edge midpoints remain on the original mesh surface.

Catmull-Clark is a quadrilateral-based subdivision scheme, which converts any potential polygonal mesh into a smooth quadrangulated surface (Catmull and Clark, 1978). The resulting topology is smoothed by a recursive bicubic B-Spline subdivision function. Similarly, new vertices are created at the middle of original mesh edges and get displaced to new locations by computing with coefficients derived from the underlying topological connectivity. Figure 2-19 shows three subdivision results by both subdivision schemes on an octahedron—a polyhedron with eight

congruent triangular faces. Loop subdivision is shown on top and Catmull-Clark subdivision is shown at the bottom. The advantage of using these computational schemes is that they offer a 'smoothing' effect after optimization. These techniques are applicable to surface tessellation when one considers refining the mesh topology for a smooth appearance.

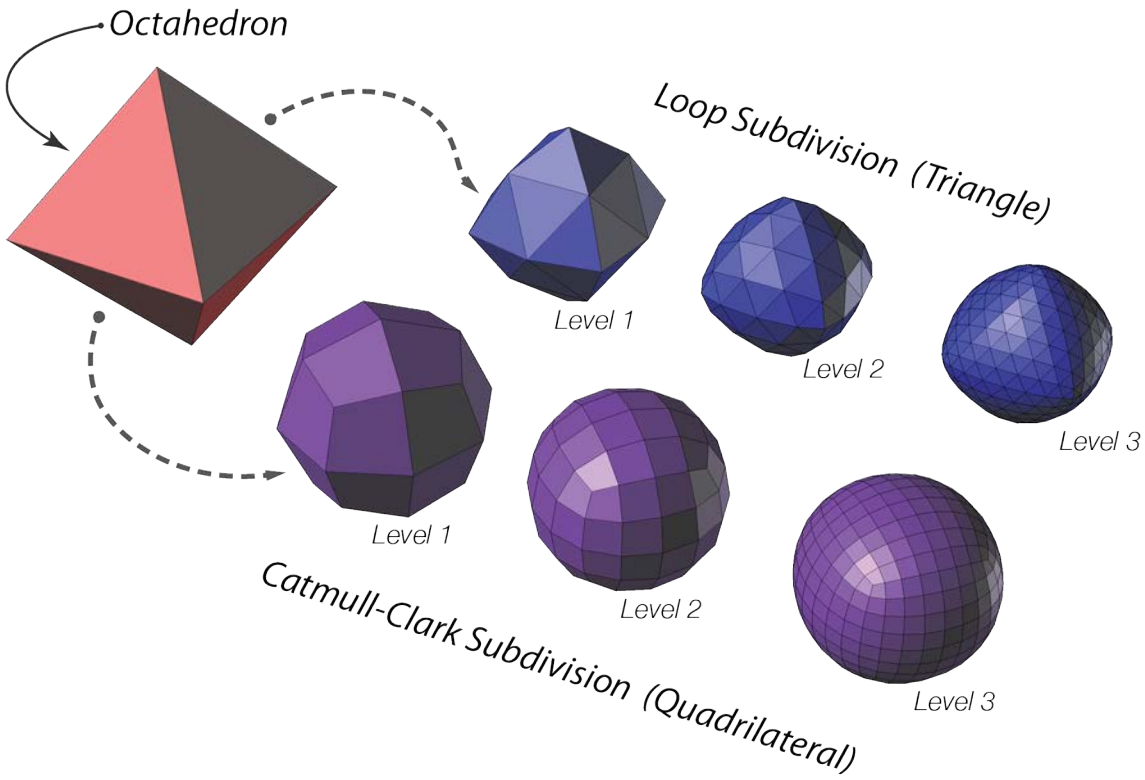


Figure 2-19 Mesh refinement

Loop subdivision (Loop, 1987) and Catmull-Clark subdivision (Catmull and Clark, 1978)

2.2.4 Planar quadrilateral mesh

A particular type of planar mesh, called the *Planar Quadrilateral (PQ-) Mesh*, offers potential for freeform design and fabrication applications (Pottmann et al., 2006a-b; 2007b). The clearest advantageous characteristic of a PQ-mesh is its quadrilateral planar faces, which is preferable for fabrication. It is straightforward to specify 'planarity' using a threshold. When this threshold is close to zero, the face can be considered to be planar. The threshold can be regarded as a measure of warping. The counterpart in the real world is in the way sheet material is viewed as planar. Most

sheet material has a degree of warping and as long as it is within a controllable threshold, the sheet can be treated as planar.

To construct a PQ-mesh, principal curvature properties of a surface are first examined. For every point on a surface, there are two principal curvatures; these correspond to the maximum and minimum values of the curvature at this point. Thus, at each point there are two principal curvature directions. For any surface, a principal curvature line is a line that always maintain tangential to the principal curvature directions on the surface. Therefore, at each point there are two principal curvature lines intersecting at right angles. The network of these two principal curvature lines can then be employed to form a mesh with quadrilateral face elements. See Figure 2-20. With further optimizing the planarity of each quad face element, a planar-quadrilateral mesh (PQ-mesh) can be derived (Liu et al., 2006). The PQ-mesh is treated as a discrete analogy of this network formed by principal curvature lines.

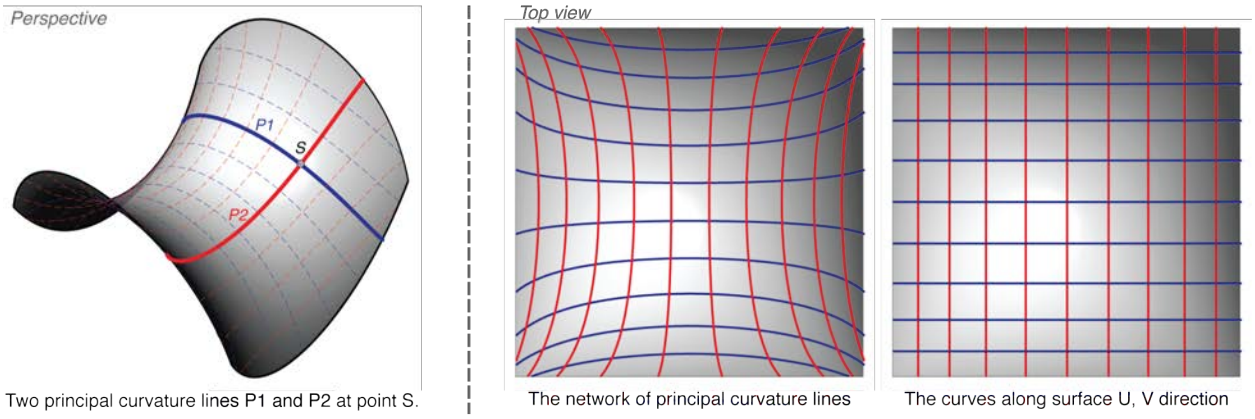


Figure 2-20 (*Left*) Principal curvature lines of a saddle surface
(*Right*) Comparison of principal curvature lines and *UV* curves

Meshing a surface with planar quadrilaterals is not easy, and relies, mainly, on a prescriptive procedure, which operates on the given surface. Figure 2-21 illustrates a sequence of meshing results after *PQ*-mesh optimization. The optimization relies heavily on how close the initial coarse mesh can generalize to—that is, increasingly resemble—the given surface. The generalization begins with the principal curvature analyses and the network of principal directions, which can then be used to create a coarse initial mesh. In Figure 2-21, given a coarse mesh in the left-most image, it is possible to generate an output mesh with planar quadrilateral faces by iterative subdivision and optimization.

However, it is not always possible to generalize an initial mesh. This still remains an unsolved research problem. For instance, it is difficult to reverse engineer a coarse representative mesh for the surface shown in Figure 2-22 (Pottmann et al., 2007a), which has a large variation in curvature. Currently, the initial coarse mesh is supplied by manual input.

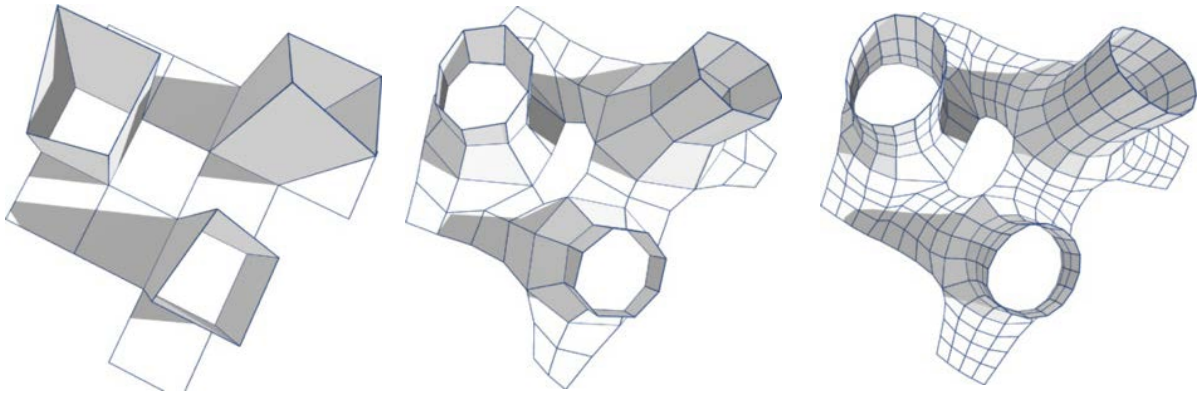


Figure 2-21 The generative process of *PQ* meshes
by iterative applications of Catmull-Clark subdivision and *PQ* perturbation. Image after Liu (2006)

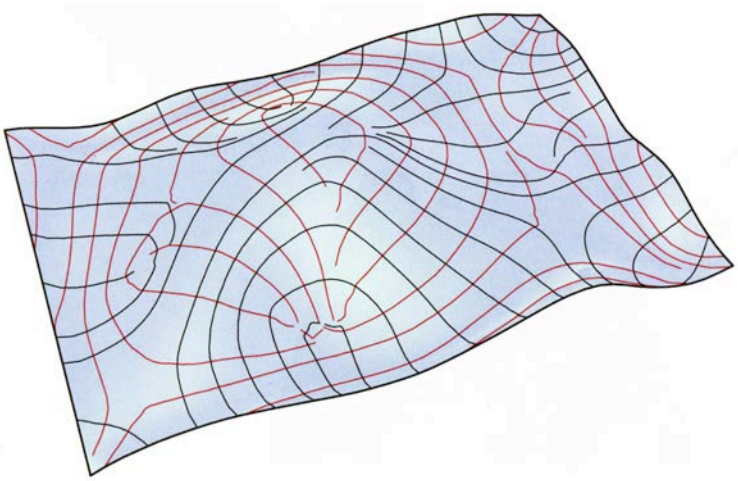
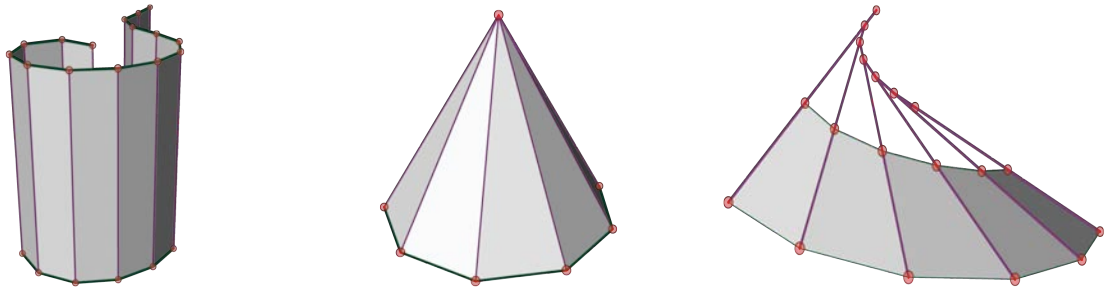


Figure 2-22 Large variations of cell sizes and directions from the network of principal curvature lines are not suitable as the basis for the layout of a *PQ* mesh.
Image from *Architecture Geometry* (Pottmann, et. al, 2007a)

2.2.5 Developable surfaces

Another kind of mesh comprising just planar faces is the *developable surface*, which is amenable to fabrication. For instance, a developable surface is characterized by the property that it can be mapped isometrically onto a plane. An *isometric mapping* preserves distance (as well as Gaussian curvature and surface area) (Portmann et al., 2007a). The planar image of a developable surface is called its *development*.

It is instructive to think of developable surfaces in terms of envelopes of surfaces. The *envelope* of a one-parameter family of surfaces is tangential to each surface in the family along the characteristic curve in that surface. A developable surface is the envelope of a one-parameter family of planes (Liu et al., 2006; Pottmann et al., 2007b). It turns out that developable surfaces are also *ruled surfaces* in which through every point there is always a straight line that lies on the surface. There are three basic kinds of ruled surfaces: i) rulings are parallel (that is, the surface is developed from a cylinder); ii) rulings are concurrent (that is, the surface is developed from a cone with its apex as the point of concurrency); and iii) rulings are tangential to a spatial curve. (Such a surface is also termed a *tangent surface.*) Figure 2-23 illustrates the three basic kinds of ruled surfaces.



a. Cylindrical surface b. Conic surface c. Tangent surface of a spatial curve

Figure 2-23 Basic kinds of ruled surface

2.2.6 Bubble mesh

The *bubble mesh* is an automatic mesh generation algorithm, which presents a distinct way of triangulating non-manifold geometry using a sphere packing technique (Shimada 1993; Shimada and Gossard, 1995). This method was devised for spheres (bubbles) packing via physical-based relaxation optimization; results show its strength in reducing the number of ill-shaped triangles and offering control over the dimension and directionality of mesh face elements. The resulting mesh face can be

triangular or quadrilateral. This technique is useful for a number of types of analysis that require geometric objects to be described in a well-structured discrete fashion, for instance, finite element, structural and heat transfer analyses (Kenji and Gossard, 1998).

Figure 2-24 illustrates surface triangulation via bubble packing. Briefly, bubble meshing involves three stages: (1) retrieving topological and geometric conditions; (2) packing the given domain with bubbles; and (3) generating the mesh.

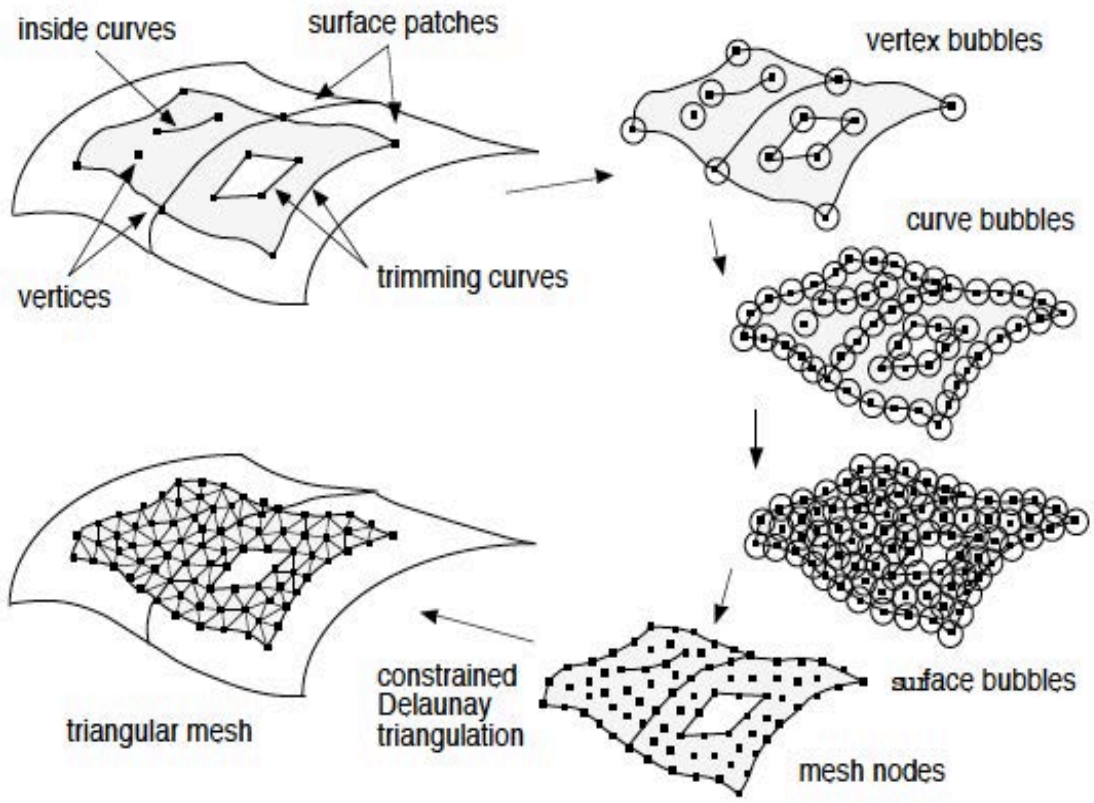


Figure 2-24 Surface Triangulation via bubble packing. Image from Shimada and Gross (1998)

For the first stage, the input surface is examined and peripheral conditions such as surface boundary and trim curves are identified.

Second, bubbles are placed in dimensional order to resolve topological and geometric constraints. In this stage, bubbles are packed on vertices first. Once the vertices have been packed, boundaries (edges) are examined next. Finally, bubbles are injected into faces and are optimized by physical-base relaxation (or force-balancing) and population control. This process is briefly described. An inter-bubble force is introduced to identify the following packing conditions for optimization: (1) two

bubbles overlap creating a repelling force; (2) two bubbles touch each other and remain stable; (3) two bubbles are too far apart resulting in an attraction force. Through inter-bubble forces, a physically-based simulation is utilized to search for an equilibrium configuration where all bubbles are placed in a stable state. Furthermore, in order to choose a sufficient number of bubbles for a given domain, adaptive population control is employed. Population control examines the ratio of local overlapping bubble populations. Significantly overlapping bubbles are removed, and bubble clusters lacking sufficient bubbles are increased.

The last stage in the optimization is to connect the centers of the tightly packed bubbles by a *constrained Delaunay triangulation* (Seidel, 1988). This ensures the best topological connections and conforms align with the surface boundary.

The advantage of utilizing bubble mesh is in the precise control of size, anisotropy and directionality of the generated mesh elements. During the meshing process, a guiding tensor field is employed to specify desired anisotropy and directionality for subsequent meshing operations. For instance as shown in Figure 2-25, a variety of two-dimensional quadrilateral meshings are generated by varying isotropic and anisotropic characteristics using a 2x2 tensor field (Viswanath et al., 2000; Shimada, 2011). Similar application can be extended to three-dimensional meshing using a 3x3 tensor field with various polygonal mesh face elements, such as quadrilaterals or hexagons (Shimada et al., 2000; Viswanath et al., 2000; Yamakawa and Shimada 2003; Vyas and Shimada, 2009).

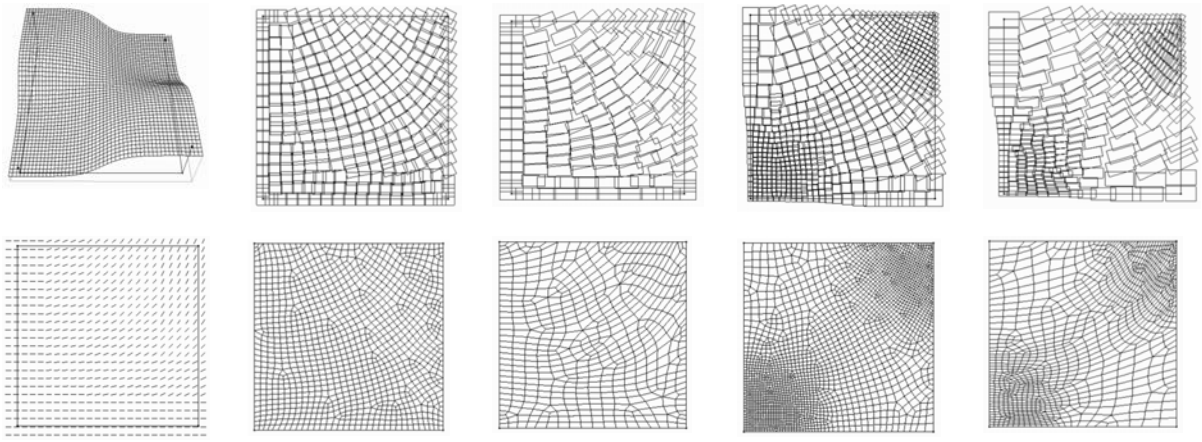
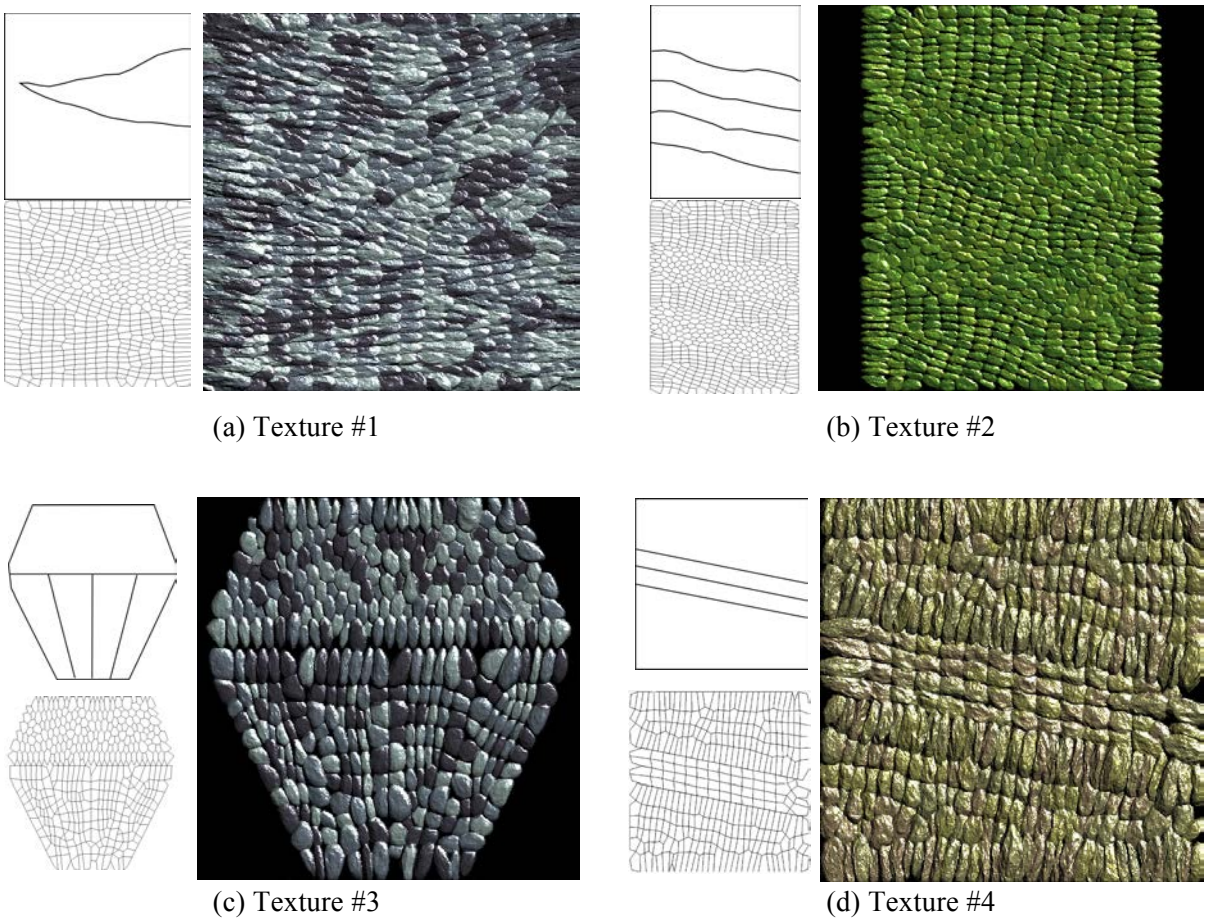


Figure 2-25 Meshing control of size, anisotropy and directionality by 2x2 tensor field for a two-dimensional meshing problem. Image after Viswanath et al. (2000)

In Figure 2-26, Itoh et al. (2003) presented potential applications by first tessellating regions of interest into a set of peudo-voronoi polygons and later triangulated using the Loop subdivision scheme (Loop, 1987) to generate organic textures. In this research, the capability of controlling mesh anisotropy and directionality facilitates the process of organic texture generation and in terms provides a computational handler for users to customize realistic textures with ease.



(a) Texture #1 (b) Texture #2

(c) Texture #3 (d) Texture #4

Figure 2-26 Organic texture generation
For each example: (*Top-Left*) Input boundaries; (*Bottom-Left*) Psuedo-Voronoi polygons; and (*Right*) Generated texture. Image from Itoh et al. (2003)

2.3 Fabricating Architectural Geometry

Digital fabrication techniques, borrowed from aerospace and automobile industries, make possible realizations of complex geometries (Kolaveric, 2005a-b; 2008). A basic strategy for digital

fabrication in current architectural practice is to approach it as top-down decomposition. Quite simply, decomposition considers how a target freeform surface is segmented with easy to construct face components; to this end, simple geometric elements are preferred (e.g., triangular panels). Two-dimensional cutting techniques can be then easily employed for shop production and on-site final assembly, within controllable time and precision. In general, to realize freeform structures, two aspects of fabrication need to be examined: firstly, decomposition strategies; and secondly, applicable techniques for fabrication. These two aspects are closely related and should be considered in any design process. It appears that the closer these parameters are integrated, the better the results achieved in practice.

Figure 2-27 illustrates two designs by Norman Foster and Partners: the Elephant House canopy and the City Hall in London. Both demonstrate the advantages of integrating physical constraints with underlying geometry construction in real architectural applications.

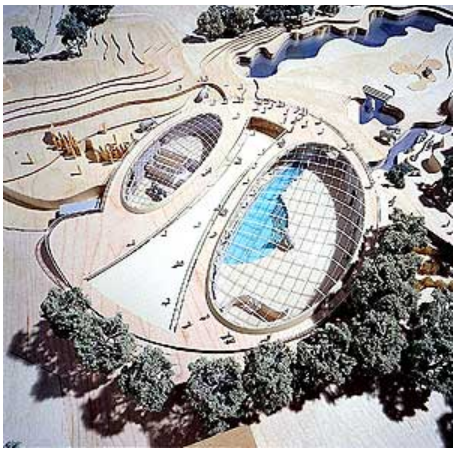
a.  b. 

Figure 2-27 a. Elephant House Canopy[2] ; b. London City Hall by Normal Foster and Partners[3]

In the canopy design, decisions are deferred until late in the process while making structural member generation feasible via the precision and efficiency derived from the principal geometric setup. The base geometry for the canopy is the torus, which can be mathematically constructed by the revolution of a circle around an axis. Mathematically, the revolution gives the surface a nice feature for discretization—namely, a planar quadrilateral patch can be derived directly from the principal

[2] Image after Foster & Partners from *arcSpace.com*, last accessed on April 5th, 2010.
[3] Image after Foster & Partners from *+math.org*, last accessed on April 5th, 2010.

curvature lines. The constructive principle of revolution makes fabrication feasible and manageable, even for this type of dome-like surface.

The City Hall project, on the other hand, demonstrates how parametric schemes used on a cone surface development can be realized for flat panel fabrications. The initial idea for the City Hall is a pebble-like form, which was later approximated by a collection of partial cone strips. A cone strip is a family of the cylindrical surface, which can be easily decomposed into planar quadrilateral faces. These two designs clearly highlight the advantages of incorporating constructive principles throughout the process from design to fabrication. The constructive principle herein captures the underlying mathematical form of the target surface and in turn makes feasible its ultimate manifestation.

Essentially, converting a curvilinear surface into a set of constructible components can be treated as a meshing process, which has been well researched from both engineering and architectural disciplines (Liu et al., 2006; Pottmann et al, 2006a-b; 2007b). In the following sections, we consider three types of manifest examples in relation to the manufacturing techniques: (1) flat polygonal panels, including both triangular and quadrilateral panels; (2) single-curved panels; and (3) double-curved panels. Many of the images shown in this section are adapted from Schodek et al. (2004) who provide a comprehensive documentation on how computer-aided manufacturing applications are implemented within various design/production contexts and fabrication environments.

From a fabrication perspective, fabrication techniques can be categorized into four basic types: (1) two-dimensional cutting; (2) three-dimensional subtraction; (3) three-dimensional deposition; and (4) formative generation (Kolaveric, 2005a; 2008). There is a close relationship between these categories and tessellation types. For instance, the cutting-based approach is suitable for planar construction. The three-dimensional techniques can be applied to single-, or double-curved panel fabrication.

2.3.1 Flat polygonal panels

These are panels constructed from planar faces. Since triangular faces are always flat, they are easily fabricated using two-dimensional cuts; for example, profile cuts using a CNC[4] machine. Tessellating a three-dimensional surface into planar panel components is analogous to approximating a three-dimensional geometry with two-dimensional counterparts, which are feasible by using digital fabrication machinery. From a practical point of view, embedding the fabrication constraints, such as

[4] Computer Numerical Control

material size or dimension of CNC cutting bed into the geometry construction and optimization process can in turn facilitate physical production in time, cost, and so on. Figure 2-28 illustrates the CNC cutting diagram with three different techniques, namely, Oxy-Fuel, Plasma and Laser cutting.

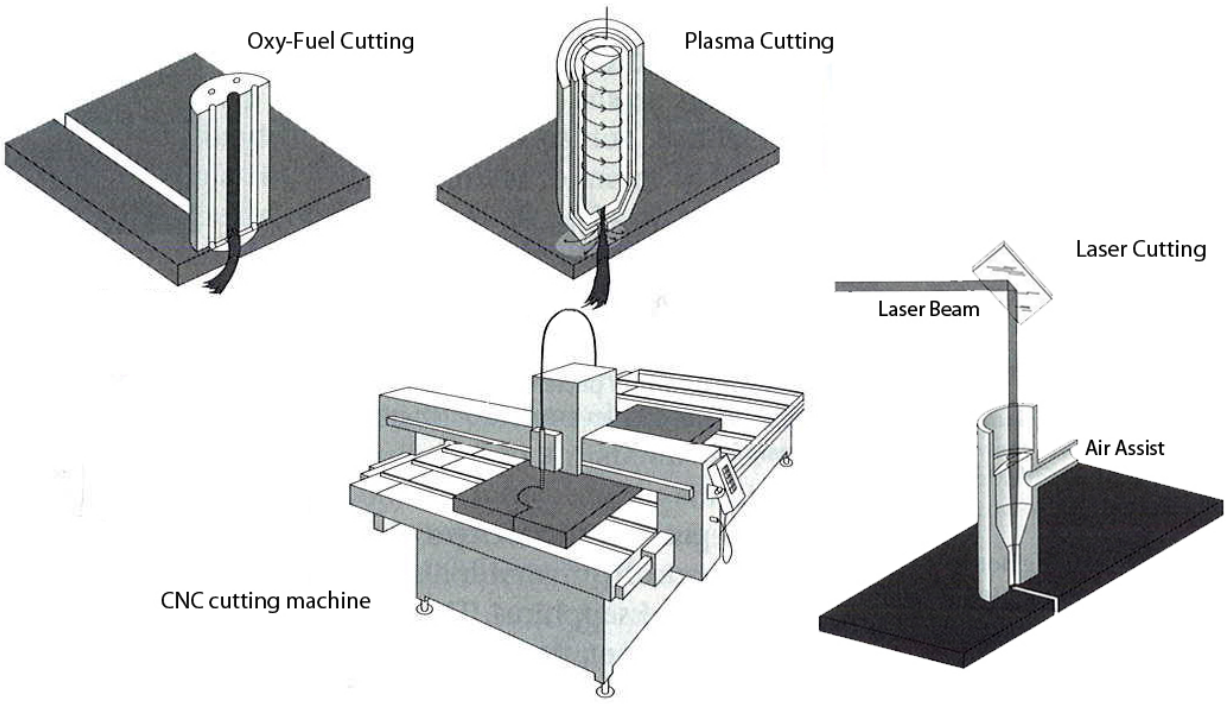


Figure 2-28 CNC cutting machine with three types of cutting techniques
(*Top-Left*) Oxy-Fuel cutting (*Top-Middle*) Plasma cutting (*Bottom-Right*) laser cutting
Image from Schodek et al. (2004: pp. 264-5)

There are examples of using such panels on large-scale projects, for example, the BMW Belt project by Coop Himmelblau in 2007 (Iwamoto, 2009). See Figure 2-29. The BMW Belt project features a double cone surface, formed by placing two cone surfaces apex to apex. The surface geometry is tessellated with triangular faces. Using a digital fabrication process, each triangulated mesh is constructed as a flat glass pane, in this way, realizing the double cone geometry.

Figure 2-30 illustrates utilizing a CNC Laser router to pre-fabricate structure frames by two-dimensional profile cutting.

Figure 2-29 BMW Belt—a double cone surface, Munich, Germany[5]

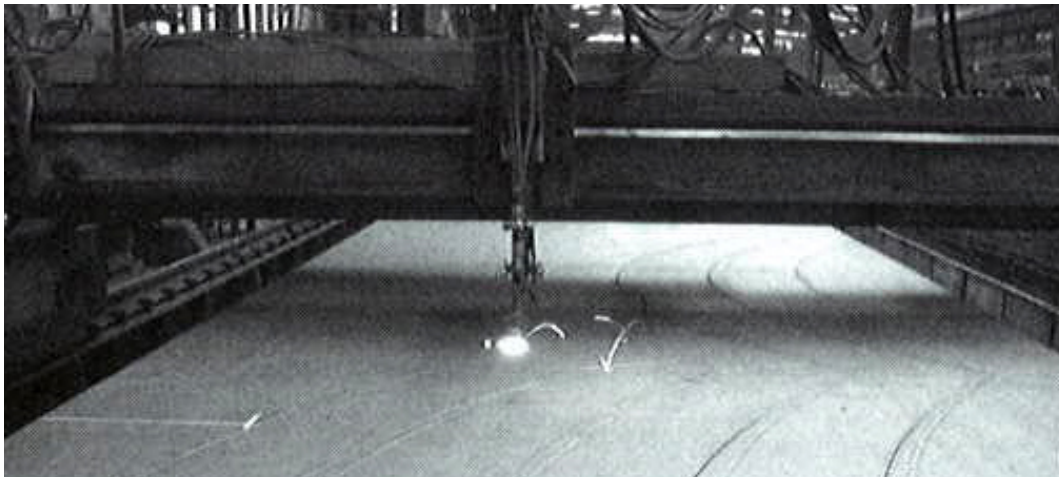


Figure 2-30 Profile cutting on a sheet steel for structure frame construction
Image from Schodek et al. (2004: pp. 75)

2.3.2 Single-curved panels

Panels with plane curves can be fabricated by combining a simple cut with a bend (deformation). Figure 2-31 illustrates a three-roll machine for bending sheet material such as sheet metal or plate. Other sheet material, such as wood, can also be applied heat bending to form a single-curved panel.

[5] Image after BMW Belt from *http://www.bmw-welt.com/*, last accessed on Oct 10th, 2011.

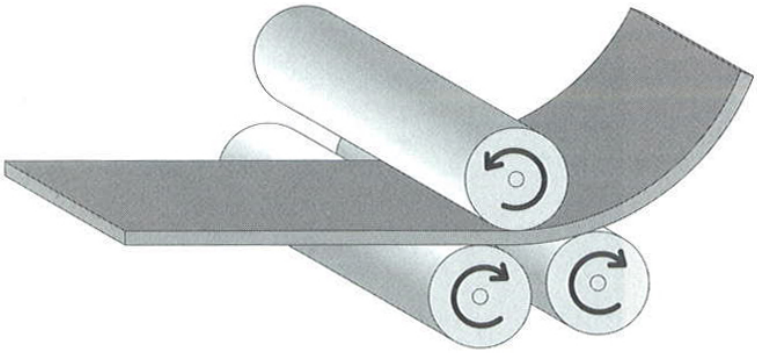


Figure 2-31 Diagram of a three-roll machine bending sheet metal or plate
Image from Schodek et al. (2004: pp. 244)

In fact, a single-curved surface belongs to a family of developable surface. For instance, a cylindrical surface is a single-curved surface (See Figure 2-23a). Many of Frank Gehry's signature freeform projects feature this type of surface (Shelden, 2002; Glympha and Shelden, 2004; Burry and Burry, 2010). At it simplest, any surface that can be formed by bending a flat sheet of material can be called a single curved surface. In Figure 2-32, single curved panels are illustrated with underlying plane curves, which are colored shaded red. In this example, the structure frames are also developed from planar curves, and thus, can be easily fabricated by two-dimensional fabrication techniques.

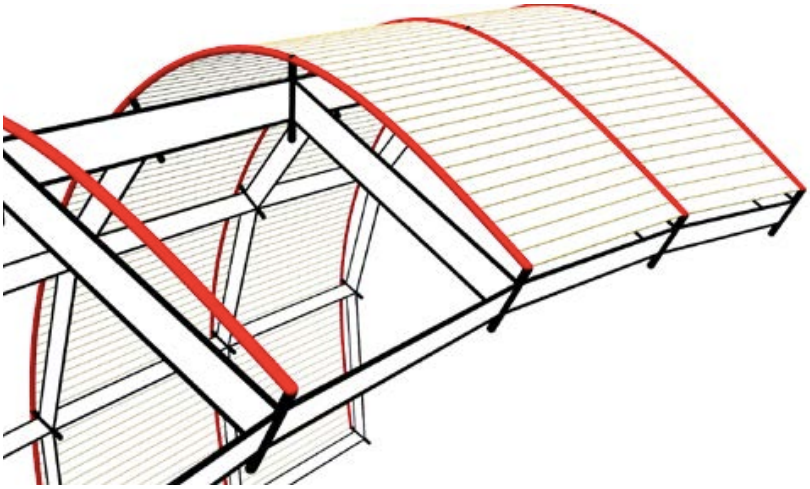


Figure 2-32 Single-curved strips with straight structure frames
Image after Pottmann (2010)

Figure 2-33 shows a single-curved application in a real architectural project—the Disney Concert Hall by Frank Gehry completed in 2003. The freeform facade is finished by sheet steel panels.

Figure 2-33 (*Left*) Disney Concert Hall by Frank Gehry (*Right*) Close view of the steel panels
Image source: http://en.wikipedia.org/wiki/Walt_Disney_Concert_Hall

2.3.3 Double-curved panels

These panels are geometrically more complicated than a flat or single-curved panel and cannot be fabricated by simple cut-and-bend, and require advanced digital fabrication techniques. In general, double-curved panels are usually featured with continuous curvilinear surfaces and can be manufactured by subtraction, deposition, or deformation.

Subtraction removes parts from volumetric material to form the desired shape; this is mainly achieved by CNC machinery, which have many degrees of freedom (DOF). The more degrees of freedom a machine has, potentially, the more complex the form that can be produced. Figure 2-34 illustrates a CNC routing machine with 5 DOF corresponding to five axes of movement. In general, a CNC machine is controlled by coded instructions, which describe the tool-bit paths in sequential order. Using a CNC machine featuring more degrees of freedom, complex forms such as double curved panels can be produced. Figure 2-35 shows a panel directly milled out from natural stone material using CNC milling.

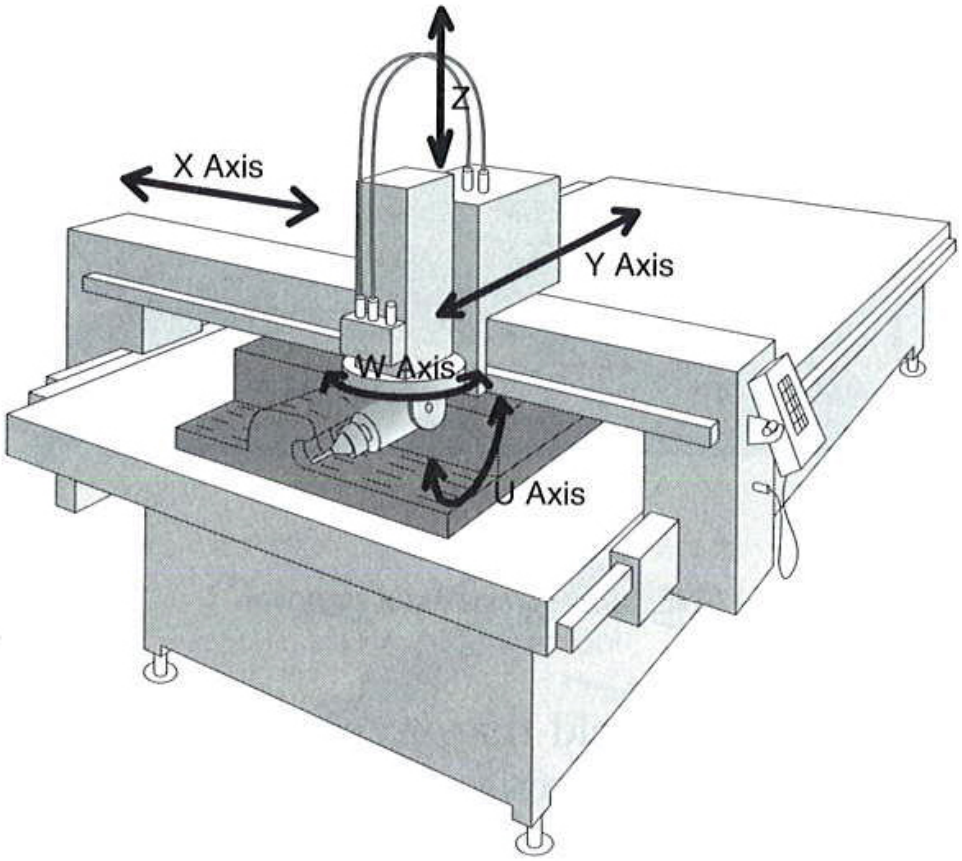


Figure 2-34 A five-axis CNC router illustrated with major axes of movement

Image after Schodek et al. (2004: pp. 242)

Figure 2-35 Complex shaped panel construction by CNC milling

(*Left* & *Middle*) Milled surface panels; and (*Right*) Façade mock-up with supporting steel ribs

Image after Schodek et al. (2004: pp. 62)

Instead of milling a double curved panel directly, a formative process by casting or deformation can be applied. Usually, a mold is created first. In more details, fabricating these panels in a formative process relies highly on the quality of mold making. In contemporary architecture practice, a formative approach is usually taken for complex surface construction. Similarly, a CNC routing machine is employed for mold making for both positive and negative compartments. Molds are milled out from volumetric material, such as foam. Constraints in this approach include the thickness of the volumetric material, the length of the milling bits, and degrees of freedom of the CNC machine in relation to the target surface topology. With milled molds, the fabricating process can proceed with either deformation or deposition.

