MULTI-PANEL UNFOLDING WITH PHYSICAL MESH DATA STRUCTURES

A Thesis

by

YOU WU

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

May 2017

Major Subject: Visualization

ABSTRACT

In this thesis, I demonstrate that existing mesh data structures in computer graphics can be used to categorize and construct physical polygonal models. In this work, I present several methods based on mesh data structures for transforming 3D polygonal meshes into developable multi-panels that can be used in physical construction. Using mesh data structures, I developed a system which provides a variety of construction methods. In order to demonstrate that mesh data structures can be used to categorize and construct physical polygonal models, this system visualizes the mathematical theory and generates developable multi-panels that can be printed and assembled to shapes similar to original virtual shapes. The mesh data structures include ones that are orientable: Quad-Edge, Half-Edge, Winged-Edge; and also one that is non-orientable: Extended GRS.

The advantages of using mesh data structures as guides for physical construction include: There is no restriction on input design model as long as it is manifold, it can be of any genus with n-sided polygon faces; Different mesh data structures provide more options to better fit the input design while taking the physical constraints and material properties in consideration; Developable panels are easy to obtain from thin planar materials using a laser-cutter; When we use mesh data structures, it is also intuitive to assemble such planar panels using mesh information.

Laser-cut developable panels based on mesh data structures provide, therefore, a cost-efficient alternative to 3D printing when dealing with large structures.

To my family, friends, and my dog Charlie.

ACKNOWLEDGMENTS

I would like to thank my committee chair, Professor Ergun Akleman for his support and guidance throughout my graduate study and the course of this research. I am truly grateful for his trust and patience. It has been a great pleasure to have the opportunity to work in TopMod lab. I would also like to thank my committee members, Professor Negar Kalantar Mehrjardi and Professor Richard R. Davison, Jr., for giving me suggestions and perspectives that enriches the purpose of my research. Thanks to my labmate Shenyao Ke for his dedication to our research project and for being a supportive friend and fellow engineer who's always asking the right question and willing to share all his wisdom.

Thanks to Professor Negar Kalantar Mehrjardi, Professor Alireza Borhani Haghighi, and the undergraduate students in College of Architecture who were brave and adventurous enough to be the first users of our software; they have contributed so much to improve the user experience.

I would also like to thank my parents for giving me love unconditionally and for supporting my life decisions and allowing me to pursue my dream. A series of thank yous go to my friends: Cong Wang, Shih-Lung Woo, Qiao Wang, Beth Nam, Aicha Boujnikh. You are always there for me no matter what and I will always cherish our friendships. Thanks to my dog Charlie and my cat Ellie for being the best pets ever, life wouldn't be the same without their company.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a thesis committee consisting of Professor Ergun Akleman and Professor Richard R. Davison, Jr. of the Department of Visualization and Professor Negar Kalantar Mehrjardi of the Department of Architecture.

The "Unfolding" software was developed by Peihong Guo, You Wu, and Shenyao Ke, of which the mesh data structure operations were fully implemented by the student.

Physical objects in Figure 6.4 were assembled by Shenyao Ke. Models in Figure 6.5 and Figure 6.6 were assembled by Shenyao Ke and You Wu. The sculptures in Figure 6.7 were created and photo-shot by students of the Department of Architecture in a beginning level architecture studio class at Texas A&M University.

All other work conducted for the thesis was completed by the student independently.

Funding Sources

Graduate study was supported by a research fellowship from National Science Foundation.

NOMENCLATURE

2D	Two Dimensional
3D	Three Dimensional
CAD	Computer Aided Design
NURBS	Non-uniform rational Basis spline
GRS	Graph Rotation System
Half-Edge	Half-Edge mesh data structure
Quad-Edge	Quad-Edge mesh data structure
Winged-Edge	Winged-Edge mesh data structure

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION AND MOTIVATION

In this chapter, I will explain the motivation behind this research work, and then introduce the goal of this project and the framework it is based on. Key concepts and algorithms that this research uses will also be discussed as well as the motivation of using them.

With the increasing interest in unusual buildings and free-form designs, the Computer Graphics community has been exploring new technologies to facilitate the construction of large scale structures. The Walt Disney concert hall is constructed with hundreds of small steel panels by wrapping them around the structure (see Figure 1.1). These panels were measured and bended in certain directions to confirm to the curvature. This method can be very labor intensive and the construction process highly depends on the design. There's a need for a generic economical solution for these large scale sculptures and buildings.

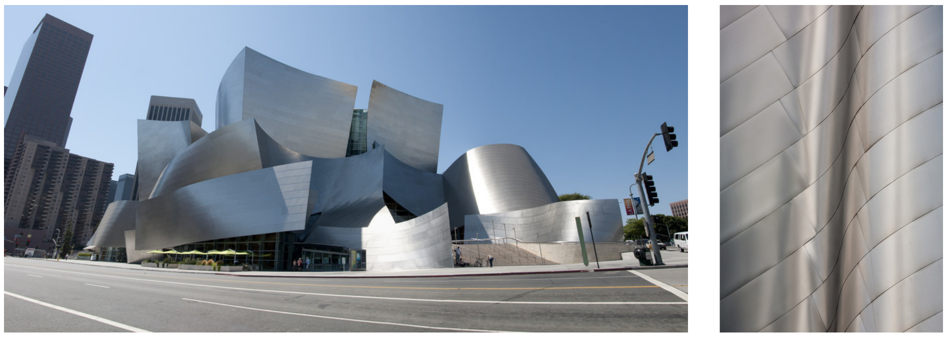


Figure 1.1: Walt Disney concert hall and the steel panels.

Since we're going for an economical solution, 3D printing is not an answer. The limitations of current 3D printing technology are the scale of printed object and the time and cost of printing process and material.

So laser cut provides a great alternative. It is a common machine in architecture studios. There's a wide range of materials that it can use. By assembling the cutted panels together, we can build larger shapes.

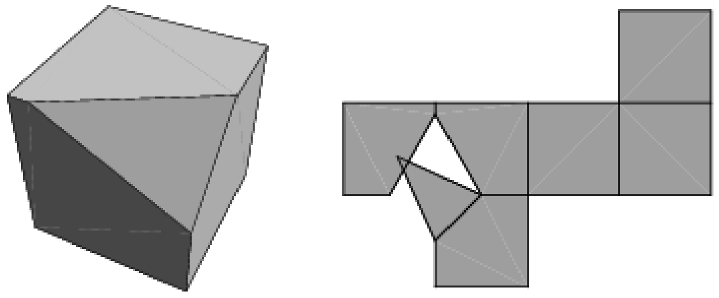


Figure 1.2: Examples of a cube with a cutted corner unfolded to single-panel.

Since laser cutter operates on flat surfaces, we will need to unfold the 3D mesh to 2D pieces. Single-panel unfolding presents a greater challenge because: first of all, there's not even proof that convex polyhedron can all be unfolded using only its edges; Secondly, even for those shapes that can be unfolded flat, there could be overlapping with itself (see Figure 1.2). Considering this is a simple cube with a cutted corner, it's obvious that more complex shapes especially those free-form designs will be either not unfold-able or overlapping itself.

Hence, our solution is to unfold it into multiple panels which also gives us simpler pieces. And with laser-cutting, we're able to construct large scale structures in an economical way.

Multi-panel Construction has been discussed in several research works [10, 11, 12, 13], where their suggested methods focused on mass production and reduction of the time and fabrication cost involved in it. However, mass production is not a strong requirement for developable components [14]. Using planar panels, we can fabricate developable components in an economical way even if each of them is different from another. Such planar panels can be individually manufactured by cutting them off paper, thin metal, or other planar materials using laser-cutter.

The goal of this project is to unfold 3D virtual shapes to multiple planar panels for physical construction of large scale shapes similar to original virtual shapes. I will present an algorithm that uses different categories of mesh data structures that are widely used in Computer Graphics as guides for physical construction. Given an arbitrary, user specified 3D mesh, constructed using any design interface, this algorithm generates a multi-panel unfold-able developable surface that still preserves the same topology as the original mesh. Since we focus on multi-panel construction, the resulting panels are relatively simpler pieces that can be easily packed and assembled together.

When dealing with physical construction, in order to eliminate unnecessary inner tension, the panels and assemblies need to be developable, which means we need to make sure that the faces are planar. In order to compensate the thickness of physical material, we will be using Bézier surface instead of a sharp crease on corner. This would allow for easier assembly. By redistributing the forces sharp creases would get along a wider surface, the physical structure is inherently more stable.

2. PREVIOUS WORK

My research mainly focuses on multi-panel unfolding and construction. Since our goal is to build large scale sculptures while keeping the production time minimal and the cost economical enough for non-industrial usage, we've chosen laser cutter to be the fabrication machine.

2.1 Fabrication Tool

Digital Modeling and Fabrication is a process that by using 3D modeling design software or computer-aided design one joins the design with production. It is widely used in construction industry where accuracy is crucial [15].

An increasingly common machine within the workshops of architecture schools, professional model makers and even design practices, the laser cutter is an important tool owing to the array of functions it offers [14].

The laser cutter is a machine that uses a laser to cut materials such as chip board, matte board, thin sheets of wood. It is firstly introduced in 1965 to drill holes in diamond dies [16]. Throughout the years, researchers have been improving this technique by innovating new cutting tools and methods [17], increasing cutting efficiency [18], and introducing new types of laser [19, 20].

Modern laser cutters are highly integrated with CAD systems. 2D vector based design software(such as Adobe Illustrator [21] and AutoCAD [22]) are used to produce lines on a grid and then send them to laser cutter. Lines defined in different configurations can either cut through the material or score it.

Laser cutter usually requires an additional step of either automatic [17] or manual assembly when constructing large scale or complex designs.

2.2 Modeling for Fabrication

Laser cutter cuts and scores the material based on the distance between laser module and the material [15]. In order to precisely control the resulting pattern, planar materials are preferred. This means that we need to unfold our 3D polygonal mesh into planar panels which is developable.

A developable surface is a surface with zero Gaussian curvature, which is essentially the envelope of a single-parameter family of planes [23]. In other words, it is a surface that can be formed by bending or rolling a planar surface without stretching or tearing. For some technical applications, this property is crucial if we want to construct objects using paper, sheets of tin etc.

Computer aided geometric design emerged when there's increasing demands in designing and modeling free-form surfaces. This area is combined with applied geometry and architecture. Researches done in this area generally focus on approximating 3D free-form surfaces and geometric processing.

2.2.1 Modeling with continuous developables

Geometric modeling with continuous developables is an old topic, the general geometric design problem has not been solved. The contributions have been limited due to the nonlinear nature of the developability constraint that prevented interactive design.

In 1968, Ferris [23] presented an explicit representation for a developable surface which is defined as the envelope of single-parameter family of planes. However, his approach does not result in a Bézier or B-spline formulation, and it does not utilize control planes in a manner that is similar to the way as in curve design.

Since then, rational Bézier and B-spline basis is widely used for designing developable surface which also leads to an extensive amount of literature discussing the developable Bézier surfaces based on both quadratic and cubic curve [24, 3, 1, 2].

