GRAPH ROTATION SYSTEMS FOR PHYSICAL CONSTRUCTION OF LARGE STRUCTURES

A Dissertation

by

QING XING

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

DOCTOR OF PHILOSOPHY

December 2011

Major Subject: Architecture

GRAPH ROTATION SYSTEMS FOR PHYSICAL CONSTRUCTION OF LARGE STRUCTURES

A Dissertation

by

QING XING

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

DOCTOR OF PHILOSOPHY

December 2011

Major Subject: Architecture

ABSTRACT

Graph Rotation Systems for Physical Construction

of Large Structures. (December 2011)

Qing Xing, B.S., Taiyuan University of Science and Technology;

M.S., Korea Advanced Institute of Science and Technology

Co–Chairs of Advisory Committee: Dr. Ergun Akleman
Dr. Wei Yan

In this dissertation, I present an approach for physical construction of large structures. The approach is based on the graph rotation system framework. I propose two kinds of physical structures to represent the shape of design models. I have developed techniques to generate developable panels from any input polygonal mesh, which can be easily assembled to get the shape of the input polygonal mesh.

The first structure is called plain woven structures. I have developed the “projection method” to convert mathematical weaving cycles on any given polygonal mesh to developable strip panels. The width of weaving strips varies so that the surface of the input model can be covered almost completely. When these strip panels are assembled together, resulting shape resembles to a weaving in 3-space.

The second structure is called band decomposition structures. I have developed a method to convert any given polygonal mesh into star-like developable elements, which we call vertex panels. Assembling vertex panels results in band decomposition structures. These band decomposition structures correspond to 2D-thickening of graphs embedded on surfaces. These band decompositions are contractible to their original graph. In a 2D-thickening, each vertex thickens to a polygon and each edge thickens to a band. Within the resulting band decomposition, each polygon corresponds to a vertex and each band corresponds to an edge that connects two vertex polygons.

Since the approach is based on graph rotation system framework, the two structures do not have restrictions on design models. The input mesh can be of any genus. The faces in the input mesh can be triangle, quadrilateral, and any polygon. The advantages of this kind of large physical structure construction are low-cost material and prefabrication, easy assemble. Our techniques take the digital fabrication in a new direction and create complex and organic 3D forms. Along the theme of architecture this research has great implication for structure design and makes the more difficult task of construction techniques easier to understand for the fabricator. It has implications to the sculpture world as well as architecture.

To my family and friends.

ACKNOWLEDGMENTS

I would like to thank my committee chairs, Professor Ergun Akleman and Professor Wei Yan, and my committee members, Professor Weiling He, Professor Gabriel Esquivel, and Professor Jianer Chen, for their guidance and support throughout the course of this research. I would like to thank Professor Scott Schaefer for substituting Professor Chen in my final dissertation defense. I appreciate the helpful comments and suggestions that were kindly provided by Professor Schaefer. Thanks also go to my friends and colleagues, and the faculty and staff, for making my time at Texas A&M University a great experience.

I also want to extend my gratitude to the students in the Department of Architecture at Texas A&M University, who participated in the research. Michael Tomaso and Ryan Collier used our weaving software and provided helpful feedback. Shiyu Hu helped me with the .eps file output and built the soccer ball band decomposition model with me. Ronny Eckels, Lauren Wiatrek, Brent Gohmert, Tim Durbin, Catlan Fearon, and Kristy Lee worked on the bunny project. Their contributions are gratefully appreciated.

Finally, thanks to my parents who have been an ongoing source of encouragement and support, and to my friends for their inspiration and help.

TABLE OF CONTENTS

LIST OF FIGURES

FIGURE Page

FIGURE Page

CHAPTER I

INTRODUCTION AND MOTIVATION

Freeform structures are one of the most striking and active trends in contemporary architecture. This direction has been pioneered by architecture firms such as Gehry Partners and Foster & Partners who exploit the utilization of digital technologies in computer graphics to architectural design and construction.

Current construction process of such unusual freeform structures is called "panelization" [59], since prefabricated components, or semi-finished parts, are used to assemble walls, panels, and ceilings [13]. Panelization process consists of 5 stages:

1. **Surface Design:** The architect designs a freeform surface in virtual space with computed-aided design software, such as *Maya* [46].

2. **Mesh Design:** For physical construction of the structure, an team of engineers converts this initial virtual surface to a polygonal mesh that approximate the shape of surface.

3. **Frame Construction:** For the initial stage of physical construction, a team of construction workers built a steel frame along the edges of that polygon mesh on the site.

4. **Panel Fabrication:** The faces of that mesh are called "panels". The panels are prefabricated off-site and then transported to the construction site. These panels can be curved or planar.

This dissertation follows the style of *IEEE Transactions on Automatic Control.*

5. **Panel Installation:** Workers install the panels into the steel frame based on the connectivity information.

Examples of physical structures that are built by using curved panels include Convention Center Disney Concert Hall in Los Angles, Art Gallery of Ontario in Toronto, Swiss Re Tower in London, and Milan Exhibition Hall. Although the use of such prefabricated panels has a number of advantages for the simplification of the construction, the number of distinctly shaped panels can be very high since each prefabricated panel can have a unique shape on the curved surface. This is a problem since the fabrication of huge number of distinctly curved panels can be very expensive. Recently, a method to reduce the number of curved panels have been introduced to reduce the construction cost [21].