Deformation is a process of continuously changing the body of a material by force. This process is usually irreversible. Figure 2-36 provides two illustrations showing how a layer of sheet materials can be deformed, on the left, by using a positive CNC cut form, and on the right, by a combination of positive and negative CNC cut forms.

Layer materials can be glass-based and other composites, wood, plastics etc. Figure 2-37 shows a thermoforming process applied to an acrylic sheet by a CNC cut positive mold.

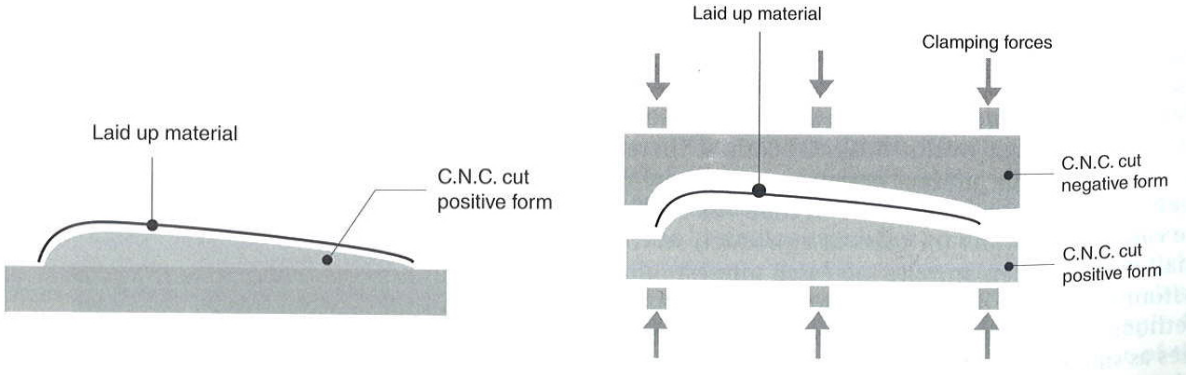


Figure 2-36 Diagram of constructing thin-shell surface by using laid-up materials
(*Left*) Positive CNC-cut form; and (*Right*) Clamped positive and negative forms
Image after Schodek et al. (2004: pp.307)

Figure 2-37 (*Left*) Mold milling (*Right*) Thermoforming acrylic sheets
Image after Schodek et al. (2004: pp. 72)

Casting is another type of the formative process, in which liquefied material, such as concrete, gypsum, metal, etc., is injected into the cavity of the mold for solidification. Figure 2-38 illustrates a foam mold milled out by a CNC machine (on the left) with concrete used for casting the final product (shown on the right). In a sense, this kind of mold making with deformation or casting is a relatively labor-intensive manufacturing technique, but provides a great flexibility to accommodate complex shaped geometry construction.

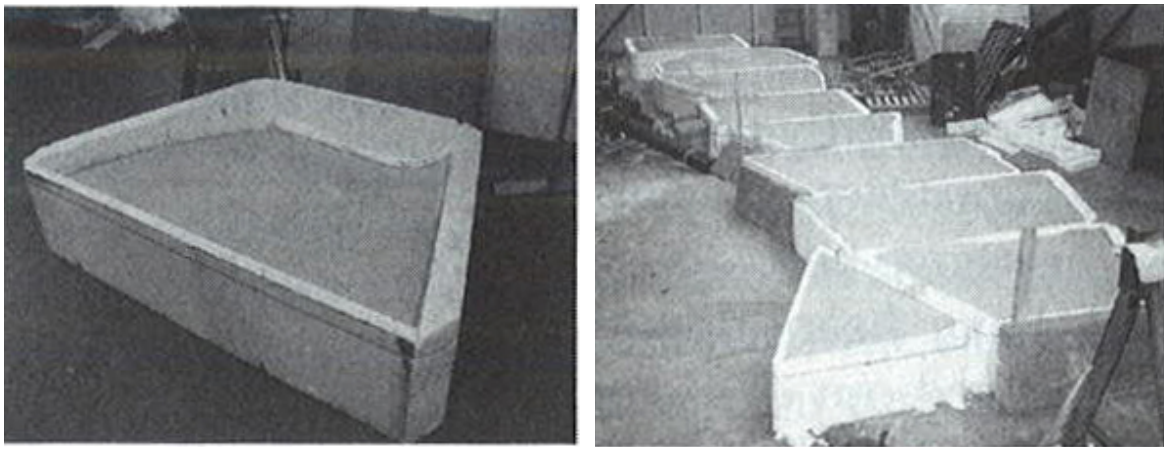


Figure 2-38 (*Left*) CNC-milled foam mold (Right) Concrete casting
Image after Schodek et al. (2004: pp. 334)

For certain intricate freeform designs such as the Innsbruck railway station by Zahad Hadid, and the Kunsthaus Graz by Peter Cook and Colin Fournier, shown in Figure 2-39, using planar-

component based fabrication is infeasible. To realize such complex forms, a formative process, such as casting, is needed and, unsurprisingly, production cost is extremely high. However, in order to fabricate physical construction, which faithfully represents the original complex design, a project-based approach is inevitable. Notwithstanding, a design with well-structured constructive principles can facilitate the digital process, and in turn, optimize the final manifestation. For instance, the cost of using a formative approach for double curved panel construction highly depends on the cost of mold making. To reduce cost, there have been developments toward a more sustainable direction through research in reusable molds, and flexibly adjustable molds, which can be adjusted for various panels with the same surface topology (Pronk et al., 2009; Eigensatz et al., 2010b; Boers[6], 2008). The key to utilizing reusable and flexible molds lie at the capability of optimizing types of surface tessellation elements by distinct surface characteristics such as dimension, curvatures, and so forth, which are derived from the underlying constructive geometry.

Figure 2-39 (*Left*) Innsbruck railway station by Zahad Hadid[7]
(*Right*) Kunsthaus Graz by Peter Cook and Colin Fournier[8]

2.4 Summary

Digital fabrication techniques have radically changed practice in the way we manifest architecture. This change has driven the evolution of building design from a mode of mass production into one of mass customization (Kolevairc, 2005a). Through CNC machinery, fabricating thousands of unique components is almost equivalent to fabricating thousands of identical components (Pine 1993). This

[6] S. Boers, http://optimalforming.com/, last accessed on Oct 10th, 2011.
[7] Iwan Baan, http://www.iwan.com/photo_index.php?category=photography, last accessed on Oct 10th, 2011.
[8] Kunsthaus Graz, http://en.wikipedia.org/wiki/Kunsthaus_Graz, last accessed on Oct 10th, 2011.

is achieved by employing machining instructions from digital fabrication data, which is generated directly from a digital model or by some underlying constructive principles. This does not, however, mean that designers are free to pursue whatever forms they choose without paying heed to requirements of fabrication. Instead, well considered evaluations, e.g., exploring sizes of panels specific to the dimensions of the chosen CNC machine, investigating warping with respect to material constraints, will accelerate production and minimize fabrication costs.

This type of digital-driven process emphasizes a more integrative and generative workflow, reuniting the originally separated professions of design and construction, and is becoming more prominent in architecture in the past few years (Kolarevic, 2009). In this dissertation, a parametric modeling process is adopted to solve surface tessellation with irregular boundary conditions. The contention is that with the well-organized underlying structure, further pattern-based propagation can become more manageable while exploring freeform surface design, which heavily relies on the discrete elements from the surface tessellation. The methodology presented in this dissertation is based on the meshing process and intends to be later extended with considerations for physical construction. To achieve this goal, the relationship between underlying surface tessellation with pattern-based propagation is investigated and serves as an essential handler for supporting future physical construction.

Chapter 3

Parametric Pattern Generation

In this chapter, preliminary studies on parametric pattern generation are described. Two kinds of patterns are considered: Archimedean patterns (Grünbaum and Shephard, 1987), and interwoven patterns that are inspired by the work of Erwin Hauer (2007). Both kinds of patterns are constructed through a sequence of parametric operations on a given base template. The findings from these investigations serve as the foundation for the parametric pattern-based surface tessellation.

To convert a surface into a pattern of polygonal forms it is common to employ approaches that are based on the *U* and *V* domain parameters. To retrieve a range of parametric values in one domain, an parametric interval with notation, $[t_0, t_1)$, is used. t_0 represents the starting parameter and t_1 the ending parameter for the interval. It is straightforward to divide an entire surface by iteratively evaluating parametric intervals, $[u_0\ u_1)$ and $[v_0\ v_1)$, of a target surface for generating subsurface patches. This approach is efficient and straightforward when the target surface remains untrimmed. The images shown in Figure 3-1 exemplify different surface segmentations obtained by varying the *UV* parameter indices for either quadrilateral or triangular configuration. The image on the left is the surface segmentation derived from uniform parameter intervals; the middle and right images are conversions from uniform *UV* segmentation into triangles with respective operations. In this example two types of triangulations are derived from splitting the original quad faces by diagonal lines.

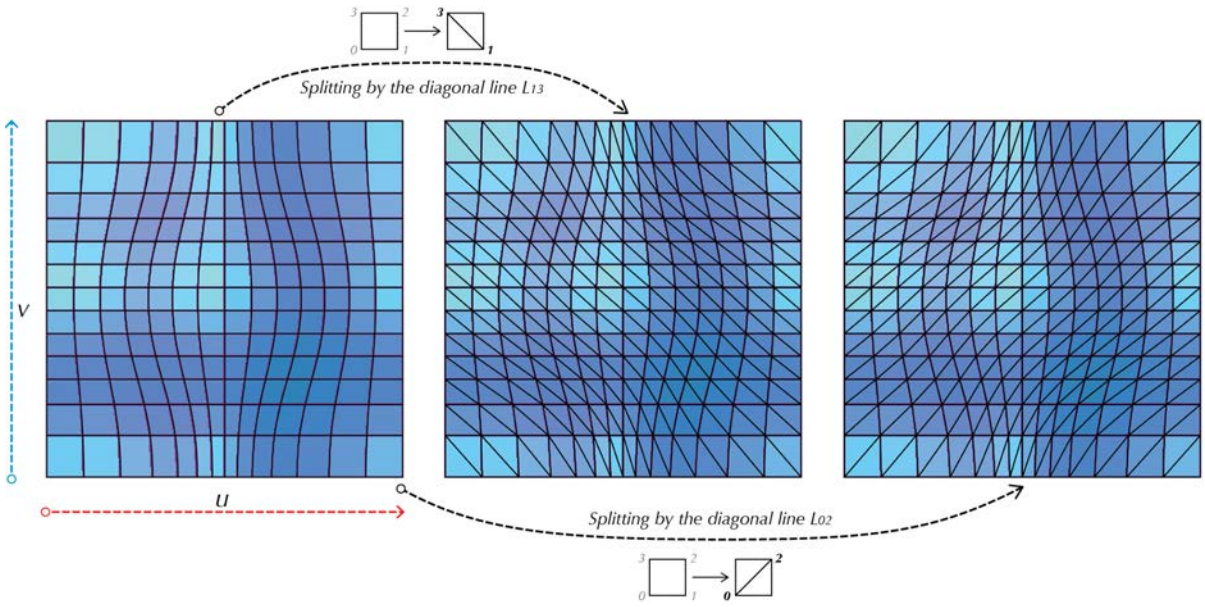


Figure 3-1 UV-based segmentation and quad-triangle conversion

(*Left*) Uniform UV segmentation (*Middle*) Conversion from quads to triangles-type 01

Right) Conversion from quads to triangles-type 02

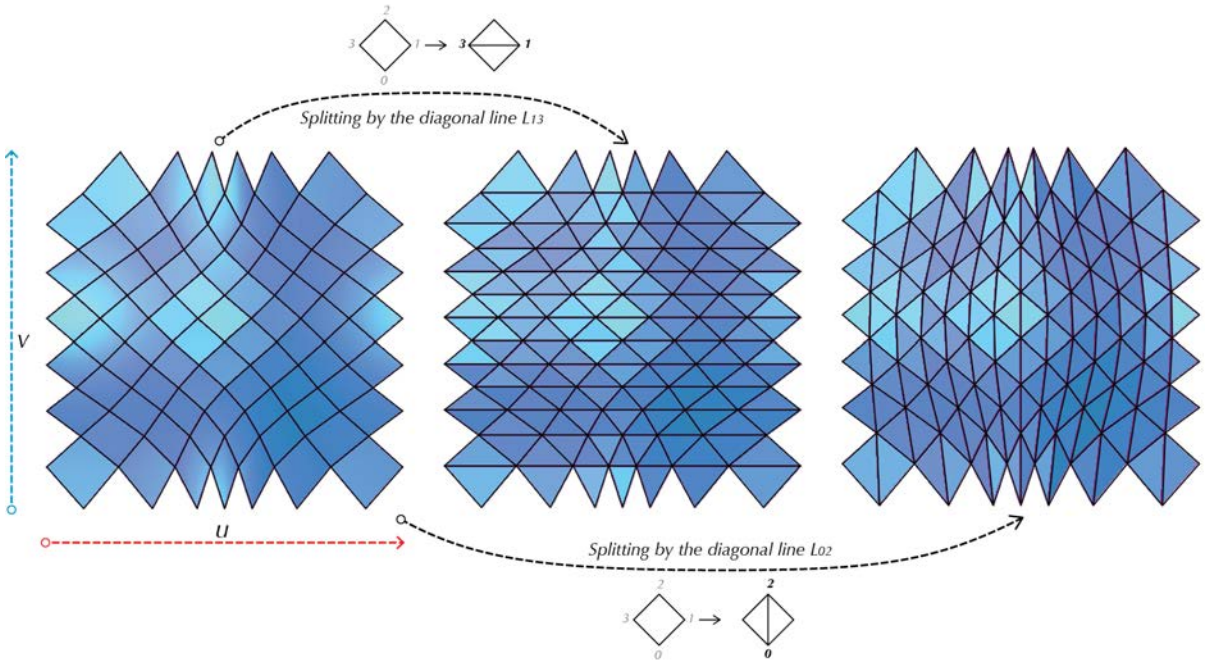


Figure 3-2 Diagrid pattern with quad-triangle conversion

(*Left*) Quadrilateral segmentation of the diagrid pattern (*Middle*) Type_01 by a horizontal split

(*Right*) Type_02 by a vertical split

By filtering the triangulated patterns with a checkerboard pattern, another type of segmentation can be derived; this is called the *diagrid* pattern shown in the left most image of Figure 3-2. A diagrid pattern is made out of diagonal lines from the original quadrilateral faces, thus creating distinct flows along directions other than the original *UV* domains. The middle and right images in Figure 3-2 correspond to diagrid subdivision by splitting the pattern into half; one uses a horizontal split along the *U* direction, and the other uses a vertical split along the *V* direction.

By manipulating the parameter intervals, other segmentation results can be achieved. Figure 3-3 is similar to the example shown in Chapter 1 (Figure 1-6), in which the parametric intervals along the *U* direction are re-parameterized by tracking surface curvature changes; in this example, the higher the curvature change the smaller the interval. The parametric intervals along *V* are gradually increased from the bottom to the top. By iteratively splitting the surface from quads to triangles (see the middle image in Figure 3-3) and triangles to quads (see the right image in Figure 3-3), various patterns emerge by simply reconfiguring the mesh elements.

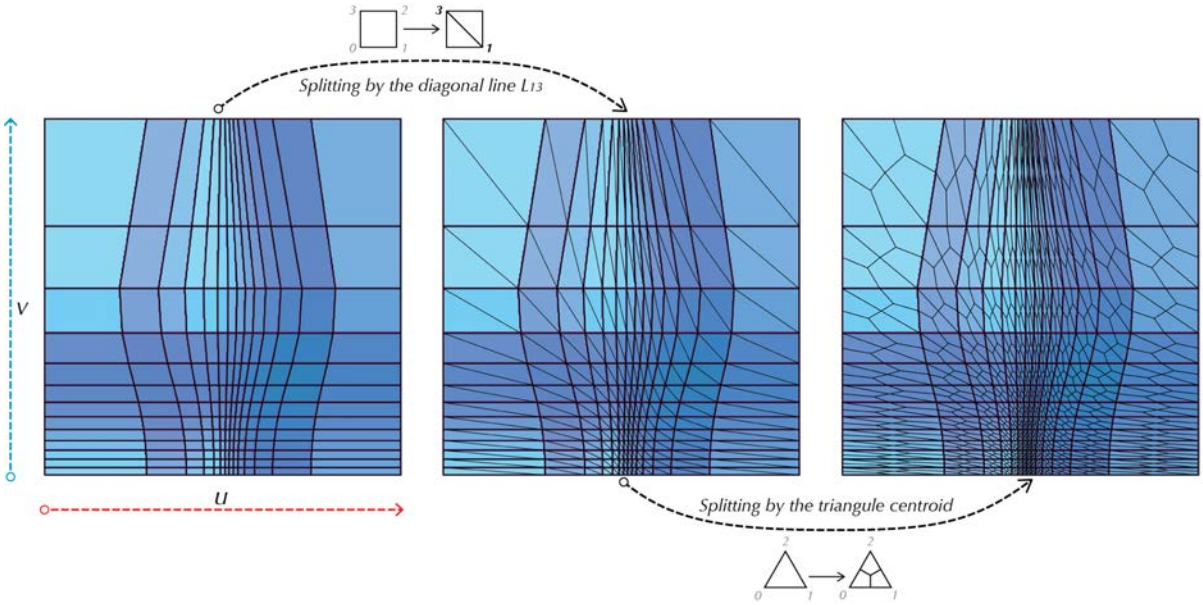


Figure 3-3 Iterative subdivision of a surface
(*Left*) Surface produced from customized intervals
(*Middle*) Conversion from quadrilaterals to triangles
(*Right*) Conversion from triangles to smaller quadrilaterals

In addition to subdivision by triangles and quadrilaterals, Archimedean patterns are also considered as examples of pattern-based tessellations. Two regular patterns, namely, the square and triangular tilings can be parametrically constructed by using the *UV* parameter intervals. From these two fundamental regular polygonal patterns, other regular and semi-regular patterns can be constructed by additional parametric operations.

3.1 Archimedean Patterns

An *Archimedean tiling* (or pattern) is a two-dimensional *vertex-transitive edge-to-edge plane-filling* pattern made up of one or more polygonal tiles called *prototiles*. Vertex-transitive indicates that the same configuration of regular polygons is repeated at each vertex. This includes identical (1) number of faces, (2) number of edges of each face, and (3) order of these faces surrounding the vertex. These patterns fill the plane. There are a total of eleven Archimedean tilings of which three are regular and the remaining eight are semi-regular (Grünbaum and Shephard, 1987).

In the purest sense, a regular tiling pattern comprises just one kind of regular polygon as the prototile; for instance, the triangular tiling has six equilateral triangles at each vertex, the square tiling has four squares and the hexagonal tiling has three regular hexagons. See Figure 3-4.

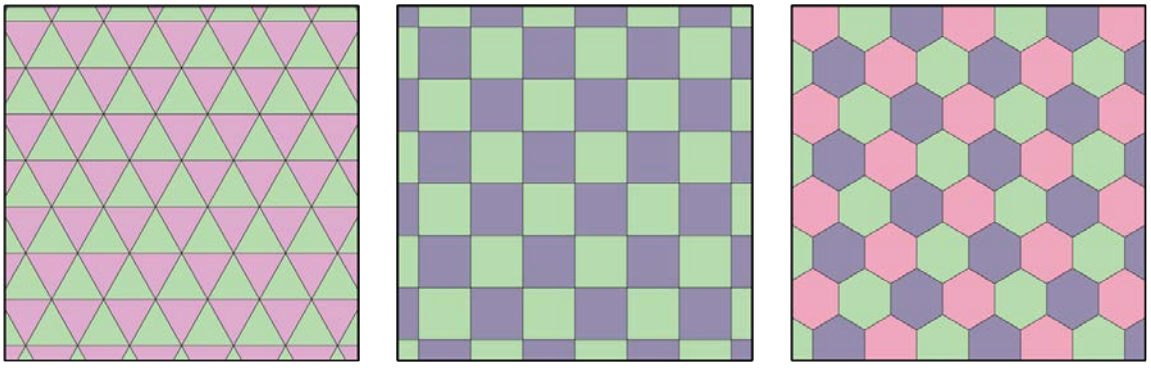


Figure 3-4 The three regular Archimedean tilings in the plane

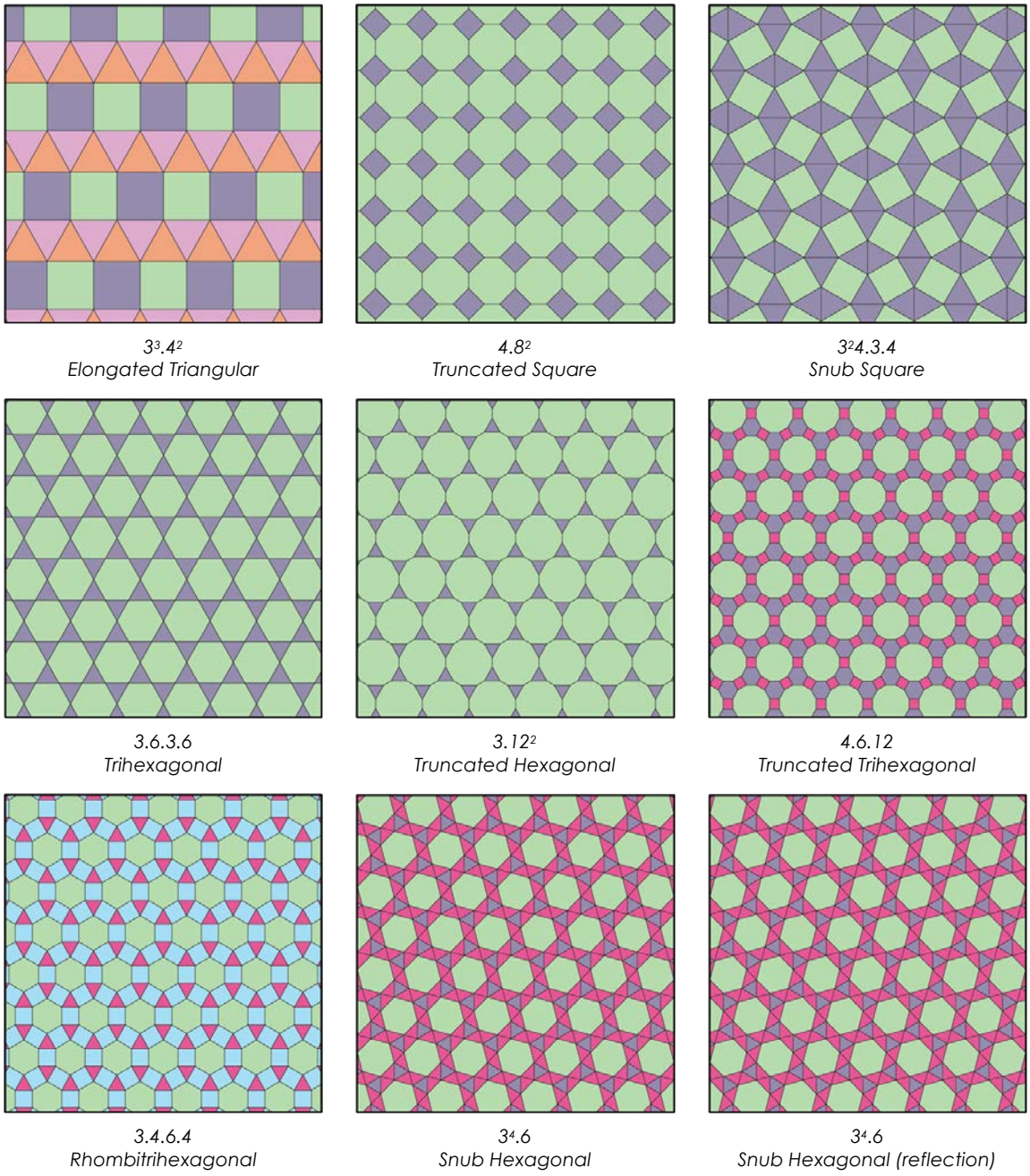


Figure 3-5 The eight semi-regular Archimedean planar tilings in which the snub hexgaonal tiling is shown as a pair of enantiomorphs

For the rest semi-regular tiling patterns, multiple polygons serve as prototiles. To distinguish the types and number of polygons used in a pattern, the following standard notation is employed. For instance, the triangular tiling is represented by 3.3.3.3.3.3 (or 3^6) or six triangles about a vertex, the

square tiling by 4.4.4.4 (or 4^4) or four squares about a vertex, and the hexagonal tiling by 6.6.6 (or 6^3) or three hexagons about a vertex. The semi-regular tilings are likewise notated. These are: 4.8^2, $3^3.4^2$, $3^2.4.3.4$, 3.6.3.6, 3.12^2, 4.6.12, 3.4.6.4, and $3^4.6$ (2). The order of illustrating these prototiles follows the complexity of construction. For example, 4.8^2 and $3^2.4.3.4$ can be both constructible from a square tiling by different operations. The 3.6.3.6 and 3.12^2 are derived from the triangular tiling; the 4.6.12 and 3.4.6.4 tilings from the hexagonal tiling. The $3^4.6$ tiling occurs in distinct right- and left-handed versions, which are also referred to as *enantiomorphs*. See Figure 3-5, which shows the eight semi-regular tilings showing the two forms of the Snub Hexagonal (3^46) tiling.

In the following, firstly, constructive rules embedded in two-dimensional tiling patterns are introduced and exemplified with pattern generation on a three-dimensional surface by segmenting the corresponding two-dimensional parametric space. The objective of this preliminary study is to investigate the underlying topological construction for alternative surface tessellation.

Next, how 'Archimedean' style tilings can be constructed on a three-dimensional surface are illustrated. For simplicity, any such tiling that maintains the vertex condition is referred to as *Archimedean* irrespective of the nature of the surface albeit planar or otherwise, or whether the polygons of the same kind are identical. In this regard, it is important to note that any *UV*-based surface subdivision has a close relationship to an Archimedean tiling, in particular, to the three regular patterns. A quadrilateral (4^4) tiling can be directly derived from a *UV*-based subdivision of any untrimmed surface. A triangular (3^6) tiling can be converted from a quadrilateral tiling, or, be procedurally constructed from the underlying *UV* domain. A hexagonal (6^3) tiling is a dual of a triangular tiling; a *dual* pattern is constructed by replacing a polygonal face by a vertex at its centroid and connecting, by an edge, the vertices corresponding to edge-on-edge faces. That is, every vertex of the polygonal form corresponds to a new polygonal face constructed from the centroid of all neighboring faces and vice versa.

Given a surface such as the one shown in Figure 3-6, parameterized modules can be created to generate a regular tessellation pattern using two parameters, which specify dimensional constraints for the designated subdivision modules along the *U* and *V* directions. The three regular Archimedean pattern shown in Figure 3-7 are constructed this way.

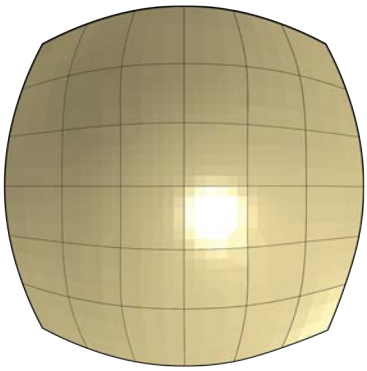


Figure 3-6 Input surface for Archimedean tessellation

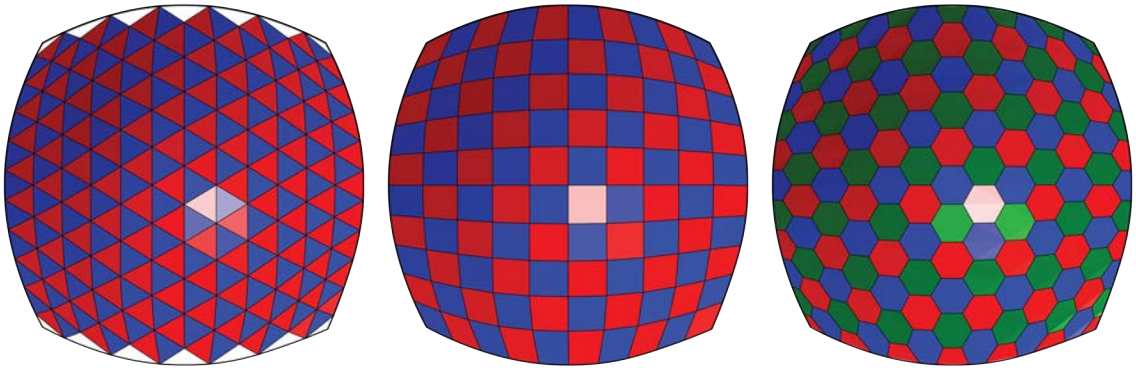
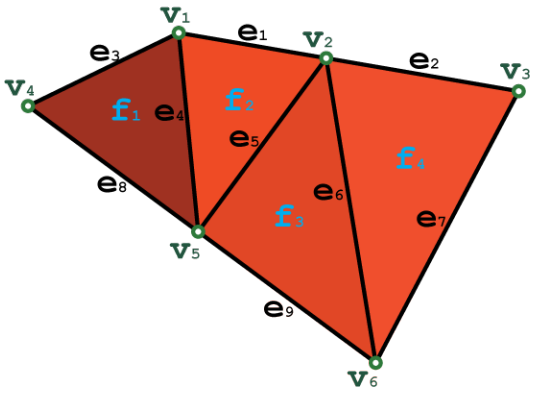


Figure 3-7 The regular Archimedean patterns
(*Left*) Triangular Pattern (*Middle*) Quadrilateral Pattern (*Right*) Hexagonal Pattern

The eight semi-regular Archimedean patterns are each closely related to a regular tiling. In the sequel, how these patterns are constructed parametrically is demonstrated. For this three new operators are introduced: truncation, insertion and alternation. But first, we need a data structure.

3.1.1 Data Structure

At a minimum, any two-dimensional pattern or 2-manifold polyhedral shape can be represented by a collection of vertices, edges, and faces, where the elements are organized by their connectivity; a pair of vertices bound an edge, edges bound faces and a face specified as a list of vertices. See Figure 3-8.



Face	Vertex List
f_1	v_1, v_4, v_5
f_2	v_1, v_5, v_2
f_3	v_2, v_5, v_6
f_4	v_2, v_6, v_3

Edge	Vertex List
e_1	v_1, v_2
e_2	v_2, v_3
e_3	v_1, v_4
e_4	v_1, v_5
…..	
e_9	v_5, v_6

Figure 3-8 Connectivity between vertices, edges, and faces

In order to be able to compute and store information of mesh element connectivity information we consider possible relationships between the mesh elements, namely, between vertices, edges and faces. These are described in the three tables below.

Relations

	Vertex	*Edge*	*Face*
Vertex	*VV*	*VE*	*VF*
Edge	*EV*	*EE*	*EF*
Face	*FV*	*FE*	*FF*

Semantics of relations

	Vertex	*Edge*	*Face*
Vertex	*Vertices of incident edges*	*Incident edges*	*Incident faces*
Edge	*Bounding vertices*	*Adjacent edges (of the same face)*	*Incident faces*
Face	*Bounding vertices*	*Bounding edges*	*Adjacent faces*

Syntax of relations

	Vertex	*Edge*	*Face*
Vertex	*V<V>*	*V<E>*	*V<F>*
Edge	$E\{V\}^2$	$E\{<E>\}^2$	$E\{F\}^2$
Face	*F<V>*	*F<E>*	*F<F>*

For implementation, the relationships *V<E>*, $E\{F\}^2$, and *V<E>* shown highlighted in the tables above are represented in the data structure. By maintaining these three relationships in the data structure, all connectivity information can be retrieved. For instance, the incident faces of a vertex, *V<F>*, can be retrieved by combining the *V<E>* and $E\{F\}^2$ relations. Likewise, bounding edges of a face, *F<E>*, can be retrieved by combining *F<V>* and *V<E>*. Upon completion of the mesh construction, all edges at a vertex are sorted counter-clockwise about the vertex normal. See Figure 3-9.

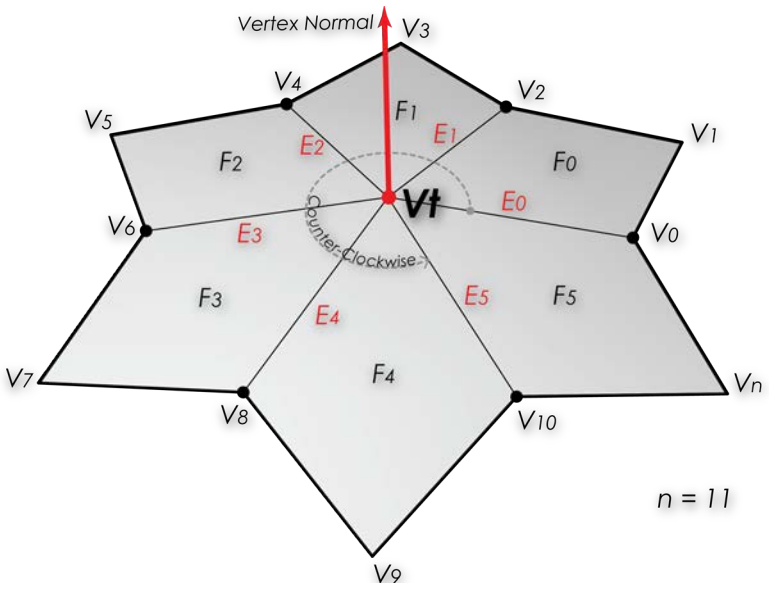


Figure 3-9 Sorted edges and faces about the vertex normal

V_0, V_2, V_4, V_6, V_8, …, V_n are sorted in counter-clockwise order around ***Vt*** about the vertex normal, shown colored shaded red. Likewise, E_0, E_1, E_2, E_3, E_4, E_5 at ***Vt*** are sorted in the same order

We are now ready to illustrate the three tiling operators using this data structure.

3.1.2 Truncation Operator

The truncation operator takes a parameter, t, which specifies a point (aka *truncating vertex*) on the edges around a given vertex to locate new vertices. By iteratively evaluating these truncating vertices, a new polygonal face around the current vertex can be constructed. Each face, incident to the current vertex, is replaced by a new face, which is formed by the truncating vertices on its bounding edges. Figure 3-10 illustrates truncation on the hexagonal (6^3) pattern. The middle image in Figure 3-10 shows vertex truncation and the right image shows face replacement. By applying the truncation operator on the hexagonal pattern (6^3), two semi-regular Archimedean pattern can be constructed, the truncated hexagonal pattern (3.12^2) and trihexagonal pattern (3.6.3.6). See the left and middle images in Figure 3-12. The truncated hexagonal pattern (3.12^2) is constructed with the setting $t = {}^1/_3$ and the trihexagonal pattern (3.6.3.6) with $t = {}^1/_2$.

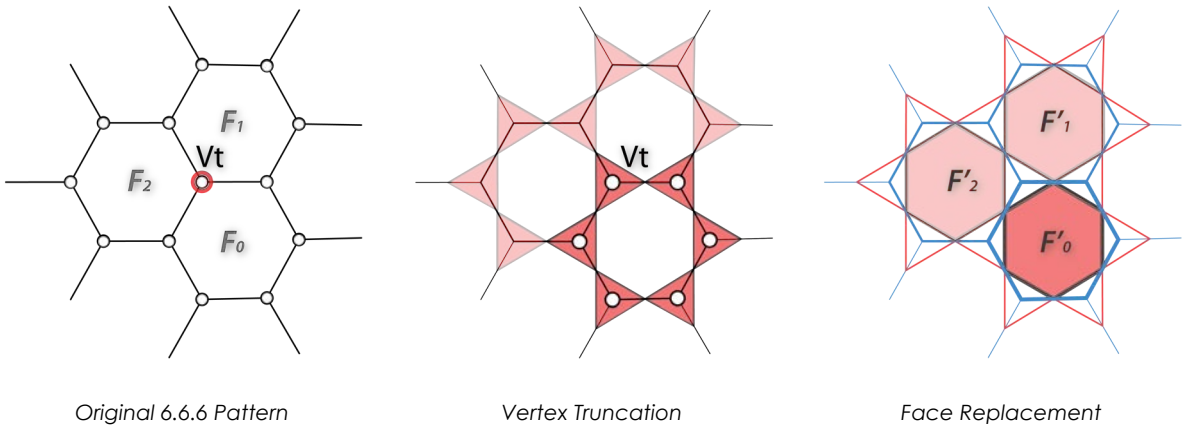
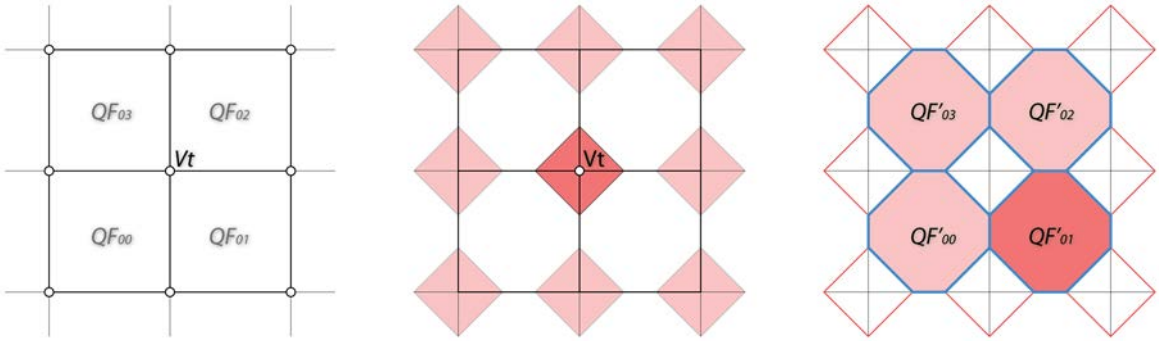


Figure 3-10 Truncation operation on a hexagonal pattern
(*Left*) Original 6^3 pattern (*Middle*) Vertex truncation by inserting truncation points on connected edges (*Right*) Face replacement by connecting truncation points on boundary edges

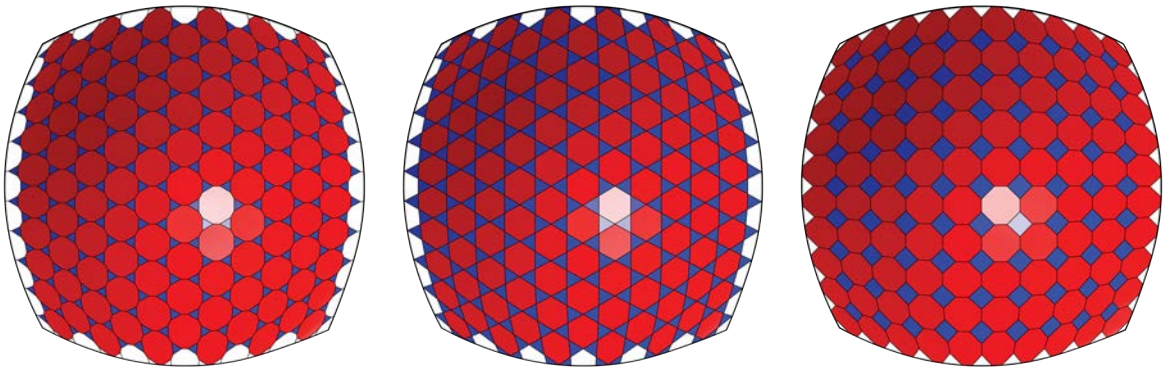
Likewise, the truncated square tiling (4.8^2) is constructed from the square tiling (4^4) by truncation by setting $t = {}^1/_3$. In this example, edges of each quadrilateral face in the square tiling are divided into three segments such that the (1) remaining edge segments and (2) new edges by connecting truncation points from neighboring edges form octagonal elements for the truncated

square tiling (4.8^2). See Figure 3-11. The middle image shows the new quadrilateral elements created by truncating vertices and the right image shows the octagon face replacement.



Original 4.4.4.4 Pattern *Vertex Truncation* *Face Replacement*

Figure 3-11 Truncation operation on a square tiling pattern
(*Left*) Original 4^4 pattern
(*Middle*) Vertex truncation by inserting truncation points on connected edges
(*Right*) Face replacement by connecting truncation points on boundary edges



(3.12²) *(3.6.3.6)* *(4.8²)*

Figure 3-12 Semi-regular patterns derived by the truncation operation from 6^3 and 4^4 patterns
(*Left*) Truncated hexagonal pattern (3.12^2) by setting t = $^1/_3$ from a 6^3 pattern
(*Middle*) Trihexagonal pattern (3.6.3.6) by setting t = $^1/_2$ from a 6^3 pattern
(*Right*) Truncated square pattern (4.8^2) by setting t = $^1/_3$ from a 4^4 pattern

Notice that when t is less than $^1/_2$, two truncation points are constructed; otherwise, only one truncation point is created. The extreme case for the truncation parameter t is when t equals 1. The resulting pattern is the dual of the original pattern, in which each new polygonal face is derived from the barycenter of the connected faces at a vertex. For example, using truncation with $t = 1$ on a triangular tessellation pattern, where each vertex has exactly 6 connected faces, will generate a hexagonal pattern. In other words, a hexagonal pattern is the dual of a triangular pattern and vice versa.