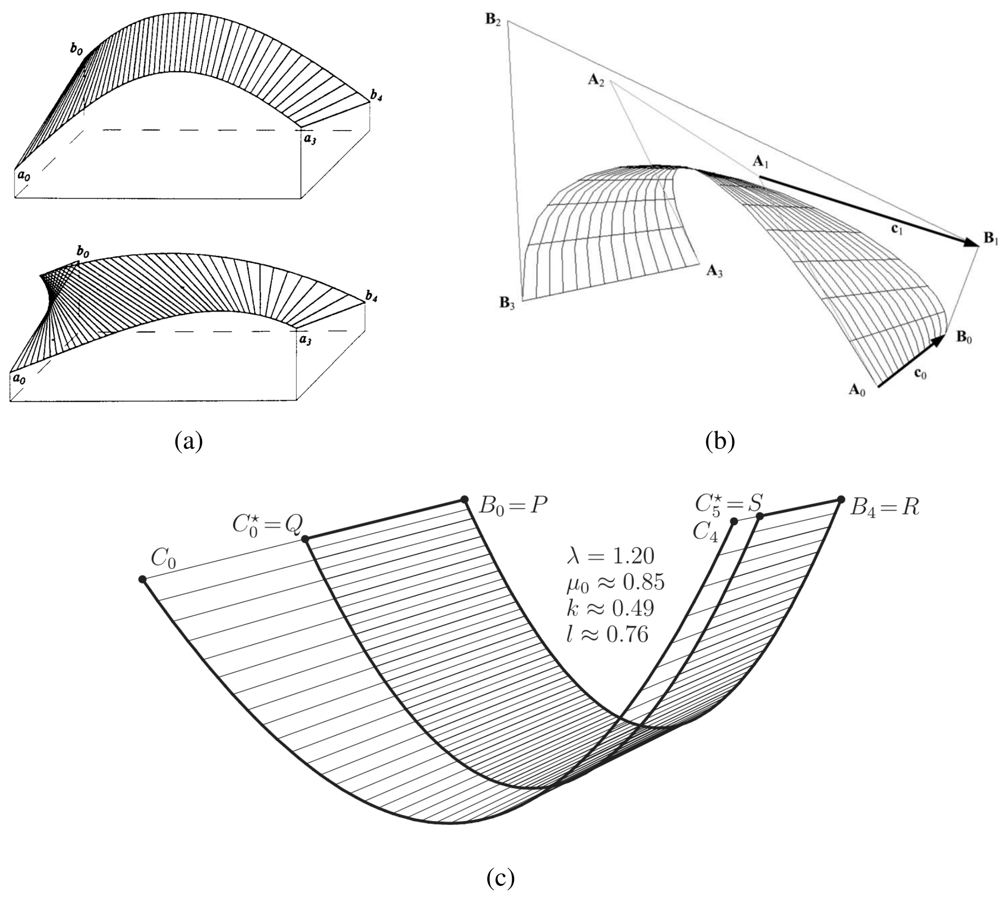


Figure 2.1: Examples of developable Bézier surfaces that are: (a) Admissible and non-admissible under the condition given by [1] (b) Cubic and have 5 degrees of freedom using method described in [2]. (c) Free of singular point using algorithm described in [3].

In [1], Aumann gave the necessary and sufficient conditions under which developable Bézier surfaces can be constructed with two boundary curves free of singular points (see Figure 2.1(a)).

As known [25, 24], the surfaces which consist of parts of planes, cylinders, cones and surfaces generated by the tangents of a (in general twisted) curve are developable. Every developable surface is generated by a one parameter set of straight lines ('generators'). Every one of these lines has the property, that there exists one and only one tangent plane for all of its points. So a necessary condition of a Bézier surface for being developable is the existence of a one parameter set of straight lines on the surface [24].

In 1998, Maekawa and Chalfant [26] performed geometric modeling for spline developables for the special case that boundary curves lie in parallel planes.

Chu and Séquin [2] extended the study to the design of developable Bézier surface from two boundary curves. In a way, the authors present a more intuitive design interface while introduced nonlinear constraints on surface. With one boundary curve freely specified, five more degrees of freedom are available for a second boundary curve of the same degree (see Figure 2.1(b)). Therefore it provides more degrees of freedom for design, no limitations in surface modeling, and simpler implementation solutions. However, only solutions for quadratic and cubic cases are derived in this paper and the characterizing equations becomes more complex for boundary curves of low degree.

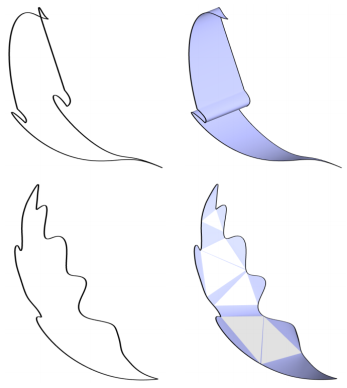
In 2004 [3], Aumann introduced a new algorithm that gives an efficient and simple method for computing a developable Bézier surface of arbitrary degree which is free from singular points (see Figure 2.1(c)). Like the algorithm of Chu and Séquin, this algorithm is also de Casteljau based. Extending on Chu and Séquin's algorithm, it uses three of the five degrees of freedom for a second boundary curve c to fix the starting point C_0 of c. The remaining two allow an easy control of singular points.

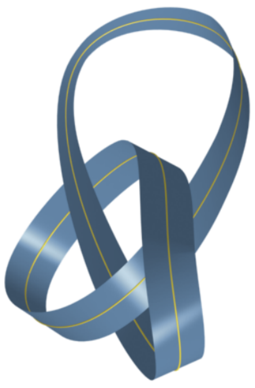
Later in the book by Pottmann and Wallner [27], the authors tried to avoid the non-linear constraints in [2, 3]. Via duality such constraints can be avoided, however it results in a less intuitive plane-based control structure.

2.2.2 Modeling with discrete developables

Trying to avoid the non-linear constraints, architects start leaning towards researches that focus on geometric modeling with discrete developable Bézier surfaces.

Most of recent works approach the problem from the perspective of design. Bo and


(a)


(b)

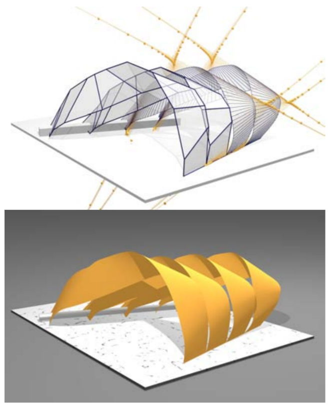

(c)

Figure 2.2: Examples of Computer Aided Geometric Design research results: (a) Developable paper petals and leaves design for a tulip using the sketch based system described in [4]. (b) Developable surfaces of a closed paper strip generated from geodesic curves as shown in [5]. (c) Control structures and the approximating developable surface using the algorithm introduced in [6].

Wang [5] present a new way of modeling developable surface to simulate paper bending. This method exploits the underlying geodesic on the developable surface. By manipulating the geodesic, the authors were able to provide shape control for modeling paper bending shapes as shown in Figure 2.2(a). Although their approach was flexible, the resulting design is limited to paper composition shapes. Therefore, designing a free-form model with this technique still presents a challenge.

In the same year, Rose et al. [4] introduced a more designer-friendly approach which essentially takes in any polyline 3D boundary and generates the developable surface that interpolates the boundary. Figure 2.2(b) shows the polyline and interpolated developable surface.

There is also a commercial software that is designed by Paul Haeberli called Lamina Design. It is used to convert any given 3D surface to a set of developable panels [28]. Lamina builds the physical structure by approximating the 3D shape with various 2D assembly parts. The design mesh is approximated by a number of 2D parts that are numerically cut and attached to fabricate the final structure.

A few of recent researchers have incorporated well-known geometric modeling algorithms such as subdivision to approximating the design. In [6], Liu et al. showed a new approach of constructing and approximating design using conical meshes which are quadrilateral meshes with planar faces as shown in Figure 2.2(c). The authors introduced conical meshes' geometry properties and demonstrated their superiority over other types of meshes from the perspective of architectural design. One property that makes it particularly useful for layer composition constructions is that when offsetting the conical mesh face planes by a constant distance, it yields a planar mesh of the same connectivity, which is again a conical mesh. The authors also proposed a new algorithm for converting quadrilateral mesh to conical mesh.

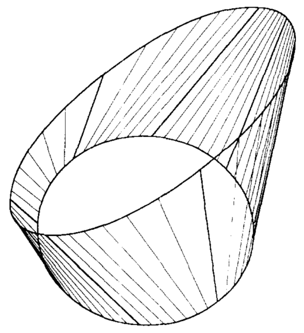
Extending on the ideas of [6], Pottmann et al. [2008] use splines for modeling developable surfaces. They use optimization to achieve approximate developability, using integrals for target functionals.

2.2.3 Modeling developables with the dual representation

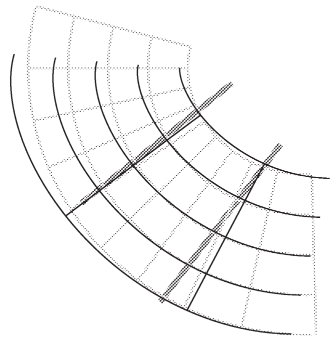
A different approach to modeling developables is to work with the dual representation. Here a developable is represented as the envelope of its tangent planes and is thus recognized as the projective dual of a space curve.

The dual representation was firstly proposed by R.M.C. Bodduluri and B. Ravani [29]. They presented a direct explicit representation for developable surfaces that is CAD-oriented, using the Bézier and B-spline bases. By examining geometric duality between points and planes, the authors consider a single-parameter family of planes (whose envelope is the developable surface) as the dual of a curve with a Bézier or B-spline form.

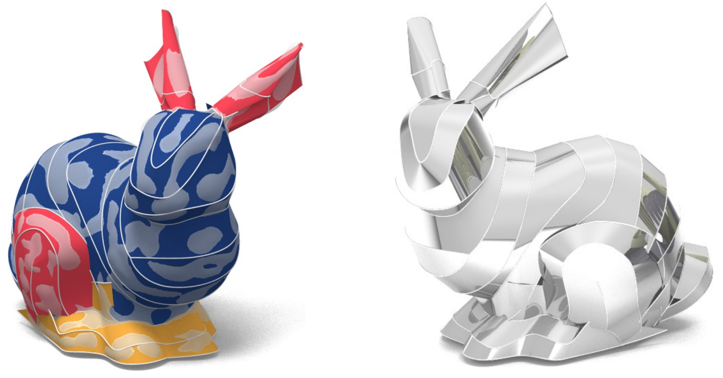
Pottmann and Farin [7] extended the results of Bodduluri and Ravani [29] in various



(a)



(b)



(c)

Figure 2.3: Examples of Approximated developable surfaces in the context of dual representation: (a) Developable piecewise (2, 1)-surface as approximation of a developable surface which connects a circle and an ellipse [7]. (b) Approximation of a developable surface (light grey) by a developable spline surface [8]. (c) Piecewise-developables which approximate bunny using Tang et al.'s algorithm [9]. Composite surfaces and the rendered result are shown.

ways: First, arbitrary NURBS representations are considered and a rigorous projective treatment are provided. They also showed how to convert developable NURBS surfaces from the dual form into a standard tensor product point representation. In particular, they were trying to solve basic design problems such as connecting two curves with a developable surface (see Figure 2.3(a)).

A separate contribution [30] by Hoschek and Pottmann in 1995 describes the fundamentals and the main ideas on interpolation and approximation with developable surfaces. They also showed some initial results on this topic.

Since then, Pottmann and Wallner [8, 27] have done extensive study on approximation algorithm for developable NURBS surfaces (see Figure 2.3(b)). The dual representation, however, is not intuitive and it is difficult to control singularities.

Most recently, Tang et al. [9] present a new approach to o geometric modeling with developable surfaces. They use a combined primal-dual spline representation for developables. They represent developables as splines and express the nonlinear conditions relating to developability and curved folds as quadratic equations. An approximated bunny is shown in Figure 2.3(c).

2.2.4 Other approaches

There has also been interest in developable surfaces for mesh parametrization [31] and mesh segmentation [32].

In [33], Massarwi et al. introduced an algorithm for approximating a 2-manifold 3D mesh by a set of developable surfaces. The approximation quality is controlled by user. The approximation output is a developable surface which is a generalized cylinder represented as a strip of triangles not necessarily taken from the original mesh.

2.2.5 Practicality of fabrication

With the design and construction of more and more unusually shaped buildings, the computer graphics community has started to explore new methods to reduce the cost of the physical construction for large shapes.

A key problem with construction of free-form shapes is the approximation of the design surface by a union of panels, which can be manufactured with a selected technology at reasonable cost. Surface approximation by planes is originally introduced by Cohen-Steiner et al. [34]. Since then, various extensions have been proposed for additional surface types, e.g., spheres and cylinders [35], quadrics [36], or developable surfaces [31]. These methods focus on optimizing for surface segmentation.

There exist an extensive amount of research on unfolding with planar panels. In [37], they looked into the planar quadrilateral facets for the Jerusalem Museum of Tolerance project by Gehry Partners, in collaboration with Schlaich Bergermann & Partners, engineers. D-Charts [31] is another algorithm developed to segment meshes into developable charts. Fabric and paper based physical structures are produced by using this algorithm.

In 2009, Akleman et al. showed how to create plain-weaving over an arbitrary surface [38]. By cross other cycles (or themselves) by alternatingly going over and under, they are able to create a plain-weaving pattern. They proved that it is possible to create such cycles, starting from any given manifold-mesh surface by simply twisting every edge of the manifold mesh. They have developed a new system that converts plain-weaving cycles to 3D thread structures. Using this system, it is possible to cover a surface without large gaps between threads by controlling the sizes of the gaps.

As an extension of [38], Xing et al. [39] showed that it is always possible to create a single-cycle plain- weaving starting from a mesh on an arbitrary surface. To extend the 2D cycles to 3D threads, they extended the original projection method in [38] to be suitable to handle non-twisted edges too.

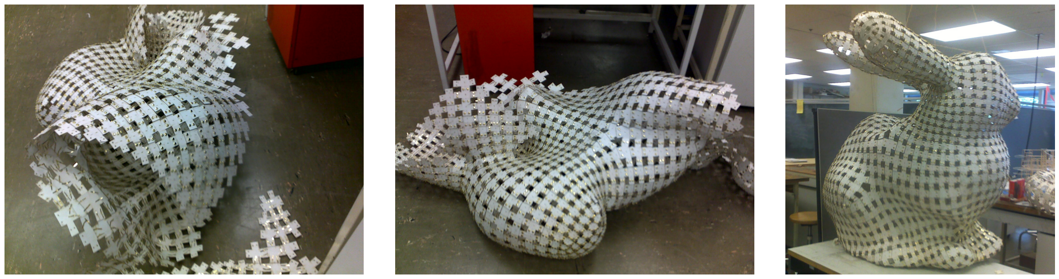


Figure 2.4: Construction of a large bunny using physical version of GRS.

In [40, 41], the authors focused on physical construction using developable components such as thin metals or thick papers. This new approach is based on GRS, which guarantees the topological consistency of the process. By connecting vertex components together, they were able to construct large-scale shape economically.

More recently Akleman et al. [42] introduced a new method for large-scale shape construction. By using quad edge data structure, the authors have developed a system to unfold any polygonal mesh into laser-cutter fitted developable panels. This method is particularly suitable to construct complicated sculptural and architectural shapes from anisotropic materials that can only be bent in one direction.

3. PRELIMINARY

A polygonal mesh consists of a set of vertices, edges, faces and neighboring relations between them [43, 44]. Based on these relations, a data structure defines how each element is stored and what references to its neighborhood it needs. Mesh data structure plays an important role in geometric algorithms, such as mesh adaptation and mesh enhancement [45].

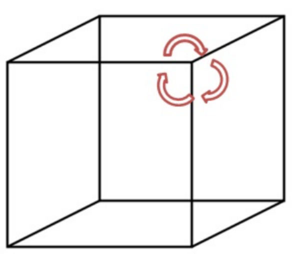
Over the past decades, there has been a surge of interest in mesh data structure in discrete and computational geometry society. Several different major data structures have been implemented for static representation and dynamic handling of arbitrary polygonal meshes [46, 47]. Most of these approaches focus on using mesh data structure in computer geometry and digital modeling, few looks at the problem of bring these data structures to physical world [40].

In this chapter, I will discuss the theoretical framework that this research is built upon, followed by the mathematical background of mesh data structures.

3.1 GRS

In combinatorial mathematics, graph rotation systems encode the embedding of graphs onto orientable surfaces, by describing the circular ordering of a graph's edges around each vertex [48, 49]. Such rotation at vertex is shown in Figure 3.1(a). By tracing the edges, we can identify the face(see Figure 3.1(b)). In [49], the graph rotation system is defined as follows:

Definition. A *rotation* at a vertex v of a graph G is a cyclic permutation of the edge-ends incident on v. A rotation system for a graph G is an assignment of a rotation to every vertex in G.


(a) The rotation of edges at a vertex.

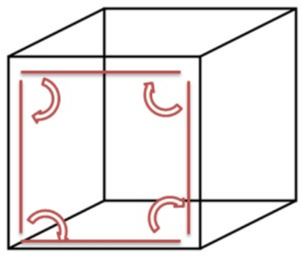

(b) Tracing edges belonging to a face.

Figure 3.1: An illustration of the classical GRS.

Heffter-Edmunds theorem asserts that there is a bijective correspondence between the set of pure rotation systems of a graph and the set of equivalence classes of embeddings of the graph in the orientable surfaces [50]. Hence, this theorem made GRS a powerful tool to guarantee topological consistency which we need to ensure that the resulting physical construction is similar to original virtual shapes.

A graph embedded in an orientable surface corresponds to a rotation system, namely, the one in which the rotation at each vertex is consistent with the cyclic order of the neighboring vertices in the embedding [48]. GRS has been explicitly used as a physical mesh data structure as shown in Figures 2.4 to construct large shapes [40, 41]. GRS have also been implicitly used in computer graphics for representing and manipulating orientable (and non-orientable) 2-manifold surfaces in the guise of various data structures, such as half-edges [45], quad-edges [51], winged-edges [44].

3.2 2D-Thickening

To convert GRS to 3D geometry for physical construction, we will be using a method called 2D-thickening.

As shown in Figure 3.2(a), a finite graph G is embedded in a closed oriented surface S. Each vertex of G thickens to a polygon (or a disk) and each edge OF g thickens to a band (See Figure 3.2(b)). Thus, each polygon corresponds to a vertex and each band corresponds to an edge that connects vertex regions.

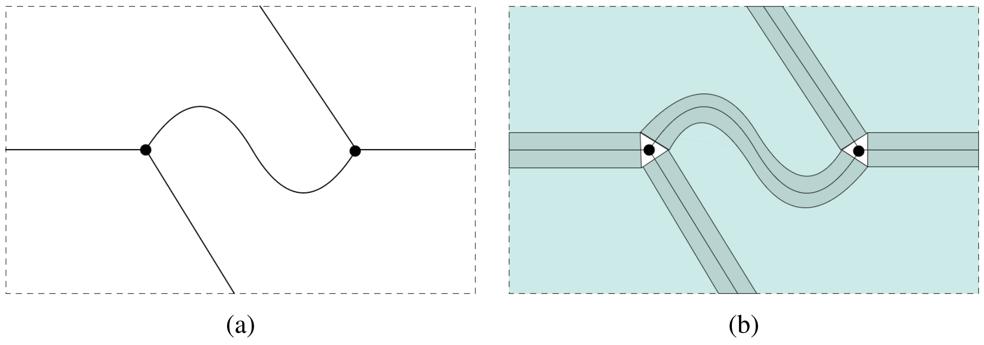


Figure 3.2: An example of the 2D-thickening in topological graph theory. A graph G in a torus and the associated 2D-thickening.

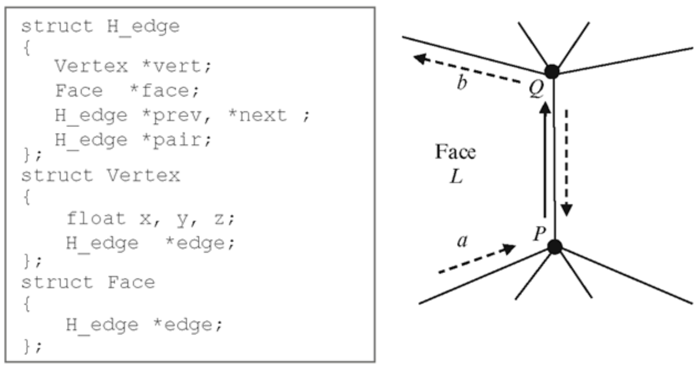
By thickening the graph G, we convert an abstract graph which is embedded on surface to bands that have thickness and size and still represent the same graph G. Hence it is the fundamental method that enables physical construction.

3.3 Half-Edge

A half-edge is a half of an edge and is constructed by splitting an edge down its length. Half-edges are directed and the two edges of a pair have opposite directions.

For Half-Edge mesh data structure, each edge is split into two half-edges as shown in Figure 3.3. Each half-edge stores the vertex Q it points to, the face L it is on, the next half-edge b and prev half-edge a, and the half-edge pair. Note that implementation might vary, here I'm referencing OpenMesh [52].

To employ it onto a mesh, we treat each half-edge as a separate component. For each half-edge, the vertices, half-edges, and face it references to consist a panel that essentially



```
struct H_edge
{
   Vertex *vert;
   Face  *face;
   H_edge *prev, *next ;
   H_edge *pair;
};
struct Vertex
{
    float x, y, z;
    H_edge  *edge;
};
struct Face
{
    H_edge *edge;
};
```

Figure 3.3: Half-Edge data structure and an illustration.

is a face band (see Figure 3.4). And the half-edge pair is the link to other panels so we will need connectors on the border edges for assembly purposes.

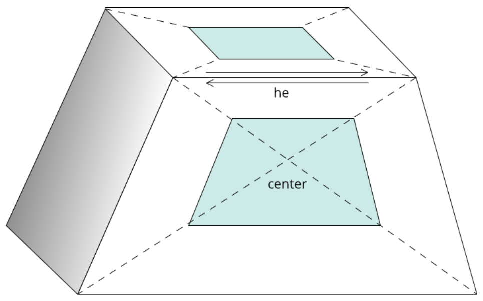


Figure 3.4: 2D-thickened Half-Edge data structure.

3.4 Quad-Edge

For Quad-edge mesh data structure, each edge is split into 4 quad-edges. Each quad-edge of an edge contains pointer to next quad-edge, faces, and neighboring vertices.

Since the edges are the only standard elements in a 2-manifold mesh consisting of two edge-ends and two half-edges [51]. Any edge can therefore be easily thickened to a

```
struct Q_edge
{
   Vertex *start, *end;
   Face   *left,  *right;


};
struct Vertex
{
    float x, y, z;
    Q_edge *edge;
};
struct Face
{
    Q_edge *edge;
};
```

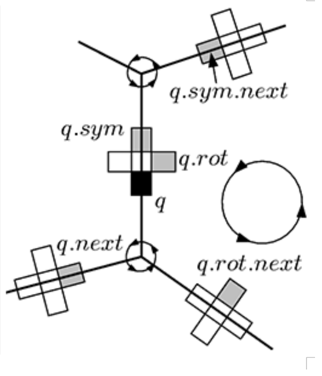


Figure 3.5: Quad-Edge data structure and an illustration.

rectangular edge-band.

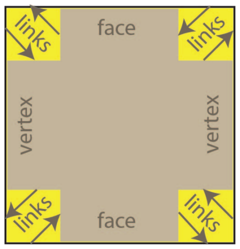
Quad-edge entities are usually depicted as crosses as shown in Figure 3.5 [51]. By encapsulating the cross shape inside of a square (see Figure 3.6(a)), we show that each quad-edge panel is actually a quadrilateral (as the name quad edge suggests). This additional depiction shows that the corners(regions drawn in yellow) of the quadrilaterals can be considered as links. In quad-edge data structure, quad-edge panels that are linked together describe faces and vertices of 2-manifold meshes as shown in Figure 3.6(b). As it can be seen in examples in Figures 3.6(b), closed cycles of directed links form boundaries for faces and vertices. The face and vertex regions are shown as two white squares and six yet-uncompleted cycles in Figure 3.6(b).

Based on this discussion, we can classify any multi-panel construction as quad-edge based, if the panels and construction satisfy the following three conditions:

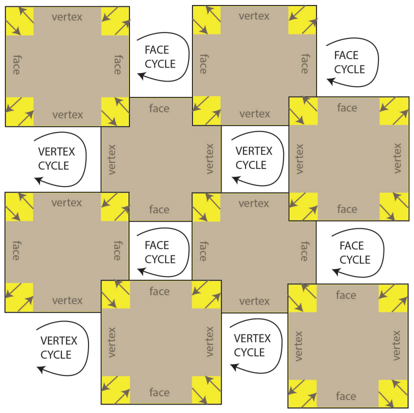
1. Panels must have exactly four corner connections;

2. Only two panels must be connected by each connection; and

3. Every connection must be used.



(a) A single panel



(b) Linked panels

Figure 3.6: Visual examples that demonstrate the concept of quad-edge. (a) provides a depiction of quad-edge as a cross. We have drawn this cross inside a yellow-colored square to demonstrate that each quad-edge element can also be considered a quadrilateral, in which corners correspond to links to other quad-edge panels. These panels are linked to describe faces and vertices of a 2-manifold mesh. (b) shows a group of linked panels.