Truncation can be described by the following pseudo-code.

"*For a given initial pattern of prototiles (faces), which are specified, essentially, by lists of vertices, edges per vertex, and faces, new faces, edges and vertices are created as follows. Each vertex is replaced by a new face formed from truncating vertices on edges incident to the original vertex. Edges of every existing prototile (face) are replaced by new edges created from the inserted truncating vertices on the original edges. Original faces are updated to include the truncating vertices on the boundary edges of the face. The truncation process is based on examining the connectivity at existing vertices and around each face.*"

[Pseudo-code for Truncation]

```
TRUNCATION (M, M', T)
 1  for each Vertex, V, in the mesh vertices, M<V>: // 1. Vertex Replacement //
 2       new VtList <- TRUNCATEPTSATVT (V, T)
 5       PF_new <- new PolyFace(VtList)
 6       update PF_new → M'
 7
 8  for each Face, F, in the mesh faces, M<F>:   // 2. Face Replacement //
10       new VtList <- TRUNCATEPTSONFACE (F, T)
11       PF_new <- new PolyFace(VtList)
12       update PF_new → M'
```

[Macro for Truncation at Vertex Edges and Faces]

```
TRUNCATEPTSATVT (V, T)
 1  new VtList    // new Vertex List by Truncating Vertex Edge//
 2  for each Edge, E, in the Edge list, V<E>:
 3      add interpolated Point(s) on Edge, E, by T → VtList
 4  return VtList


TRUNCATEPTSONFACE (F, T)
 1  new VtList    // new Vertex List by Truncating Face//
 2  for each Vertex pair, [Vi, Vi+1], in, F<V>:
 3      add interpolated Point(s) by Verteices, [Vi, Vi+1] with T BI → VtList
 4  return VtList
```

3.1.3 Insertion operator

The insertion operator begins with the insertion of a new polygonal element at an edge. The next step is to construct new faces at bounding vertices and incident faces of the current edge. For instance, in order to create a truncated trihexagonal (4.6.12) tiling pattern we apply an insertion operator on a triangular tiling (3^6). The first step starts with new quadrilateral face generation at each edge, from which new vertices are constructed by offsetting the edge midpoint toward the centroids of incident faces. A parameter, *t*, in this operation is utilized to control the offsetting distance, as well as the dimension of the quadrilateral face element, as shown in Figure 3-13. The second step constructs a face replacement for each incident face (of current edge) by connecting offset vertices from the bounding edges (the left image of Figure 3-14). Lastly, each vertex is replaced by new polygonal face, which is formed by offset vertices from connecting edges at the current vertex, as show in the right image in Figure 3-14.

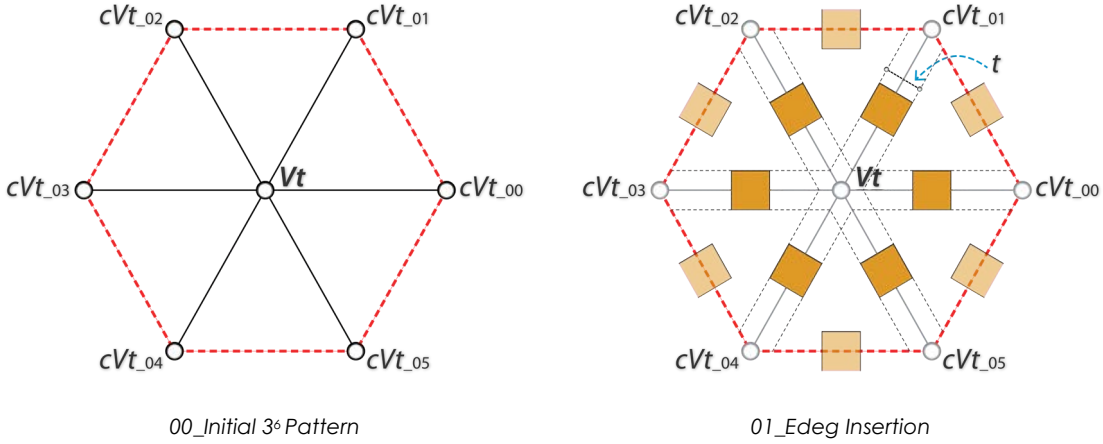


Figure 3-13 Insertion Step 1: Edge insertion

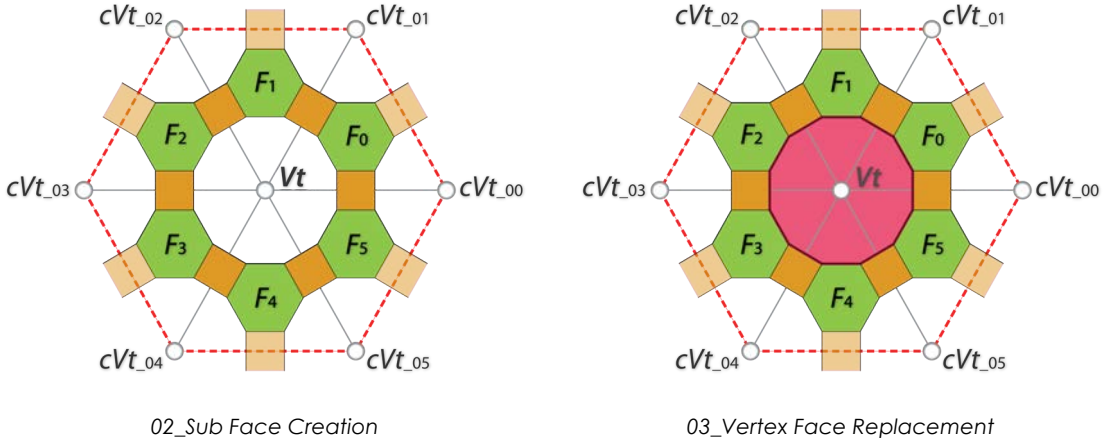


Figure 3-14 Face and Vertex Replacement

(*Left*) Step 2: Sub-face creation (*Right*) Step 3: Vertex Face replacement

Likewise, the rhombi-trihexagonal (3.4.6.4) tiling pattern can be constructed by insertion of a regular 6^3 pattern. In this case, the control parameter, t, is set equal to $^1/_2$. A quad face is constructed for each edge. For each connected face, a triangular face is constructed. Each bounding vertex is replaced by a hexagonal face. Figure 3-15 illustrates the two semi-regular patterns, which are created by insertion. Notice that the (4.6.12) tiling pattern (shown on the left in Figure 3-15) could also be derived from a trihexagonal (3.6.3.6) pattern solely by truncation. In a sense, insertion offers an alternate approach to the pattern generation by rendering different parametric controls over the procedural construction of corresponding polygon elements.

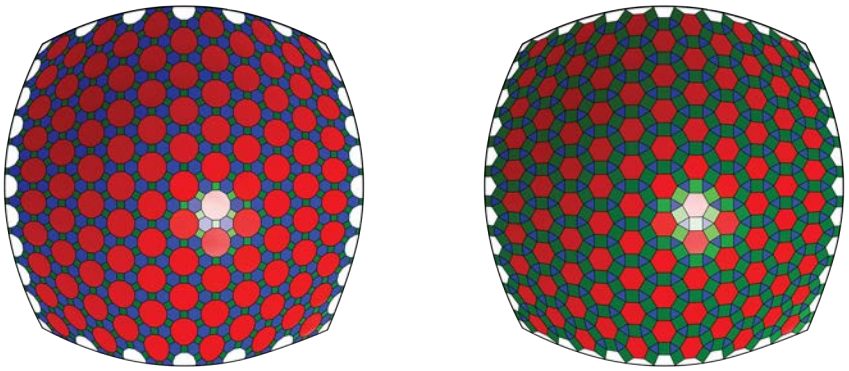


Figure 3-15 Semi-regular pattern by the scaled-insertion operator
(*Left*) Truncated trihexagonal pattern (4.6.12) (*Right*) Rhombi-trihexagonal pattern (3.4.6.4)

This process can be described by the following pseudo-code:

"*For a given initial pattern of prototiles (faces), which are specified, essentially, by lists of vertices, edges per vertex, and faces, new prototiles are created by edge insertion as follows. First, quadrilateral faces are inserted on each edge. The parametric relation of the insertion face to the edge is determined by a parameter, T, which specifies the offset distance from the edge and the dimension of the insertion face is the double of the offset distance. Next, vertex and face replacement is, respectively, performed by examining the edges incident at the vertex, or the edges bounding the face.*"

[Pseudo-code for Insertion]

INSERTION (***M, M', T***)

```
1  for each Edge, E, in the mesh edges, M<E>: // 1. New edge face insertion //
2      VtList <- EDGEFACEINSERTION(E, T)
3      PF_new <- add new PolyFace(VtList)
4      update PF_new → M'
5
6  for each Vertex, V, in the mesh vertices, M<V>: // 2. Vertex Replacement //
7      VtList <- EDGEOFFSETPTSATVERTEX(V, T)
8      PF_new <- add new PolyFace(VtList)
9      update PF_new → M'
10
```

```
12  for each Face, F, in the mesh faces, M<F>: // 3. Face Replacement //
13      VtList <- EdgeOffsetPtsOnFace(F, T)
14      PF_new <- add new PolyFace(VtList)
15      update PF_new → M'
```

[Macro for EdgeFaceInsertion and EdgeOffsetPts at Vertex and on Face]

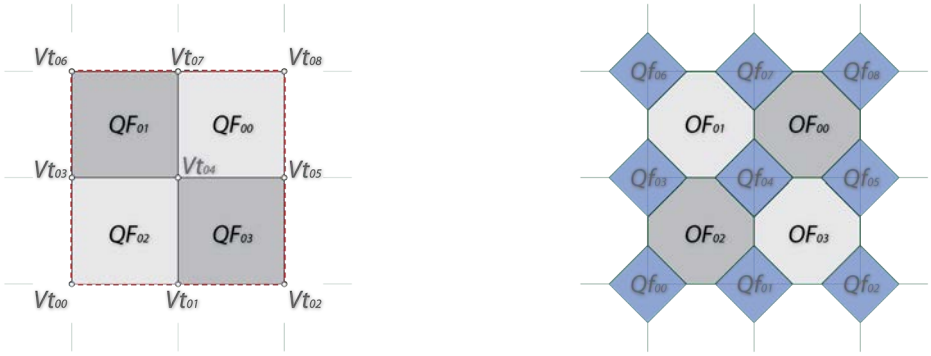
```
EdgeFaceInsertion (E, T)
 1  new VtList    // new Vertex List for Edge Face Insertion //
 2  for each Face, F, in the Edge face list, E<F>:
 3      add offset edge points toward connected face, F, by T → VtList
 4  return VtList

EdgeOffsetPtsAtVertex (V, T)
 1  new VtList    // 1. new Vertex List by EdgeOffset points at vertex//
 2  for each Edge, E, in, V<E>:
 3      add offset Point(s) from E by T → VtList
 4  return VtList

EdgeOffsetPtsOnFace (F, T)
 1  new VtList    // 1. new Vertex List by EdgeOffset points on face //
 2  for each Vertex pair, [V_i, V_i+1], in, F<V>:
 3      add offset Point(s) by Verteices, [V_i, V_i+1] with T → VtList
 4  return VtList
```

3.1.4 Alternation Operator

The alternation (or snub) operator creates a snub pattern from a truncated pattern, for example, a snub square pattern ($3^2$4.3.4) from a truncated square pattern (4.8^2). First, the alternation operation removes the alternative vertices for a prototile in the pattern, and in the process tags the other alternative sequence of vertices for removal. Next, a triangular face is introduced at each vertex

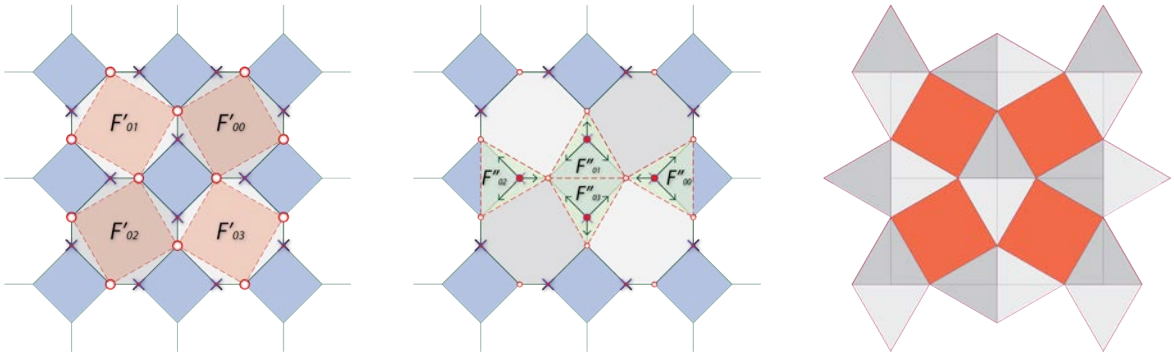
tagged for removal using the alternately selected vertices, in this way, generating a new snub configuration.

Alternation applies only to polygonal faces with an even number of vertices and the minimum number of vertices is six. Figure 3-16 and Figure 3-17 illustrate the construction. This includes (1) truncating the square tiling pattern, as shown in Figure 3-16; (2) removing vertices from the truncated polygons to form new polygonal faces, as shown in the left image in Figure 3-17; (3) inserting new triangular faces at vertices tagged for removal, as shown in the middle image in Figure 3-17. The resulting snub square pattern is shown on the left in Figure 3-18.



00_Inital Square Tiling *01_Truncated Square Tiling (4.8^2)*

Figure 3-16 Alternation operation: Truncation



02_Alternating Vertices for new Quad Faces *03_ Gap Triangular Faces at Vertices for Removal* *04_Final Result, Snub Square Tiling ($3^2$4.3.4)*

Figure 3-17 The constructive process of the snub operation from a truncated square pattern (4.8^2)

Similarly, the snub hexagonal ($3^4.6$) pattern can be derived from a truncated trihexagonal (4.6.12) pattern with an alternation operator. In this case, each dodecagonal face is reduced to a hexagon and there are a total of four triangles and one hexagon at each new vertex, as shown in the right image in Figure 3-18.

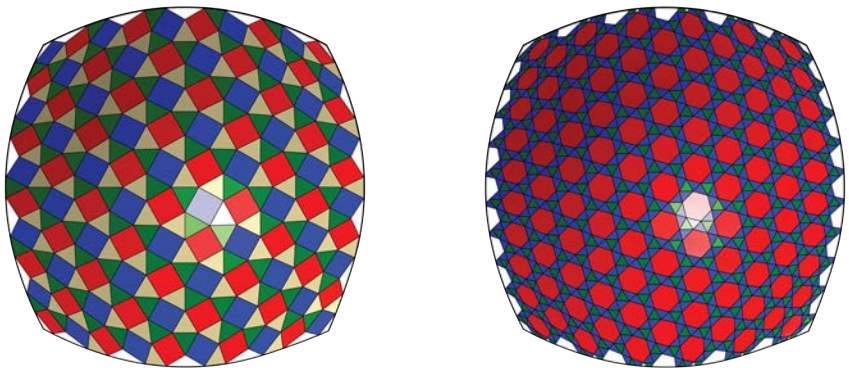


Figure 3-18 (*Left*) Snub square tiling (*Right*) Snub hexagonal tiling

The pseudo-code below describes the alternation operation once the truncated pattern has been created.

> " *Given an initial pattern of prototiles (faces) essentially specified by a list of vertices, a list of edges per vertex, and a list of faces, new prototiles are created by vertex alternation as follows. For each target face (which contains even number of vertices), a sub-face replacement is created by alternatively reducing the vertices into half. The half of tagged vertices for removal is then harvested for the vertex replacement. The minimum number of vertices of a target face is six, which in turn induces a triangular face. In this operation, a Boolean parameter, Alt, is utilized to identify the pattern of the alternation and thus two mirrored versions are created.*"

[Pseudo-code for Alternation]

ALTERNATION (***M, M'***, ***Alt***)

```
1  for each Face, F, in the mesh faces, M<F>: // 1. Sub Face Creation //
2      VtList <- SUBFACEBYVERTEXALTERNATION(F, Alt)
3      PF_new <- add new PolyFace(VtList)
4      update PF_new → M'
5
```

```
for each Vertex, V, in M<V>:   // 2. Tagged Vertex Replacement //
    if (V.IsTagged2Remove()) then:
        VtList <- VERTEXFACEREPLACEMENT(V)
        PF_new <- add new PolyFace(VtList)
        update PF_new → M'
```

[Macro for SubFace by Vertex Alternation and Vertex-face Replacement]

```
SUBFACEBYVERTEXALTERNATION (F, Alt)
new VtList          // new Vertex List for Face-Vertex Alternation //
for each Vertex, V, in the Edge face list, F<V>:
    if (Alt) then : add V → VtList
    else : V <- Tagged for removal     // tagged vertex for removal //
    Alt = !Alt; // change the sign of the alternating pattern
return VtList

VERTEXFACEREPLACEMENT (V)
new VtList    // 1. new Vertex List by Truncating Face//
for each Vertex, V', in, V<V>:
    if (!V.IsTagged2Remove()) then: VtList <- V'
return VtList
```

In summary, inspired by the two-dimensional Archimedean tiling patterns, the three parametric operators presented above can be promoted to create a corresponding tessellation pattern on a given three-dimensional freeform surface. A surface tessellation problem is essentially a pattern-based subdivision problem. The constructive procedures presented for topological manipulations simplify the process of three-dimensional tessellating operations. This preliminary studies serves as the basis to customized patterns that can be constructed by examining the local topological relationships for later intricate pattern generation.

3.2 Interwoven Pattern

Interwoven patterns exhibit weave behavior that create a sense of continuous visual appearance. To construct an interwoven pattern, for any sequence of faces, the ending geometric component in one face has to be smoothly transformed into the connecting geometric component of the next face in the sequence. By following this constructive principle, we can create a continuous intricate pattern based on a simple underlying layout. A few examples of interwoven patterns are considered, which are derived from basic geometric patterns such as quadrilaterals and hexagons. The patterns shown below are inspired by the designs of architectural screen walls by Erwin Hauer (2007). The first three patterns are based on circular trims in quadrilateral faces, followed by a self-interlocking pattern, which can be derived from either the quadrilateral or hexagonal boundaries.

3.2.1 Trimming-Based Patterns

The three patterns, named *ED_03*, *ED_04*, and *ED_05*, in Figure 3-21, Figure 3-22, and Figure 3-23 are constructed by a sequence of trimming operations and spatial transformations.

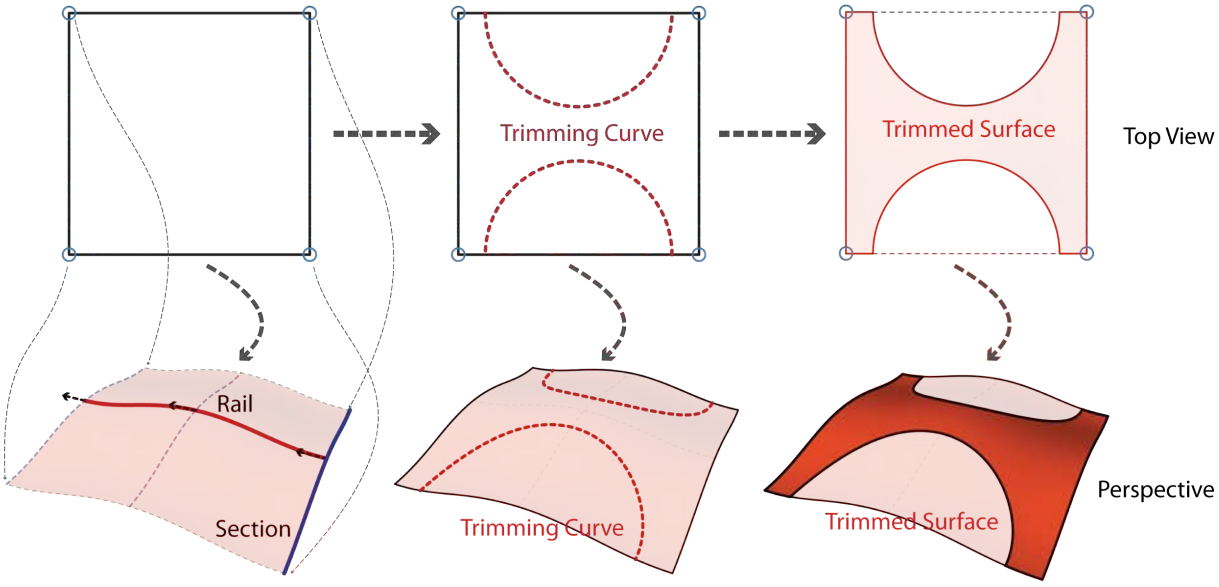


Figure 3-19 Construction of the interwoven pattern ED_03 by trimming a quadrilateral boundary

The top row in Figure 3-19, illustrates in top view the steps for creating a trimmed pattern. The bottom row shows how the corresponding surface manipulations in three-dimensional space. For the

base trimmed-surface module, Figure 3-20 illustrates additional transformation rules—such as, in this case, rotation along the central axis and mirror operation via the planar quad boundary —being applied to generate the second half of this module.

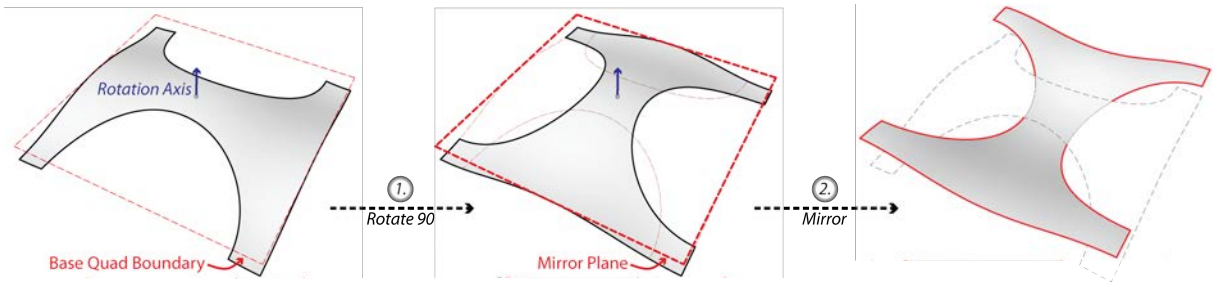


Figure 3-20 Construction of the interwoven pattern *ED_03* by transformation, rotation and mirror

Pattern ED_03

The resulting interwoven pattern *ED_03* is shown in Figure 3-21. On the left side of the figure is the interwoven module, which consists of two parts—upper and lower module components colored in shades of green and blue respectively. The thickness of the component is derived from the offset operation along the normal direction to the base quadrilateral face.

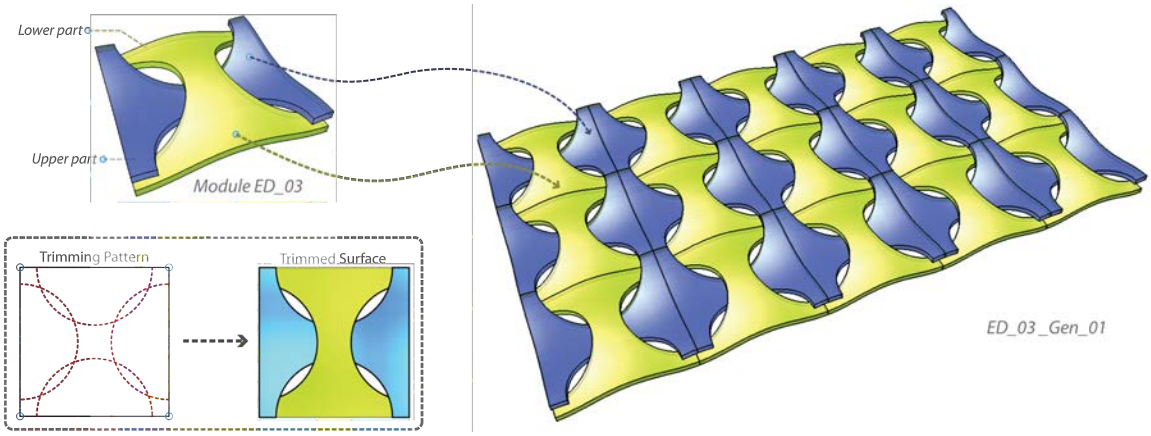


Figure 3-21 Interwoven pattern *ED_03* (Quadrilateral-based pattern)
Inspired by Erwin Hauer (1952)'s continuous screen, *Design 03*, Church at Leising, Vienna, Austria

Pattern ED_04

This particular interwoven pattern, which is shown in Figure 3-22, is inspired by Erwin Hauer's *Design 1* and by Rinus Roelofs' sculptural example, *Connecting Holes* (Roeflofs, 2010). Similar to *ED_03*, *ED_04* takes the quadrilateral face as a primary boundary, but applies a different elliptical trim pattern to the four corner vertices, the midpoints of the four edges, and the centroid of the face.

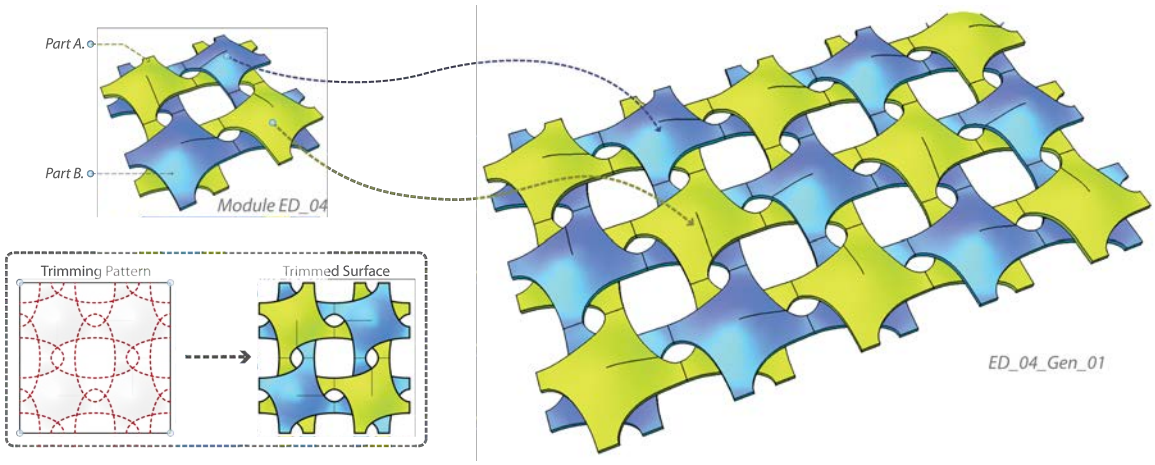


Figure 3-22 Interwoven pattern *ED_04* (Quadrilateral-based pattern)
Inspired by Erwin Hauer (1950)'s continuous surface, *Design 1*

Pattern ED_05

ED_05 is an interwoven or chain pattern based on the quadrilateral grid. This pattern exploits the same elliptical trim pattern as *ED_04*, although employing it at different locations of the quad boundary. In this case, the elliptical trim pattern is used on the vertical boundary edges and the vertical centerline of the quad boundary. Pattern *ED_05*, in Figure 3-23, demonstrates how two identical chain components interlock with each other. Although the two patterns, *ED_04* and *ED_05*, employ the same trim patterns—two eclipses intersecting orthogonally, they generate very distinct results by the variation of the corresponding location to the boundary.

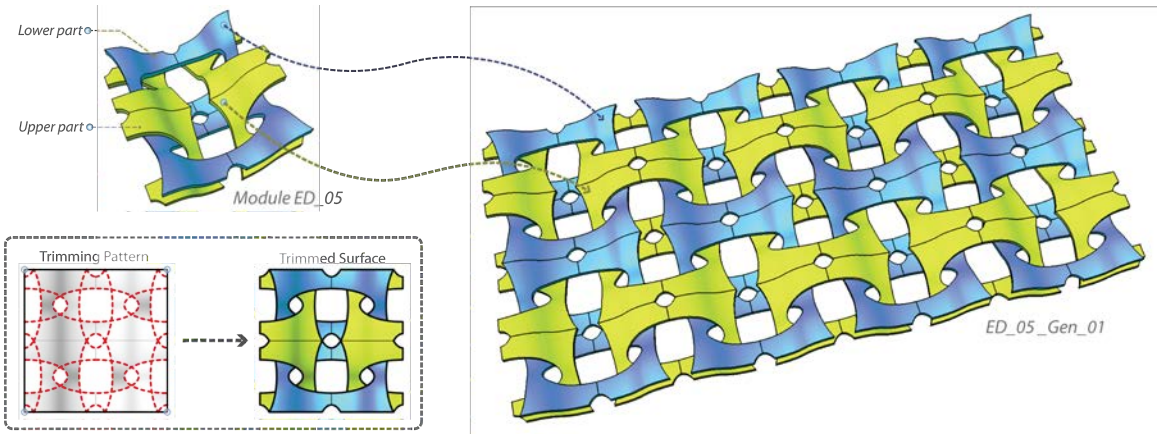


Figure 3-23 Interwoven pattern *ED_05* (Quadrilateral-based pattern)

Two two-dimensional patterns are illustrated in Figure 3-24. On the left is shown a pattern based on the hexagonal grid using a process similar to that described above. The difference in the pattern generation here is that instead of selecting partial subdivision edges from the base quad faces, two groups of sub-polylines are produced using the checkerboard pattern controlled by the surface isoparameters. The other interwoven pattern shown at the right of Figure 3-24 is derived from the regular honeycomb tiling, where each hexagonal cell generates a set of curves. The aggregations of these curves create the dynamic interwoven visualization, even though these are simply planar two-dimensional curves. For this type of interwoven pattern, the self-interlocking feature makes it different from the rest of examples, as shown above. Self-interlocking is used here to identify the interwoven pattern, which has a continuous flow from part to part without interruption.

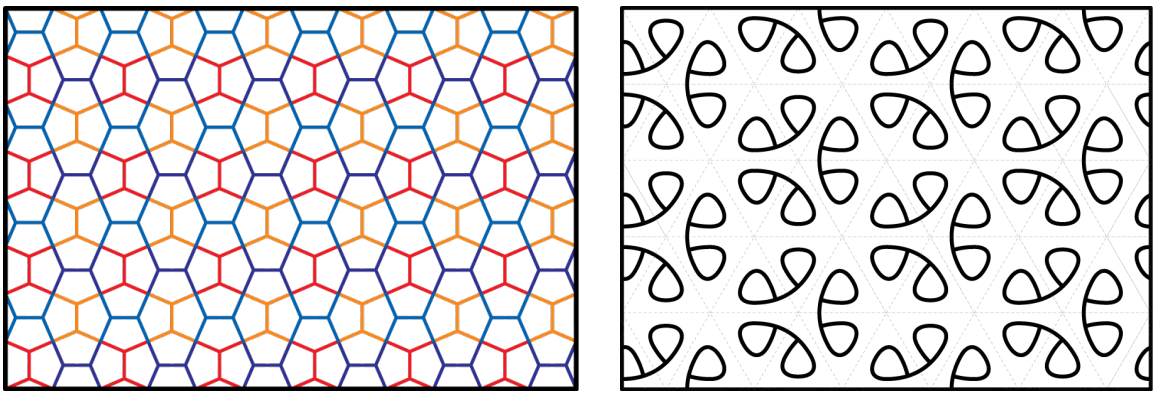


Figure 3-24 (*Left*) Hexagonal tiling by interweaving two perpendicular hexagonal grids; (*Right*) Curvilinear weave based on a hexagonal grid

3.2.2 Self-Interlocking Patterns

For self-interlocking patterns, the major difference from the trimming-based pattern is that each module is self-continuous. Instead of treating a module of two separate parts, which joining only at the external connecting edges with adjacent modules, this type of pattern joins parts internally. This characteristic creates a more intricate continuous movement from local module to entire modular propagation. Figure 3-25 illustrates the trimming operation applied on a target surface with customized trim curves at four corners.

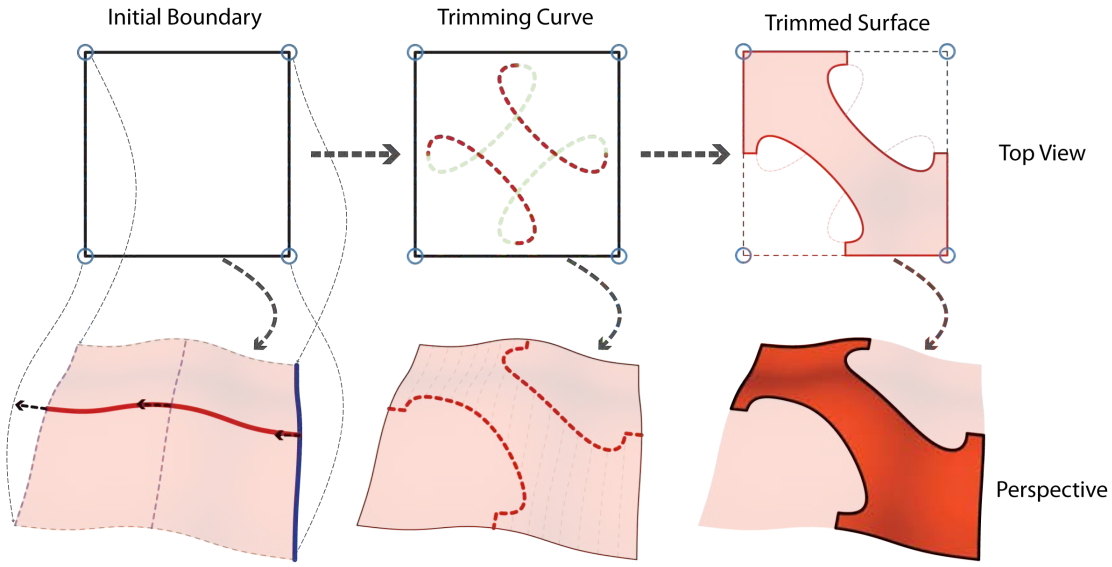


Figure 3-25 Constructive process of a self-interlocking pattern

Figure 3-26 demonstrates an example of a self-interlocking pattern. In this example, since each module is a self-continuous pattern, only one shade is utilized.

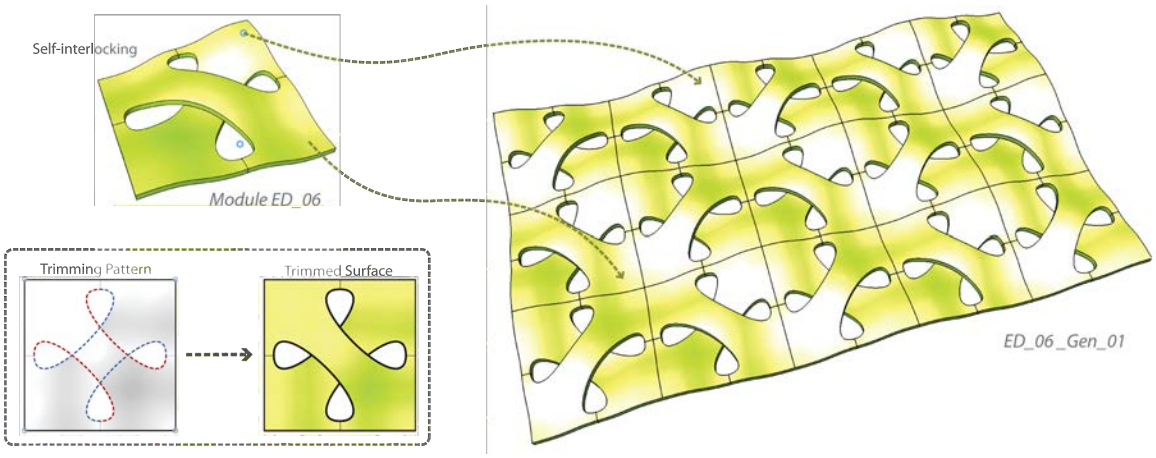


Figure 3-26 Interwoven Pattern *ED_06* (Self-Interlocking)

To go step further with these two operations, trimming-based interweaving and self-interlocking interweaving, Figure 3-27 and Figure 3-28 are two interwoven examples derived from an underlying hexagonal pattern. In light of the underlying topological connectivity, the hexagonal interwoven pattern can be spited into three groups and thus three different colors are used in Figure 3-27.

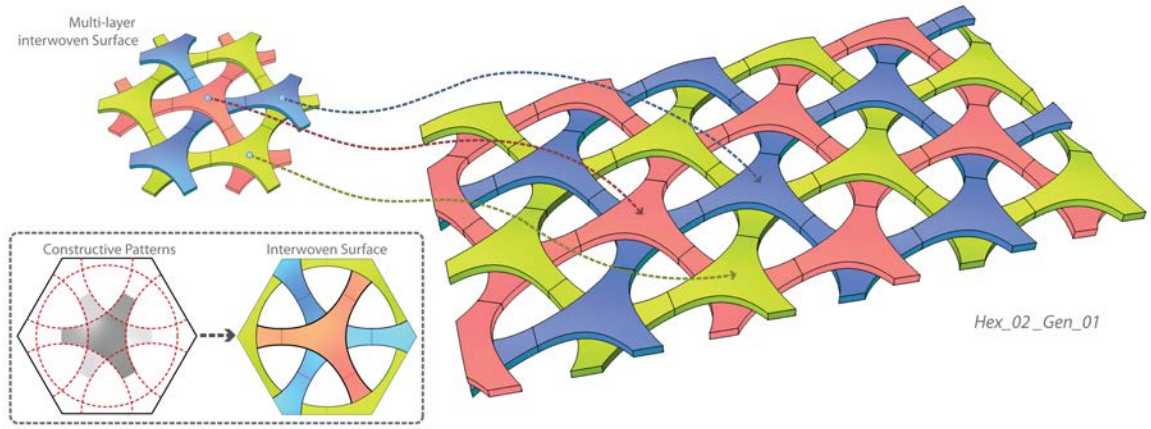


Figure 3-27 Interwoven Pattern Hex_01

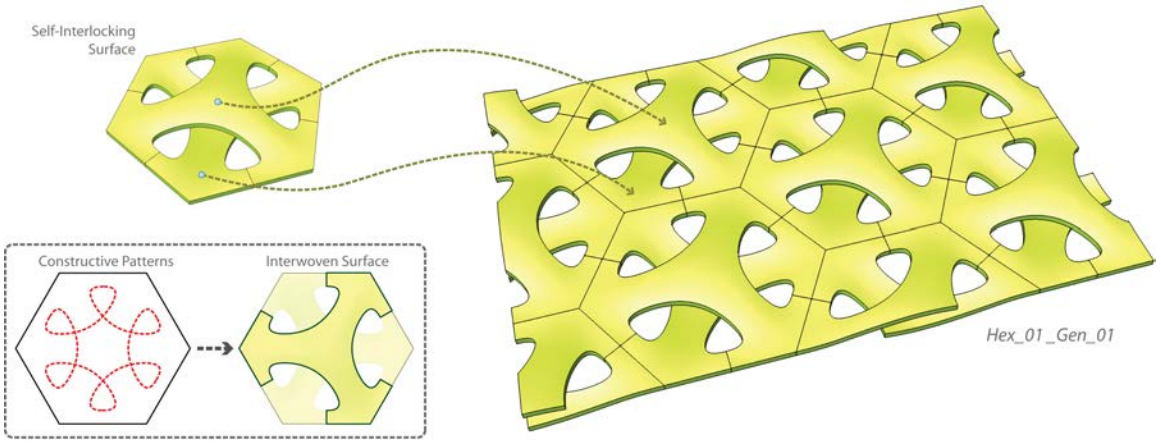


Figure 3-28 Interwoven Pattern Hex_02 (Self-Interlocking)

To summarize, both interwoven and self-interlocking patterns are constructed based a polygonal boundary, which corresponds to essential face elements in a surface tessellation. The order of the face elements serve as the important index for creating continuous weaving visual appearance.

The pseudo-code below describes the interweaving operation.

" Given a set of polygonal face elements, interwoven modules are created by subtracting parts from the original face element with designated patterns such as circular opening and corresponding transformation. For generating a continuous interweaving visual appearance, vertices of each face element are essential, namely, the order of the vertices of a polygonal face is sorted so that the constructive principle can be employed consistently to propagate through the entire set of faces in a coherent manner."

[Pseudo-code for Interweaving]

INTERWEAVE (***F***)

```
1  for each Face, F: // Face-based modular construction with sorted vertices //
2      if v in F<V> is sorted :
3          IWF1 <- MODULECONSTRUCTION (F<V>, true)
4          IWF2 <- MODULECONSTRUCTION (F<V>, false)
5          IWFnew <- IWF1 + IWF2
```

[Macro for Module Construction]

```
MODULECONSTRUCTION (F<V>, isModuleOne)
 1  new TrimCurveList // 1. New trimming curve List
 2  index = (isModuleOne)? 0 : 1
 3  for i = 0 to F<V>.Length:
 4      C <- new TrimCurve(F<V>[i], F<V>[(i+1)% F<V>.Length])
 5      if (i % 2 == index) then : // 2. Add trimming curve alternatively
 6          add C → TrimCurveList
 9  return new MODULEBYTRIMMING (F<V>, TrimCurveList)


MODULEBYTRIMMING (F<V>, tCrvs)
 1  S <- new Module Surface(F<V>)  // 1. new Module Surface construction
 2  S.Trim(tCrvs) // 2. remove surface area by designated trimming curves
 4  return S
```

In a sense, how designated modular counterparts interweave with each other is treated as a design operation and can be given by any sequence of instructions, such as trimming → Rotation → Mirror (Figure 3-19 and Figure 3-25). The above examples, from Figure 3-21 to Figure 3-28, demonstrate potential of incorporating trimming on different modular boundaries and corresponding transformation for alternate pattern constructions—such as, interwoven patterns shown in this Chapter. The hypothesis is that all face elements are properly configured such that the procedural operations can be built upon the order of the underlying topology. For instance, a well-structured quadrilateral face can be a face with vertices configured in the counter-clockwise order from a lower-left corner. Given a set of well-structured faces, a continuous interweaving appearance can be constructed by the simplified procedure with the underlying vertex topology. Notwithstanding, such a topological condition can only hold true while tessellating a completely untrimmed surface; thus regular tessellation pattern. With the increasing complex boundary conditions that often occur when designers utilize a freeform surface for the architectural design, regular tessellation pattern cannot be easily constructed, for instance, irregular polygons at trimming edges. The evolving complex boundary conditions need special treatment to correlate the subsequent pattern generations. In the Chapter 4, boundary conditions are first analyzed and this dissertation uses quadrilateral as an

example to illustrate the difficulty of customizing the tessellation problem when the boundary condition complex grows. The objective is to generate a quadrilateral mesh with minimized irregularities for subsequent pattern-based construction. More details in tessellating surface with irregular boundary conditions for procedural construction are given and discussed in Chapter 5.