Hence, to employ it onto a mesh, we created edge bands(dark blue region in Figure 3.7), and we extend the edge band to cover the links(on the corners of edge band) that point to neighboring quad-edges.

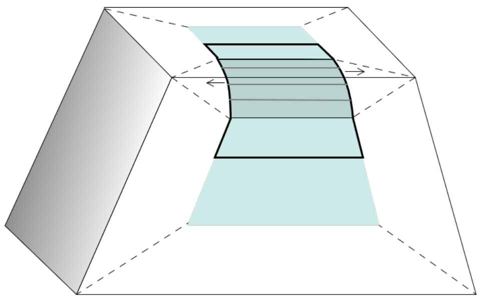


Figure 3.7: 2D-thickened Quad-Edge data structure.

3.5 Winged-Edge

For Winged-edge mesh data structure as shown in Figure 3.8, each edge stores the end vertices P and Q, the left face L and right face R it is on, and the left and right next and prev edges a, b, c, d. It is given such name because of the extra edges it stores.

```
struct W_edge
{
   Vertex *start, *end;
   Face   *left,  *right;
   W_edge *left_prev,  *left_next;
   W_edge *right_prev, *right_next;
};
struct Vertex
{
    float x, y, z;
    W_edge *edge;
};
struct Face
{
    W_edge *edge;
};
```

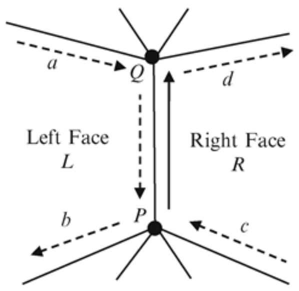
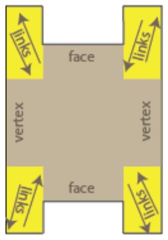


Figure 3.8: Winged-Edge data structure and an illustration.

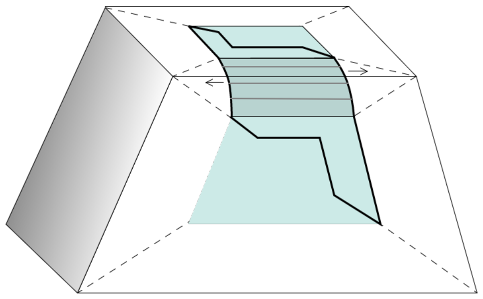
Similar to quad-edge, each edge consists of two edge-ends and two half-edges, therefore they can be thickened to edge-band. With the additional links to neighboring edges, we thicken them to wings. By extending the edge band (see Figure 3.9(a)), we show that each winged-edge panel is actually a quadrilateral with wings. This shows that the wings(regions drawn in yellow) of the quadrilaterals can be considered as links.

3.6 Extended GRS

As mentioned in previous works, GRS can also be used to represent and manipulate non-orientable 2-manifold surfaces. An important concept in GRS is edge twisting. An edge has type 0 if it is untwisted and type 1 if it is twisted [48].

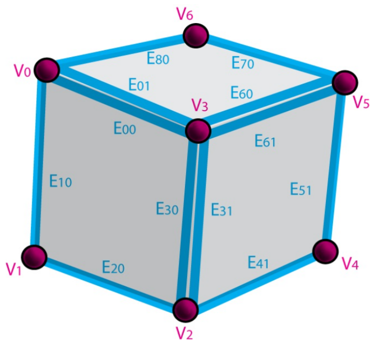


(a) A single panel in 2D

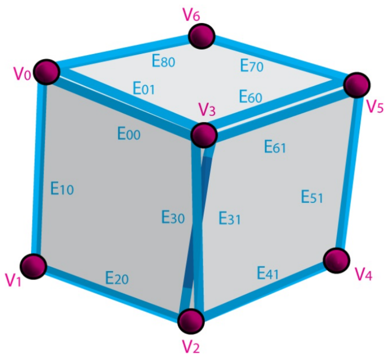


(b) in 3D

Figure 3.9: 2D-thickened Winged-Edge data structure.



(a) The initial mesh whose edges are not twisted



(b) After twisting one edge v_2v_3

Figure 3.10: Face boundary walk before and after twisting an edge.

The concept of edge twisting is extended in [38], the formal definition of extended GRS is described as:

Definition. An *extended graph rotation system (EGRS)* is a graph rotation system with extended edge twists. Note that the face-tracing algorithm can be applied to an EGRS.

To construct a plain-weaving pattern on orientable surface, we apply *Face-Tracing Algorithm* with the extended edge-twisting operations. The following theorem is a foundation for our development of cyclic plain-weaving (see [38] for a proof of the theorem):

Theorem. Let $\rho_0(G)$ be a graph rotation system with no twisted edges, which corresponds to an embedding of the graph G on an orientable surface S_h. Let A be an arbitrary subset of edges of G. If we twist all edges in A positively, or if we twist all edges in A negatively, then the resulting EGRS induces a cyclic plain-weaving on S_h.

As mentioned above, a fundamental algorithm on graph rotation systems known as *Face-Tracing*. This algorithm applied to a graph rotation system $\rho_0(G)$ on a surface S returns a collection of graph cycles that are the boundary-walks of the faces in $\rho(G)$. For instance, for the face with twisted edge in Figure 3.10, the face boundary walk results in a cyclically ordered set

$$K_1 = \{E_{00}, E_{10}, E_{20}, E_{30}, E_{61}, E_{51}, E_{41}, E_{31}\}$$

Such edge twisting method has been used to construct non-orientable surface virtually [38].

4. UNFOLDING: CONSTRUCTION WITH ORIENTABLE-MANIFOLD MESH DATA STRUCTURE[*]

In this chapter, I will discuss in detail the algorithm of using three orientable-manifold mesh data structures such as Half-Edge, Quad-Edge, and Winged-Edge for construction purposes.

I will also show the process of using these data structures to unfold any polygonal mesh into laser-cut developable panels that can be assembled to construct physical structures that approximate original 3D polygonal meshes.

4.1 Half-Edge

As discussed in Preliminary section 3.3 at page 16, we observe that to employ Half-Edge to mesh using the 2D-thickening method, we can simply create the face bands by scaling the faces(see Figure 4.1). The detailed steps are described as follows:

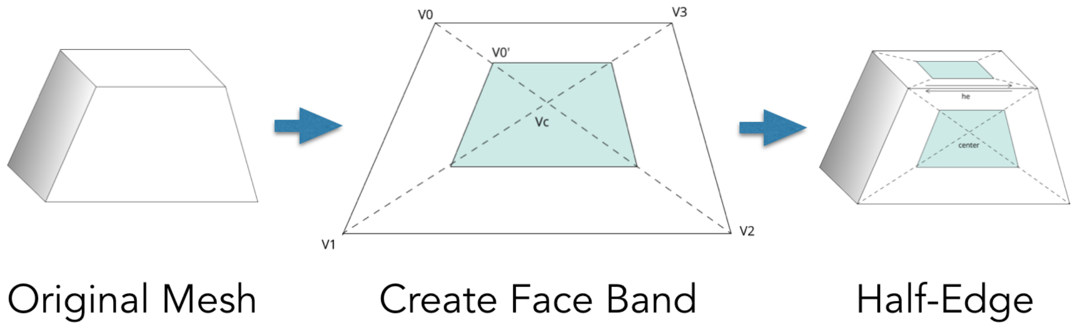


Figure 4.1: Flowchart of construction with Half-Edge.

[*]Part of this chapter is reprinted with permission from "Construction with physical version of quad-edge data structures" by Ergun Akleman, Shenyao Ke, You Wu, Negar Kalantar, AliReza Borhani, Jianer Chen, in *Computers & Graphics*, vol. 58, pp. 172-183, 2016, Copyright 2016 by Elsevier Ltd. DOI link: http://doi.org.ezproxy.library.tamu.edu/10.1016/j.cag.2016.05.008

Step 1: Scale Face.

For each corner vertex v_n on a N-sided face $f = v_0, v_1, ..., v_n$ of the old mesh, create a new vertex v'_n which is scaled towards the face center v_c as in Figure 4.2. The new vertex position is calculated as

$$v'_n = sv_n + (1 - s)v_c \tag{4.1}$$

where s is the scaling factor that controls the thickness of face bands.

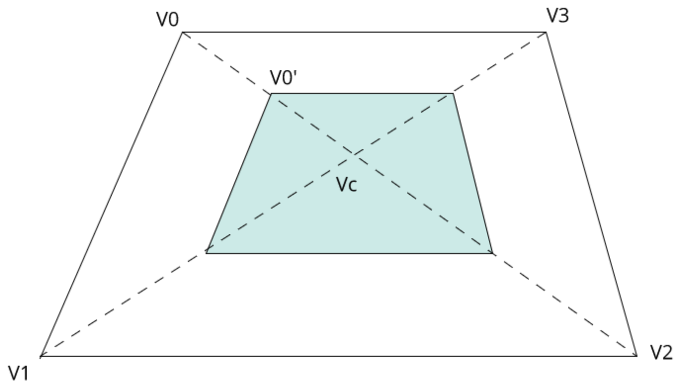


Figure 4.2: Illustration of creating face bands.

Step 2: Create Face Band.

For each face of the old mesh, create a new face (as in Figure 4.2) by connecting all the new vertices $v'_0, v'_1, ..., v'_n$ of the original face f.

4.2 Quad-Edge

As discussed in Preliminary section 3.4 at page 17, we know that to employ Quad-Edge to mesh using the 2D-thickening method, we take in the original mesh, then we create face bands and edge bands, then using face band as a reference, we create flaps to cover links(connectors). We use the same face band generating algorithm as Half-Edge.

A flowchart of the whole process is shown in Figure 4.3).

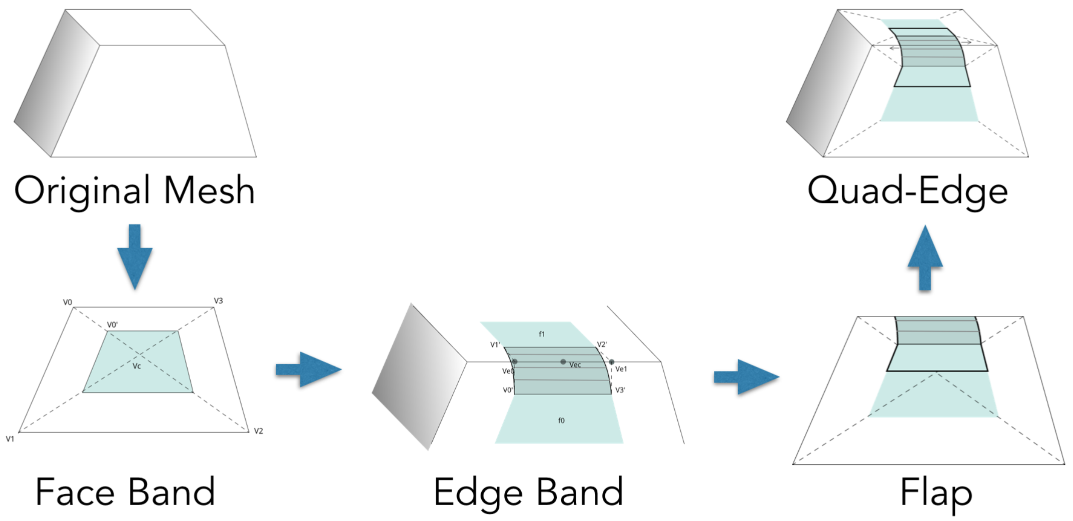


Figure 4.3: Flowchart of construction with Quad-Edge.

The detailed steps are described as follows:

Step 1: Create Face Band.

Scale face and create face bands as described in Step 1 and 2 in section 4.2.

Step 2: Create Edge Band Control Mesh.

For edge bands, we use Bézier surface to preserve the curvature while keeping it developable. The prerequisite of constructing Bézier surface is to create control mesh.

For each edge of the old mesh, create two vertices v_{e0}, v_{e1} which is scaled towards the edge center v_{ec} as shown in Figure 4.4. The new vertex position is calculated as

$$v_{e0} = sv_s + (1-s)v_{ec} \tag{4.2}$$

$$v_{e1} = sv_e + (1-s)v_{ec} \tag{4.3}$$

where v_s, v_e are the original edge endpoints and s is the scaling factor that controls the thickness of edge bands.

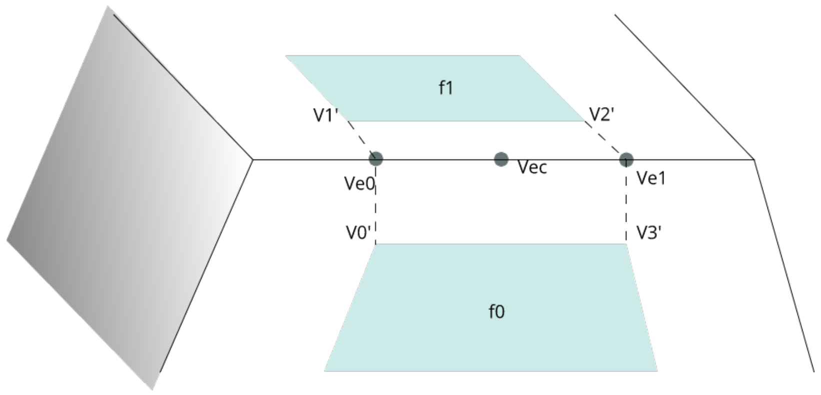


Figure 4.4: Bézier control points on edge.

For each edge of the old mesh, find the neighboring faces' new corner vertices v'_i, v'_{i+1} and v'_j, v'_{j+1} that was scaled from edge endpoints v_e, v_s. Combined with the new edge vertices v_{e0}, v_{e1}, they form the control mesh. For example, in Figure 4.4, v'_1 and v'_2 from f_1, v'_0 and v'_3 from f_0, and v_{e0}, v_{e1} form the control mesh.

Since face band is scaled towards face center, based on *SAS Similarity Criterion*, we derive that the face edges $\overline{v'_1 v'_2}$ and $\overline{v'_0 v'_3}$ are parallel to original edge $v_s v_e$. Hence the control points reside on three parallel lines. Such control mesh gives us a cylindrical Bézier mesh which we prefer.

Step 3: Create Edge Band.

Now using this cylindrical control mesh, we can interpolate the Bézier surface by discretely sampling the two Bézier curves. From the side view (see Figure 4.5), we can see that we have three control points on one side, which forms a quadratic Bézier curve.

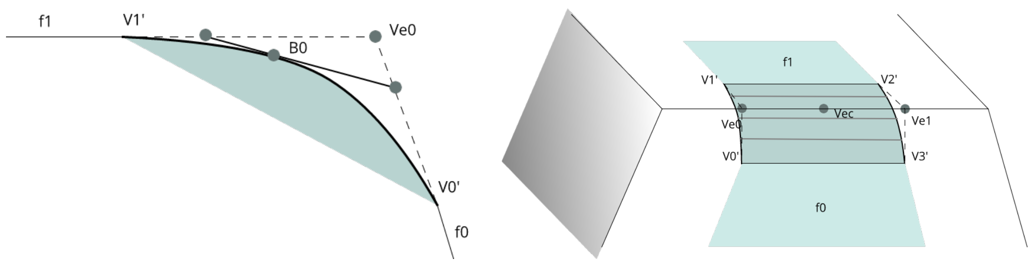
A quadratic Bézier curve is the path traced by the function $B(t)$,

$$B(t) = (1-t)^2 v_0' + 2(1-t)t v_{e0} + t^2 v_1', 0 \le t \le 1. \tag{4.4}$$

where v_0', v_{e0}, and v_1' are control points.

Based on different number of samples, we calculate multiple $B(t)$ positions and create new vertices. Note that the number of samples and the sampling scheme is exactly symmetrical on two sides of the Bézier surface, only by doing this we are able to keep it cylindrical.

Create new faces by connecting these new interpolated vertices in order. Further connect them with face edges, we get the edge band (see dark blue region in Figure 4.5).



(a) Bézier surface (side view).   (b) Bézier surface as edge band.

Figure 4.5: Illustration of creating edge band.

Step 4: Create Flap.

Flap is designed to overlap with neighboring panels. We prefer the flap to be parallel to the edge band and still can be scaled up or down relative to the face band. Hence, the first step is to scale the face band vertices. Similar to creating face bands(see section), we obtain new vertices by scaling inwards

$$v''_n = tv'_n + (1-t)v_c \tag{4.5}$$

where v_c is face center.

As discussed in section 3.4, each Quad-Edge entity is created by encapsulating the cross shape inside of a square. Each Quad-Edge panel is essentially a quadrilateral. Hence the corners of the quadrilaterals can be considered as links.

So we extend the v''_0 and v''_3 outwards to neighboring face band edges and find the intersection of line $\overline{v''_0v''_3}$ with the face band edges $\overline{v'_0v'_1}$ and $\overline{v'_2v'_3}$.

$$\begin{aligned} v_{f0} &= intersect(\overline{v''_iv''_j}, \overline{v'_iv_{i+1}}) \\ v_{f1} &= intersect(\overline{v''_iv''_j}, \overline{v'_jv_{j-1}}) \end{aligned} \tag{4.6}$$

where we're assuming the vertex numbering follows a counter-clockwise order as shown in Figure 4.6.

Once we have the intersection points v_{f0} and v_{f1}, we can create a flap with intersection points and edge band boundary.

4.3 Winged-Edge

As discussed in Preliminary section 3.5 at page 20, we know that to employ Winged-Edge to mesh using the 2D-thickening method, we go through a similar process as Quad-Edge. We use the same face band and edge band generating algorithm as Quad-Edge.

The detailed steps are described as follows:

Step 1: Create Face and Edge Band.

Create face bands and edge band as described in Step 1 to 3 in section 4.2.

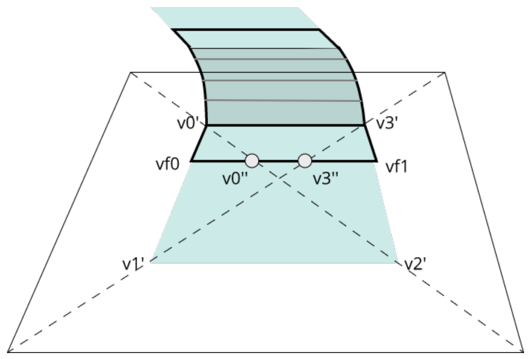


Figure 4.6: Illustration of creating flap on Quad-Edge.

Step 3: Create Flap.

The only thing different from Quad-Edge is the flap. As stated in Preliminary section 3.5, the additional links to neighboring edges can be thicken to wings. By extending the edge band, we create the flaps.

Hence first step is to scale the face band vertices towards face center using equation 4.5. Since the links region are on the wings, we extend the Quad-Edge quadrilateral region of v_0', v_0'', v_3'', v_3' to include also the wing which is v_2'' and v_2'. An illustration is shown in Figure 4.7.

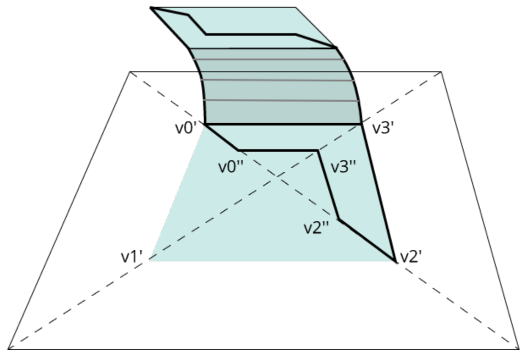


Figure 4.7: Illustration of creating flap on Winged-Edge.

5. WOVEN: CONSTRUCTION WITH NONORIENTABLE-MANIFOLD MESH DATA STRUCTURE

In this chapter, I will discuss the process of using non-orientable-Manifold Mesh Data Structure which is extended GRS(woven) to weave any convex polygonal mesh into laser-cut developable panels that can be assembled to construct physical structures that approximate original 3D polygonal meshes.

I will also show the process step by step of using woven structure for construction purposes.

As discussed in Preliminary section 3.6 at page 20, we observe that we can apply *Face-Tracing Algorithm* with the extended edge-twisting operations to obtain plain-weaving pattern. (see Figure 5.1).

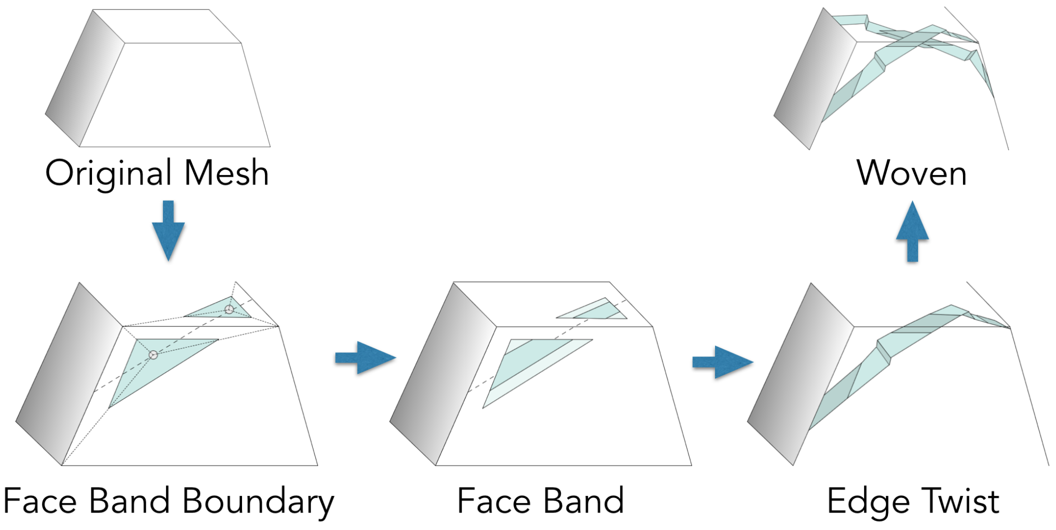


Figure 5.1: Flowchart of construction with Extended GRS(Woven).

The detailed steps are described as follows:

Step 1: Create Face Band Boundary.

To create the face band, first we need to define the boundary. We follow the face tracing algorithm with the edge twisted, and scale the face corners to form the boundary. Since we want the resulting strip to be as straight as possible, we calculate the edge centers V_{ei} and V_{eij}

$$\begin{aligned} V_{ei} &= (V_i + V_{i+1})/2 \\ V_{eij} &= (V_i + V_j)/2 \end{aligned} \tag{5.1}$$

then the scaling pivot V_{ci} is the middle of edge center line

$$V_{ci} = (V_{ei} + V_{eij})/2 \tag{5.2}$$

where V_i, V_j, V_{i+1} are face corner vertices as shown in Figure 5.2. Note that this scheme follows the De Casteljau's algorithm for splitting cubic Bézier curve.

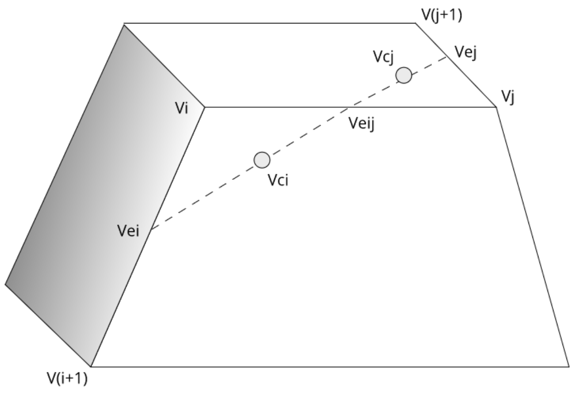


Figure 5.2: Illustration of face band scaling pivot.

After we establish the pivot points, we scale twisted edge and its neighboring edge's vertices v_i towards that pivot (as shown in blue triangle area in Figure 5.3),

$$v_i' = sv_i + (1 - s)v_{ci} \tag{5.3}$$

where s is scaling factor.