Interwoven patterns that are derived from base patterns show the application of procedure-based approaches to design exploration. A major motivation for such interwoven pattern generation experiments is to be able to demonstrate a parametric modeling process that identifies procedures involved in pattern generation. This will not only serve as essential groundwork for subsequent surface tessellation, but also provide strategies to help designers in developing their own parametric modeling toolkits.

Chapter 4

Boundary-Driven Tessellation

In order to solve the problem of tessellating surfaces with irregular boundary conditions, the meshing process is treated as the foundation to extend for future architectural applications, such as surface panel design and physical construction. In this chapter, an automatic meshing generation workflow is described, which is based on a goal-driven process with the following three stages (see Figure 4-1). The workflow depends on the formation of three major *boundary-driven* components: BDTensor, BDCurve, and BDMesh. Boundary-driven refers to the computation in relation to the interpolation of featured surface boundary conditions. These terms are defined in the sequel. But first, the workflow includes following three stages:

Feature Selection

The first stage is to identify featured boundaries from the given surface to be meshed. This step utilizes both the existing and trimmed boundaries from the surface of interest. By default, all boundary curves of a given surface are considered for the computation. Notwithstanding, specific customized curves, such as partial curves(s) from the surface boundary and/or on the surface domain, can also be specified. Further details on how to incorporate customized curve(s) in the meshing process are discussed in Section 4.1.

Mesh Construction

The meshing process initiates from a seed, a starting BDTensor node, within the target surface domain. This can be given by user input, or stochastically chosen from the initial feature boundaries. Accordingly, a network of BDCurves is constructed and sorted by curve-to-curve intersections. The last step in the second stage is to fit the mesh faces by iteratively traversing the sorted curve network. More details are discussed in Sections 4.2 to 4.4.

Goal-Driven Optimization

After an initial mesh has been constructed, the goal-driven optimization is executed to search for a qualified meshing result with the designated constraints. In this stage, few sub-processes are devised to fine-tune the meshing results. Due to irregularities in the potential boundaries, certain violated mesh face elements are further refined at this stage; for instance, triangular faces with skewed angle(s) are removed (Tri-Face Removal). In some cases, unintended polygonal face elements occur at where multi-directional boundaries meet and these will need to be further decomposed into smaller quadrilateral elements (aka quadrangulation[9]). In addition, a mesh-smoothing algorithm is also implemented to regulate the dimensions of the mesh elements through the constructed mesh topology.

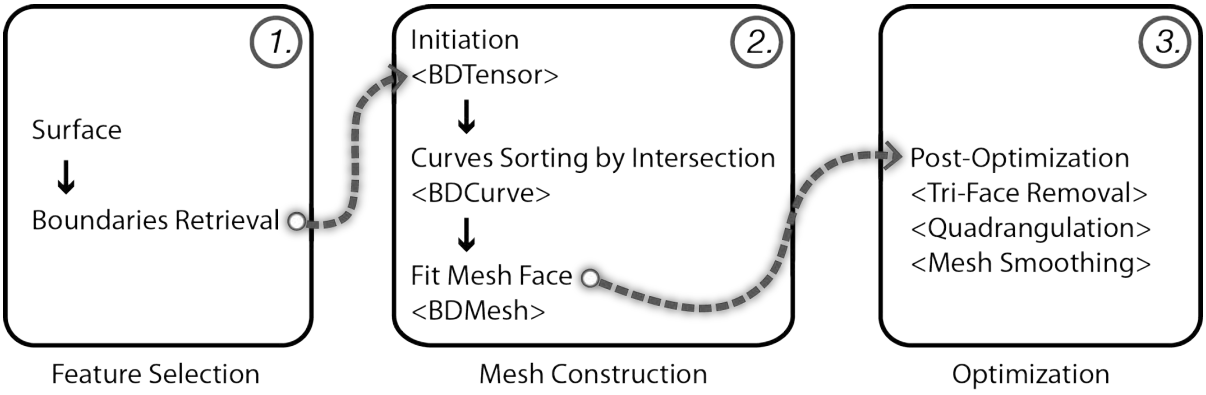


Figure 4-1 The proposed workflow for boundary-driven mesh optimization

For the demonstration purpose, the quadrilateral as the target pattern of the meshing result is considered. Tessellating a surface with only quad elements, especially with complex boundary

[9] Quadrangulation is a process of enforcing every mesh face element in a mesh to be a quadrilateral face and thus four-sided.

conditions provides a challenge that is dissimilar to the triangular mesh, since almost any surface can be tessellated with triangles. In this dissertation, a quadrilateral-meshing algorithm is proposed, which is derived from examining the influences of both the inherent and customized boundaries as a way of providing an alternative for freeform surface manifestation.

4.1 Interpolating Boundary-Driven Tensor

A geometric information object, namely, a multi-directional *tensor* is introduced to describe the boundary-driven computation. This tensor is an object that holds both directional information and corresponding scalars of a location of interest on a given surface. In particular for this study, the tensor object pertains to three directional vectors and corresponding scalars, which are interpolated by examining the relative distance relationship from the current location of interest to featured boundaries. These vectors are resampled values from the (1) tangent direction derived from the featured boundary curve(s), (2) normal direction at the location of the interest on the surface, and (3) binormal direction obtained from the cross product of the first two direction vectors. See Figure 4-2. *P* is the point on the boundary curve of the surface *S*. *T* is the interpolated tangent direction at *P* and *N* is the normal direction at *P* on *S*. *B*, the binormal, is the cross product of *T* and *N*.

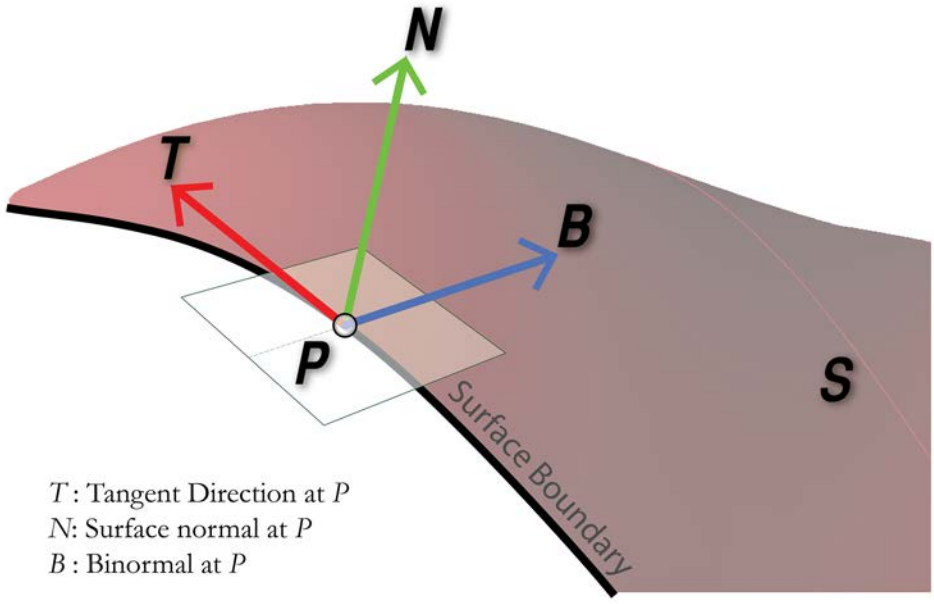


Figure 4-2 A tensor object, *P*, at surface boundary

Due to the fact that a tensor in this study is retrieved from a location of interest relative to the given surface boundaries, the term boundary-driven tensor (*BDTensor*) is employed. In a sense, BDtensor contains information for linear interpolation between multiple vectors and scalars of each individual entity. The output of a tensor node yields directional projections, which supports navigation through the given surface domain according to the influences from the feature boundaries. In Figure 4-2, *T*, *N* and *B* are three vectors maintained by a BDTensor object at *P*. By default, all the scalars are set to 1.

For any location within the surface boundary, a BDTensor is computed by its current location in relationship to the featured boundary curves. To compute the influences from featured boundary conditions, the *Inverse Distance Weighting* (IDW) method (Shepard, 1968) is employed in which a given number of interpolated values from the feature boundaries are resampled. The equation for a BDTensor at a node *u* is given by equation (4-1):

$$T(u) = \sum_{i=0}^{N} \frac{w_i(u) * u_i}{\sum_{i=0}^{N} w_i(u)}, \quad w_i(u) = \frac{1}{d(u,u_i)^{\rho}}, \quad 0 \le i \le N \tag{4-1}$$

$T(u)$ is the BDTensor at target node u. N is the number of source nodes utilized for interpolation; it can be potentially less than the number of boundary edges. Each w_i is a weighting function. Each u_i is a local interpolating node on a feature boundary edge, and d represents the distance function from a boundary node u_i to the target node u. ρ is the power parameter to smooth out the influences of the sampling boundary nodes.

Figure 4-3 illustrates a BDTensor *P*, which is interpolated by locally influential nodes, *pA*, *pB*, *pC*, and *pD*, on the feature boundaries, respectively *E_A*, *E_B*, *E_C* and *E_D*. Each node has directional vectors calculated in two conjugate directions with an associated weight. These directional vectors are remapped onto their local reference coordinate system where the *Z*-axis is normal to the node location. With tensor objects on hand, a *BDCurve* can be constructed by iteratively moving the node toward its next location along the interpolated direction. Figure 4-3 illustrates the difference between the underlying curve (derived from the uniform iso-parameters) and the interpolated curve (computed by the local boundary influences). The isoparametric curves are shown shaded green using a dashed pattern. *BDCurves* are shown colored blue or red with arrows. More details regarding how a BDCurve is constructed are discussed in Section 4.3. But first, tensor field initiation is described.

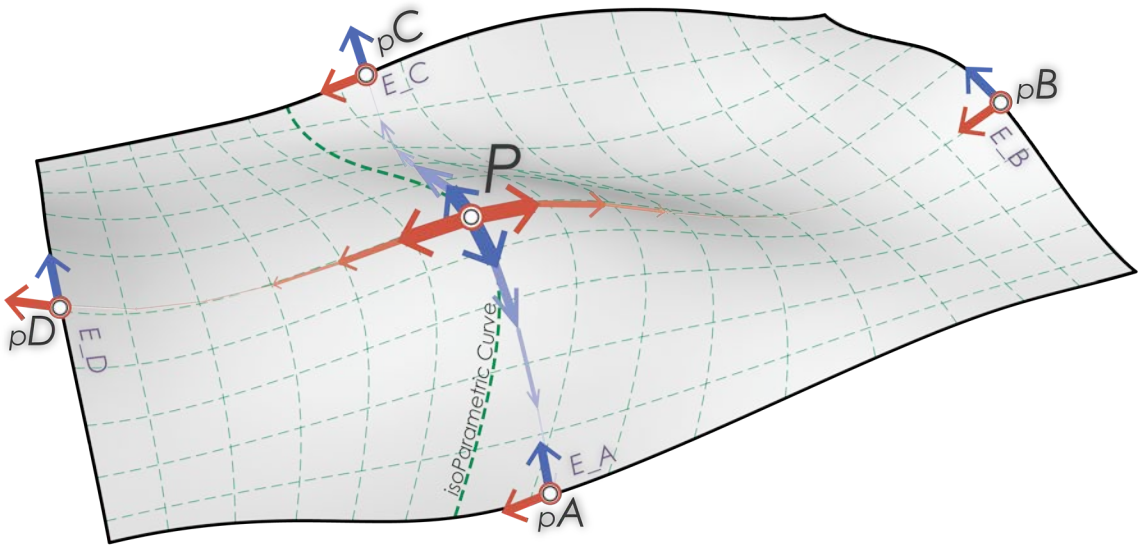


Figure 4-3 Conjugate curves
derived from boundary-driven computations and the underlying iso-parametric grid

4.2 Tensor field initiation

For a given surface of interest, the first step in the boundary-driven computation initiates the BDTensor interpolation. The process starts with a sampling grid system from the underlying surface *UV* domain. By evaluating the vectors from both the featured boundaries and inherited surface curvature properties at the location of interest, a tensor node is created. During the interpolation process, the weighted vectors are interpolated by parameterizing the influences from both featured boundaries and the surface curvature properties. Figure 4-4 illustrates three variations of tensor interpolation results from (1) boundary-driven analysis, (2) curvature-driven analysis, and (3) integration of both analyses (the images shown here use the same testing surface as Figure 4-3 but from a top view instead of a perspective view).

The top row image in Figure 4-4 is the result by evaluating tensors from the featured boundaries only. The bottom row images in Figure 4-4 demonstrate the results from integrating of both boundary and curvature analyses. The left image on the bottom row shows the tensors interpolated only from the surface curvature analysis; the right image is the parameterized result by integrating the influences from both boundary-driven interpolation and surface curvature analyses. All visualized vectors are remapped to the local reference coordinate system in a conjugate relationship.

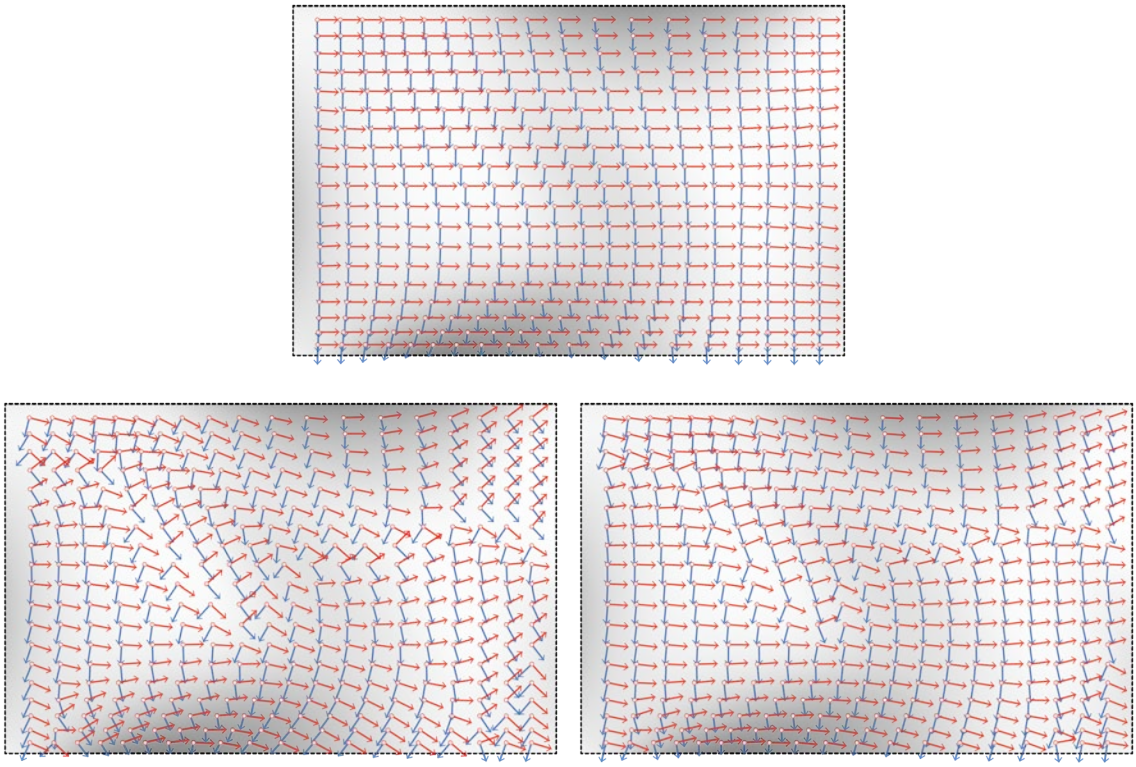


Figure 4-4 BDTensor field generation

(*Top*) Tensor field generated from boundary-driven analysis

(*Bottom-left*) Tensor field generated from only surface curvature analysis

(*Bottom-right*) Tensor field generated by integrating influences from both boundary-driven and surface curvature analyses

In addition to the whole-boundary analysis, boundaries utilized for tensor interpolation can also be specifically specified. For example, Figure 4-5 shows a customized tensor field interpolation from a customized source, namely, an additional curve on the target surface. Notice that the result shown here can also be derived from a trimmed surface with the same input curve. One of the advantages of taking additional curve(s) as the input parameter is to provide flexibility for exploring/customizing various alternative tessellations. For instance, tessellations can be derived from single to multiple boundaries inherent in the given surface; or, it can be derived from a set of designated curves on the given surface domain for customized pattern generation. Promoting this input source as an individual parametric handler makes the optimization process amenable to various possible scenarios that may occur during the iterative design exploration process. The image on the left in Figure 4-5 illustrates an additional curve for tensor field interpolation; the right side image shows the resulting tensor field calculated from the customized source.

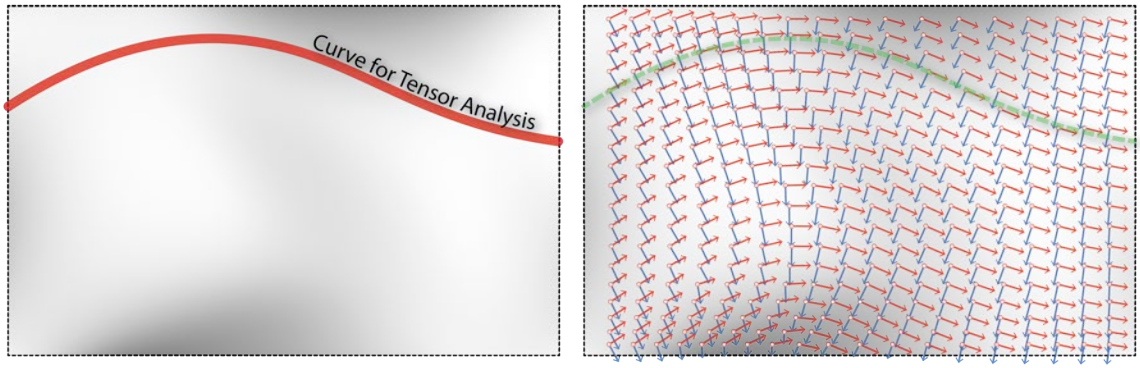


Figure 4-5 Customized tensor field generation by additional input curve on the target surface
(*Left*) Customized input curve for BDTensor interpolation
(*Right*) Tensor field interpolation result

During the interpolation process, not all boundaries are necessarily taken into consideration. In some cases, only partial boundaries are utilized. For instance in Figure 4-6, the target surface is a surface with an interior trimmed opening.

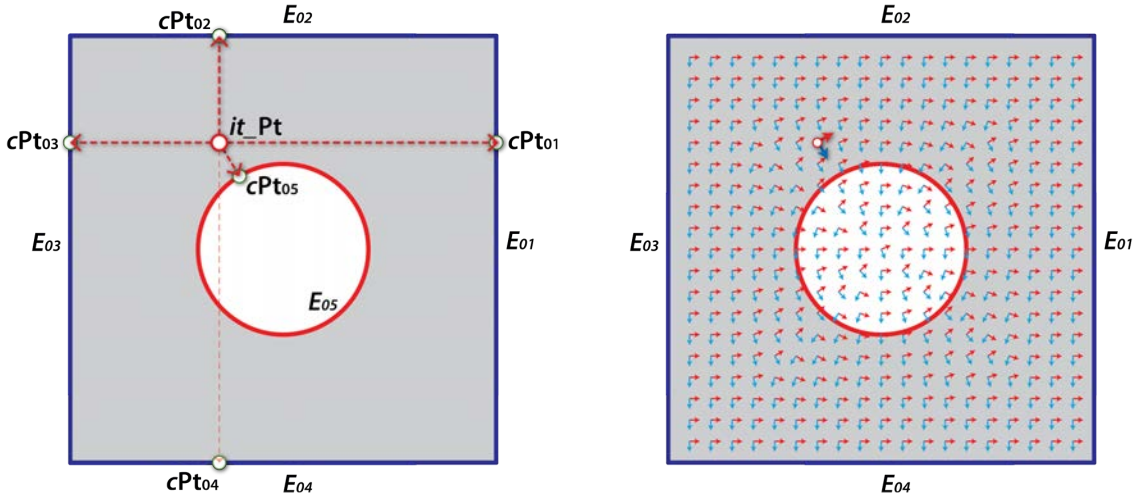


Figure 4-6 BDTensor interpolation by selected boundaries
(*Left*) Boundaries selection by evaluating point-of-interest visibility
(*Right*) Initial tensor grid visualization

A tensor, *it_Pt*, is considered as one of the initial tensor nodes in the meshing process. While processing all the influences from the featured boundaries, a visibility test is performed to filter out

only useful boundaries for the tensor computation. The shortest distance from the location of interest, *it_Pt*, to the featured boundaries are examined initially. cPt_{01} is the closest point location on Edge E_{01}, cPt_{02} for the E_{02} and so on. Among these closest points, cPt_{04} is currently invisible due to interference from the interior-trimming boundary, E_{05}, and thus is excluded from the interpolation process. The objective of this filtering process is to reduce the calculating errors that may occur by taking all boundaries into consideration.

In addition, the initial BDTensor interpolation is set to cover the entire untrimmed surface domain. The main reason for this is to ensure coverage of the sampling tensor field propagation, even for the area that are trimmed at the boundary edges. The right side image in Figure 4-6 shows that even for the trimming areas, there are sampling tensors calculated for subsequent BDCurve interpolation.

4.3 Boundary-Driven Curve Generation

A Boundary-driven curve (*BDCurve*) is approximated by a collection of BDTensors tangent to the underlying interpolated tensor field on the target surface.

To initiate a BDCurve creation, an initial interpolation node will be given either from the user input or randomly picked from the target surface. A *viable* node for BDCurve initiation is the node within the valid surface domain. For instance, in Figure 4-7, an initial node *P* is picked. The traveling distance *d* indicates how far along the current tensor direction will the next node proceed. By iteratively traversing the surface domain by guided direction from the underlying tensor field, two BDCurves will be constructed in a conjugate relationship, shaded in solid blue and red with shadow on the right side of Figure 4-7. The smoothness of the constructed curve can be improved by decreasing the stepping size, namely, the traveling distance *d* from the current location to the next destination. The shorter the traveling step is, the smother the BDCurve will be. The curve interpolation terminates as the traveling node reach the constrained radius on the existing surface boundaries.

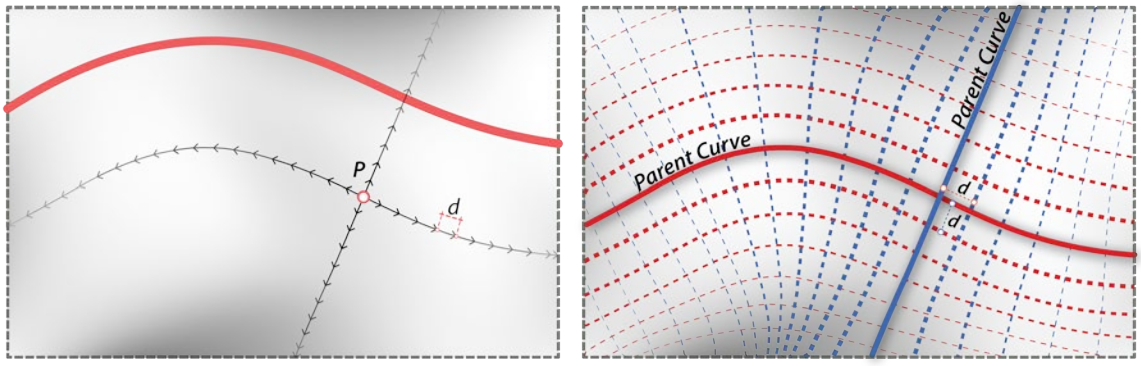


Figure 4-7 BDCurve interpolation process

After the initial pair of BDCurves have been created these two curves are used as the parent curve to derive subsequent BDCurves. To start, an offset distance, which specifies the distance from one curve to the consecutive one, is chosen. This distance is potentially larger than the traveling distance. By emitting the sampling nodes on the constructed BDCurves, the corresponding offset BDCurves are created. The right side image of Figure 4-7 illustrates the result of BDCurve network creation from a given start node *P*, via the parent BDCurve initiation, to the final offspring BDCurves propagation.

The following pseudo-code describes the BDCurve construction:

[Pseudo-code for BDCurve construction]

BDCurve (N_{start} , ***Dir, d***)

```
1  new List<N>; // new list for interpolated node along specified direction //
2  N ← N_start
3  While isWithinSurfaceBoundary(N):
4      add N → List<N>        // add current node to the Node list //
5      if isClose2Boundary(N, d):
6          N ← FindClosestPointOnBoundary(N);
7          add N → List<N>    // add last node to the Node list //
8      else :
9          dir_Curr ← InterplaterDir(N, Dir) // interpolate navigating dir //
10         N ← N + d* dir_Curr  // Update node to next location //
11 Curve_new ← new BDCurveConstructor(List<N>)
```

Figure 4-8 shows two different curve networks derived from (1) *UV*-based parameterization, and (2) boundary-driven interpolation. At first glance, the *UV*-based curve network (the left side image of Figure 4-8) is similar in appearance to the boundary-driven curve network (the right side image of Figure 4-8). However, they are very different in their formation. The former is interpolated solely from the underlying iso-parameters, the latter is computed from inherent boundary conditions. In comparison, the boundary-driven network conforms to the inherent surface boundary conditions more strictly than the *UV*-based curve network and thus produces less sheared quadrilateral faces. This property ensures a well-configured framework for the tessellation pattern, particularly, with considerations to the boundary edges.

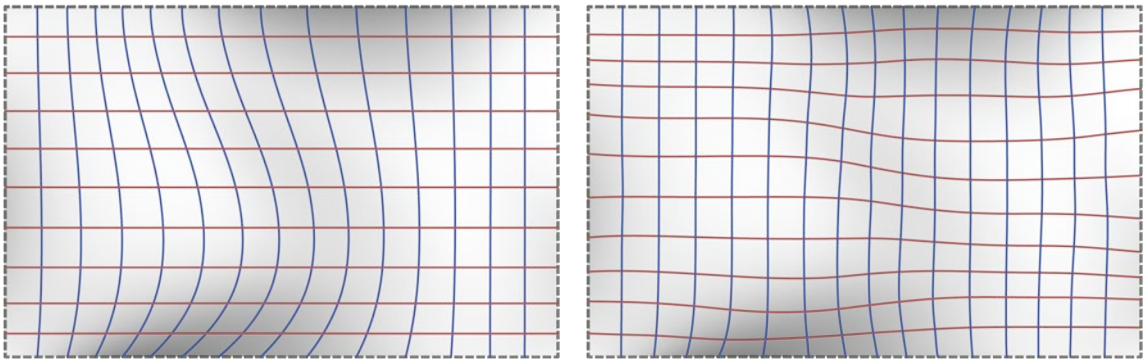


Figure 4-8 (Left) *UV*-based curve network; (Right) BDCurve network

While interpolating the curves from featured boundary conditions, the underlying surface curvature analysis can also be examined for further optimization. The images in Figure 4-9 demonstrate various curve generations by remapping the influences from featured surface boundaries and inherent surface curvature analysis. The examples shown here are based on the same tensor fields illustrated in Figure 4-4. In brief, the influence from the surface curvature is gradually increased from the top to the bottom in the figure below. Also, the value of the dimensional constraint for the curve network generation is decreased from the left to the right such that the generated curve pattern shown on the right is denser than the one on the left.

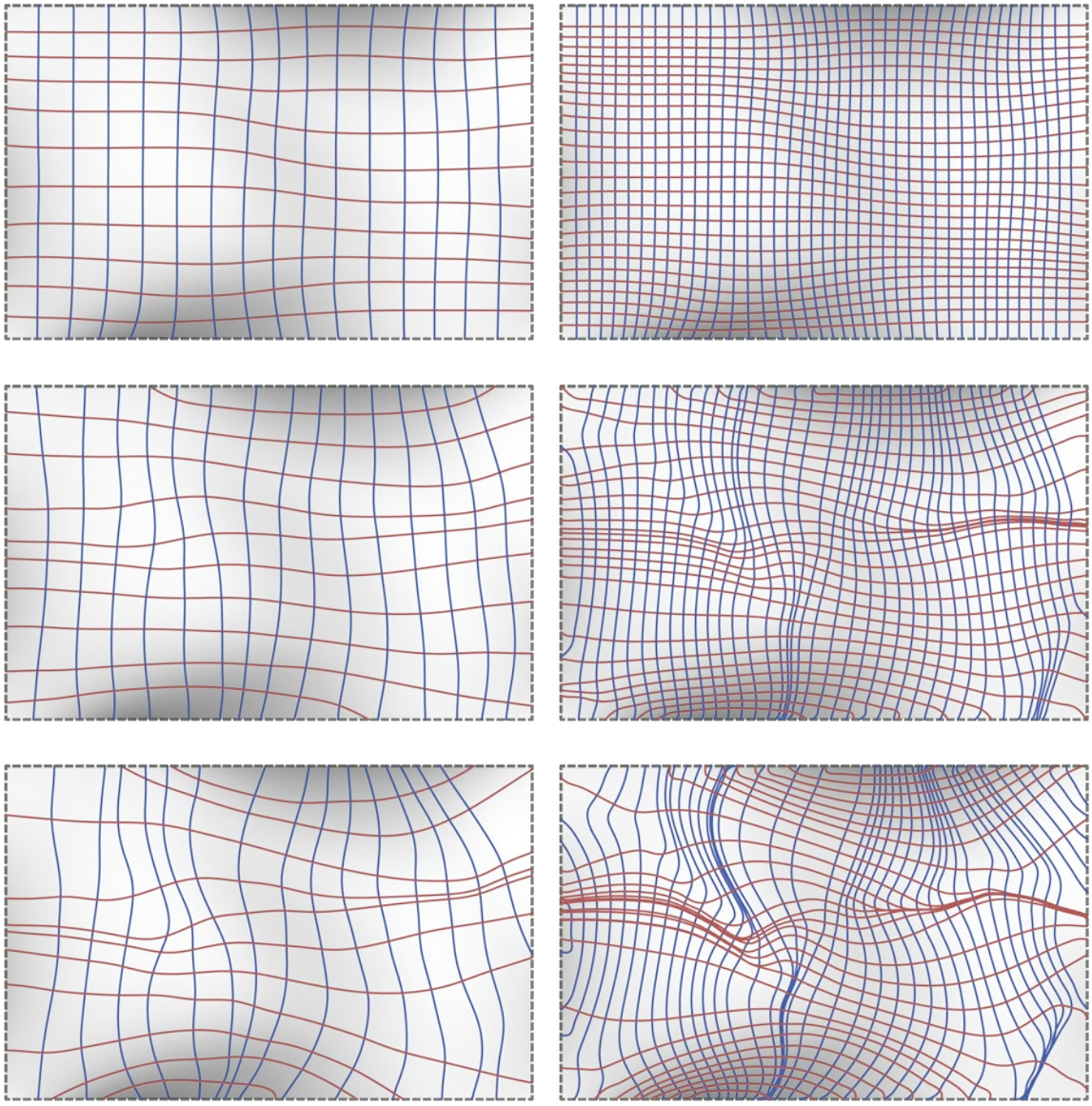


Figure 4-9 BDCurve network generated by interpolation influences of (1) the featured boundaries and (2) underlying surface curvature

As previously mentioned, additional curves can also be treated as additional input for the tensor field interpolation. Figure 4-10 shows the BDCurve network derived from a customized tensor field depicted in Figure 4-7. The capability of supplying customized sources for boundary-driven interpolation provides flexibility in exploring potential pattern designs.

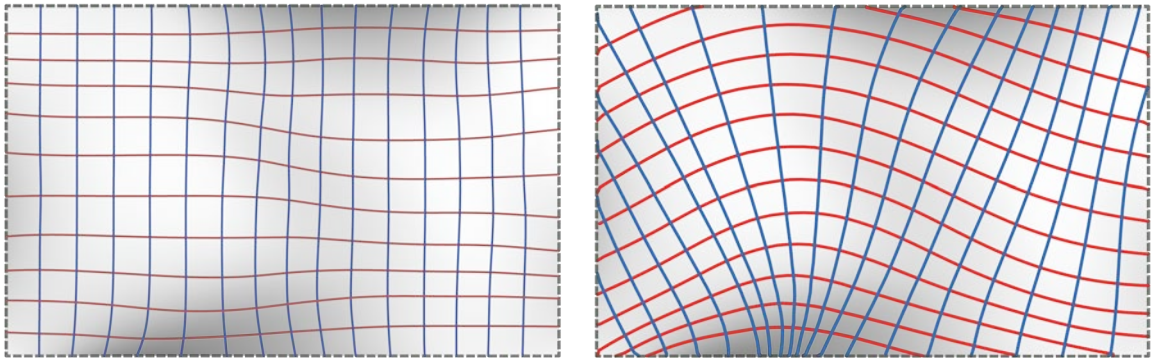


Figure 4-10 BDCurve networks derived from
(*Left*) the original surface boundaries (*Right*) a customized boundary source

When BDCurves are created, they are grouped by their origin, namely the directions along which they were derived. For the constructed BDCurves, curve-to-curve intersections are evaluated. These intersections are used to formalize the unsorted *BDCurves* through the formation of an interconnected network. There are three types of intersections, including (1) intersections between two conjugate curves, (2) intersections between these conjugate curves with original boundaries, and (3) intersections of two original edge curves (which are the original corner vertices).

The data scheme for intersecting nodes and the associative BDCurves is now described. As shown in Figure 4-11, the intersecting node, P_1, is a data object, which maintains the information of the intersection event between two curve entities, $Curve\_01$ and $Curve\_02$. This node keeps tracking the closest neighboring nodes by the parametric order on the curves to which these intersection events belong.

The *parametric order* of each node along the associative curve is determined by its parameter, t, which is often utilized for interpolating points on the governing curve domain. For practical reasons, the parametric domain of a given curve is normalized; thus, end points of a normalized curve have t = 0.0 and t = 1.0 respectively. In Figure 4-11, the graph node P_1 (t = 0.3) has a predecessor node P_0 (t = 0.2) and a successor node P_2 (t = 0.4) on $Curve\_01$; likewise, P_1 also maintains connectedness information to its predecessor and successor nodes, P_3 and P_4, along $Curve\_02$. By default, there are a total of four neighboring nodes connected to each node, including two predecessor and successor nodes from two intersecting curves; the exception occurs while these nodes are generated by (1) intersections between conjugate curves with original boundaries and (2) intersections between two original edge curves.

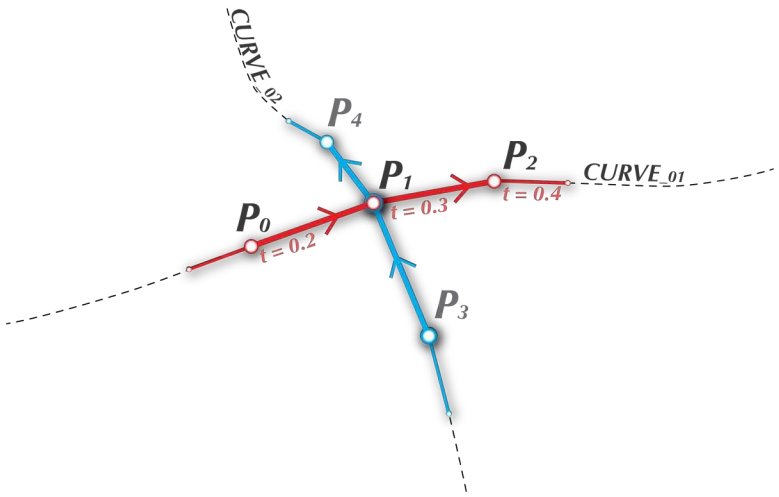


Figure 4-11 A mesh node as the data type to maintain parametric order on the curve to which it belongs

In Figure 4-12, there are three kinds of curves: the original edge curves (shown shaded in green) and two directional curves in a conjugate relationship (shown shaded in blue and red). In the connected structure, the graph nodes (intersection points) represent mesh vertices. For example, in Figure 4-12, *ItP_X₃Y₂* is a mesh node created by intersecting *BDCrv_X₀₃* with *BDCrv_Y₀₂*, and connected to its neighboring nodes, *ItP_X₂Y₂* and *ItP_X₄Y₂*, in their parametric order on *BDCrv_Y₀₂*, and is also connected to mesh nodes, *ItP_X₃Y₁* and *ItP_X₃Y₃*, along *BDCrv_X₀₃*.

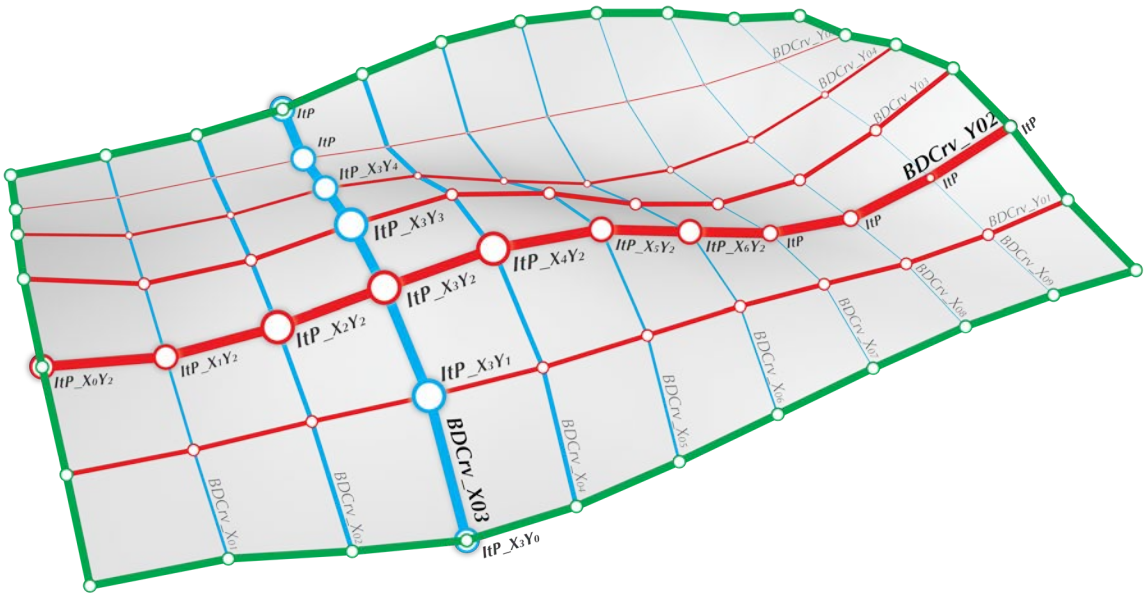


Figure 4-12 Curve-curve intersection to construct boundary-driven mesh nodes

In summary, the generated mesh nodes in the network are not necessarily the same as the sampled BDTensors created in the first step. Instead, they are remapped nodes on the curve network, which governs the formation of the boundary-driven mesh (*BDMesh*). BDMesh governs the topological information of the discrete model optimized with the feature boundary conditions. More detail on how these sorted nodes are revisited to build the corresponding mesh edges and faces by the mesh topology solver is discussed in Section 4.4.

4.4 Meshing with the Boundary-Driven Curve Network

To optimize a target surface with discrete elements, the feature boundary conditions to formalize a representative curve network is investigated, from which curve-to-curve intersections are performed to sort the underlying topological relations. In a sense, an intersecting node is essentially the mesh vertex and provides an easy access to construct the final mesh elements with the interconnected topological relationships.

4.4.1 Mesh Topology Solver

With the topologically sorted mesh nodes, a mesh topology solver is executed to construct the corresponding mesh faces and edges from this interconnected curve network. The algorithm initiates a search by visiting existing mesh nodes in the network and consecutively determines the shortest path between its current neighboring nodes to form corresponding faces.

For example, Figure 4-13 illustrates the topology solver: V_4 is the origin of the search, and is connected to V_1, V_3, V_7, and V_5 in counter-clockwise order. (When mesh nodes are sorted, their topological relations are also structured in a counterclockwise fashion along the current normal direction at the current location). The shortest path approach is adopted to fit a best-matched mesh face.

In Figure 4-13, to construct faces connected to mesh node V_4, the process looks at potential paths connecting pairs of its neighboring nodes, for example, [V_7, V_5]. The goal is then to find the shortest path from V_4 to V_5 via V_7. To create the mesh face, F_1, the algorithm searches depth first by looking at the connected neighbors of V_7, and finds three potential paths consisting of neighbors, V_6, V_{10} and V_8. By continuously advancing to their consecutive nodes in the network, a shortest path of [V_4, V_7, V_8, V_5] can be found and nodes found at this path are then utilized as the vertices for form a new mesh face. This searching process terminates immediately once a shortest path has been found, for instance, path

of $V_4 \rightarrow V_7 \rightarrow V_8 \rightarrow V_5$, or, when a face element that shares the same nodes in the initial search set already exists. By sweeping through all mesh nodes in the network, the initial mesh is constructed.

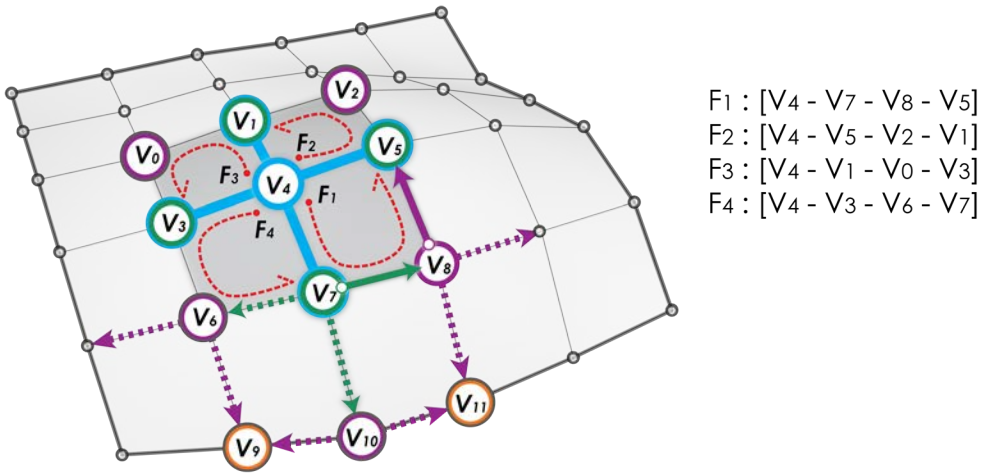


Figure 4-13 Fitting mesh faces by shortest path search

The pseudo-code below describes the mesh face fitting process and the recursive function for shortest path search:

[Pseudo-code for BDMesh Topology construction]

```
/* Constructing the mesh topology by shortest path search on the sorted curve network */
MeshTopologyConstruction (Nodes):
 1  for each sorted node, N, in the curve network;
 2      for each pair of connected nodes, [N_start, N_end] of current Node(N):
 3          if P ← ShortestPathExist(N, N_start, N_end):
 4          then MF_new ← MeshFace(P)
 5               UpdateMeshFace(MF_new) in the mesh topology
```

[Pseudo-code for Shortest Path Search]

```
/* Recursive function for the shortest path search */
ShortestPath (N_source, N_start, N_target):
 1  add N_source to List<N>
 2  while (N_target is NOT found) && (List<N> is NOT empty):
 3      new List_Temp<N>
 4      for each node, N_Current, in List<N>:
 5          for each neighboring node, N_Child, of N_Current<N>:
 6              if N_Child == N_Current:
 7                  return currently found path – P
 8              else:
 9                  add N_Child → List_Temp<N>
10      update List<N> ← List_Temp<N>
```

4.4.2 Mesh Refinement

Owing to possible complexity of boundary conditions associated with arbitrary surfaces, the preliminary boundary-driven mesh (BDMesh) will be potentially composed of triangles, quadrilaterals, and other polygonal face elements. To ensure a quad-dominant mesh, additional mesh refinement functions are required to remove skewed triangles, and to construct quad-dominant faces from arbitrary polygonal faces. Lastly, a mesh-smoothing operator is introduced to relax the constructed mesh topology. The conditions and procedures for refining the mesh elements to a well-structured quad-dominant mesh are discussed below.

Removing Skewed Triangles

To control whether a triangle face is skewed for removal is specified by a threshold parameter, which is calculated by the ratio of the smallest and largest interior angles of a triangle face. The threshold parameter may also be set by user input. When the ratio is less than the specified value, the triangle is tagged for removal. Figure 4-14 illustrates skewed triangle removal carried out by vertex replacement. The vertex with the largest interior angle of the triangle is removed, and is replaced by its nearest vertex in the triangle. All topological entities associated with this tagged vertex, such as edges and faces, are updated accordingly. For instance, after replacing the vertex (colored in dark grey) by the

existing vertex (shaded in light grey) in the network, edges e_1 and e_5 are removed and a new edge, e_{new}, is updated for mesh face F_1.

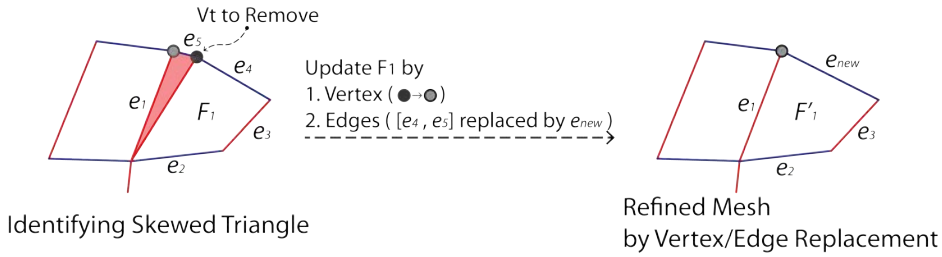


Figure 4-14 Skewed triangle removal

Mesh Quadrangulation

When tessellating the given surface with the constructed curve network, polygonal face elements may be produced at points where multiple boundaries meet. These polygonal faces are subdivided to convert the initial mesh into a quad-dominant mesh containing only quadrilateral faces in the network. The subdivision process is carried out by edge mid-point and face center vertex insertions. Figure 4-15 illustrates new quadrilateral faces being formed by recursively connecting the center of an existing polygonal face to the mid-point of the edges together with the original face vertices. A polygon with N edges will yield N corresponding quadrilateral faces. This mechanism can be applied to any arbitrary polygonal shape.

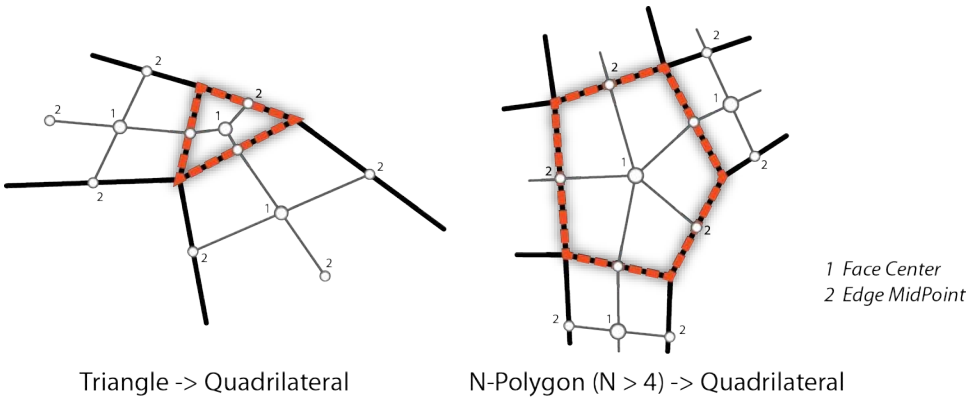


Figure 4-15 Quad meshing by face center and edge midpoint insertion
(*Left*) Quadrangulate a triangle face; (*Right*) Quadrangulate a 5-sided polygon face

Mesh Smoothing

The quality of the resulting quad-mesh can be improved by mesh smoothing, also called mesh relaxation. A Laplacian smoothing algorithm (Herrmann, 1976) is considered with local perturbation to ensure that the smoothed result conforms to the original input surface. To start, this process computes the new vertex locations by a finite difference approximation of the Laplace operator, which moves a mesh vertex toward the centroid of the connected vertices. The equation is written as follows:

$$P_{new} = \frac{1}{N}\sum_{i=0}^{N} \alpha_i P_i \tag{4-2}$$

where α_i is the weighting factor for each connected mesh vertex.

As the initial BDCurve network is generated from the conjugate relationship, the connected vertices of a BDMesh vertex will likely form a convex polyhedron. (Exceptions occur at vertices on the surface boundaries). For interior vertices bounded by convex hulls, the new locations derived from centroids of these polygons remain inside the original boundary. This property ensures homogeneous mesh generation and maintains the original anisotropic configuration. However, for peripheral vertices on the original boundaries, special treatment is needed. For example, vertices that are moved away (inside or outside) the original boundaries will need to be adjusted so that the mesh stays as close as possible to the original surface. In other words, this constraint enforces the conformity of the inherited surface boundaries. Two cases of mesh vertex replacements are illustrated in Figure 4-16. In addition, corner vertices belong to the third scenario where they will not be modified in order to keep the original boundaries intact.

After smoothing mesh vertices, local modulation of mesh vertex location is through vertex perturbation. There are two types: (1) vertex-to-edge and (2) vertex-to-face perturbations. *Vertex-to-edge* perturbation moves the mesh vertex back to the closest boundary (shown in the right side image in Figure 4-16). Likewise, *vertex-to-face* perturbs the vertex onto the input surface (shown in the left side image in Figure 4-16). By so perturbing the mesh vertices either to the nearest location on the boundaries or onto the surface, the refined *BDMesh* can better represent the given surface. It also conforms to the given boundary conditions.

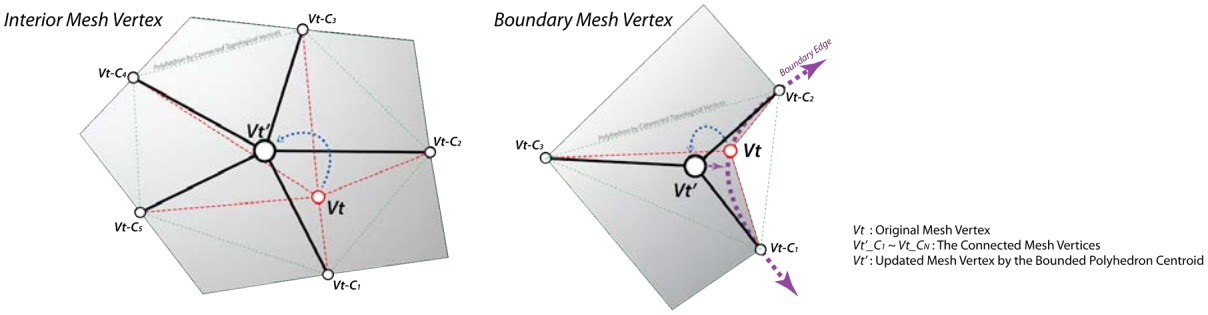


Figure 4-16 Mesh vertex replacement

(*Left*) Interior vertex: Replaced by the centroid of a convex polyhedron

(*Right*) Boundary vertex: Moved from the original boundary and then adjusted by vertex perturbation

In the following figures, two examples are demonstrated with the boundary-driven optimization. The first example is an untrimmed surface and the second is the same surface with both interior and exterior trimming. For example, Figure 4-17 illustrates the resulting smoothed surface tessellation with only the featured boundary conditions from the untrimmed surface domain. The top row images are the tessellation without mesh smoothing from both the top view (left) and perspective (right); the middle row shows the resulting mesh with mesh smoothing; the bottom row is a further quadrangulation of the same tessellated pattern.

To go a step further, another trimmed surface is examined with the proposed meshing process. Figure 4-18 is the same surface as shown in Figure 4-16 with additional trimming edges (shown shaded red curves in Figure 4-18) for the optimization. By taking the new boundary conditions, initial BDCurve patterns are first derived, as shown in the top row images of Figure 4-19; the middle row of Figure 4-19 shows the smoothed mesh result; the bottom row of Figure 4-19 is the further quadrangulation of the resulting mesh. The quadrangulation guarantees all the mesh faces to be quadrilateral. The two examples shown in Figure 4-17 and Figure 4-19 demonstrate the result of applying the boundary-driven optimization. The resulting mesh not only has more equi-dimensional mesh faces, but also has relaxed boundary edges.

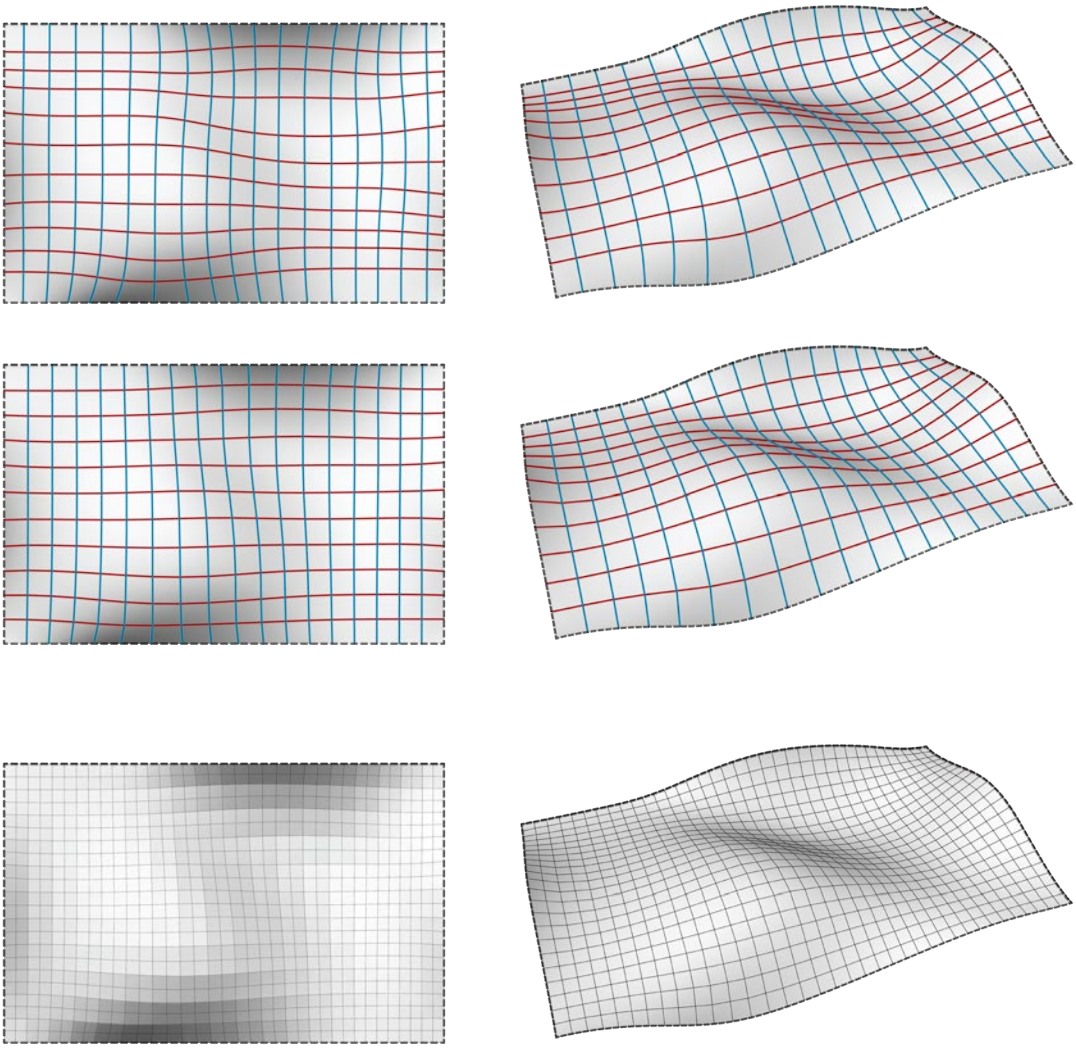


Figure 4-17 BDMesh smoothing

(*Top*) BDMesh without mesh smoothing (*Middle*) BDMesh with mesh smoothing

(*Bottom*) BDMesh with mesh smoothing and further quadrangulation

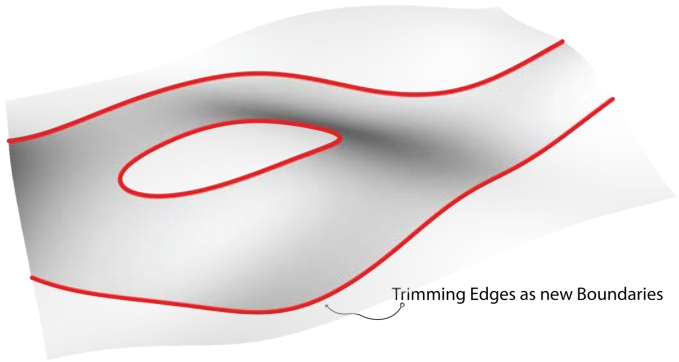


Figure 4-18 New boundary condition introduced by the trimming operation

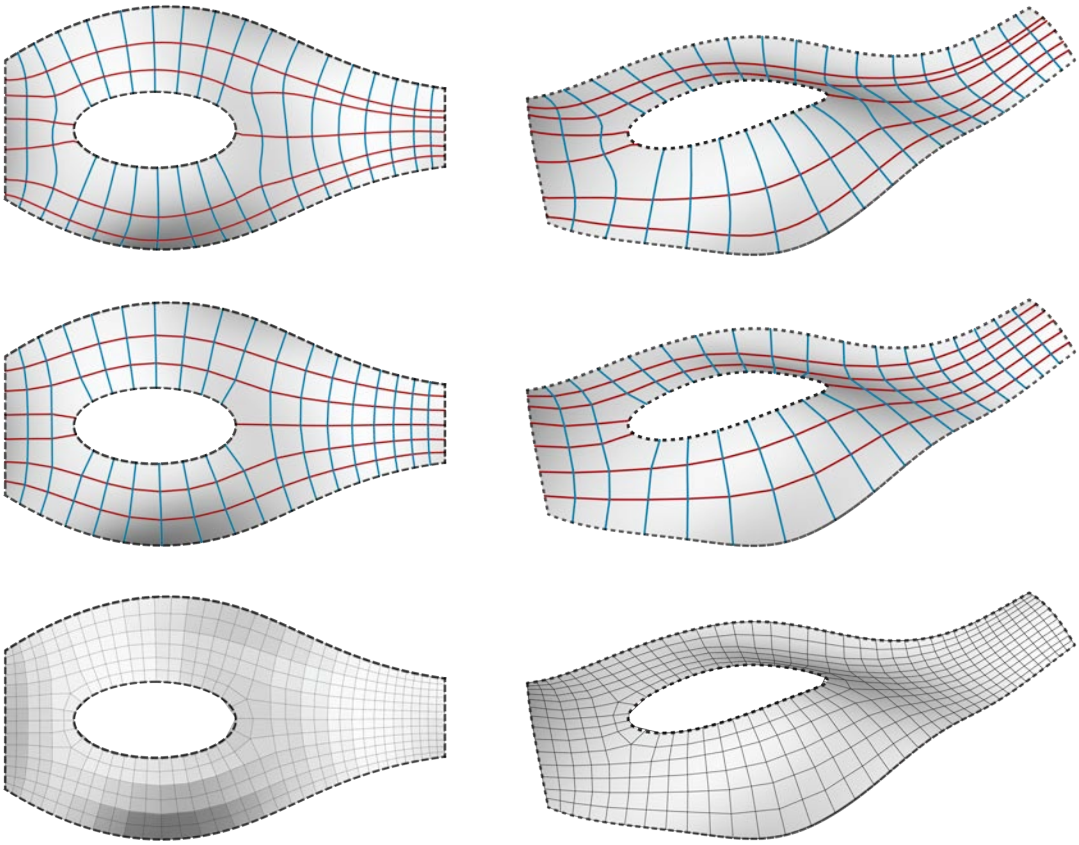


Figure 4-19 BDMesh a trimmed surface with smoothing
(*Top*) BDMesh without mesh smoothing (*Middle*) BDMesh with mesh smoothing
(*Bottom*) BDMesh with mesh smoothing and further quadrangulation

4.5 Mesh Analysis

In this section, the goal-driven optimization processes for solving issues from surface boundary conditions are highlighted. Results derived from the BD-driven optimization are checked against the conventional *UV*-based tessellation for comparison and analysis.

Surfaces shown in Figure 4-3 and Figure 4-17 are first utilized to compare the differences in the discretized models that are derived from *UV*-based and BD-driven approaches. For instance, when a surface remains untrimmed, it is straightforward to apply both approaches to generating quad-dominant meshes. However, while BDMesh continues to generate quad-dominant mesh even as the complexity of the boundary conditions increase, *UV*-based tessellation fails at the boundaries, where irregular polygonal elements occur, for instance, irregularly trimmed polygonal faces.

To perform the analysis, some properties in relation to the mesh elements are first identified. These properties serve as the analytic indices for comparison. These include: (1) face warping; (2) face area; (3) edge length; and (4) surface conformity. In one sense, these indices are employed to distinguish the geometrical properties exhibited in the discretized models and imply applications for future physical construction. For instance, *face warping* refers to the degree of distortion of a quadrilateral face and this can be utilized as the index for manufacturing with planar, single- or double-curved panels. *Face area* indicates the potential fabrication constraints that are embedded in the machinery or materials such as the minimal/maximal dimension of an applicable glass pane. Likewise, *edge length* denotes the minimal/maximal structural frame elements. Lastly, *surface conformity* is the deviation from the discretized model to the original surface and is checked to ensure the closeness from the meshing result to the original input surface.

To start, the untrimmed surface is examined. Table 4-1 shows the fundamental information from both the *UV*-based and BD-driven models, including the number of vertices, edges and faces. Notice that the number of elements is regulated as the foundation for comparison and thus both models have the same number of vertices, edges and faces. Table 4-2 provides the analytical data by computing four differential index values of interest—warping, area, edge length and surface conformity. The differential index, d_i, is computed by averaging the difference between each sampling value to the mean value. The calculating function is given by equation (4-3):

$$d_i = \frac{1}{N}\sum_{i=1}^{N}\left(x_i - \mu\right), \quad \text{where} \quad \mu = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{4-3}$$

x_i represents single sampling value and μ is the mean value of the current set of sampling values.

	UV-Mesh	*BD-Mesh*
***NF**, Number of Face*	864	864
***NE**, Number of Edges*	1788	1788
***NV**, Number of Vertices*	3456	3456

Table 4-1 Mesh configuration

	UV-Subdivision	*BD-Mesh*
$\boldsymbol{Diff}_w$, Warping ratio ***Difference***	0.015156	0.014681
$\boldsymbol{Diff}_{Fa}$, *Face Area* ***Difference***	0.274045	0.242099
$\boldsymbol{Diff}_{EgL}$, Edge length ***Difference***	0.131189	0.098543
$\boldsymbol{C}_{Surf}$, ***Conformity*** to the input Surface	$2.6526e^{-5}$	$2.2582e^{-5}$

Table 4-2 Mesh analysis by checking against
(1) face warping; (2) face area; (3) edge length; and (4) surface conformity

The shaded warping difference results are illustrated in Figure 4-20. In one sense, for an untrimmed surface like Figure 4-20, it is straightforward to divide using simply *the UV* parameters. At first glance, the *UV*-based tessellation conforms to surface continuity nicely; however, due to the underlying surface characteristics such as surface curvature and boundaries, most face elements are double-curved faces with potential vastly varied areas and shear angles. See the top row image in Figure 4-20. In this figure, the faces colored shaded red represent the faces with the minimal degree of face distortion and the shaded blue elements are with maximum face distortion. The bottom row image shows the other model optimized by BDMesh. By comparing the results from both approaches, the proposed BDMesh result indicates the potential in minimizing the overall difference among generated mesh elements. In other words, the *UV*-based subdivision has more diverging elements with distinct properties across the entire surface domain. In addition, the discrepancies among the generated BDMesh elements such as edge length ($Diff_{Egl}$) and face area ($Diff_{Fa}$) are also minimized, as

shown in Table 4-2. Overall, the differential indices from the BD-driven approach is better that the conventional *UV*-based approach. Yet, some properties, such as face warping, are still limited to the underlying surface curvature.

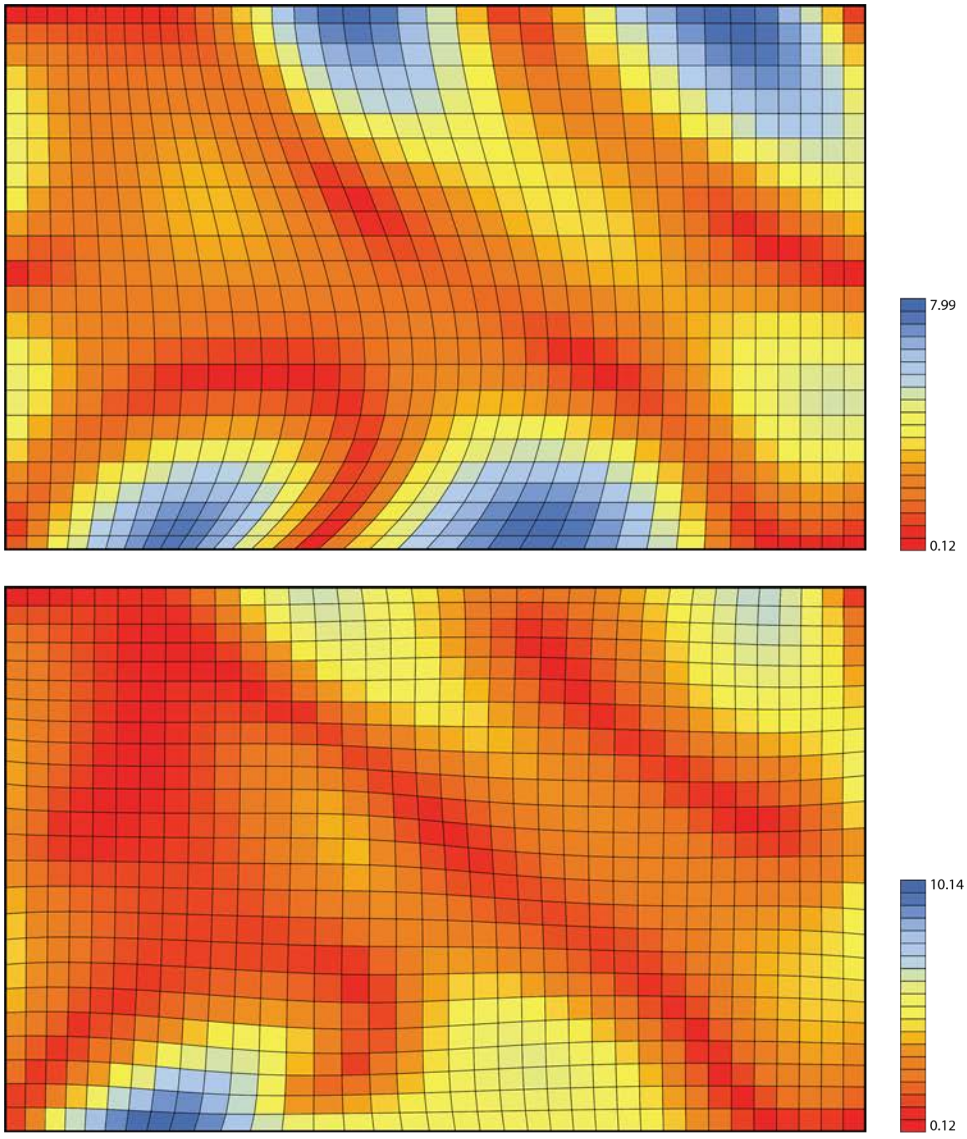


Figure 4-20 Mesh Warping analysis

Similar comparisons are also made by gradually introducing the trimming operations onto the same surface. The trimming operations used here are trimming curves illustrated in Figure 4-18. The analytical results are shown in Figure 4-21. On the left column, the first trimmed surface example is

given, and on the right column, the second trimmed surface with both interior and exterior trimming. Notice that for both examples, *UV*-based results have the same shaded colors due to the original untrimmed face analysis. From a physical construction point of view, the trimmed elements are simplified as the same type of face construction yet with an additional cut. The bottom row images in Figure 4-21 demonstrate two different results, in which meshing elements are optimized by the featured boundary conditions and therefore shaded distinctly. For both trimmed surface examples in Figure 4-21, the whole boundary conditions are taken into consideration. As shown in the bottom-right image, the most distorted elements occur at the rounded corner of the interior trimming edge. This is due to the fact that BDMesh imposes the constraint on keeping both the integrity of the surface boundary and the dominant pattern—namely a quadrilateral face. When fitting surfaces with only quad elements, it is indispensable to have some of these irregular regions particularly at the boundary where direction alters at a sharp angle, such as at corner areas. Overall, BDMesh results demonstrate the strength in generating mesh elements with a conjugate relationship.

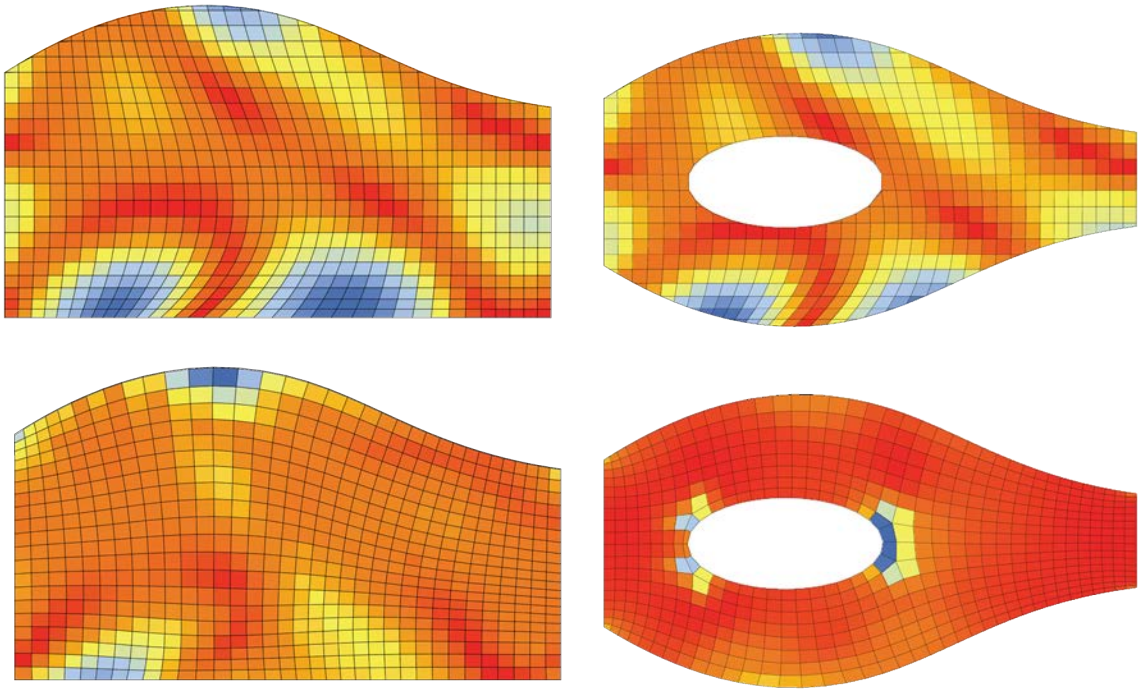


Figure 4-21 Meshing trimmed surfaces with face warping analysis.
(*Left*) Trimmed Surface Type 1 (*Right*) Trimmed Surface Type 2
(*Top*) UV-based subdivision (*Bottom*) BD-Driven optimization

Chapter 5

Application:

Pattern-Based Tessellation

In this chapter, an application of the pattern-based tessellation is given. The objective is to demonstrate the boundary-driven approach to the surface tessellation problem, and extend this mesh result with customized patterns derived from the Archimedean and interwoven constructions presented in Chapter 3.

Overall, the implementation of pattern-based surface tessellation can be divided into following three steps:

(1) Meshing the target surface with the given boundary conditions

An initial discrete version of the target surface is derived from the given boundary conditions, which may be derived from the inherited surface boundaries or customized by user input. The output of this step is a quad-dominant mesh, which features a conjugate relation network.

(2) Pattern generation

Patterns generated by such constructive rules as illustrated in Chapter 3 are provided for potential polygonal pattern generation, which serves as the basis for subsequent surface panel development.

(3) Construction of surface panel components

In the last step, panel components, such as, panels, structural beams or connecting joints, are procedurally constructed from the corresponding mesh elements—faces, edges and vertices. This step demonstrates the conversion from conceptual tessellation pattern generation to real building component manifestation.

5.1 Application platform

For practical purposes, a series of operational modules are presented and implemented in an *object-orientated programming* (OOP) fashion using Microsoft .NET C# with RhinoCommon *Software Development Toolkit* (SDK) and Grasshopper. RhinoCommon SDK is an extended .NET version of the OpenNURBS library, which was founded by Robert McNeel for supporting CAD, CAM, CAE and three-dimensional graphical software development (McNeel, 2010). In the implementation, RhinoCommon SDK is used to support the geometry data imported from Rhinoceros® 3D, a 3D modeling environment for NURBS surface construction. Grasshopper is a plugin developed on Rhinoceros® 3D and utilized as the platform to construct the relationships among various operational modules to perform the boundary-driven optimization. The proposed modules are encapsulated as Grasshopper components using RhinoCommon and Grasshopper SDKs for fast prototyping. Figure 5-1 illustrates a BDTensor module consisting of five input parameters (on the left-hand side of the component) and three output parameters (on the right hand side of the component).

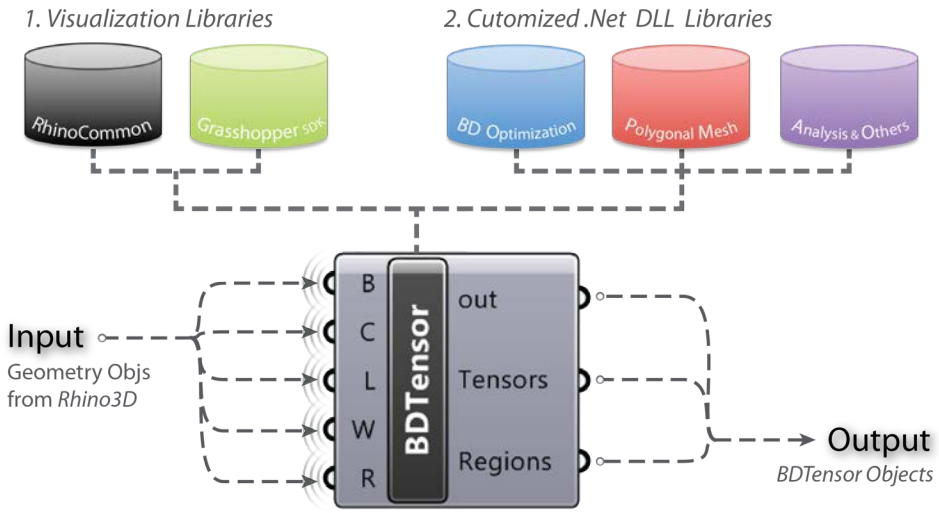


Figure 5-1 BDTensor encapsulated as a Grasshopper component

The current implementation is deployed as a dynamic-link library (dll) and ported to a collection of customized components in Grasshopper. The implemented library manages the links to RhinoCommon SDK for geometry data interoperation, and provides customized objects to compute and store the required information for surface tessellation. By utilizing the OOP format, the implemented library can be potentially applied in any other .NET Framework based platform with the least amount of effort. Figure 5-2 shows three major boundary-driven components, which are connected as a directed graph network captured from the main graphical user interface (GUI) of Grasshopper. Curves connecting from one end to the other represent the data flow displayed in a left-to-right order.

In Figure 5-2, *Input Surface (AudiSurf)* is the target surface for the tessellation; *Input Curves* are potential curves for customized tensor field interpolation and they can be part of the surface boundaries, or additional curves on the given surface; *Input Nodes* are initial seeds for the BDCurve interpolation. BDTensor, BDCurve and BDMesh are grasshopper components implemented for boundary-driven optimization.

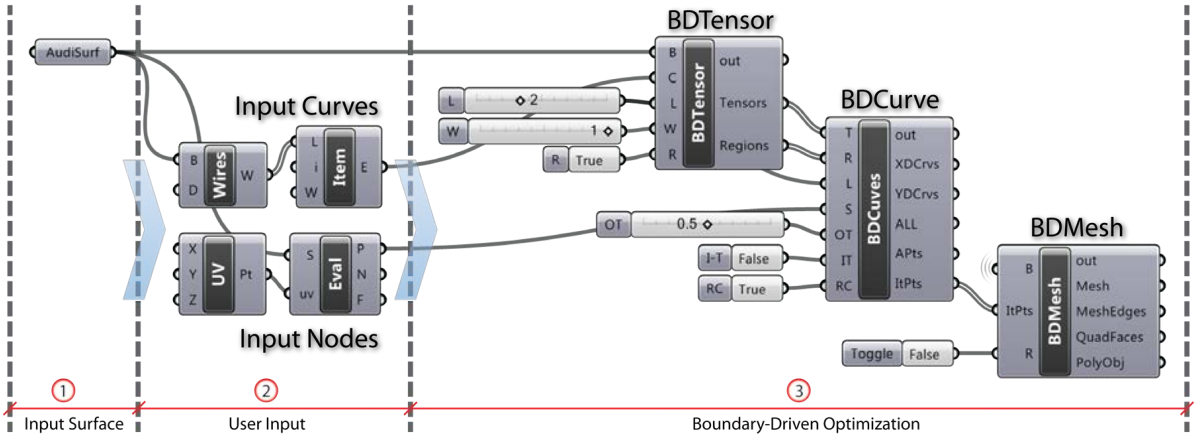


Figure 5-2 Boundary-Driven components in Grasshopper GUI

To summarize, the system workflow initiates from the given surface and is procedurally examined by individual components from the BDTensor, BDCurve to BDMesh (via BDTopology) with optional input parameters, such as additional curves and initial seed for BDCurve construction. In Figure 5-3, the constructive relationships among these operational modules during the optimization process are presented. As shown in the figure, the *Surface* object is the dominant source, and is the seed for initiating the entire process.

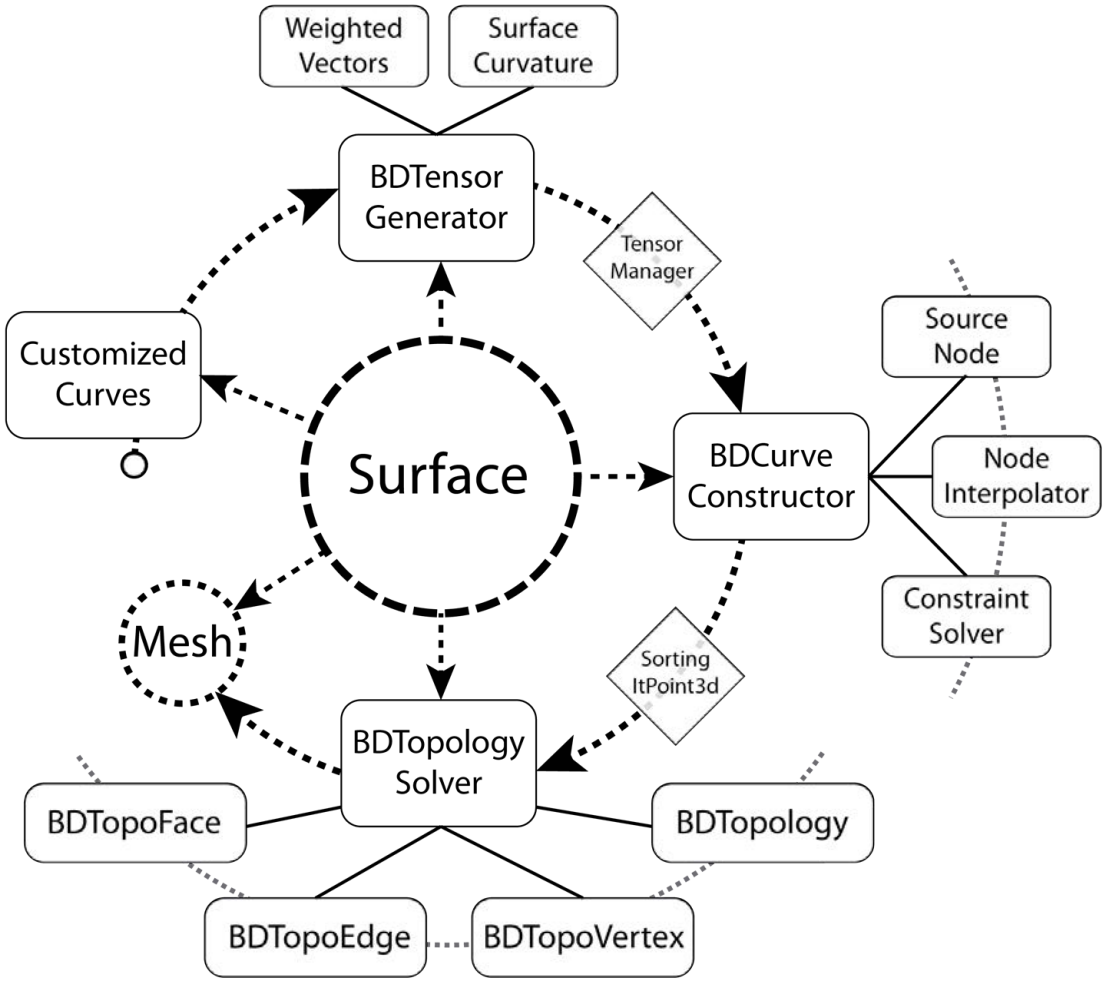


Figure 5-3 Elements of the boundary-driven optimization

For each modular component, there are auxiliary objects generated to maintain information related to the respective operation. For example, boundary curves interpolated on the surface can be treated specifically for tensor field generation; these input curves should be valid segments on the target surface domain. Two BDTensor subcomponents, weighted vectors and surface curvature, are first computed from the sampling locations on the target surface and can be potentially adjusted by the parametric control rendered to users.

After the initial tensor field has been constructed from the featured boundary conditions, the BDCurve interpolation initiates with the given initial seed, a valid node within the surface domain, or randomly picked by the system, to form the conjugate curve network.

Once the curve construction is complete, the curve-to-curve intersection operator is triggered to topologically sort the built curve network. An intermediate object, *ItPoint3d*, is here utilized as an information holder to store the local connectivity information among curve-to-curve intersections, which is subsequently utilized for BDMesh vertex construction.