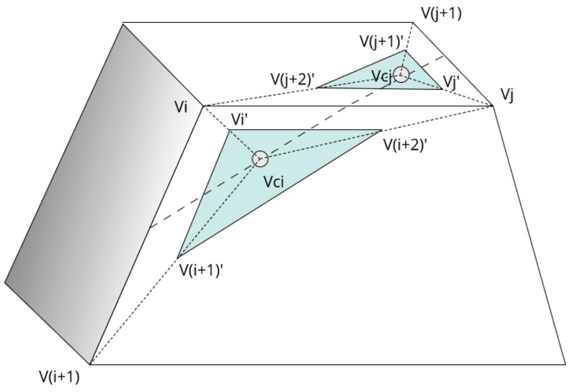


Figure 5.3: Illustration of face band boundary.

Step 2: Create Face Band.

Once we have the boundary, we scale them towards edge center line to create face band (see Figure 5.4).

$$\begin{aligned}
v_{i0} &= tv_i' + (1 - t)v_{ei}' \\
v_{i1} &= tv_{i+1}' + (1 - t)v_{ei}' \\
v_{i2} &= tv_{i+2}' + (1 - t)v_{eij}' \\
v_{i3} &= tv_i' + (1 - t)v_{eij}'
\end{aligned} \tag{5.4}$$

where v_{ei}' and v_{eij}' are the intersection of edge center line $v_{ei}v_{eij}$ and face band boundary.

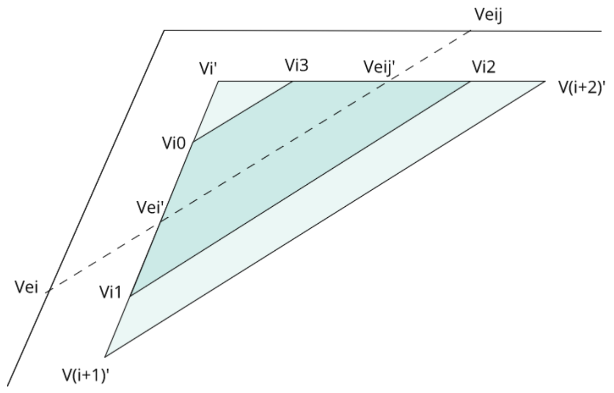


Figure 5.4: Illustration of face band.

Step 3: Create Edge Band.

Edge band is created by connecting neighboring edge pairs. For example, in Figure 5.5, edge band is the dark blue region $\{v_{i3}, v_{i2}, v_{j2}, v_{j3}\}$.

Since face band is scaled towards scaling pivot, based on *SAS Similarity Criterion*, we derive that the face edges $\overline{v_{i3}v_{i2}}$ and $\overline{v_{j2}v_{j3}}$ are parallel to original edge $\overline{v_iv_j}$.

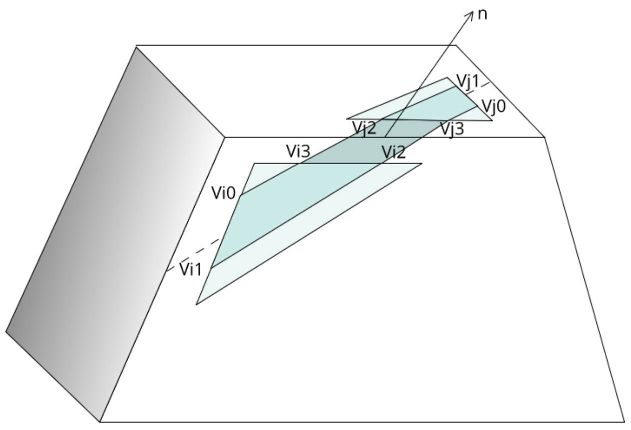


Figure 5.5: Illustration of strip edge patch.

Step 4: Thickness Offsetting.

A strip is formed with 3 edge bands and 2 face bands interlacing together as shown in Figure 5.6. Once we have the strip, we need to offset their position to compensate the thickness of material. We do it by separately offsetting each segment of the strip along the edge patch normal. Hence the middle segment always connect to top segment, then connect to the bottom segment.

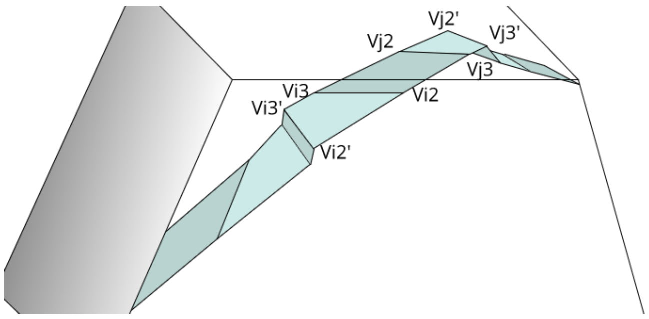


Figure 5.6: Illustration of a strip.

First we calculate the edge band normal of each segment n,

$$\overrightarrow{n} = \overrightarrow{v_{i2} - v_{i3}} \times \overrightarrow{v_{j2} - v_{i3}} \tag{5.5}$$

where $\times$ is cross product.

Then we offset each segment based on normal $\overrightarrow{n}$, a segment is defined as $\{v'_{i3}, v'_{i2}, v'_{j2}, v'_{j3}\}$ where each vertex is calculated as

$$v'_{in} = (v_{in} + v_{in-1})/2 \tag{5.6}$$

where v_{in-1} is the v_{in}'s neighboring vertex in a counter-clockwise naming scheme.

To offset it by thickness r,

$$v''_{in} = v'_{in} + r\,\overrightarrow{n} \tag{5.7}$$

Step 5: Weaving.

Continue the face-tracing algorithm with all the edges twisted, and do Step 1 to 4 on each edge, we obtain the weaving pattern as shown in Figure 5.7.

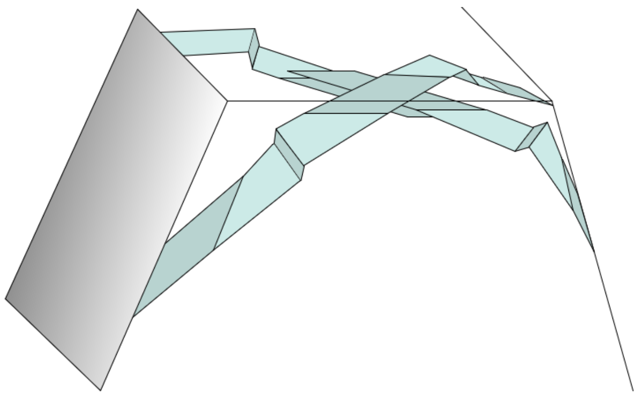


Figure 5.7: Illustration of three strips weaving.

6. IMPLEMENTATION AND RESULTS

6.1 Implementation

The developed software is called Unfolding[53]. It is open sourced on GitHub.

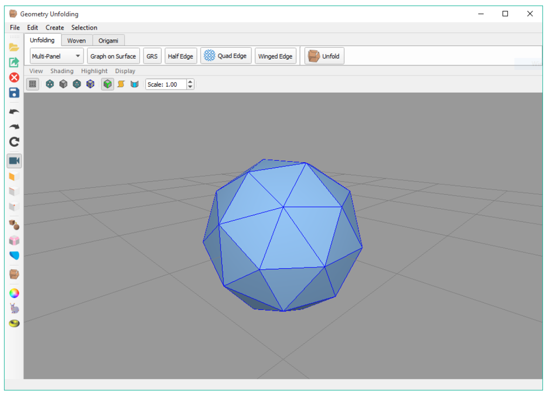


Figure 6.1: A screen-shot of the software interface.

6.1.1 Specifications

The software supports Modern OpenGL(3.3+) and high DPI display. It is written in C++ for maximum efficiency. User interface is developed with Qt framework. It runs on Windows while it is theoretically cross-platform with a few tweaks to conform to MacOS's clang and Linux's gcc compilers.

6.1.2 Software

The development of this system is based on the framework that has been built by Guo [54]. Guo's framework reads in OBJ model and represents them using half-edge mesh structure. The unfolding module which projects 3D developable panels to 2D planar pieces and the rendering module which handles graphics display are implemented by a fellow graduate student.

Since we use Qt to handle the user interface, it allows a great range of freedom as for the design. Hence it can as technically detailed or as user-friendly as it can be.

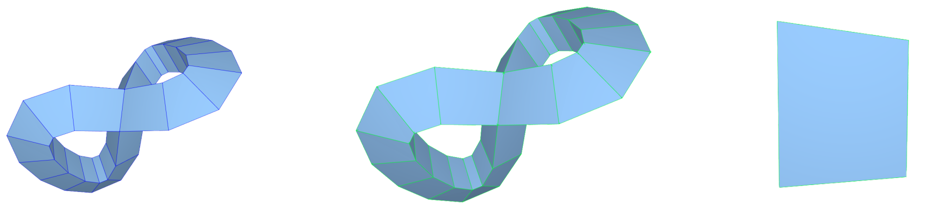
For the purpose of serving mainly as a research and educational software, the User Interface is straight-forward and research-driven. Exact terms are used on the label, such as Half-Edge and Quad-Edge. However, we hide the unnecessary complex mathematical variable names from the user. For example, instead of "degree of curve", we use the notion of "sample", which conveys the same information and is much easier to understand.

6.2 Results

In this section, I will show the software outputs and the physical constructed pieces. Construction details will also be discussed.

6.2.1 Results of Half-Edge

Figure 6.2 shows the screenshots of the process of constructing a genus-2 quadrilateral manifold with Half-Edge. Figure 6.2(a) shows the original mesh, Figure 6.2(b) is the generated developable surface, Figure 6.2(c) shows one panel. The face band scale here is 100 %.

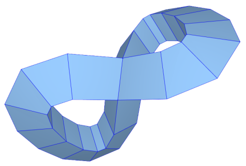


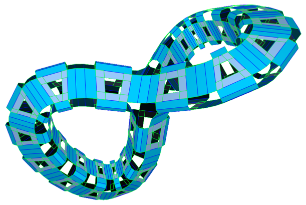
(a) Original Mesh (b) Developable surface (c) Single Panel

Figure 6.2: Constructing a genus-2 quadrilateral manifold with Half-Edge.

6.2.2 Results of Quad-Edge

Figure 6.3 shows the screenshots of the process of constructing a genus-2 quadrilateral manifold with Quad-Edge. Figure 6.3(a) shows the original mesh, Figure 6.3(b) is the generated developable surface, Figure 6.3(c) shows one panel. The face band scale here is 50 %, flap scale is also 50 %.


(a) Original Mesh


(b) Developable surface

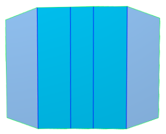
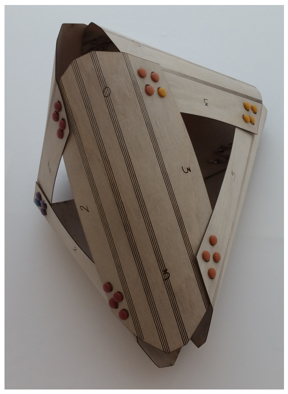
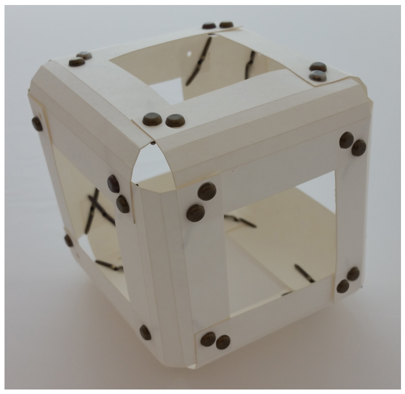

(c) Single Panel

Figure 6.3: Constructing a genus-2 quadrilateral manifold with Quad-Edge.


Tetrahedron


Cube

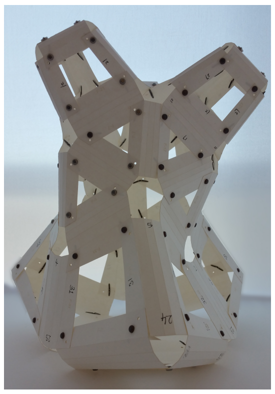
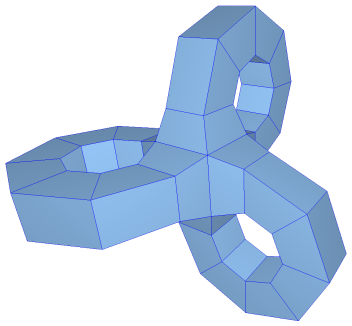
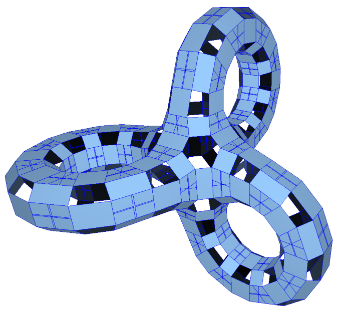

Concave

Figure 6.4: Platonic solids and concave shapes constructed with Quad-Edge.