Following this step, the topology solver is activated to build the mesh topology, which will reassemble the target surface using the discrete polygonal elements—in this case, quad-dominant faces. Intermediate products including the BDTopology, BDTopoVertex, BDTopoEdge and BDTopoFace are all computed and managed by the BDTopology solver. These products respectively pertain to the essential information that relates to the overall topological connectivitiy as well as to the vertex-to-vertex, vertex-to-edge, and vertex-to-face relationships. The resulting topological entities represent the final mesh elements. In order to handle any potential polygonal shapes during the optimization process, a *Polygon* object is also provided as the geometry data entity to manage the geometry information that can host a polygonal shape with more than 4 vertices and can be used later by the mesh optimization process, for example, quadrangulation, mesh smoothing, etc.

5.2 The target surface: West façade of Zaha Hadid's Next Gene Museum

For demonstration purposes, a proposed design project by Zaha Hadid is remodeled, namely, the Next Gene Museum in Taipei, Taiwan. Based on presented design documents provided by Zaha Hadid Architect (2008), the formation of the conceptual building mass was derived from a series of mass-cutting operations (as shown in the top-left image in Figure 5-4). The proposed target surface is the west façade of the building envelope (as shown in the bottom image in Figure 5-4).

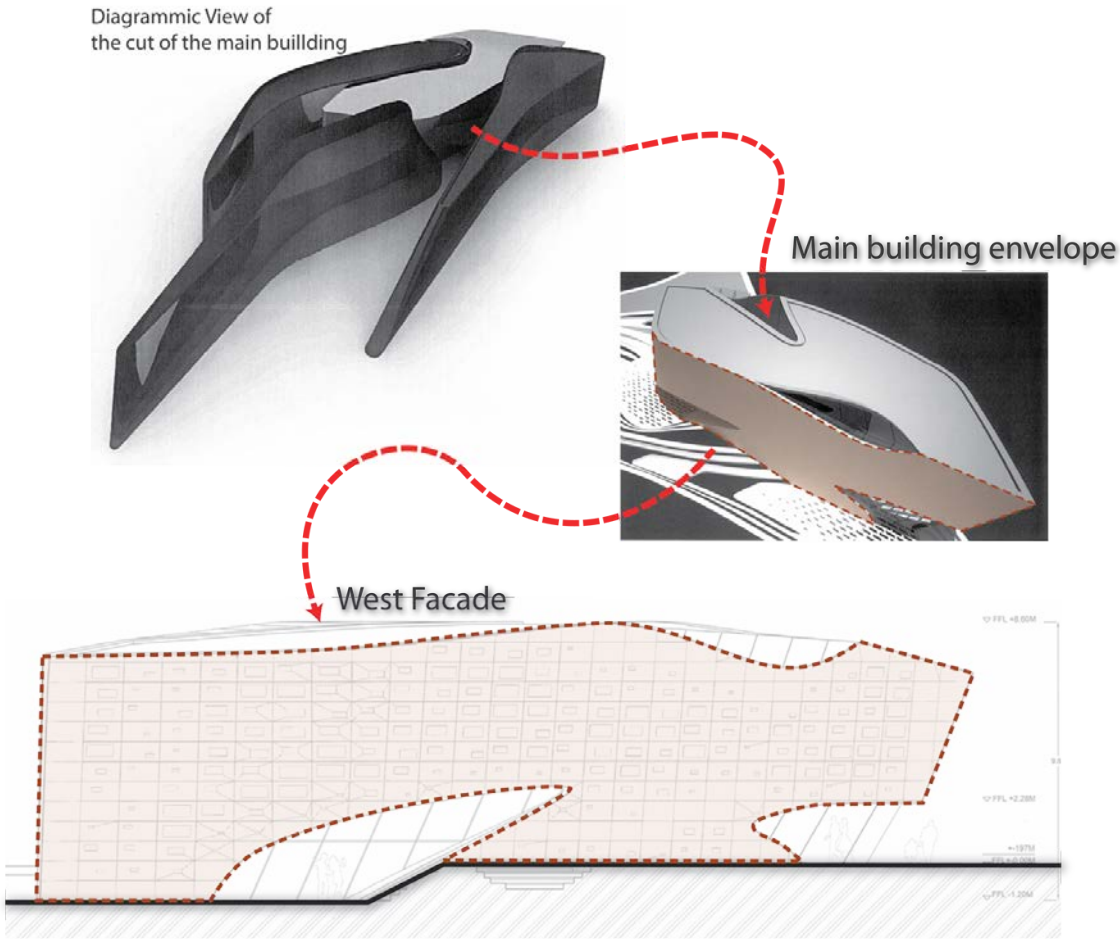


Figure 5-4 Conceptual massing of Zaha Hadid's Next Gene Museum in Taipei, Taiwan
Image is modified by author from Zaha Hadid (2008) with illustrated annotations

While manifesting the building envelope with considerations for physical construction, the proposed design was approximated by a collection of partial cone strips. As shown in Figure 5-5, the geometry for the west facade was constructed from the same cone surface with the respective trimming operations. For experimental purposes, the same procedure is mimicked to develop the experimental object, namely, a NURBS surface with trimmed boundaries.

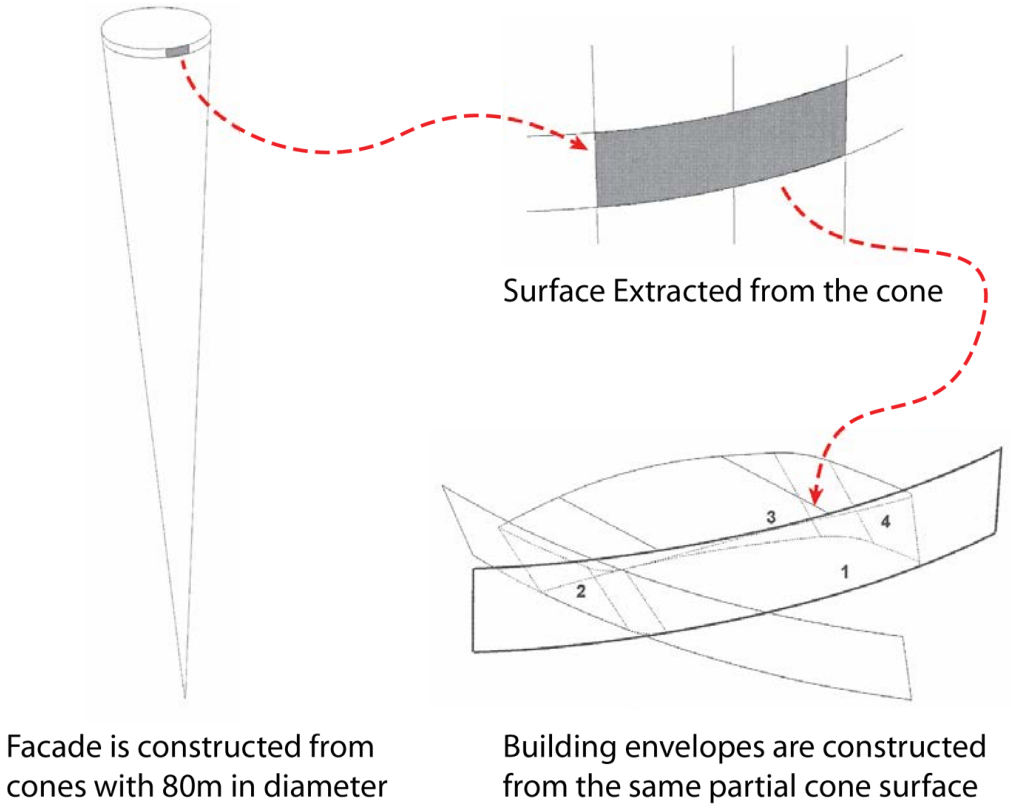


Figure 5-5 Cone surface reconstruction
Image adapted and modified by author
from an image of the Next-Gene Museum by Zaha Hadid (2008)

By increasingly adding trimming operations onto the target surface—a partial cone strip, the objectives are to (1) investigate the continuously changing influences as the increasingly complex boundary condition grows and (2) to examine the optimized mesh results along the changes from the given boundary conditions. The steps and results of tessellating the surface with boundary-driven optimization are presented next.

To start remodeling the west façade of the Next Gene Museum, a series of trimming operations was taken on an initially untrimmed cone surface. The proposed trimming curves are drawn as red-dashed line patterns shown in the top row image of Figure 5-6 and numerically labeled by their respective operation order in the process. In total, there are six trimming curves. The newly formed boundary conditions contain both new edge segments from the trimming curves and partial edge

segments from the original untrimmed boundaries (as shown in the bottom row of Figure 5-6). The original boundary segments are illustrated as solid grey lines.

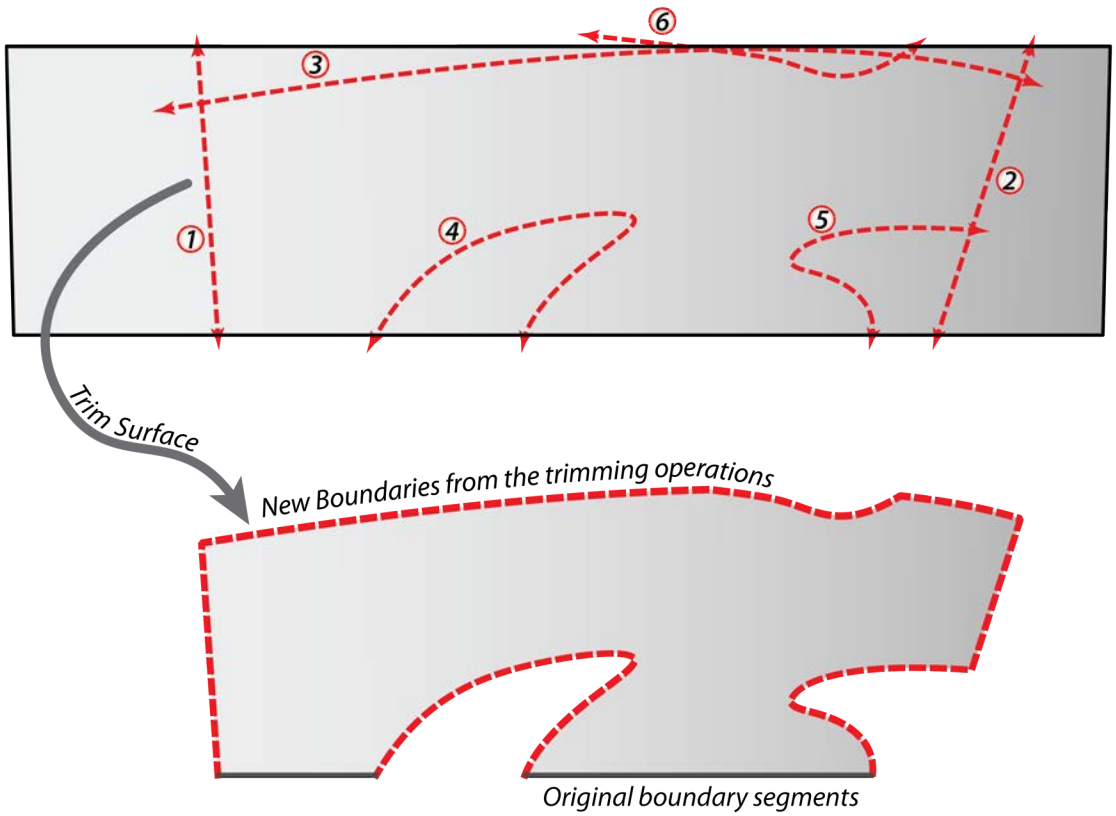


Figure 5-6 Trimming operations
for remodeling the west façade of the Next Gene Museum by Zaha Hadid (2008)
(*Top*) Order of the trimming operations (*Bottom*) Resultant trimmed surface

5.3 From single to multiple boundary consideration

Given a trimmed surface as shown in Section 5.2 (Figure 5-6), the first step of the process identifies the boundaries of the input surface, shown as solid red line segments in Figure 5-7. There are a total of 8 boundary edges, BE_0 to BE_8, and 8 corner vertices, VT_0 to VT_8. Corner vertices are important identifiers while smoothing the mesh results as they are treated as fixed vertices, which remain at their current location to keep the shape intact. The remaining vertices will move according to vertex-to-surface conditions, including the edge-vertex condition or interior vertex condition, which in turn

determine whether the vertices are attached to the closest boundary edge, or move to the closest surface location.

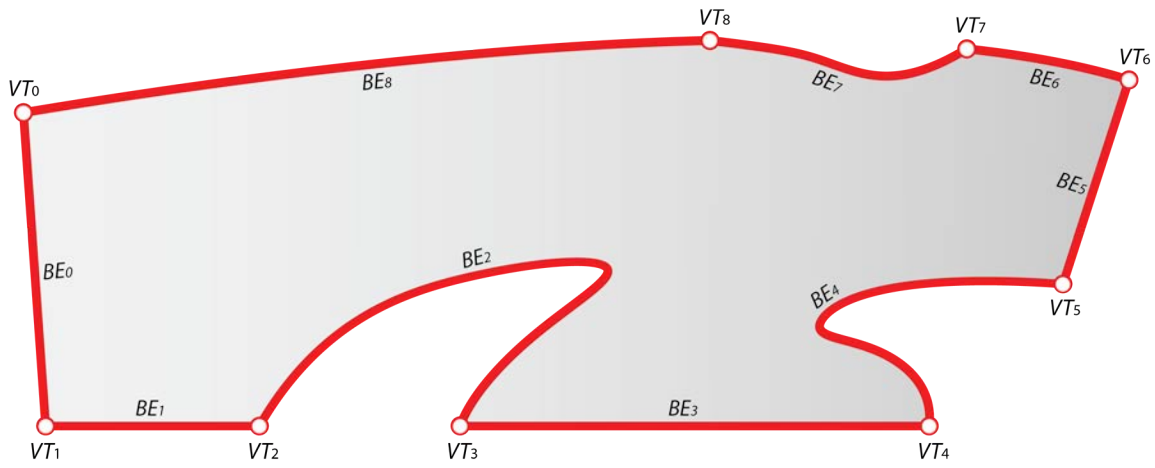


Figure 5-7 Surface boundaries identification and corner vertices extraction

After the boundary conditions have been identified, the initial tensor field and corresponding BDCurve network is constructed. As shown in Figure 5-8, the initial conjugate BDCurve network is constructed by the given curve offset constraint. The two directional curves are drawn in red and blue solid line pattern respectively.

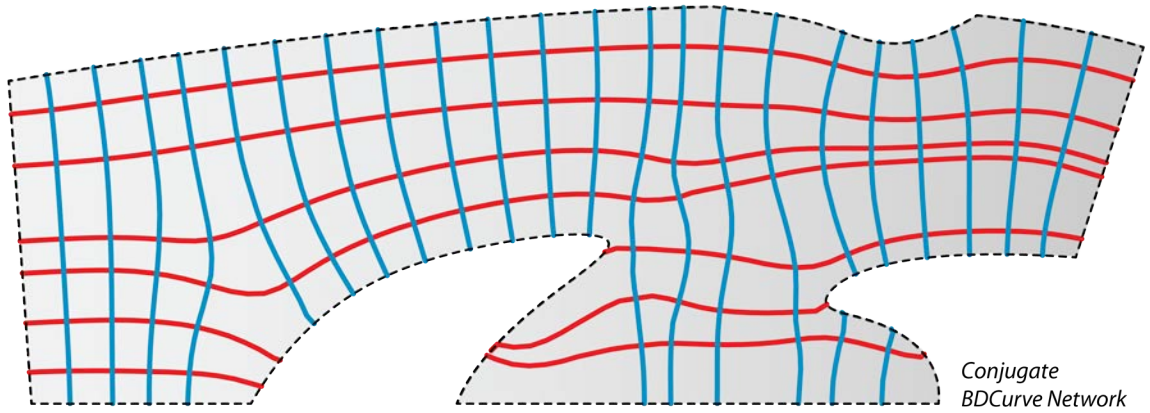


Figure 5-8 Initial BDCurve network

After curve-to-curve intersection, the underlying topological network is sorted and then the preliminary BDMesh object is constructed, as illustrated in the top row image in Figure 5-9. With this initial mesh result, the further optimization process is performed to relax the current mesh configuration, as shown in the bottom row image in Figure 5-9. The mesh edges after relaxation become more balanced and demonstrate optimized dimensional control over the constructed mesh network. Notice that the preliminary mesh result may potentially consist of other polygonal shapes besides quadrilaterals. In this example, both triangles and pentagons are generated during this process. A further quadrangulation can then be performed to guarantee a quad-dominant meshing result.

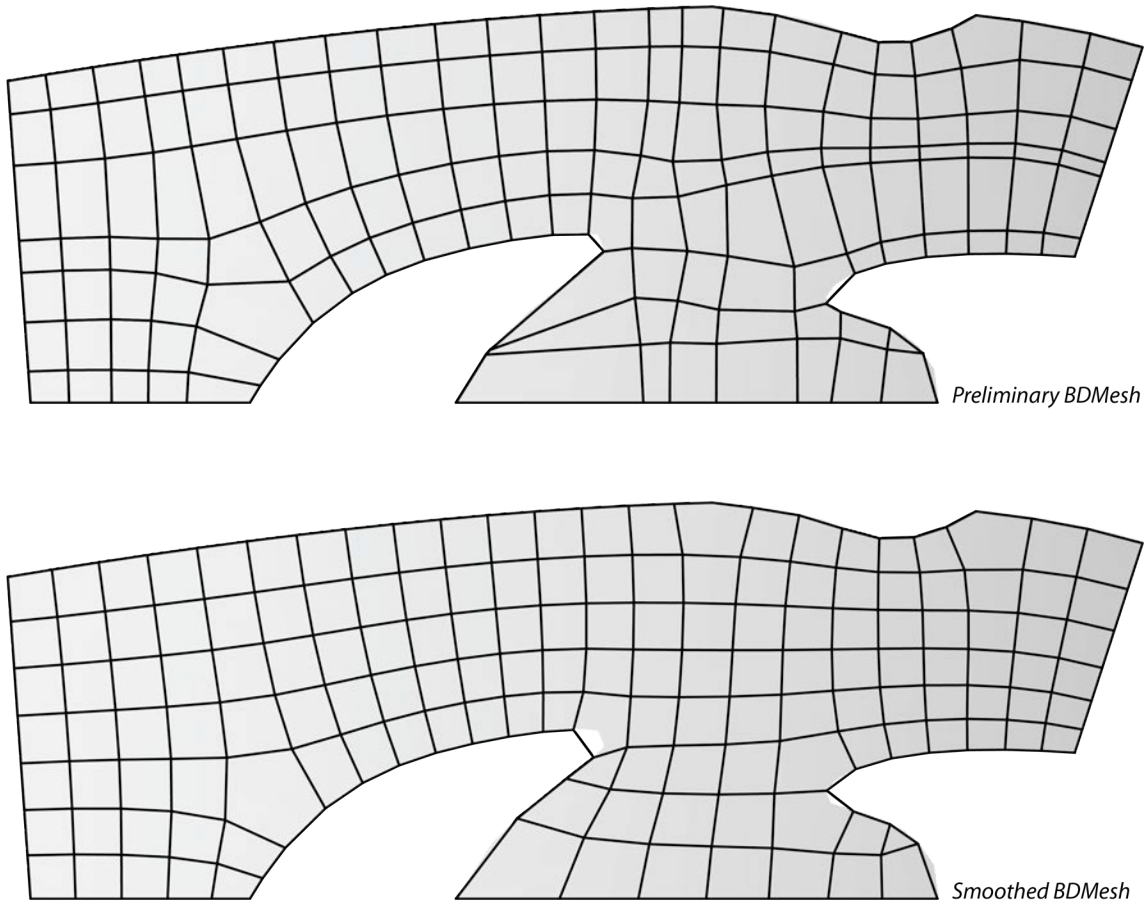


Figure 5-9 Boundary-driven optimization
(*Top*) Preliminary BDMesh (*Bottom*) Smoothed BDMesh result

Given a tessellation scheme that caters to the inherent boundary conditions, a series of meshing results with increasingly complex boundary conditions are examined. These are illustrated in Figure

5-10. A total of six steps are displayed in a counter-clockwise order from the top-left to the top-right corner. For each step, a new boundary condition is introduced by an additional cut, which is illustrated as a solid grey line with arrows on both ends. The operational sequence follows the same trimming order shown in Figure 5-6. In Figure 5-10, the existing boundary edges are drawn in the red dashed pattern.

For instance, starting from the fourth step of the illustrated workflow (shown at the right-bottom image in Figure 5-10), irregular regions start to emerge where multiple boundary sources meet and thus polygonal face elements, such as pentagons, are generated at these regions. The polygonal shapes are treated as the intermediate meshing elements and are later refined as quad-dominant elements. The top-right image of the Figure 5-10 is the preliminary result of meshing target surface with all featured boundaries. The results shown in Figure 5-10 are all optimized with mesh smoothing.

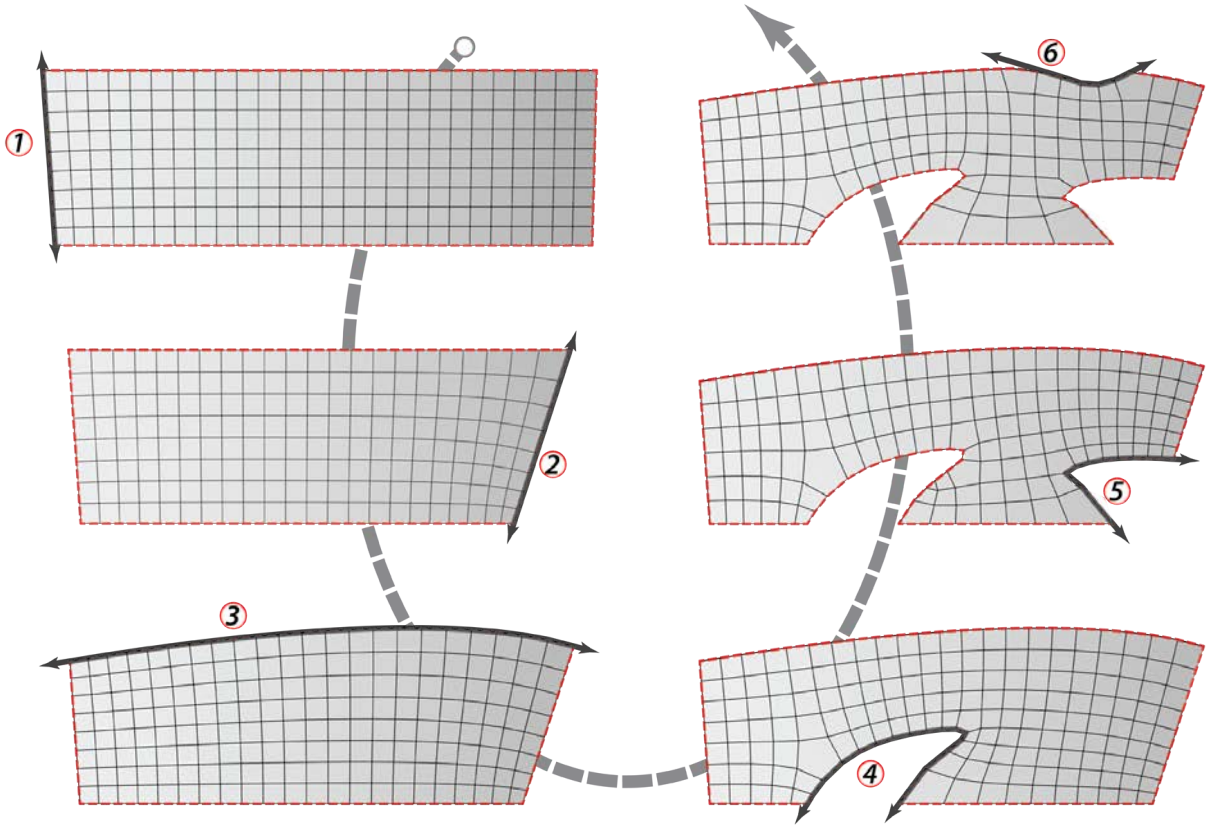


Figure 5-10 BDMesh results from increasing complex boundary conditions ordered from the top-left to the top-right corner

In brief, only triangular and quadrilateral face elements are valid elements for the conventional mesh object. As discussed earlier, an additional *Polygon* object is introduced as an information holder, employed to accommodate any arbitrary polygonal faces that may occur during the tessellation process. In a sense, polygonal objects, which potentially consist of more than four bounding vertices, are regarded as preliminary geometry entities, which pertain to the important topological relationship, and may serve for further optimizations and potential applications.

Figure 5-11 illustrates a preliminary quad-dominant mesh result, optimized with the featured boundary conditions. The meshing result demonstrates a balanced local relationship among generated mesh elements with irregularities minimized at regions where multi-directional edges meet. This renders a well-configured network for further pattern-based generation and potential freeform application.

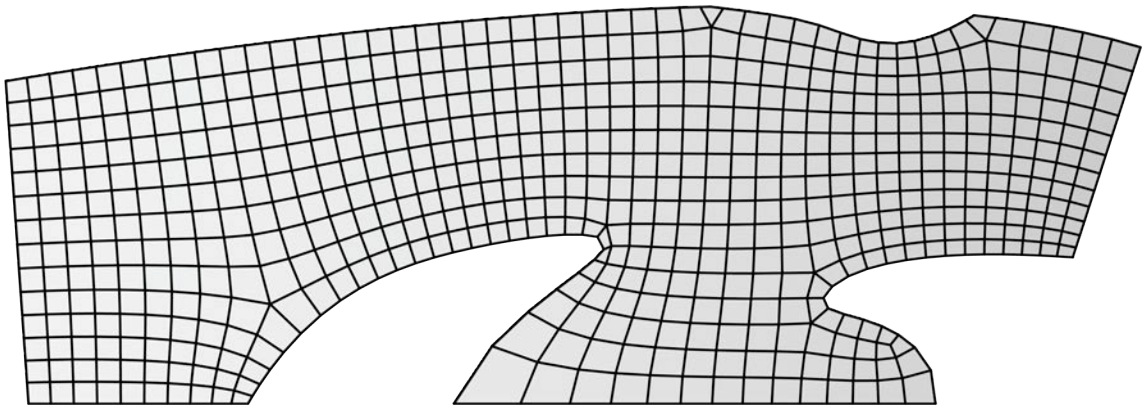


Figure 5-11 Quad-dominate meshing result

5.4 Pattern generation by using topological operators

In this section, Archimedean patterns are utilized to explore pattern-based generation by reconfiguring mesh elements derived from the proposed boundary-driven optimization. The consideration of whole boundary optimization is to investigate the coherent configuration that will fit the tessellated surface with designated patterns without creating arbitrary incomplete/trimmed elements at the boundaries.

5.4.1 Truncation operator

Given the quad-dominant mesh shown in Figure 5-11, the truncation operator presented in Chapter 3 is considered first. Recall that the truncation operator generates vertex replacement at each vertex, v, by defining new truncating vertices specified by the parameter t and is applied to on all connected edges. Then, for each face element in the constructed network, a new face is accordingly generated. As shown in Figure 5-12 through Figure 5-16, a series of truncated patterns are generated with the parameter values for t, ranging from $^1/_3$ to 1.

In Figure 5-12, a truncated square pattern (4.8^2) is constructed with $t = {}^1/_3$. In the constructed pattern, each original edge is first divided into three equi-dimensional segments. Then, each mesh vertices is replaced with a new quadrilateral face and each original quad face is replaced with an eight-sided polygonal shape—namely an octagon—connecting the newly truncated vertices on the original bounding edges.

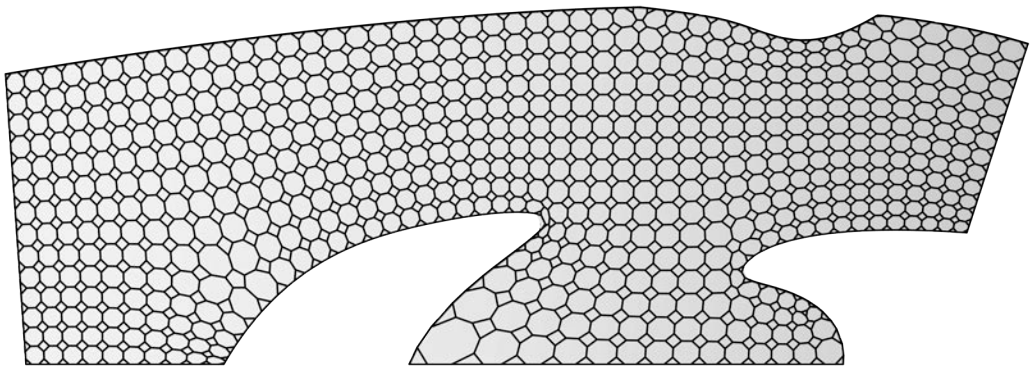


Figure 5-12 Archimedean Pattern (4.8^2) generated by the truncation operator with $t = {}^1/_3$

By increasing the value of the parameter t from $^1/_3$ to $^1/_2$, each vertex replacement face touches neighboring faces at the midpoint of the incident edge. The resulting face replacement element is a quadrilateral face. As shown in Figure 5-13, the resulting pattern has the same appearance as the commonly seen diagrid pattern.

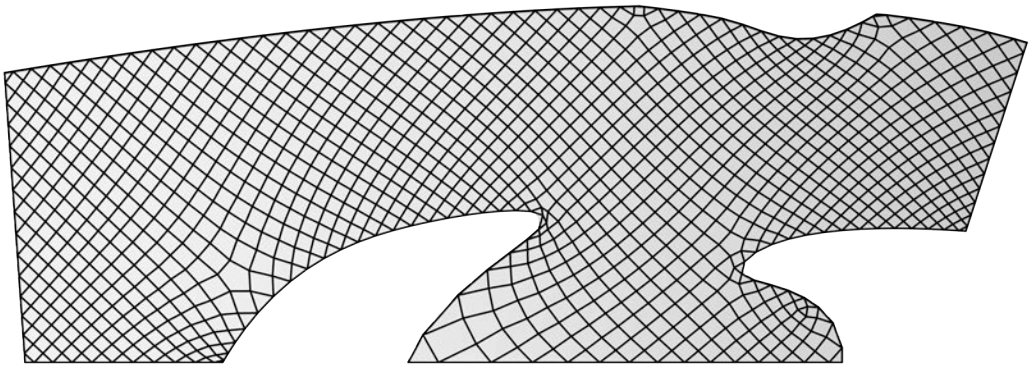


Figure 5-13 Diagrid pattern generated by the truncation operator with $t = {}^{1}/_{2}$

To achieve a more balanced mesh object, the smoothing operator can also be applied here to derive a better-smoothed mesh network. Figure 5-14 shows the smoothed mesh result of the diagrid pattern. Due to the irregular regions that are inherent from the underlying boundary conditions, the truncated pattern generates irregular polygons (such as pentagons in this case) at singular vertices, which, in this case, are those connected to n vertices (where $n \neq 4$).

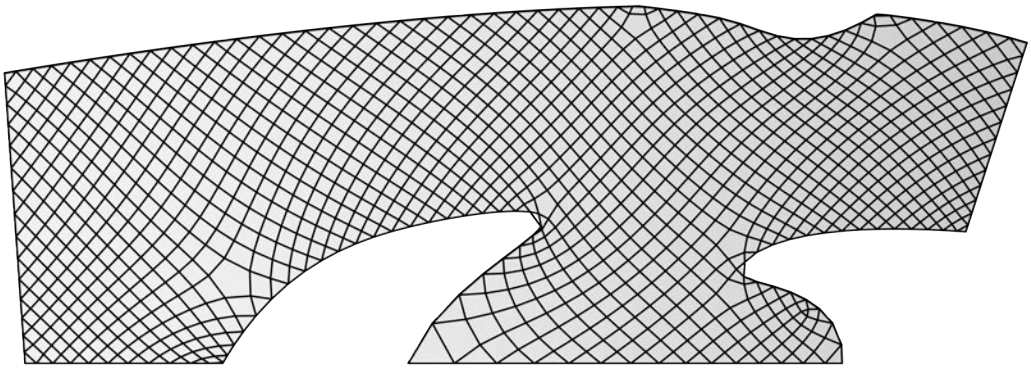


Figure 5-14 Smoothed diagrid pattern generated by the truncation operator with $t = {}^{1}/_{2}$

When the value of the t parameter is increased up to ${}^{2}/_{3}$, a similar truncated square pattern (4.8^2) is created. However, in this case, the order of the quad and octagonal prototile construction is reversed. See Figure 5-15. In other words, a new octagon face will be created at each vertex and a new quad face at each face. The difference can be identified from the element created on the boundary edges, where the new configuration has half octagon shapes instead of half quadrilateral faces (triangular faces) in the original truncated square pattern.

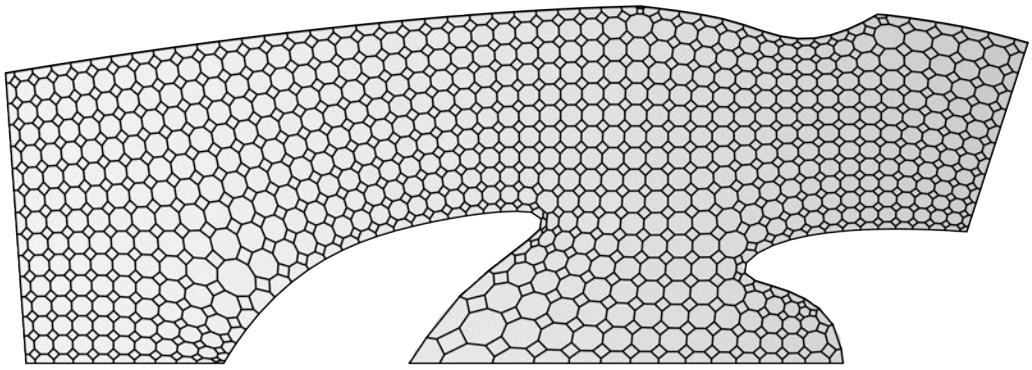


Figure 5-15 Archimedean pattern (4.8^2) generated by the truncation operator with $t = {}^2/_3$

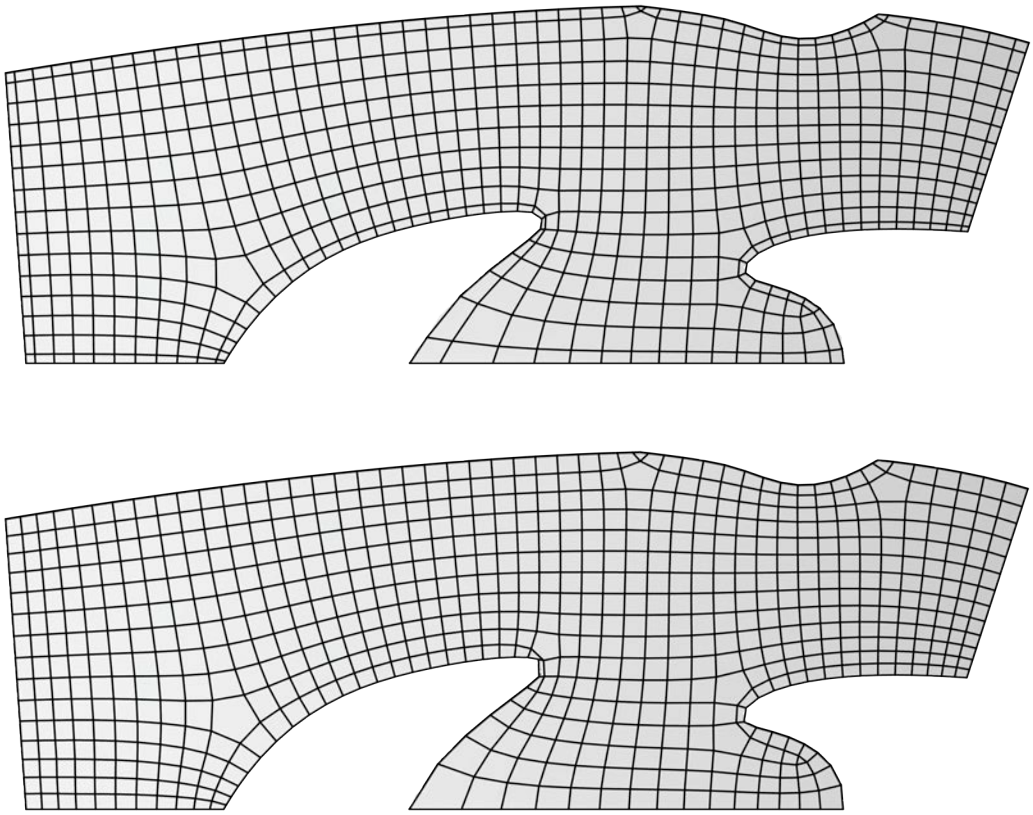


Figure 5-16 The dual pattern of the Archimedean pattern (4^4) generated by truncation with $t = 1.0$
(*Top*) Without smoothing (*Bottom*) Smoothed mesh pattern

Lastly, when the truncation operator with parameter $t = 1$ is applied, the dual of the original mesh is generated, as shown in the top row image in Figure 5-16. In this case, for each vertex, the replacement element is constructed by the neighboring mesh face centroids. Likewise, a mesh

smoothing operation can be applied to adjust the dimension of the smaller quad elements around the boundary edges, or corners, as shown in the bottom row image in Figure 5-16. At first glance, the dual mesh is a similar quad-dominant mesh. However, exceptions occur at the irregular regions, where polygonal shapes other than quadrilaterals are generated.

5.4.2 Insertion operator

A second operator is explored with the same initial quad-dominant mesh object. In light of the underlying topological connectivity, certain mesh results will look similar to those derived by the truncation operation, but with different orders in the prototile configuration. The parameter t with its values ranging from 0 to 1 is again considered in a similar manner this time to explore pattern variations using the insertion operator.

Figure 5-17 demonstrates the pattern generated with parameter $t = {}^1/_5$. At first glance, the configuration of the generated pattern looks similar to the truncated square pattern (4.8^2). However, there is a subtle difference between the two. In this pattern generated by the insertion operator, the quad face element is aligned with the original conjugate curve direction, and in the other it is not. All octagon elements are created at both mesh vertices and mesh faces. In the truncated square pattern (4.8^2), the octagon shape is only created at each mesh face or each mesh vertex when t is set to be larger than ½.

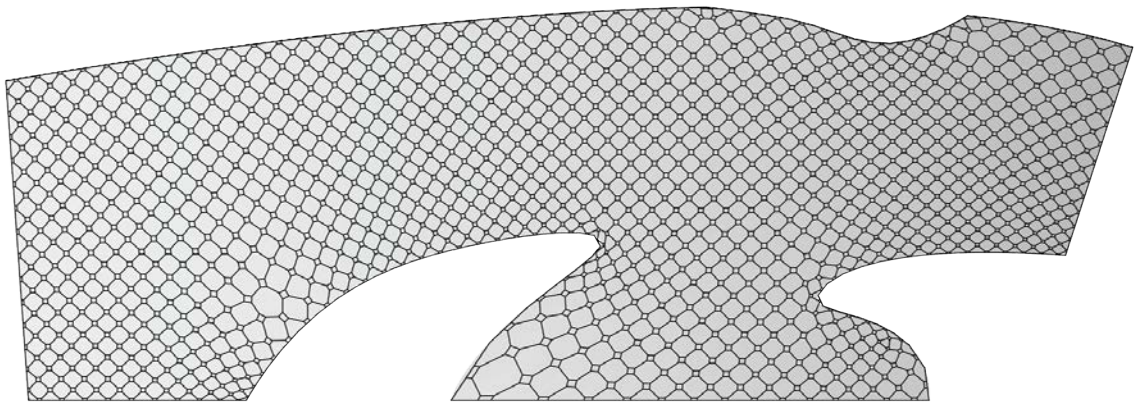


Figure 5-17 Archimedean pattern generated by the insertion operator with $t = {}^1/_5$

As the value of parameter t is increased from ${}^1/_5$ to ${}^1/_2$, the new face elements on the edges touch each other at corners, and subsequently create smaller face replacements with the same topological

configuration. See Figure 5-18. This is similar to a quad mesh subdivision in which each mesh face is replaced by 9 sub-faces. The new faces created at mesh edges are shared with the neighboring faces.

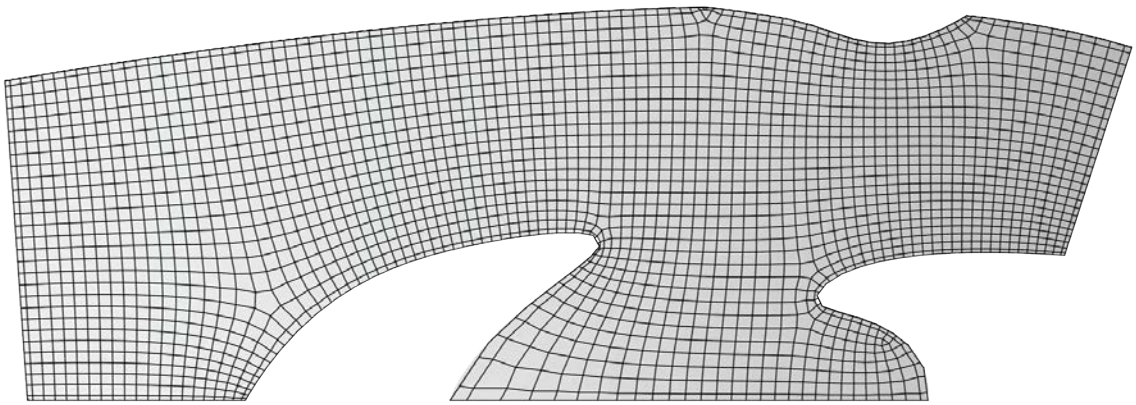


Figure 5-18 Archimedean pattern generated by the insertion operator with $t = {}^1/_2$

When the value of the t parameter is increased from ${}^1/_2$ to ${}^4/_5$ a similar pattern as the one derived when the value t equals ${}^1/_5$. In this case, instead of creating quad face elements at edges and octagon elements on the vertices and faces, the order of quad and octagon replacements are interchanged—namely, octagon elements are now created on edges, and quad faces on vertices and faces. See Figure 5-19. The difference can be identified by the elements created at corner vertices.

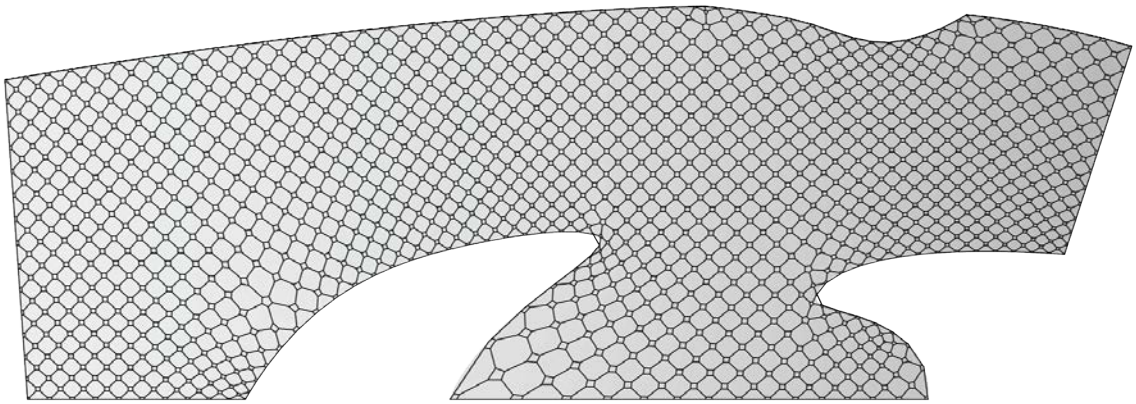


Figure 5-19 Archimedean pattern generated by the insertion operator with $t = {}^4/_5$

Lastly, as the value of the parameter t is increased to 1, a diagrid pattern is created. In this case, each edge element will be replaced with a quadrilateral element (or a diamond shape element), which is constructed by connecting two neighboring face centroids and bounding vertices of the current edge.

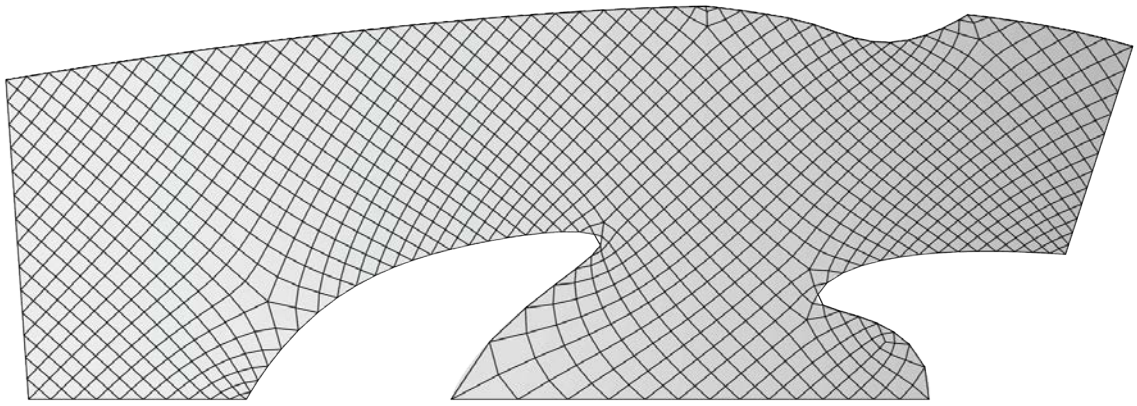


Figure 5-20 Archimedean pattern generated by the insertion operator with $t = 1$

5.4.3 Alternation operator

The alternation operator generates a new Archimedean pattern by reducing the number of original vertices into half. Due to the nature of this operation, the number of the vertices of the target polygon shapes is at least six, or more, for a successful conversion. For example, a hexagon is replaced by a triangle; an octagon by a quadrilateral, and so forth. Any polygonal shape that has less than 3 vertices (a triangle) is not valid. As discussed in Chapter 3, the snub square pattern ($3^2.4.3.4$) and the snub hexagonal ($3^4.6$) pattern can be respectively derived from a truncated square pattern (4.8^2) and a truncated trihexagonal (4.6.12). In this section, the alternation operator is applied to the truncated pattern with the parameter $t = {}^1/_3$ (same as the result shown in Figure 5-12). The alternation pattern is illustrated in Figure 5-21. Since the surface has limits defined by its boundary, it is possible to show alternate parametric variations of the pattern. Figure 5-22 illustrates a variant of Figure 5-21, which is subject an additional localized rotation about each vertex in the prototile generation. Similar variations can be obtained from other parametric patterns.

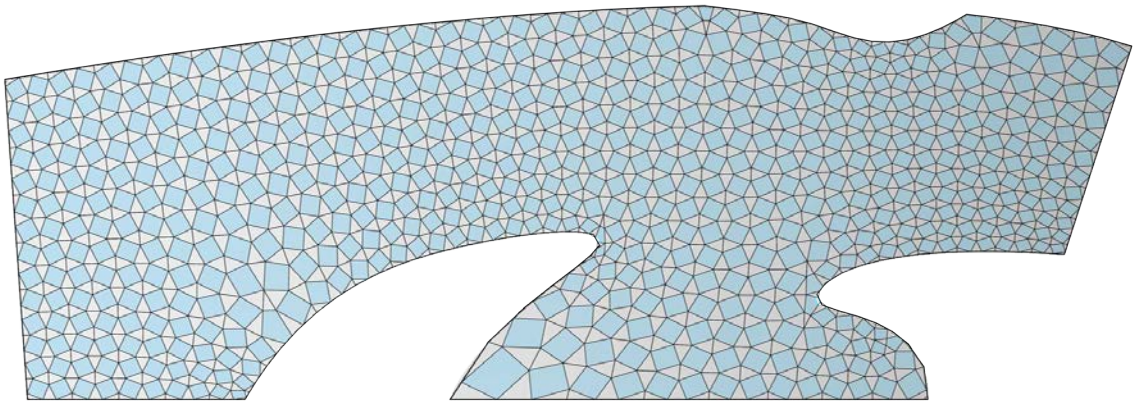


Figure 5-21 Archimedean pattern generated by the alternation operator on the truncated pattern with $t = {}^1/_3$

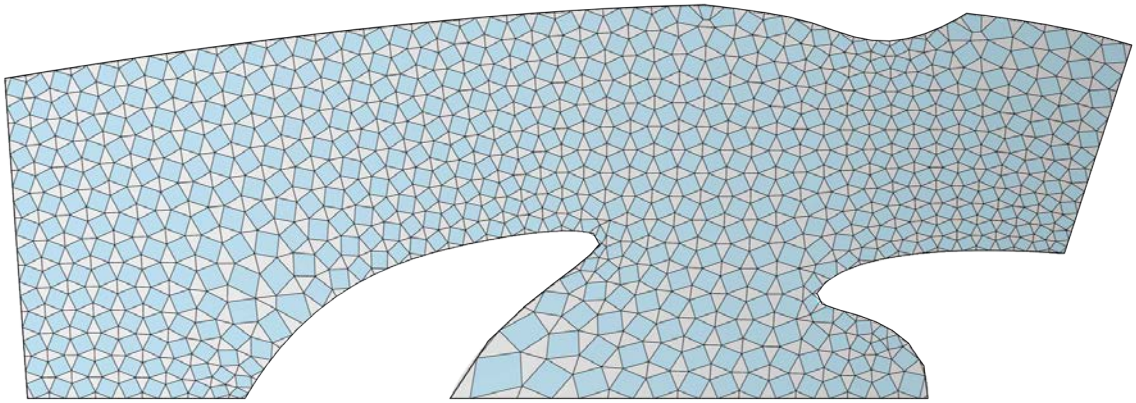


Figure 5-22 A variant of Archimedean pattern shown in Figure 5-21

5.4.4 An application from the Archimedean pattern

Given the tessellated pattern derived from the Archimedean operators, surface components such as structure frames, panels, etc., can be procedurally constructed. In this section, an application of procedural construction of surface components is demonstrated.

Given the discrete model of the surface, the first step is to examine the relations among the underlying topology for the designated constructive operations. In this section, the pattern derived from the insertion operator (Figure 5-20) is employed as the example to demonstrate the procedures involved for surface component construction. The process starts from the preliminary BDMesh result, shown in the left image in Figure 5-23. By applying the insertion operator with $t = 1.0$, the diagrid

pattern is first created (the middle image of Figure 5-23). The resulting pattern is identical to the Archimedean pattern demonstrated in Figure 5-20.

The following step is to tag all vertices by their origins, from which they are derived. Recall that, as *t* is set 1.0, the insertion operator will replace an interior edge element with a new polygon face formed by connecting two edge end points with two neighboring face centroids. By cross-examining the origins of each newly created vertex, new vertices are separated into three groups: (1) alternate edge end point 01, (2) alternate edge point 02, and (3) face centroid. For each pair of edge end points, they are alternately divided into two groups so that all neighboring vertices are separated into two distinct groups. This step is particularly designed for subsequent vertex modulation and can be potentially varied or provided by the end user with other factors taken into considerations. Figure 5-24 illustrates the sorted vertices in three different groups.

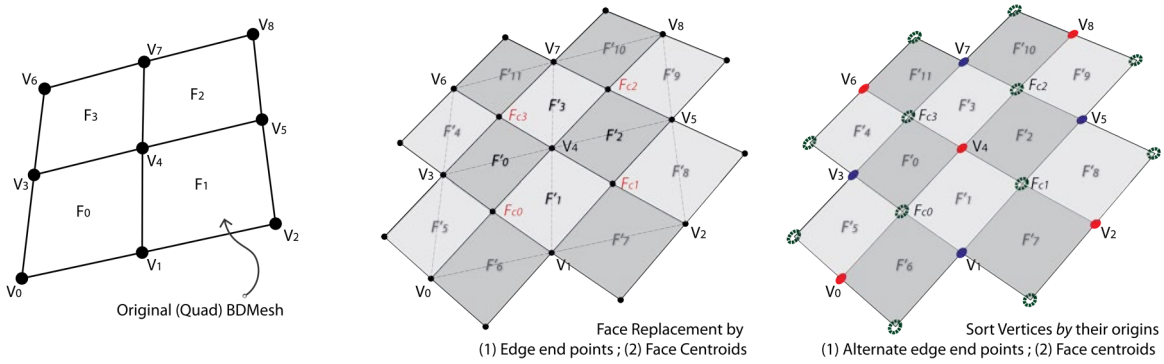


Figure 5-23 Procedure of sorting mesh vertices by their origins
(*Left*) Quad Mesh (*Middle*) Diagrid pattern by Insertion operator with $t = 1.0$
(*Right*) Sorted mesh vertices

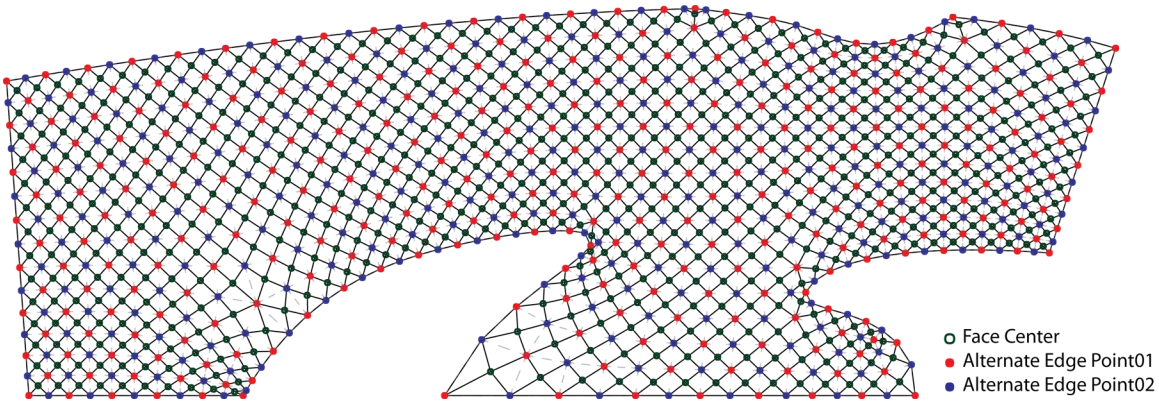


Figure 5-24 Sorted mesh vertices overlaid with the underlying mesh topology

With the base diagrid pattern and sorted vertices, vertex modulation is considered as a means of adjusting the location of each mesh vertex along its normal direction. The left image of Figure 5-25 illustrates vertex modulation along the normal direction. In brief, for each vertex in the group of alternate edge end point 01 (colored shaded red with arrow), they are moved along the normal direction; vertices in the group of alternate edge end point 02 (colored shaded blue with arrow) are moved in the reversed normal directions. For the remaining vertices in the face centroid group, they remained at their current locations. With the based pattern updated by respective modulation, frame components are constructed by investigating underlying mesh topology and the result is shown in the right image of Figure 5-25.

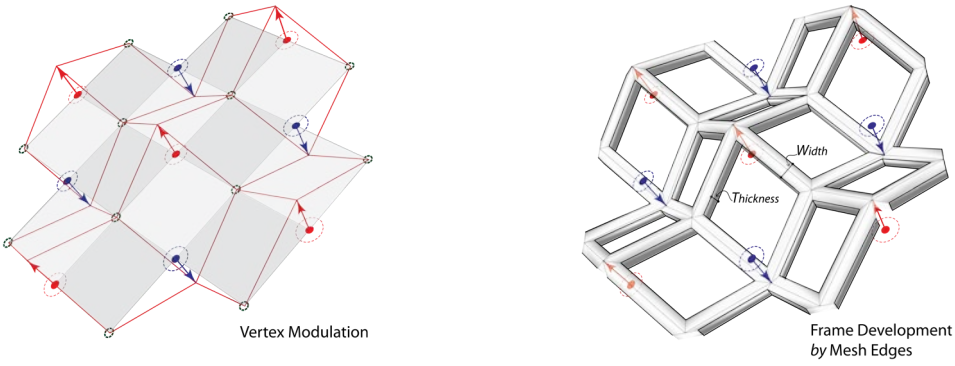


Figure 5-25 (*Left*) Mesh vertex modulation (*Right*) Structure frame construction

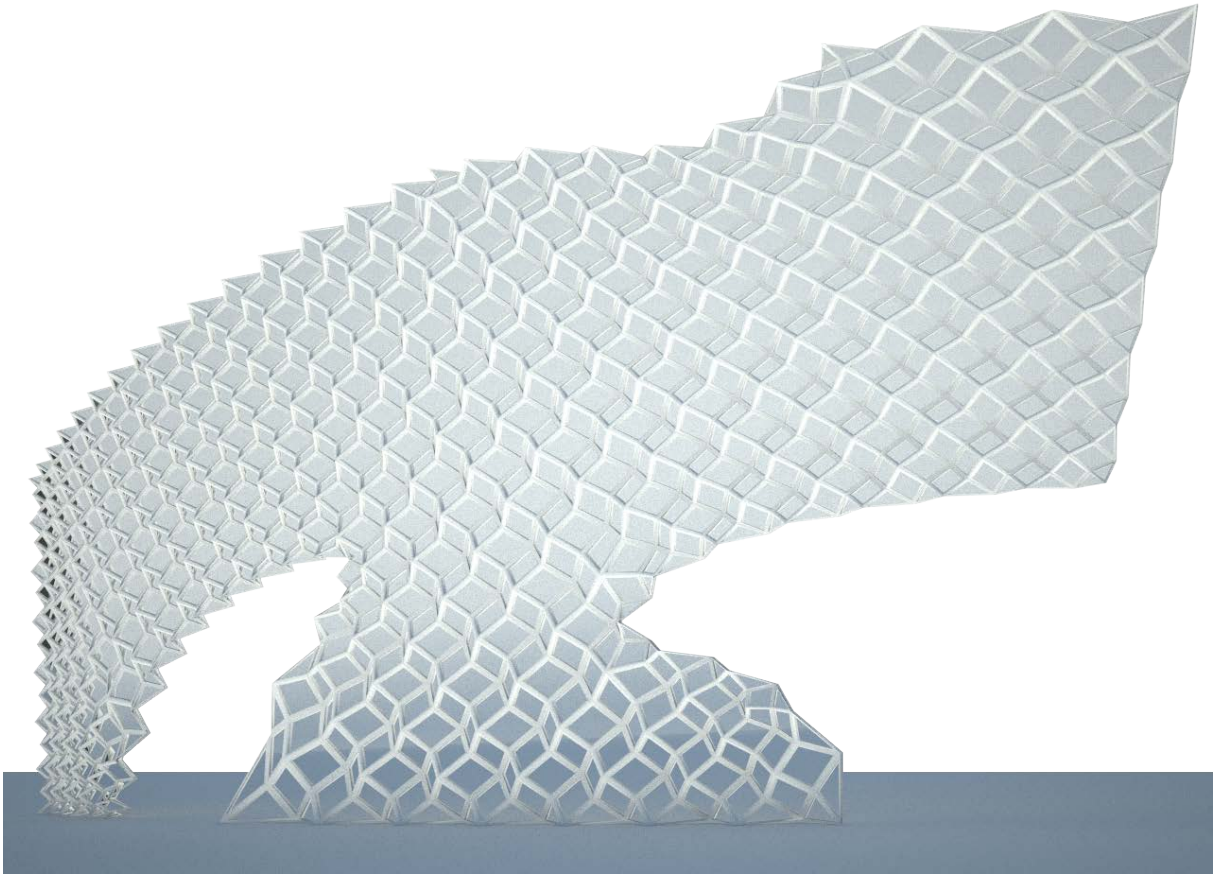


Figure 5-26 Surface rendering with structure frames and panels by the underlying Archimedean pattern

Figure 5-26 illustrates the final rendering of the surface design with structure frames and glass panels by vertex modulation shown from Figure 5-23 to Figure 5-25. In brief, the pattern-based tessellation serves as the essential groundwork for the further freeform surface applications. Even with the trimmed surface with irregular boundary conditions, the approach demonstrates the potential for the freeform surface exploration with the optimized mesh structure.

In summary, owing to the complex nature of the irregular boundary conditions, it is almost impossible to optimize any arbitrary surface by just regular regions. Notwithstanding, the algorithmic approach presented in this dissertation optimizes the meshing result by minimizing the number of irregular regions. The presented discrete model not only conforms to the boundaries it inherits, but at the same time, minimizes the irregular regions that may occur during the optimization process. Given a well-structured quad-dominant mesh, alternative pattern configurations can be procedurally explored by restructuring the underlying topological network.

5.5 Interwoven pattern generation

In this section, an interwoven pattern application is given. As discussed in Chapter 3, an interwoven pattern consists of two continuous counterparts, which are respectively formed by a sequence of faces. The ending geometric component of a face is smoothly transformed into the connecting geometric component of the next face in the sequence. The notion of "*connecting ending components of the current face to the connecting components of the next face*" suggests the order of construction as the fist constructive principle.

To construct an interwoven pattern, the following generative procedure combining both sorting and construction is considered. Given a quad-dominant mesh, a simplified module (a coarse mesh) is first created from a quadrilateral boundary (as shown in Figure 5-27). To engender a continuous appearance, the ending components of the current face are connected to the connecting components of the next face. For instance, in Figure 5-28, mesh faces, F_3 and F_7, of the module on the left are connected to mesh faces, F_8 and F_{12}, of the continuous module on the right at $[V_0, V_{15}]$ and $[V_{12}, V_{11}]$ respectively.

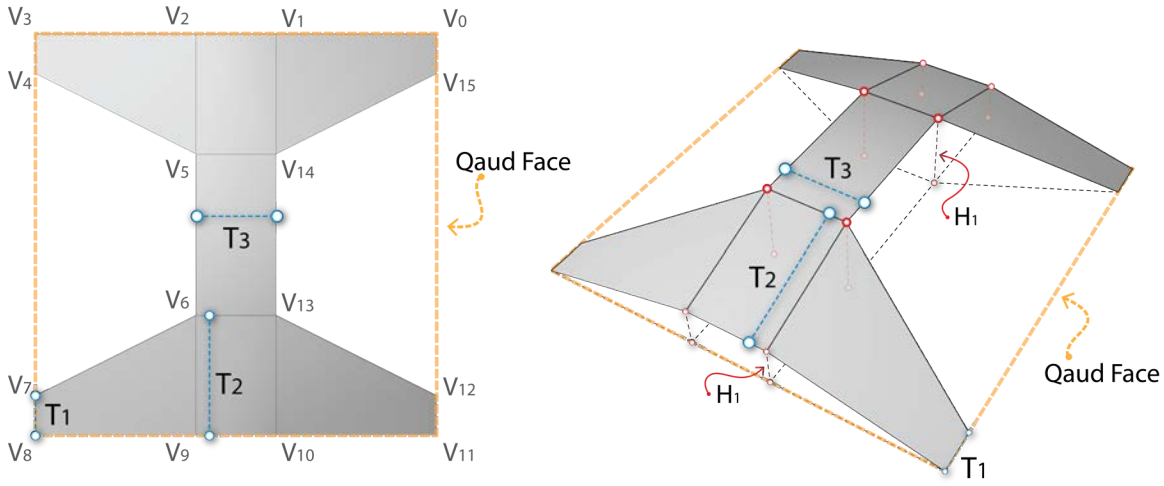


Figure 5-27 A mesh module for the interwoven pattern generation

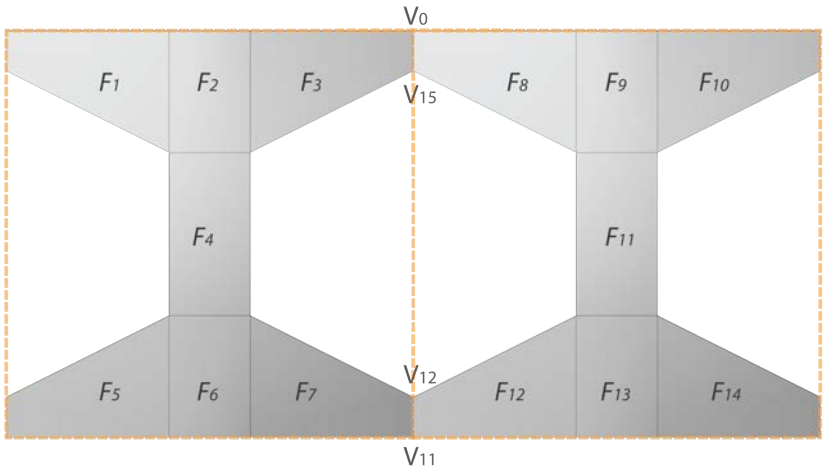


Figure 5-28 A pair of continuous modules for the interwoven pattern

With a module as shown in Figure 5-27, the next step is to investigate the underlying connectivity for procedurally constructing the continuous counterparts. In Figure 5-29, design modules are constructed by (1) the alternate vertex selection, and (2) conjugate curve directions. To start, vertices are separated into two alternate groups such that each vertex is separated from all its connected vertices. As shown in Figure 5-29, Vt_4 is the member of the *Vt-Group* 2 (colored shaded red) and connected to Vt_3, Vt_1, Vt_5, and Vt_7 in the *Vt-Group* 1 (colored shaded blue). With the sorted vertices, the construction initiates from the location, Vt_4, by following the given direction from the underlying conjugate network to build up the corresponding modules, for instance, a module bounded by Vt_4, Vt_3, Vt_0, and Vt_1 in a counter-clock-wise order. By continuously investigating the incident faces at the currently selected direction, a collection of continuous interwoven modules is constructed. In the generative process, the alternate vertex identification yields the constructive order of the vertices from the selected face, and the reference conjugate directions indicate the faces for two-directional counterparts construction.

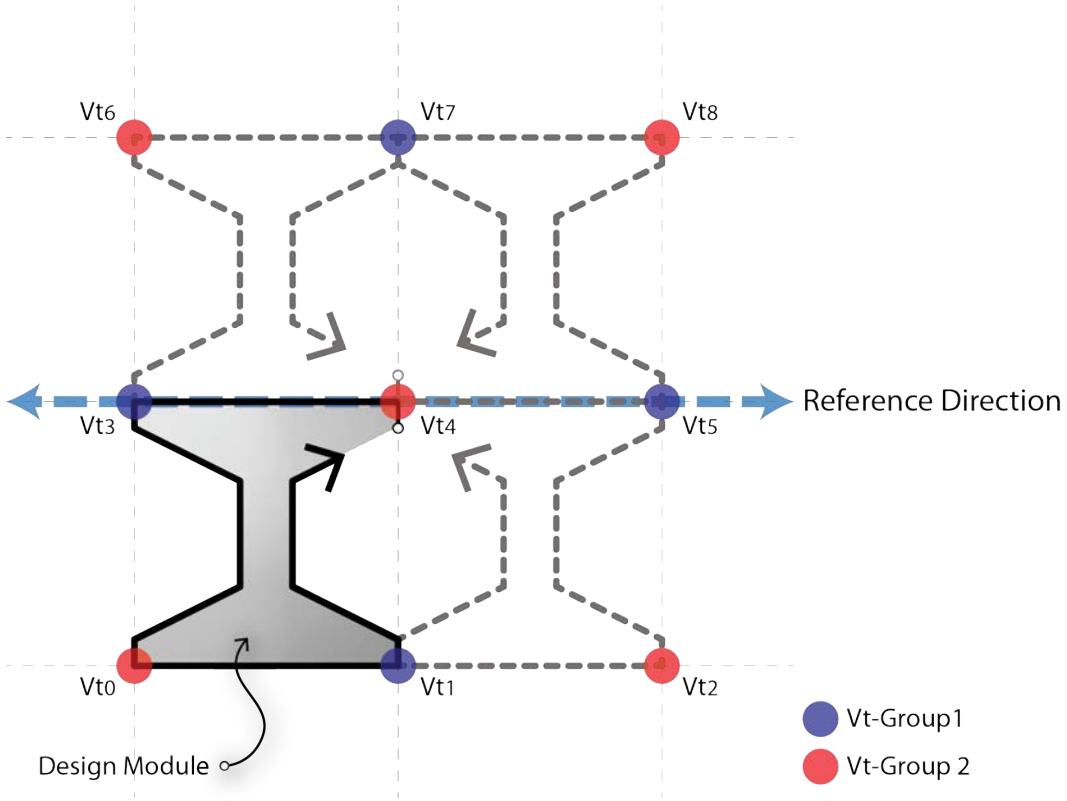


Figure 5-29 Module propagation by alternate vertex group and referenced conjugate direction

Figure 5-30 illustrates the recursive procedure utilized for sorting the alternate groups of vertices. Given a starting vertex, *Vt*, as shown in the top-left corner in Figure 5-30. The sorting process proceeds by iteratively traversing all the unseen neighboring vertices, *V*<*V*>, in the topological network. For a mesh with thirty-six vertices, a total of five iterations are required to separate all vertices into two alternate groups along two conjugate directions.

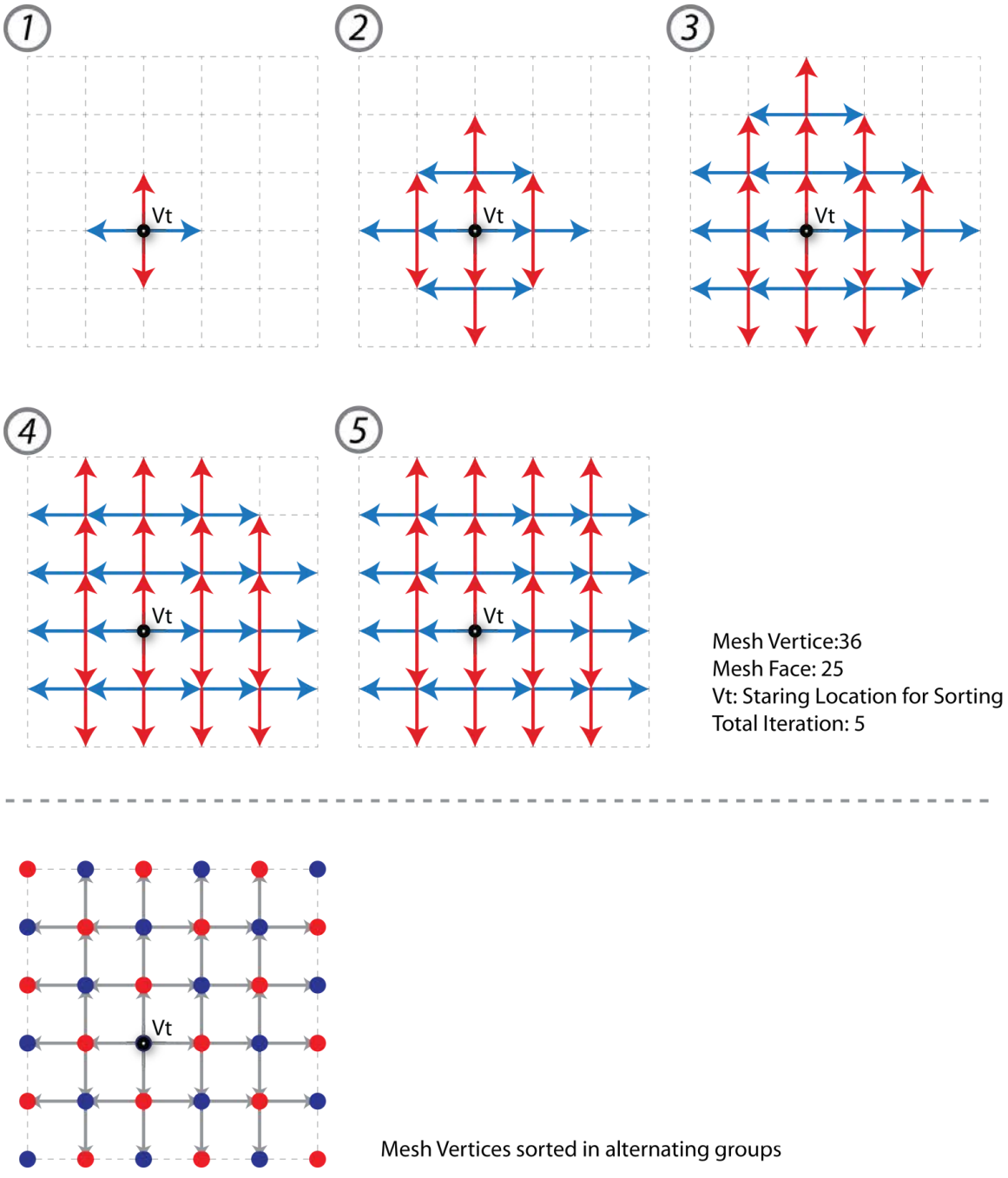


Figure 5-30 A mesh module for the interwoven pattern generation

Figure 5-31 illustrates the alternate vertex sorting by coloring vertices shaded red and blue overlaid with the initial quad-dominant mesh (same as the one as shown in Figure 5-11). Two directional edges in the conjugate network are illustrated in the bottom image of Figure 5-31. Notice that these conjugate edges are not necessary partial segments from the initial BDCurve network;

instead, they are new edge elements derived from the quadrangulation. As shown in the bottom image of Figure 5-31, the third group of edge components (colored shaded black) occurs at the irregular regions where a vertex is connected by more than four edge components. These areas require the special treatment when developing the interwoven module.

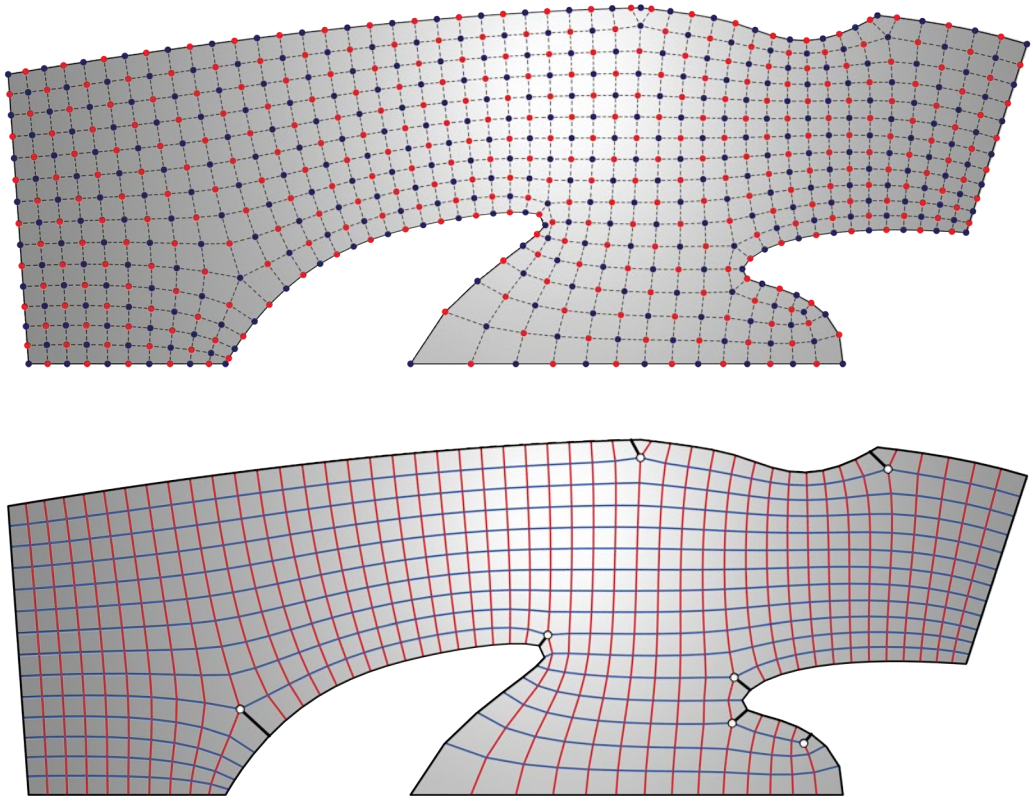


Figure 5-31 Mesh module refinement by Catmull-Clark subdivision

Figure 5-32 demonstrates the module refinement by applying the Catmull-Clark subdivision on the initial coarse mesh module. By iteratively subdividing the initial coarse mesh into smaller quad faces, the refined mesh engenders a smoother appearance for better visual transition.

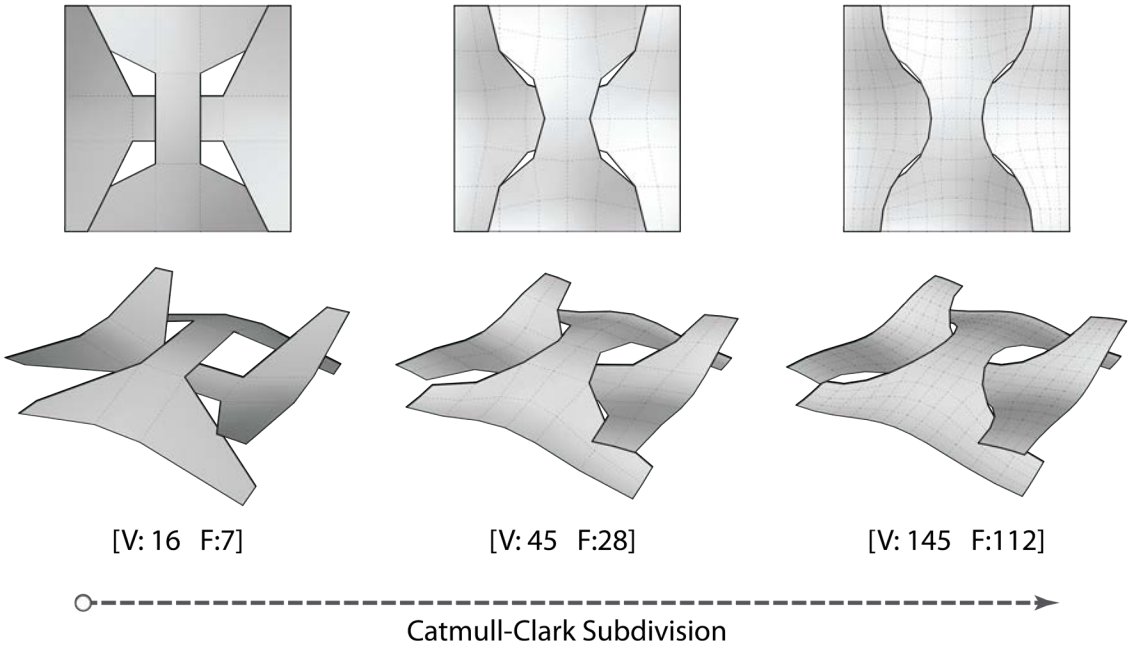


Figure 5-32 Mesh module refinement by Catmull-Clark subdivision

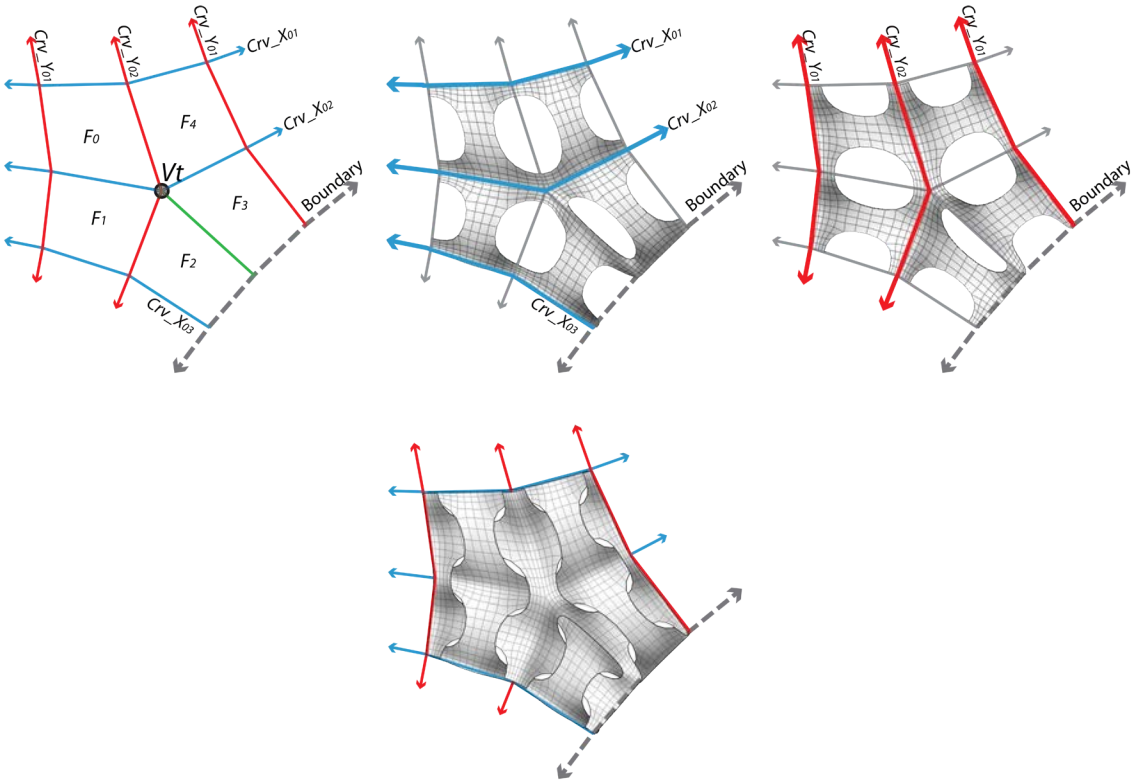


Figure 5-33 Module construction at the irregular region

Figure 5-33 illustrates a special module for the irregular region. In the proposed module, two continuous counterparts of the interwoven pattern are created along two conjugate directions, as shown in the middle and right image at the top row in Figure 5-33. The bottom row image of Figure 5-33 shows the resulting interwoven pattern at the irregular vertex, *Vt*, in the front view.

Additional renderings in perspective are shown in Figure 5-34. Two counterparts are colored in shaded red and green respectively. In Figure 5-35, the final rendering of the surface with interwoven panels is given. Briefly, the examples generated by applying the Archimedean pattern and interwoven pattern both indicate the dominant influence from the underlying tessellation.

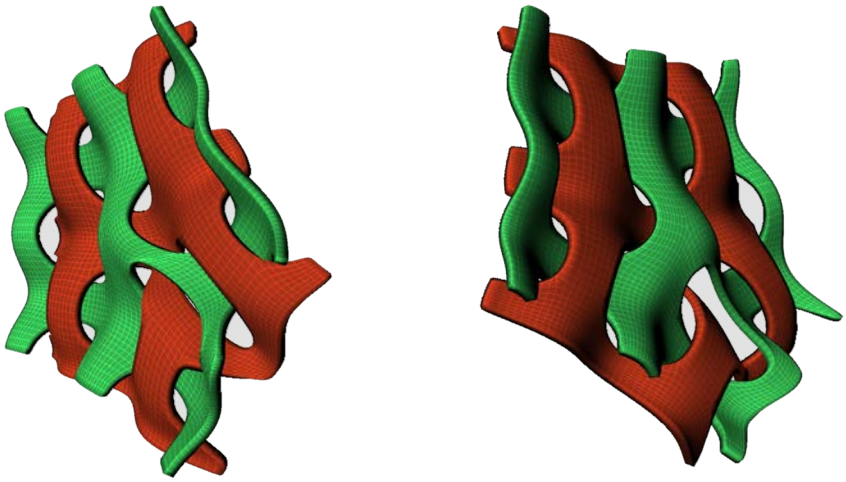


Figure 5-34 Module component in perspective

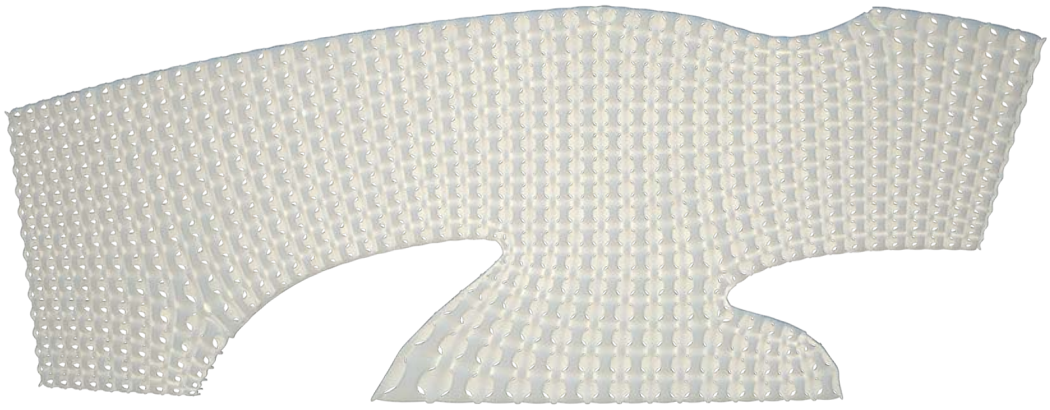


Figure 5-35 A mesh module for the interwoven pattern generation

Chapter 6

Conclusion and Future Work

This thesis is motivated by the desire to assist freeform surface manifestation by tackling the surface tessellation problem with added boundary conditions. The objective is to afford designers an innovative way of exploring panel-based freeform surface designs. In this chapter, this dissertation concludes by examining the outcomes of resolving the surface tessellation problem rooted in three major research areas: (1) meshing; (2) the parametric modeling process; and (3) architectural applications. Overall, this dissertation is directed towards promoting a boundary-driven approach as a systematic and computational means for solving issues when considering customizable pattern-based tessellation, in particular, with an interest in tackling evolving complex boundary conditions that usually grow as design progresses.