To test the construction process, we started with Platonic solids, then moved on to concave designs(see Figure 6.4). We have also tested it on high-genus meshes as shown in Figure 6.5.


(a) 3-genus mesh


(b) Developable surface

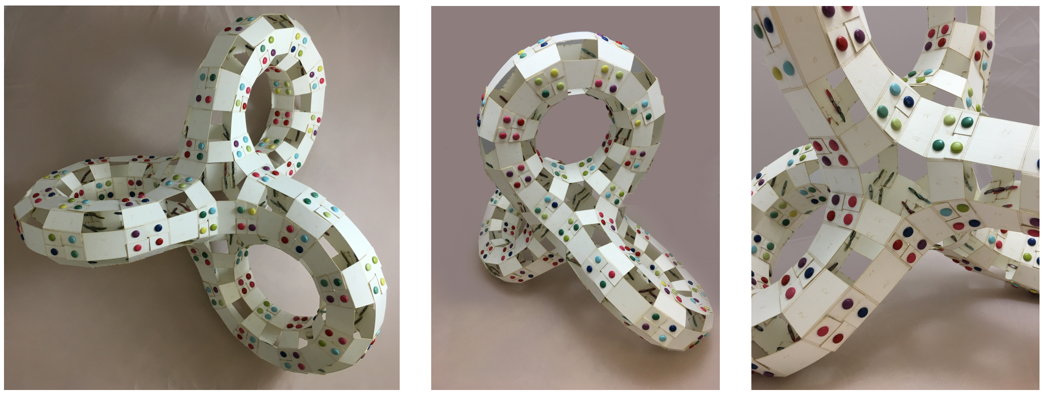
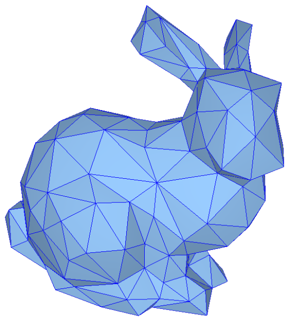
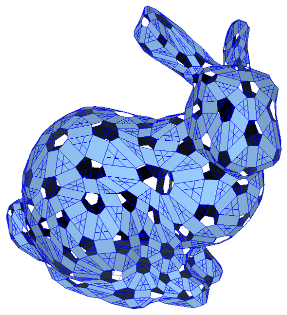
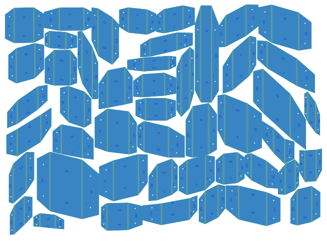

(c) Constructed with Quad-Edge

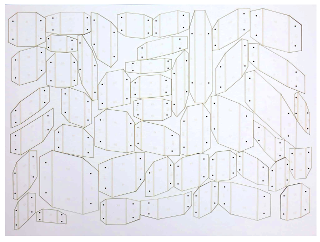
Figure 6.5: 3-genus object constructed with Quad-Edge.

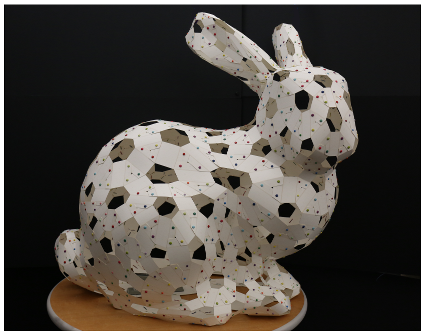
To demonstrate our method's advantages when building large scale shapes, we have also built a Stanford bunny which is approximately 4 feet tall. We start by simplifying the original bunny mesh to 625 edges (see Figure 6.6(a)). Then we use Quad-Edge to generate the developable panels(see Figure 6.6(b) and (c)), laser-cut and assembled them together(see Figure 6.6(d) (e) and (f)).


(a) Simplified Stanford bunny


(b) Generated developable bunny


(c) Screenshot of panels


(d) Laser-cutted panels


(e) Constructed bunny

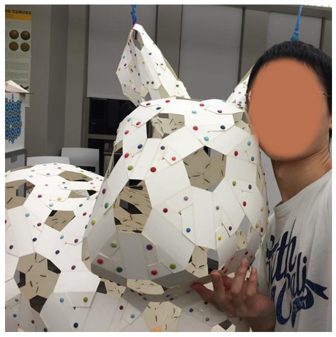

(f) Size compared to human

Figure 6.6: The process of constructing Stanford bunny. Approximately 600+ panels are separated to 12 sheets of 18x24 inch Strathmore heavyweight papers. Laser cutting took about 6 hours to complete. 4 people assembled it in 12 hours, using approximately 1200 fasteners. The total cost for all the material is approximately $60.

Our research received interests of Professor Negar Kalantar Mehrjardi and Professor Alireza Borhani Haghighi in the Architecture Department of Texas A&M University. A group of students used our software in a beginning level architecture studio to create and construct their own designs. Figure 6.7 shows a collage of student works.

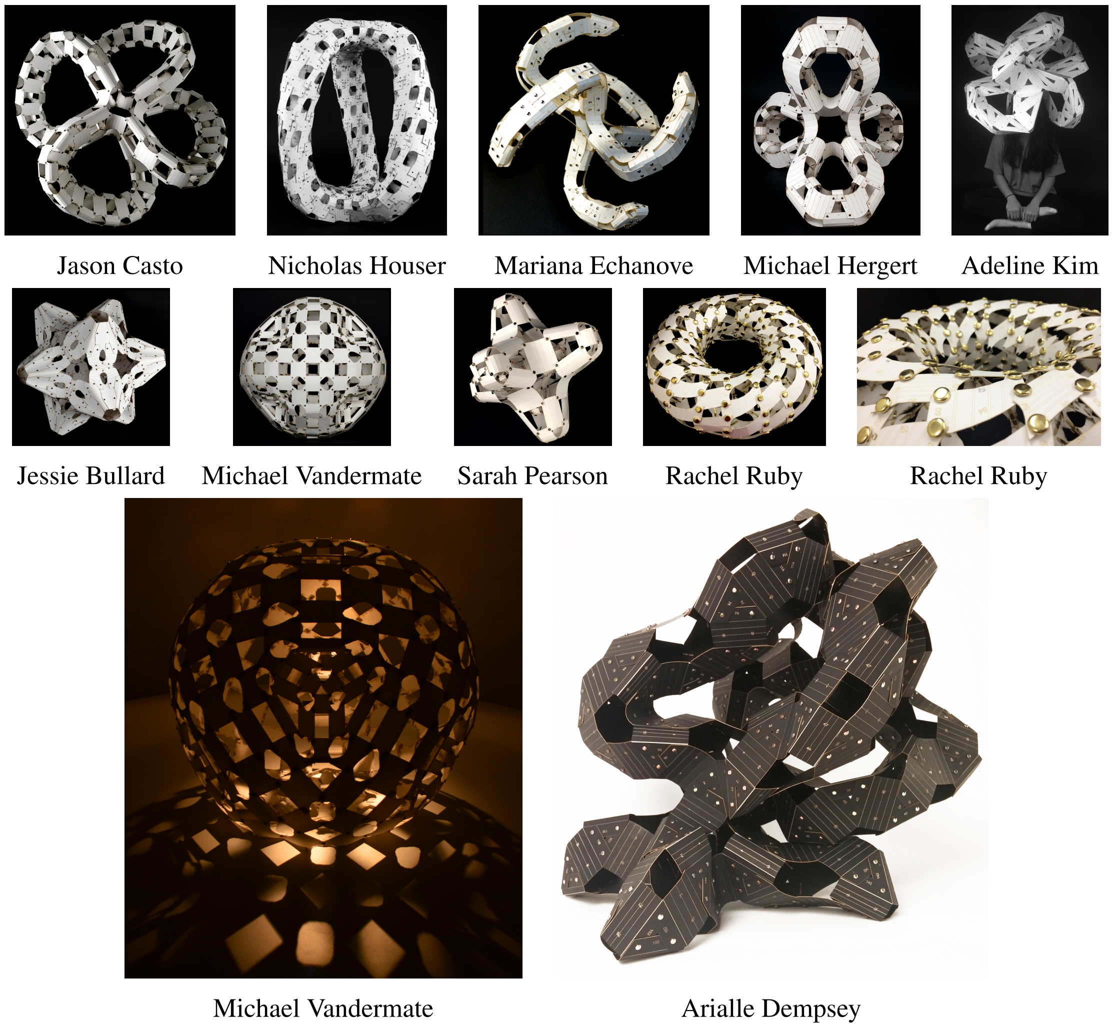
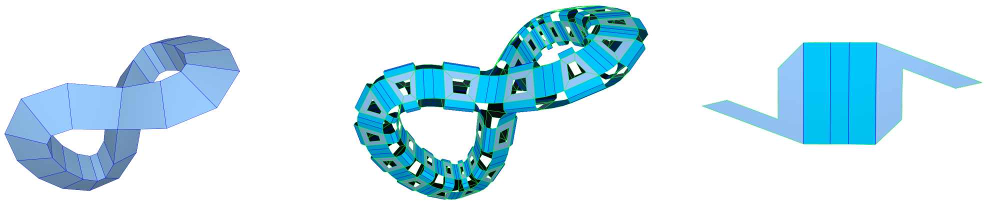


Figure 6.7: Examples of quad-Edge sculptures designed and constructed by students in a beginning level architecture studio. The names of students who designed and constructed sculptures are provided under each photograph.

6.2.3 Results of Winged-Edge

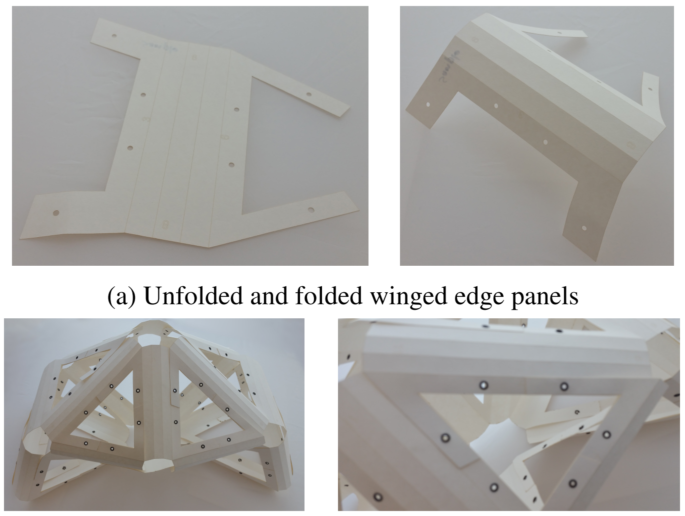
Figure 6.8 shows the screenshots of the process of constructing a genus-2 quadrilateral manifold with Winged-Edge. Figure 6.8(a) shows the original mesh, Figure 6.8(b) is the generated developable surface, Figure 6.8(c) shows one panel. The face band scale here is 50 %, flap scale is also 50 %, shift is set to left.



(a) Original Mesh (b) Developable surface (c) Single Panel

Figure 6.8: Constructing a genus-2 quadrilateral manifold with Winged-Edge.

We have also tested the process of constructing concave shapes with Winged-Edge. We expected because of its wings, it can provide a stronger structure. However, as soon as we build one, we realized that it is hard to put together. Another problem is that winged-edge panels do not pack well because of its wings (see Figure 6.9).