Briefly, in this dissertation in Chapter 2 I have laid out the important ground work in relation to previous research and discussed current applications. This has been followed in Chapter 3 by preliminary studies on constructive operators for Archimedean and interwoven pattern generation. In Chapter 4, an optimization approach is described with detail on how surface boundaries are utilized for tessellation. In Chapter 5 further applications of using the BD-driven (boundary-driven) approach for customized pattern generations are given.

In the sequel, the contributions, current research limitations and future directions are discussed. Contributions are given within the context of the technical implementation and pedagogical implications. *Technical* contributions refer to both the algorithmic BD-driven optimizing process and its corresponding

implementations for solving geometrical constraints, which stem from the given surface characteristics and the featured boundary conditions. *Pedagogical* contributions refer to the application of the parametric modeling process, upon which the technical components are procedurally constructed and integrated, to address complex geometry problem within the context of freeform architectural design. Finally, current limitations are discussed and future directions of the proposed BD-Driven approach are projected with respect to supporting sustainable development in the field of freeform architectural design.

6.1 Summery: BD-Driven Tessellation

Customizing and controlling the tessellation scheme for any arbitrary surface is of some interest to both the computational geometry and architecture communities (Liu et al., 2006; Pottmann, 2008; Pottmann et al., 2008; Eigensatz et al., 2010). Conventionally, tessellating a NURBS surface is typically limited to its underlying iso-parameterics, namely, *U* and *V*. One immediate limitation of using such iso-parametric schemes is at the surface boundaries, in which complex conditions often evolve with added design operations such as trimming (Figures 1-4 and 1-7). Not only is the underlying iso-parametric scheme insufficient for tessellating a surface with trimmed edges, but furthermore, the generated tessellation patterns are, to some degree, uncontrollable by the *UV* parameters (Figure 1-5). The contention is that the given complex boundary conditions can be taken into consideration for surface tessellation, a well-structured boundary-driven model can be constructed and thus facilitate further architectural applications. Moreover, by rendering the control of tessellating surface back to users (or designers), design exploration can be expanded, or, perhaps, better instructed for seeking potentially better or other alternatives.

In the process of approximating a surface with pattern-based components, the BD-driven approach initiates from the featured surface boundary investigation. Surface boundaries are treated as the input and the proposed computation utilizes three major BD-driven components, namely, BDTensor, BDCurve, and BDMesh to explore the solutions. Among these, the first two components, BDTensor and BDCurve, are preliminary information operators, which compute and store interpolated data from an underlying surface topology and featured boundary characteristics. BDMesh then builds up the discretized elements by iteratively sorting the constructed BDCurve network and optimizes toward a quad-dominant mesh. The goal is to provide a discretized model that conforms to surface continuity and at the same time maintains surface integrity in a coherent manner. In addition, irregular elements, such as triangles or polygonal face elements at boundary edges, or irregular regions in the quad-dominant configuration are also be minimized.

6.2 Technical Contributions: Mesh Automation

BDMesh is an algorithmic approach for automatic mesh generation from a NURBS surface with added considerations given to the featured boundary conditions. As discussed in this dissertation, meshing is essential for manifesting a freeform surface and could potentially dominate both aesthetical appearance and physical construction. In light of the desire to tessellate surfaces with potential irregular boundary conditions, this dissertation promotes BD-driven optimization as a general approach to this problem. Major technical contributes are summarized as the following three aspects: (1) automatic quad-dominant mesh generation; (2) easy-to-manage structure for fast prototyping; and (3) a divide-and-conquer approach to solving complex freeform designs.

Automatic quad-dominant mesh generation

BDMesh provides an algorithm for converting a surface into a quad-dominant mesh, which consists of mainly quadrilateral face elements. The approach eases the discretization process for users, particularly—designers pursuing freeform architecture, and provides a distinct aspect on how mesh configuration can be optimized and guided by incorporating featured boundary constraints. The resulting mesh exhibits the potential of not only generating quad mesh but also minimizing irregular regions in the constructed mesh structure.

In one sense, BDMesh affords users the capability of discretizing a surface in a relatively easy fashion. BDMesh supplies a well-structured discretized model for users to evaluate further objective and performance criteria during the explorative and iterative design process. As an example, the mesh can be regarded as a representation model to simulate the compression and tension of a built structure with the results then being harvested back to improve structural soundness.

Easy to manage for fast prototyping

A mesh is a structured network of vertices, edges and faces. In the constructed network, a pair of vertices bound an edge; a list of edges bound a face, and so forth. These connectivity relationships are essential and also useful information when extended to architectural application. For instance, given a mesh data structure, designers can easily develop glass panes for surface panels by utilizing the mesh face elements. In general, mesh elements are employed as references to develop and examine physical building components. By drawing analogies from mesh elements to corresponding design artifacts such mapping renders flexibility for designers to systematic and fast prototyping of intended architectural components.

A divide-and-conquer approach for complex freeform design

The meshing process is regarded as the essential step in free from surface manifestation. In one sense, this process simplifies the surface by a collection of polygonal components, which are later extended with added considerations that cater for physical fabrication. This simplification step can be integrated as part of the divide-and-conquer approach, in which a complex surface geometry problem is divided into two solvable sub-problems and then tackled individually. The first sub-problem relates to the pattern-based surface tessellation and the second refers to the procedural construction of the designated architectural components.

6.3 Pedagogical Contributions: Parametric Design Process

In the body of this dissertation, a parametric modeling approach is promoted as the basis for a systematic and computational design approach in which various modules are investigated. For instance, BDCurve is a parametric module that visualizes curves in two conjugate directions, which are interpolated from the feature boundary conditions. BDMesh is the mesh component, which computes the discretized model of a given surface. In a word, each module exists individually, but also contingent on each other. On one hand, these modules are individual entities due to the fact that they are domain-specific information objects, which respectively govern domain-specific information. On the other, they are also dependent on their predecessors—namely, the input. For instance, BDMesh relies on a BDCurve, and BDCurve relies on a BDTensor. The directed connectivity among dependent objects delineates the order of the constructive operations in the parametric modeling process. The implementation of the approach demonstrates a constraint-solving scheme for design exploration within a parametric modeling context.

Additionally, with the described optimization process, I further investigated customizable pattern-based generations using Archimedean and interwoven patterns (Chapter 5). These examples are built upon a discretized model from the target surface and can be treated as the successor of the BDMesh component. Overall, the presented BD-driven model generalizes the process for pattern-based exploration by supplying a well-structured network. The underlying topology (namely, a quad-dominant mesh network with minimal irregularities) affords designers a relatively easy fashion for exploring alternative pattern-based generations when considering geometrically complex surface designs.

Boundary-driven as the vehicle for pattern-based surface tessellation

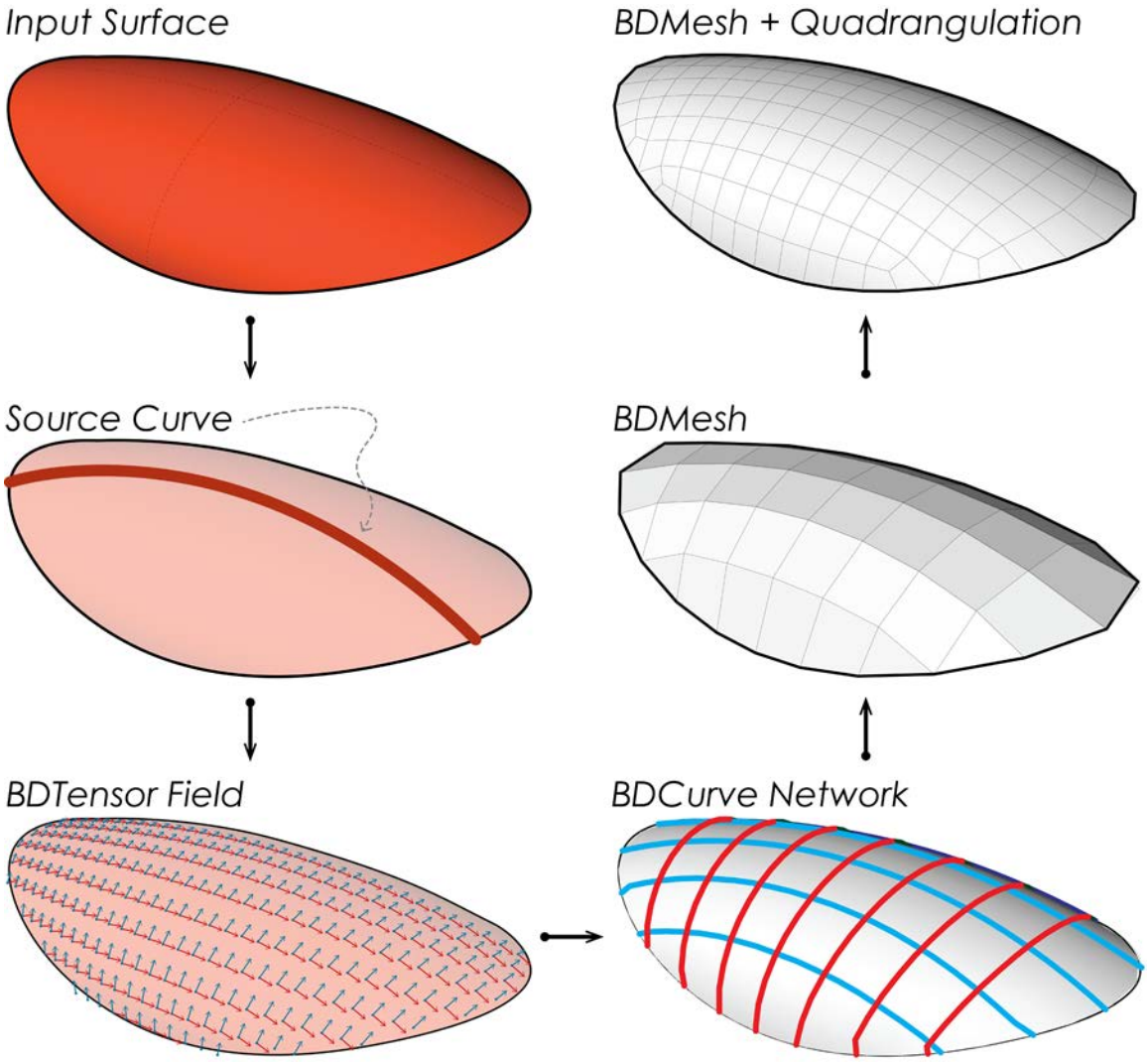


Figure 6-1 A boundary-driven optimization process using additional boundary curve(s) The process initiates from the *top-left* corner to the *top-right* corner

As discussed in the body of this dissertation, boundaries of a surface are treated as important ingredients for constructing surface tessellations. Figure 5-10 is a demonstration of evolving surface tessellations through a gradual increasing of the complexity of the feature boundary conditions. In general, this generative process can be summarized as the following six steps: (1) Surface selection; (2) Boundary identification; (3) BDTensor field initiation; (4) BDCurve network construction; and (5) BDMesh construction; and (6) Post-optimization. As new boundary conditions are introduced, the full cycle of six-

stage operations are re-executed to investigate current available solution(s). This process can be iteratively explored with any potential boundary condition. As noted earlier, "boundaries" may not necessarily be limited to the original surface edges. Additional curve(s) on a surface domain can also be treated as additional curve source(s) to customize surface tessellation pattern. In Figure 6-1, a complete iteration of using an additional curve on a trimmed surface is illustrated. Notice that this surface tessellation only takes the single curve on the surface (shown in the middle-left image of Figure 6-1) for the optimization. In this figure, the six-stage process initiates from the top-left corner to the top-right corner in a counter-clockwise order.

Parametric pattern derivation using topological operation and mesh subdivision

In Chapter 5, further applications are given to demonstrate the parametric process from the initial mesh generation to the subsequent pattern explorations. For instance, the west façade of Zaha Hadid's Next Gene Museum is utilized as an example to demonstrate extended parametric applications (using both Archimedean and interwoven patterns). Such pattern-based construction is regarded as an extension of the above parametric modeling process and thus successive to the BD-driven optimization. This and other examples stated in Chapter 1 are built upon the assumption that a well-structured tessellation pattern can lead to further sustainable architecture development. The BD-driven approach presented in this dissertation supports this view by solving computational geometry constraints, particularly, for freeform surfaces, within a parametric modeling paradigm.

6.4 Current Limitations

The main investigation in this dissertation concentrated on the formation of an optimized pattern-based tessellation with irregular boundary conditions. As shown in Chapter 5 Figure 5-4 and Figure 5-5, boundaries are often modified to pursue both visual and functional purposes (Zahad, 2008). The act of such manipulation may potentially increase the complexity of the surface boundary conditions and thus future manifestation. An important assumption is that these boundaries could serve as important design cures and vehicles during the course of freeform design. Under this freeform design context, the optimization approach utilizes boundaries as the dominant force for tackling surface tessellating issues such that the resulting segmentation can afford to fabricate constructible building components. Some limitations that apply to this research are discussed next.

Boundary conditions vs. surface curvature

In some cases, the boundary-driven tessellation may not always yield an optimal solution; for instance, surface with curvature directions vastly divergent from the trimming boundaries. In this situation, constraining the mesh generation solely to the given boundaries may potentially increase the complexity of the generated face elements, particularly, at those constrained boundaries. As shown in Figure 6-2, the influence from surface curvature is gradually increased in the optimization process from the left to the right images. More irregular regions emerge at the boundaries where the conjugate directions are modified by the curvature directions. This result is due to the conflicts between the dominant force of quadrangulation and the underlying surface curvature.

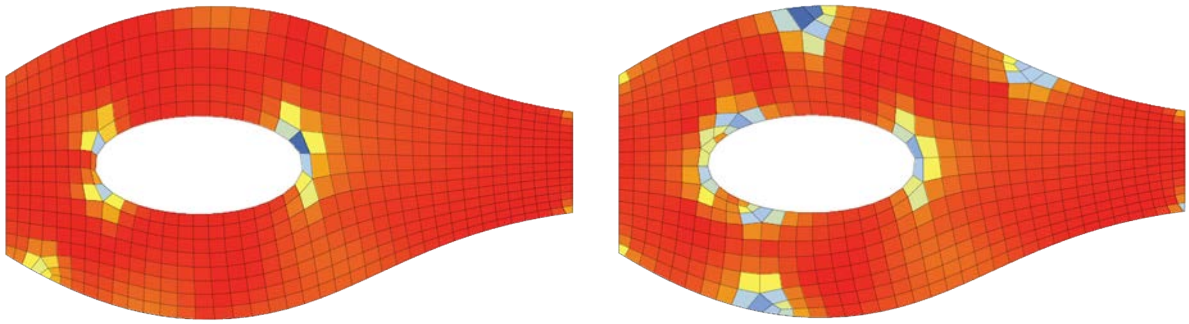


Figure 6-2 Meshing trimmed surfaces with surface curvature consideration.
The curvature influence utilized in the interpolation is gradually increased from left to the right.

BDCurve network initiation

In the optimization process, preliminary BDCurve construction is initiated from a given node on the target surface domain (Chapter 4 Figure 4-6). This initial node can be either supplied by user input or randomly picked by the system. Current limitation occurs when following two conditions exist simultaneously: (1) a target surface is featured with a relatively symmetrical configuration; and (2) the initial sampling node does not capture this symmetrical relation from the given surface. For instance, a sampling node at the center along the symmetrical direction will more likely generate a symmetrical configuration. In other words, a randomized initial node for BDCurve construction may not always be sufficient and efficient in capturing, sometimes, important visual characteristics, such as, symmetry, and this requires additional intervention from the user control. However, to identify a symmetrical relation is not an easy task and requires further research into computational geometry. Currently, for such a surface, an initial node is instructed intentionally by user input. In some cases, a bad node choice can potential

cause more irregular regions in the generated structure and this is currently compensated by post-optimization, such as skewed triangle removal or quadrangulation.

To summarize, this process is limited to some degree by certain types of surface configurations and is currently refined by user control and post optimization. For robustness in the mesh construction, an improvement can be made toward reducing the probability of bad node initiations from either user input or stochastic search. In doing so, the efficiency of generating a well-structured mesh can be improved. Nevertheless, this is not an easy task for algorithmically identifying an initial node in a geometrically complex surface. In one sense, visual guidance from users can be utilized to facilitate this optimization process.

Single surface interpolation

Current implementation is limited to single-patch surface interpolation. A single-patch surface depends solely on a set of control points and a single interpolating function. A surface that consists of multiple patches is currently not considered as a valid input surface type. A potential remedy to such surfaces with multiple patches is to separate the surface into a collection of solvable single-patch surfaces, which can then be tackled separately. A further optimization to realign seams among different surface patches can be expected to generate a watertight meshing result.

6.5 Future Directions

This dissertation presents an algorithmic approach to solving surface tessellation problem within the context of architectural freeform design. The current implementation is focused on theoretical and technical investigations into the problem. In order to demonstrate the power of utilizing such an approach for freeform architectural design, more surface tessellation and fabrication examples need to be considered. In addition, an appropriate interface between users and computational mechanisms is expected for the successful integration in the real design practice. The overview for the future directions is given as follow.

Cross- or multiple-platform usages

Current deployment is limited to dynamic-link libraries and utilized as prototype components in Grasshopper/Rhino for demonstration purposes. To make a real application with lasting contributions to the field of freeform designs, multiple- or cross-platform manipulations are required, for instance, other

platforms, such as Revit or even Processing—the former is an example of a commercial parametric building information modeling tool, the latter a general purpose graphical programming environment targeted towards artists and designers. Both platforms provide a certain amount of support towards parametrically generating constructive geometry and can be considered as potential candidates for incorporating the approach described in this dissertation. Essentially, the approach articulates how information flow regarding surface tessellation should be directed, processed and computed. In considering multi-platform support, proper data interpolation and integration schemes are required for managing the integrity of distinct data presentations in respective platforms.

User testing, control and interaction

In order to render the flexibility of utilizing this approach, adding controls at the user-interaction level are considered. For instance, improved control, from single to multiple boundaries, needs to be further addressed from a user-interaction perspective. Single to multiple boundaries are essential and are implemented for BDTensor interpolation and BDCurve network construction. The current application of employing additional curves is limited by an ability to directly program the curves. Given that control can be, interactively, manipulated by users in real time, this approach will be more amenable to real design exploration.

In addition, user testing within architectural design contexts is also considered essential for validating the approach for practical use. Feedbacks from testing experiments can be expected to improve the workflow and develop potential future functionalities.

Multiple-patch surface

As mentioned in Chapter 4 and in Section 6.4, the current implementation is limited to single-patch surface tessellation. In order to accommodate various surface configurations that may occur in the design process, a multiple-patch surface example is suggested. This involves analysis across various surface domains and the alignment of meshing elements at the boundaries where various surfaces meet.

For some geometrically complex surfaces, it is sometimes beneficial and strategic to first have a single-patch surface split into smaller surface patches, and then tessellating each patch. In doing so, the complexity of the surface can be reduced and this process will in turn facilitate the surface tessellation process.

Further extension using performance-based refinement

Planarity is probably one of the more important indices for fabrication; notwithstanding, planarity is not always applicable nor is it the only panacea. Instead, a structure to accommodate various considerations, such as geometrical, structural, performative, material, etc., is much more practical. In other words, other design or performance constraints may sometimes be considered to be more dominant and hence induce much more complex design constraints. In the future, further applications are considered to encompass performance-driven criteria in the optimization process. Essentially, a well-structured mesh is a discrete model for conducting analysis and results can also be easily harvested for further refinement.

Fabrication

As discussed in Chapter 2, there are various types of manufacturing techniques catering to distinct face-based construction. In addition one can also consider using boundary-driven tessellation patterns as the foundation for future fabrication validation. Intended development of optimizing pattern-based module for fabrication is considered as the essential steps toward a fabrication-friendly application.

To summarize, the body of this dissertation demonstrates a constraint solving exercise within the context of freeform designs. By resolving complex boundary conditions, the ultimate goal is to make technology as affordable as possible so that design creativity can be expanded without limitations. Yet, this does not imply that the freedom to pursue creativity is unlimited. Instead, it is a constrained freedom. For instance, by solving both the geometrical and fabrication constraints, an optimized solution for cost-effective fabrication can be provided for a sustainable design development.

BIBLIOGRAPHY

Abi-Ezzi, S. S. and L. A. Shirman, 1991. "Tessellation of Curved Surfaces under Highly Varying Transformations," in *Eurographics '91*, 385-397.

Abi-Ezzi, S. S. and L. A. Shirman, 1994. "Fast Dynamic Tessellation of Trimmed NURBS Surfaced1," *Computer Graphics Forum* **13**(3): 107-126.

Aish, R. and R. Woodbury, 2005. "Multi-Level Interaction in Parametric Design," in *Proceedings of International Symposium on Smart Graphics*, Frauenwoerth Cloister, Germany, 151-162.

Akleman, E., V. Srinivasan, et al., 2005. "Remeshing schemes for semi-regular tilings," in *The International Conference on Shape Modeling and Applications*, 44-50.

AKLEMAN, E., V. SRINIVASAN, et al., 2004. "Topmod: Interactive Topological Mesh Modeler," Technical Report, Texas A&M University, College Station, Texas.

Aurenhammer, F., 1991. "Voronoi Diagrams - A Survey of a Fundamental Geometric Data Structure," *ACM Computing Surveys* **23**(3): 345-405.

Bell, B. and A. Vrana, 2004. "Digital Tectonics: Structural Patterning of Surface Morphology," in *Proceedings of the 23rd Annual Conference of the Association for Computer Aided Design in Architecture*, Cambridge, Ontario, 186-201.

Berg, M. d., O. Cheong, et al., 2008. *Computational Geometry: Algorithms and Applications*, Springer-Verlag, Berlin.

Bertel, S., C. Freksa, et al., 2004. "ASPECTUALIZE AND CONQUER IN ARCHITECTURAL DESIGN," in *Visual and Spatial Reasoning in Design III* (J. Gero, B. Tversky and T. Knight), pp. 255-279.

Biloria, N. and V. Sumini, 2009. "Performative Building Skin Systems: A Morphogenomic Approach Towards Developing Real-Time Adaptive Building Skin Systems," *International Journal of Architectural Computing* **07**(04): 643-676.

Boeykens, S. and H. Neuckermans, 2006. "Improving Design Workflow in Architectural Design Applications," *International Journal of Architectural Computing* **04**(04): 1-19.

Burry, J. and M. Burry, 2010. *The New Mathematics of Architecture*, Thames & Hudson, New York.

Burry, J. R., 2007. "Mindful Spaces: Computational Geometry and the Conceptual Spaces in which Designers Operate," *International Journal of Architectural Computing* **05**(04): 611-624.

Cardoso, D. and L. Sass, 2008. "Generative Fabrication," in *Design Computation Cognition 08*, Atlanta, USA, 713-732.

Catmull, E. and J. Clark, 1978. "Recursively generated B-spline surfaces on arbitrary topological meshes," *Computer-Aided Design* **10**(6): 350-355.

Corser, R., 2010. *Fabricating Architecture: Selected Readings in Digital Design and Manufacturing*, Princeton Architectural Press, Princeton, NJ.

Cutler, B. and E. Whiting, 2007. "Constrained Planar Remeshing for Architecture," in *Graphics Interface 2007*, Montreal, Canada, 11-18.

Davis, D., J. Burry, et al., 2011. "Untangling Parametric Schemata : Enhancing Collaboration through Modular Programming," in *Proceedings of the 14th International Conference on Computer Aided Architectural Design Futures*, Liege, Belgium, University of Liege-Belgium, 55-68.

Delaunay, B., 1934. "Sur la sphère vide, Izvestia Akademii Nauk SSSR," *Otdelenie Matematicheskikh i Estestvennykh Nauk* **7**: 793-800.

Dirichlet, G. L., 1850. "Reine Angew," *Journal für die Reine und Angewandte Mathematik,* **40**: 209–227.

Dritsas, S. and M. Becker, 2007. "Research & Design in Shifting from Analog to Digital," in *Proceedings of the 27th Annual Conference of the Association for Computer Aided Design in Architecture*, Halifax, Nova Scotia, 56-65.

Dritsas, S., R. Charitou, et al., 2006. "Computational Methods on Tall Buildings - The Bishopsgate Tower," in *Proceedings of the 24th Annual Conference of Education and research in Computer Aided Architectural Design in Europe*, Volos, Greece, 778-785.

Duesing, B., 2007. "NURBS Add a Curve to CAD Modeling," last accessed on Nov 05, 2011, from http://www.rhino3d.com/clippings/Sept2007/DesignWorld_Rhino_Sept2007.pdf.

Eastman, C., 2004. "New Methods of Architecture and Building," in *Proceedings of the 23rd Annual Conference of the Association for Computer Aided Design in Architecture*, Cambridge, Ontario, 20-27.

Eastman, C. M., 1999. *Building Product Models: Computer Environments, Supporting Design and Construction*, CRC Press, Boca Raton, FL.

Eigensatz, M., M. Deuss, et al., 2010a. "Case Studies in Cost-Optimized Paneling of Architectural Freeform Surfaces," in *Advances in Architectural Geometry 2010*, 49-72.

Eigensatz, M., M. Kilian, et al., 2010b. "Paneling Architectural Freeform Surfaces," *ACM Transactions on Graphics* **29**(4): 45.

Emdanat, S., E. G. Vakalo, et al., 1999. "Solving Form-Making Problems Using Shape Algebras and Constraint Satisfaction," in *Architectural Computing: The Intelligent Machine 1: AI*, pp. 620-625.

Eves, H., 1972. *Survey of Geometry*, Allyn and Bacon, Boston, MA.

Fleischmann, P., 1999. "Mesh Generation for Technology CAD in Three Dimensions " Ph.D. dissertation, Technische Universität Wien, Vienna, Austria.

Flory, S. and H. Pottmann, 2010. "Ruled Surfaces for Rationalization and Design in Architecture," in *Proceedings of the 30th Annual Conference of the Association for Computer Aided Design in Architecture*, New York, 103-109.

Glympha, J., D. Sheldena, et al., 2004. "A parametric strategy for free-form glass structures using quadrilateral planar facets," *Automation in Construction* **13**: 187-202.

Goldberg, S. A., 2006. "Computational Design of Parametric Scripts for Digital Fabrication of Curved Structures," *International Journal of Architectural Computing* **4**(3): 99-117.

Grunbaum, B. and G. C. Shephard, 1987. *Tilings and patterns*, W. H. Freeman and Company, New York.

Hadid, Z., 2008. "Schematic Design Report for Next-Gene Architecture Museum," Technical Report, Graduate Institute of Architecture, National Chiao-Tung University, Hsinchu, Taiwan.

Hauer, E., 2007. *Erwin Hauer: Continua-Architectural Screen and Walls*, Princeton Architectural Press, New York City.

Hermann, L. R., 1976. "Laplacian-isoparametric grid generation scheme," *Journal of Engineering Mechanics* **102**(EM5): 749-756.

Herr, C. M. and J. Karakiewicz, 2007. "ALGOGRAM: Automated Diagrams for an Architectural Design Studio," in *Proceedings of the 12th International Conference on Computer Aided Architectural Design Futures*, Sydney, Australia, 167-180.

Hesselgren, L., R. Charitou, et al., 2007. "The BishopsgateTower Case Study," *International Journal of Architectural Computing* **5**(1): 61-82.

Hudson, R., 2008. "Knowledge Acquisition in Parametric Model Development," *International Journal of Architectural Computing* **6**(4): 435-451.

Hudson, R., 2009. "Parametric Development of Problem Descriptions," *International Journal of Architectural Computing* **7**(2): 199-216.

Iordanova, I., 2007. "Teaching Digital Design Exploration:Form Follows…," *International Journal of Architectural Computing* **5**(4): 685-702.

Iwamoto, L., 2009. *Digital Fabrications: Architectural and Material Techniques*, Princeton Architectural Press, New York City.

Jernigan, F., 2007. *BIG BIM little bim - The practical approach to Building Information Modeling*, 4Site Press, Salisbury, Maryland.

Jodidio, P., 2009. *Zaha Hadid: Complete Works*, TASCHEN America Llc, Los Angeles, CA.

Kahlert, E. J., 2009. "TILING SURFACES WITH STRAIGHT STRIPS," Master Thesis, The School of Computing Science, Simon Fraser University, Vancouver, BC, Canada.

Kieran, S. and J. Timberlake, 2004. *Refabricating Architecture: How Manufacturing Methodologies are Poised to Transform Building Construction*, McGraw-Hill Professional, New York.

Kieran, S. and J. Timberlake, 2008. *Loblolly House: Elements of a New Architecture*, Princeton Architectural Press, Princeton, NJ.

Kilian, A., 2006. "Design Exploration through Bidirectional Modeling of Constraints," Ph.D. dissertation, Department of Architecture, Massachusetts Institute of Technology, Cambridge, MA.

Kilian, A., 2006. "Design innovation through constraint modeling," *International Journal of Architectural Computing* **4**(1): 87-105.

Kolarevic, B., 1993. "Geometric Relations as a Framework for Design Conceptualization," Doctoral dissertation, Graduate School of Design, Harvard University, Cambridge, MA.

Kolarevic, B., 1997. "Regulating Lines, Geometric Relations, and Shape Delineation in Design," in *Proceedings of the 15th Annual Conference of Education and research in Computer Aided Architectural Design in Europe*, Vienna, Austria, 17-20.

Kolarevic, B., 2005a. *Architecture in the Digital Age: Design and Manufacturing*, Taylor & Francis, New York and London.

Kolarevic, B., Ed. 2005b. *Performative Architecture: Beyond Instrumentality*, Spon Press, New York and London.

Kolarevic, B., 2009. "Towards Integrative Design " *International Journal of Architectural Computing* **7**(3): 335-344.

Kolarevic, B. and K. Klinger, Eds., 2008. *Manufacturing Material Effects: Rethinking Design and Making in Architecture*, Routledge, New York and London.

Lee, D. T. and B. J. Schachter, 1980. "Two algorithms for constructing a Delaunay triangulation," *International Journal of Parallel Programming* **9**(3): 219-242.

Lindsey, B., 2001. *Digital Gehry*, Birkhäuser, Basel.

Liu, Y., H. Pottmann, et al., 2006. "Geometric Modeling with Conical Meshes and Developable Surfaces," *ACM Transactions on Graphics* **25**(3): 681-689.

Loop, C., 1987. "Smooth Subdivision Surfaces Based on Triangles," Master Thesis, University of Utah, Salt Lake City, UT.

Maeda, J., 2001. *Design By Numbers*, The MIT Press, Cambridge, MA.

Maeda, J., 2004. *Creative Code: Aesthetics + Computation*, Thames & Hudson, New York.

Maeda, J., 2006. *The Laws of Simplicity*, The MIT Press, Cambridge, MA.

Maleki, M. M. and R. F. Woodbury, 2008. "Reinterpreting Rasmi Domes with Geometric Constraints:A Case of Goal-seeking in Parametric Systems," *International Journal of Architectural Computing* **06**(04): 375-395.

Medjdoub, B., 1999. "Interactive 2D Constraint-Based Geometric Construction System," in *the Eighth International Conference on Computer Aided Architectural Design Futures* Atlanta, 197-212.

Meredith, M., Aranda-lasch, et al., Eds., 2008. *From Control to Design: Parametric/Algorithmic Architecture*, Actar, Barcelona, Spain.

Moustapha, H., 2004. "A Formal Representation for Generation and Transformation in Design," in *Generative CAD Systems Symposium*, Carnegie Mellon University, Pittsburgh, PA

Moustapha, H., 2006. "Architectural Explorations: A Formal Representation for the Generation and Transformation of Design Geometry," Ph.D. dissertation, School of Architecture, Carnegie Mellon University, Pittsburgh, PA.

Muller, P., P. Wonka, et al., 2006. "Procedural Modeling of Buildings," in *Proceedings of SIGGRAPH 2006*, 614-623.

Nir, E. and G. Capeluto, 2005. "Smart Cloud-of-Points Model: Point-based Digital Media and Tools for Architectural Design," in *Proceedings of the 23th Annual Conference of Education and research in Computer Aided Architectural Design in Europe*, Lisbon, Portugal, 687-694.

Oxman, N., 2007. "Get Real Towards Performance-Driven Computational Geometry," *International journal of architectural computing* **05**(04): 663-684.

Oxman, R., 2006. "Theory and design in the first digital age," *Design Studies* **27**(3): 229-265.

Oxman, R., 2008. "Performance-based Design: Current Practices and Research Issues," *International Journal of Architectural Computing* **06**(01): 1-17.

Pak, B., O. O. Ozener, et al., 2006. "Utilizing customizable generative design tools in digital design studio: Xp-GEN experimental form generator," *International Journal of Architectural Computing* **04**(04): 21-33.

Paoluzzi, A., V. Pascucci, et al., 1995. "Geometric Programming: A Programming Approach to Geometric Design," *ACM Transactions on Grphics* **14**(3): 266-306.

Piegl, L., 1991. "On NURBS: a Survey," *IEEE Computer Graphics and Applications* **11**(1): 55-71.

Piegl, L. A. and W. Tiller, 1996. *The NURBS Book*, Springer-Verlag, New York, NY.

Pine, B. J., 1993. *Mass Customization - The New Frontier in Business Competition*, Harvard Business School Press, Boston, MA.

Pottmann, H., 2008. "Geometry of Architectural Freeform Structures," in *Proceedings of the 2008 ACM symposium on Solid and physical modeling*, ACM New York, 9-9.

Pottmann, H., 2010. "Architectural Geometry as Design Knowledge," *Architectural Design* **80**(4): 72-77.

Pottmann, H., A. Asperl, et al., Eds., 2007a. *Architectural Geometry*, Bentley Institute Press, Exton, PA.

Pottmann, H., S. Brell-Cokcan, et al., 2006a. "Discrete Surfaces for Architectural Design," in *Curves and Surface Design: Avignon 2006*, 213-234.

Pottmann, H., Y. Liu, et al., 2007b. "Geometry of multi-layer freeform structures for architecture," *ACM Transactions on Graphics* **26**(3): 1-11.

Pottmann, H., A. Schiftner, et al., 2008a. "Freeform surfaces from single curved panels," in *International Conference on Computer Graphics and Interactive Techniques*, Los Angeles, California, ACM, New York, 76:70-10.

Pottmann, H., A. Schiftner, et al., 2008b. "Geometry of Architectural Freeform Structures," *International Mathematical News (Internationale Mathema- tische Nachrichten)* **209**: 15–28.

Pottmann, H. and J. Wallner, 2006b. "The focal geometry of circular and conical meshes," *Advances in Computational Mathematics* **29**(3): 249-268.

Pronk, I. A., I. V. Rooy, et al., 2009. "Double-curved surfaces using a membrane mould," in *International Association for Shell and Spatial Structures (IASS) Symposium 2009*, Valencia, Spain

Prusinkiewicz, P. and A. Lindenmayer, 1991. *The Algorithmic Beauty of Plants*, Springer, New York.

Puusepp, R. and P. Coates, 2007. "Spatial Simulations with Cognitive and Design Agents," *International Journal of Architectural Computing* **05**(01): 100-114.

Reda, I. and A. Andreas, 2008. "Solar Position Algorithm for Solar Radiation Applications," Technical Report, National Renewable Energy Laboratory, Golden, Colorado.

Roelofs, R., 2010. "About Weaving and Helical Holes," last accessed on Nov 05, 2011, from http://www.rinusroelofs.nl/projects/h-holes/pr-h-holes-00.html.

Rogers, D. F., 2000. *An Introduction to NURBS: With Historical Perspective*, Morgan Kaufmann, San Francisco.

Schodek, D., M. Bechthold, et al., 2004. *Digital Design and Manufacturing: CAD/CAM Applications in Architecture and Design*, Wiley, Hoboken, NJ.

Schumacher, P., 2009. "Parametric Patterns," in *Architectural Design* (M. Garcia), pp. 28–41, Wiley, Hoboken, NJ.

Seidel, R., 1988. "Constrained Delaunay Triangulations and Voronoi Diagrams with Obstacles," Technical Report, Inst. for Information Processing, Graz, Austria.

Sequin, C. H., 2005. "CAD tools for aesthetic engineering," *Computer-Aided Design* **37**: 737-750.

Shea, K., R. Aish, et al., 2004. "Towards integrated performance-driven generative design tools," *Automation in Construction* **14**(2): 253-264.

Shelden, D. R., 2002. "Digital Surface Representation and the Constructibility of Gehry's Architecture," Ph.D. dissertation, Department of Architecture, Massachusetts Institute of Technology, Canbridge, MA.

Shepard, D., 1968. "A two-dimensional interpolation function for irregularly-spaced data," in *Proceedings of the 1968 ACM National Conference*, 517–524.

Shimada, K. and D. C. Gossard, 1995. "Bubble Mesh: Automated Triangular Meshing of Non-Manifold Geometry by Sphere Packing," in *ACM Third Symposium on Solid Modeling and Applications*, 409-419.

Shimada, K. and D. C. Gossard, 1998. "Automatic Triangular Mesh Generation of Trimmed Parametric Surfaces for Finite Element Analysis," *Computer Aided Geometric Design* **15**(3): 199-222.

Sutherland, I. E., 1963. "Sketchpad: A man-machine graphical communication system," Ph.D. dissertation, Massachusetts Institute of Technology, Cambridge, MA.

Szalapaj, P., 2005. "The Digital Design Process in Contemporary Architectural Practice," in *Proceedings of the 23th Annual Conference of Education and research in Computer Aided Architectural Design in Europe*, Lisbon, Portugal, 751-759.

Terzidis, K., 1999. "Computers and the Creative Process," in *Proceedings of Education and research in Computer Aided Architectural Design in Europe*, Liverpool, UK, 43-50.

Terzidis, K., 2003. "Hybrid Form," *Design Issues* **19**(2): 76-80.

Terzidis, K., 2004. "Algorithmic Design: A Paradigm Shift in Architecture?," in *Proceedings of Education and research in Computer Aided Architectural Design in Europe*, Copenhegen, Denmark, 201-207.

Terzidis, K. and J. Jungclaus, 2007. "Predicting the Future: Open Source CAAD?," in *Proceedings of Education and research in Computer Aided Architectural Design in Europe*, Frankfurt, German, 815-820.

Voronoi, G. F., 1908. "Nouvelles applications des paramètres continus à la théorie de formes quadratiques," *Journal für die reine und angewandte Mathematik*, **134**: 198–287.

Wallner, J. and H. Pottmann, 2011. "Geometric Computing for Freeform Architecture," *Journal of Mathematics in Industry* **1**(1): 4.

Wallner, J., A. Schiftner, et al., 2010. "Tiling Freeform Shapes With Straight Panels: Algorithmic Methods," in *Advances in Architectural Geometry 2010*, Vienna, Australia, 73-86.

Wang, T.-H., 2008. "Rule-based Procedural Reconstruction of NURBS Surfaces for Architectural Exploration," in *Proceedings of Advances in Architectural Geometry 2008*, Vienna, Austria, 131-134.

Wang, T.-H., 2009. "Procedural Reconstruction of NURBS Surfaces," in *Proceedings of the 14th International Conference on Computer Aided Architectural Design Research in Asia*, Yunlin, Taiwan, 597-606.

Wang, T.-H., 2010. "Design Patterns for Parametric Modeling in Grasshopper," last accessed on Nov 05, 2011, from http://www.andrew.cmu.edu/org/tsunghsw-design/.

Wang, W. and Y. Liu, 2009. "A Note on Planar Hexagonal Meshes," in *Nonlinear Computational Geometry*, pp. 221-233, Springer-Verlag, New York, NY.

Wang, W., Y. Liu, et al., 2008. "Hexagonal Meshes with Planar Faces," Technical Report, Dept. of Computer Science, Hong Kong University, Hong Kong, China.

Woodbury, R., 2010. *Elements of Parametric Design*, Routledge, New York and London.

Woodbury, R., 2010. "Elements of Parametric Design," last accessed on Nov 05, 2011, from *http://www.designpatterns.ca/*.

Woodbury, R., R. Aish, et al., 2007. "Some Patterns for Parametric Modeling," in *Proceedings of the 27th Annual Conference of the Association for Computer Aided Design in Architecture*, Halifax, Nova Scotia, 222-229.