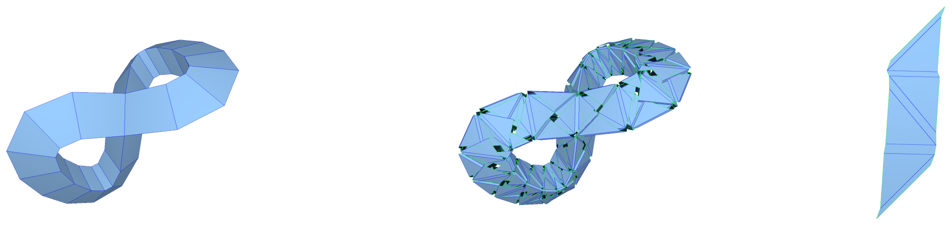
(a) Unfolded and folded winged edge panels

(b) A structure constructed by winged edge panels and its detail

Figure 6.9: Examples of models that corresponds to winged-edge mesh data structures. (a) shows flat and bent single panels and (b) shows a shape constructed by using such winged-edge panels and its detail.

6.2.4 Results of Woven

Figure 6.10 shows the screenshots of the process of constructing a genus-2 quadrilateral manifold with Woven. Figure 6.10(a) shows the original mesh, Figure 6.10(b) is the generated developable surface, Figure 6.10(c) shows one panel. The face band scale here is 80 %, thickness offset is 0.1inch.



(a) Original Mesh (b) Developable surface (c) Single Panel

Figure 6.10: Constructing a genus-2 quadrilateral manifold with Woven.

7. CONCLUSION AND FUTURE WORK

In conclusion, I presented a system to solve generic multi-panel unfolding problem. Given any user specified 3D mesh which is constructed using any design interface, this system generates a multi-panel unfold-able developable surface that still preserves the same topology as the original mesh.

In this work, we ignored the thickness of the materials when constructing with Half-Edge, Quad-Edge, and Winged-Edge. To include such an important physical property would be useful for physical construction. Fortunately, there is a simple solution that can be obtained by using two sandwich-layered panels as shown in Figure 7.1. With such sandwich-layers, it is also possible to obtain desired bending by changing the relative sizes of two panels. We are also planning to include this property in our implementation.

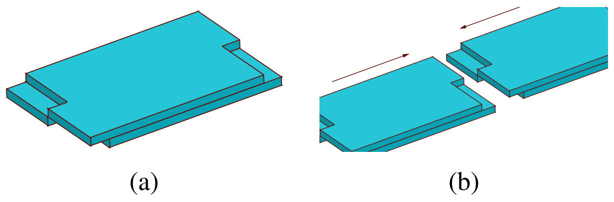


Figure 7.1: Obtaining uniform thickness everywhere using two layers of panels.

Although multi-panel solution has many advantages, single-panel solution can also be very interesting to explore. While multi-panel provides economical and simpler panels in general, single-panel provides more intuitive assembly procedure.

REFERENCES

[1] G. Aumann, "Interpolation with developable Bézier patches," *Computer Aided Geometric Design*, vol. 8, no. 5, pp. 409–420, 1991.

[2] C.-H. Chu and C. H. Séquin, "Developable Bézier patches: properties and design," *Computer-Aided Design*, vol. 34, no. 7, pp. 511–527, 2002.

[3] G. Aumann, "Degree elevation and developable Bézier surfaces," *Computer Aided Geometric Design*, vol. 21, no. 7, pp. 661–670, 2004.

[4] K. Rose, A. Sheffer, J. Wither, M.-P. Cani, and B. Thibert, "Developable surfaces from arbitrary sketched boundaries," in *SGP'07-5th Eurographics Symposium on Geometry Processing*, pp. 163–172, Eurographics Association, 2007.

[5] P. Bo and W. Wang, "Geodesic-controlled developable surfaces for modeling paper bending," in *Computer Graphics Forum*, vol. 26, pp. 365–374, Wiley Online Library, 2007.

[6] Y. Liu, H. Pottmann, J. Wallner, Y.-L. Yang, and W. Wang, "Geometric modeling with conical meshes and developable surfaces," in *ACM Transactions on Graphics (TOG)*, vol. 25, pp. 681–689, ACM, 2006.

[7] H. Pottmann and G. Farin, "Developable rational Bézier and B-spline surfaces," *Computer Aided Geometric Design*, vol. 12, no. 5, pp. 513–531, 1995.

[8] H. Pottmann and J. Wallner, "Approximation algorithms for developable surfaces," *Computer Aided Geometric Design*, vol. 16, no. 6, pp. 539–556, 1999.

[9] C. Tang, P. Bo, J. Wallner, and H. Pottmann, "Interactive design of developable surfaces," *ACM Transactions on Graphics (TOG)*, vol. 35, no. 2, p. 12, 2016.

[10] M. Eigensatz, M. Kilian, A. Schiftner, N. J. Mitra, H. Pottmann, and M. Pauly, "Paneling architectural freeform surfaces," *ACM transactions on graphics (TOG)*, vol. 29, no. 4, p. 45, 2010.

[11] C.-W. Fu, C.-F. Lai, Y. He, and D. Cohen-Or, "K-set tilable surfaces," *ACM transactions on graphics (TOG)*, vol. 29, no. 4, p. 44, 2010.

[12] H. Zimmer, F. Lafarge, P. Alliez, and L. Kobbelt, "Zometool shape approximation," *Graphical Models*, vol. 76, no. 5, pp. 390–401, 2014.

[13] M. Huard, M. Eigensatz, and P. Bompas, "Planar panelization with extreme repetition," in *Advances in Architectural Geometry 2014*, pp. 259–279, Springer, 2015.

[14] N. Dunn, *Digital fabrication in architecture.* Laurence King, 2012.

[15] G. Chryssolouris, D. Mavrikios, N. Papakostas, D. Mourtzis, G. Michalos, and K. Georgoulias, "Digital manufacturing: history, perspectives, and outlook," *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, vol. 223, no. 5, pp. 451–462, 2009.

[16] J. L. Bromberg, *The laser in America, 1950-1970.* MIT Press, 1991.

[17] R. J. Saunders, "Method and apparatus for direct laser cutting of metal stents," July 14 1998. US Patent 5,780,807.

[18] V. Niziev and A. Nesterov, "Influence of beam polarization on laser cutting efficiency," *Journal of Physics D: Applied Physics*, vol. 32, no. 13, p. 1455, 1999.

[19] D. MacFarlane, V. Narayan, J. Tatum, W. Cox, T. Chen, and D. Hayes, "Microjet fabrication of microlens arrays," *IEEE photonics technology letters*, vol. 6, no. 9, pp. 1112–1114, 1994.

[20] G. Ball and W. Morey, "Compression-tuned single-frequency Bragg grating fiber laser," *Optics letters*, vol. 19, no. 23, pp. 1979–1981, 1994.

[21] “Adobe Illustrator.” `http://www.adobe.com/products/illustrator.html`, 2017. [Online; accessed 20-February-2017].

[22] “Autodesk AutoCAD.” `http://www.autodesk.com/products/autocad/overview`, 2017. [Online; accessed 1-March-2017].

[23] L. Ferris, “A standard series of developable surfaces,” *Marine Technology*, vol. 5, no. 1, pp. 53–62, 1968.

[24] J. Lang and O. Röschel, “Developable (1, n)-Bézier surfaces,” *Computer Aided Geometric Design*, vol. 9, no. 4, pp. 291–298, 1992.

[25] H. Brauner in *Lehrbuch der Konstruktiven Geometrie*, Springer, 1986.

[26] J. S. Chalfant and T. Maekawa, “Design for manufacturing using B-spline developable surfaces,” *Journal of Ship Research*, vol. 42, no. 3, pp. 207–215, 1998.

[27] H. Pottmann and J. Wallner, *Computational line geometry*. Springer Science & Business Media, 2009.

[28] “Lamina Design.” `http://laminadesign.com/index.html`, 2017. [Online; accessed 1-March-2017].

[29] R. Bodduluri and B. Ravani, “Design of developable surfaces using duality between plane and point geometries,” *Computer-aided design*, vol. 25, no. 10, pp. 621–632, 1993.

[30] J. Hoschek and H. Pottmann, “Interpolation and approximation with developable B-spline surfaces,” *Mathematical methods for curves and surfaces*, pp. 255–264, 1995.

[31] D. Julius, V. Kraevoy, and A. Sheffer, “D-charts: Quasi-developable mesh segmentation,” in *Computer Graphics Forum*, vol. 24, pp. 581–590, Wiley Online Library, 2005.

[32] H. Yamauchi, S. Gumhold, R. Zayer, and H.-P. Seidel, "Mesh segmentation driven by Gaussian curvature," *The Visual Computer*, vol. 21, no. 8, pp. 659–668, 2005.

[33] F. Massarwi, C. Gotsman, and G. Elber, "Papercraft models using generalized cylinders," in *Computer Graphics and Applications, 2007. PG'07. 15th Pacific Conference on*, pp. 148–157, IEEE, 2007.

[34] D. Cohen-Steiner, P. Alliez, and M. Desbrun, "Variational shape approximation," in *ACM Transactions on Graphics (TOG)*, vol. 23, pp. 905–914, ACM, 2004.

[35] J. Wu Leif Kobbelt, "Structure recovery via hybrid variational surface approximation," in *Computer Graphics Forum*, vol. 24, pp. 277–284, Wiley Online Library, 2005.

[36] D.-M. Yan, Y. Liu, and W. Wang, "Quadric surface extraction by variational shape approximation," in *International Conference on Geometric Modeling and Processing*, pp. 73–86, Springer, 2006.

[37] J. Glymph, D. Shelden, C. Ceccato, J. Mussel, and H. Schober, "A parametric strategy for free-form glass structures using quadrilateral planar facets," *Automation in construction*, vol. 13, no. 2, pp. 187–202, 2004.

[38] E. Akleman, J. Chen, Q. Xing, and J. L. Gross, "Cyclic plain-weaving on polygonal mesh surfaces with extended graph rotation systems," in *Proc. SIGGRAPH*, vol. 9, 2009.

[39] Q. Xing, E. Akleman, J. Chen, and J. L. Gross, "Single-cycle plain-woven objects," in *Shape Modeling International Conference (SMI), 2010*, pp. 90–99, IEEE, 2010.

[40] Q. Xing, G. Esquivel, E. Akleman, J. Chen, and J. Gross, "Band decomposition of 2-manifold meshes for physical construction of large structures," in *ACM SIGGRAPH 2011 Posters and Talks*, p. 58, ACM, 2011.

[41] E. A. P. Hernandez, S. Hu, H. W. Kung, D. Hartl, and E. Akleman, “Towards building smart self-folding structures,” *Computers & Graphics*, vol. 37, no. 6, pp. 730–742, 2013.

[42] E. Akleman, S. Ke, Y. Wu, A. Borhani, N. Kalantar, and J. Chen, “Construction with physical version of quad-edge data structures,” *Computers & Graphics*, 2016.

[43] E. Akleman and J. Chen, “Guaranteeing the 2-manifold property for meshes with doubly linked face list,” *International Journal of Shape Modeling*, vol. 5, no. 02, pp. 159–177, 1999.

[44] B. G. Baumgart, “Winged edge polyhedron representation,” tech. rep., Stanford University, 1972.

[45] M. Mäntylä, *An introduction to solid modeling*. Computer Science Press, 1988.

[46] M. Botsch, S. Steinberg, S. Bischoff, and L. Kobbelt, “Openmesh-a generic and efficient polygon mesh data structure,” 2002.

[47] T. J. Alumbaugh and X. Jiao, “Compact array-based mesh data structures,” in *Proceedings of the 14th International Meshing Roundtable*, pp. 485–503, Springer, 2005.

[48] J. L. Gross and T. W. Tucker, *Topological graph theory*. Courier Corporation, 1987.

[49] J. L. Gross and J. Yellen, *Graph theory and its applications*. CRC press, 2005.

[50] J. Edmonds, “A combinatorial representation of polyhedral surfaces,” *Notices of the American Mathematical Society*, vol. 7, 1960.

[51] L. Guibas and J. Stolfi, “Primitives for the manipulation of general subdivisions and the computation of voronoi,” *ACM Transactions on Graphics (TOG)*, vol. 4, no. 2, pp. 74–123, 1985.

[52] "OpenMesh." `https://www.openmesh.org/`, 2017. [Online; accessed 13-March-2017].

[53] TopModLab, "Unfolding." `http://github.com/topmodlab/unfolding`, 2017. [Online; accessed 11-February-2017].

[54] P. Guo, E. Akleman, H. Ying, X. Wang, and W. Liu, "Critical points with discrete Morse theory," in *ACM SIGGRAPH 2015 Posters*, p. 67, ACM, 2015.