In some designs, it is desirable to use planar panels such as glass. One of the most famous examples of such planar panels is a spectacular glass roof covering the court of British Museum, which is designed by Foster + Partners. Recently, methods are developed to reduce the number of different shaped planar triangles and quadrilaterals [25, 58]. Since fabrication of such planar panels is relatively cheaper than fabrication of curved panels, the reduction of the number of differently shaped components is not as essential as in the case of curved panels. The biggest problem with planar panels is to match the thickness of planar components. It was recently shown that planar panels must satisfy conical mesh property to have a consistent thickness. Conical mesh property requires that for every given vertex all panels that share that vertex must be tangent to the same cone.

In this dissertation, I consider even simpler panels, which are cut from extremely thin surfaces, such as thin metal sheets or papers. Such panels can be planar or can be curved like cylinders and cones. Such surfaces, which are called developable surfaces, are especially useful since Gaussian curvature is zero everywhere. This property can allow us construct an approximation of a given surface. Moreover, if we construct a structure using developable panels such as thin metal sheets or papers, even if each panel is different in shape, they can be manufactured fairly inexpensively by cutting large sheets of thin metals or paper using laser-cutters. Other big expenses for construction of such large shapes come from the transportation of the curve panels. Since developable surfaces can be unfolded/flattened to planar surfaces, it is easy to stack and transport them.

In this dissertation, I present a new approach for economical design and construction of unusually shaped organic structures with developable panels. One of the issues with laser cutting and the fabrication of objects is that the objects tend to be flat, simplistic and angular due to the output technology. Our new approach takes the technology in a new direction and creates complex and organic 3D forms. Along the theme of architecture this research has great implication for structure design and makes the more difficult task of construction techniques easier to understand for the fabricator. It has obvious implications to the sculpture world as well as architecture. The methods I have developed for simplification of construction process is based on our research group's theoretical work on topological graph theory and relating topological graph theory to geometry using Gauss-Bonnet theorem. I have developed two techniques to automatically create easily assembled developable panels from any given manifold mesh.

- *Plain Woven Structures:* The first technique creates developable strips that can form woven structures [68]. I have developed projection method to create these weaving strips from any given extended graph rotation system, which describes a cyclic weaving [2, 6, 69, 69].

- *Band Decomposition Structures:* The second technique creates star like developable panels which we call vertex components. These are obtained by a method of 2D-thickening of a graph rotation system (GRS) [8]. Using these vertex components, one fabricator can construct structures that corresponds to the 2D-thickening of the graph rotation system embedded as a polygonal mesh of the deigned 3D model in virtual space.

Both of these structures are guaranteed to work based on the fundamental Heffter-Edmunds theorem of Topological Graph theory [30], which 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. As a direct consequence of the Heffter-Edmunds theorem, in order to assemble the structure all the work that the construction workers have to do is to attach the corresponding edge- ends of vertex components. Once all the components are attached to each other, the whole structure will correctly be assembled.

Gauss-Bonnet theorem, moreover, asserts that the total Gaussian curvature of a surface is the Euler characteristics times two π. Since the structure is made up only of developable panels, Gaussian curvature is zero everywhere on the solid parts. The Gaussian curvature happens only in the empty regions (holes) and that are determined uniquely. Hence, we can correctly form Gaussian curvature of holes. The structures can always be raised and formed in 3-space.

I also developed a method to simplify finding corresponding pieces among a large number of developable panels. It is particularly important to simplify the assembling process such that the panels can be assembled with a minimum instruction by the construction workers who do not need to have extensive experience. One of the main tasks left for the construction workers is the identification and finding corresponding

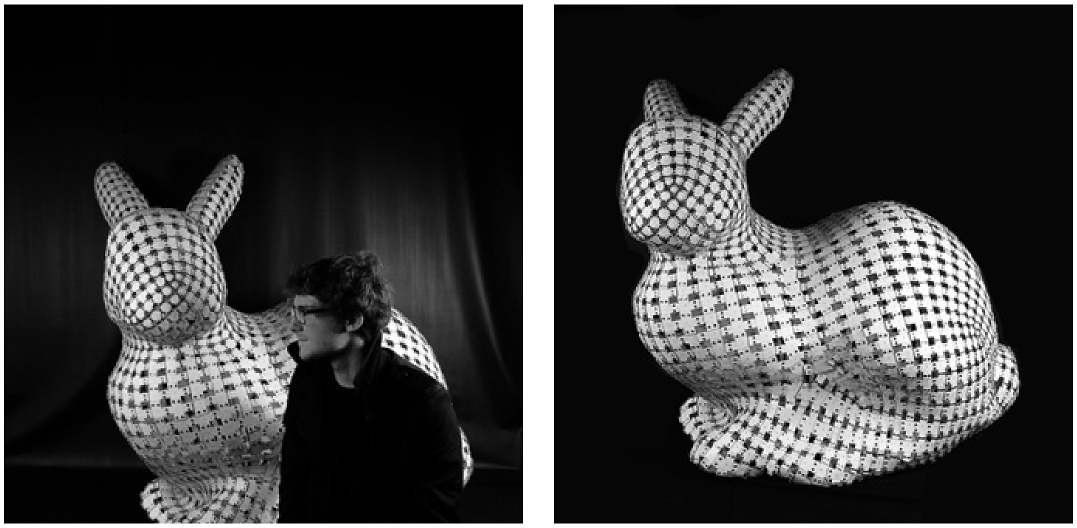


Fig. 1. This large sculpture of Bunny is constructed with laser-cut poster-board papers assembled with brass fasteners.

edge-ends. Since we print edge-numbers directly on edge-ends of vertex components, which are engraved with laser-cutters, finding the components is straightforward. I developed a strategy to simplify the process of finding the corresponding pieces. I used flood-fill algorithm to output assembling panels on materials, such as thin sheets of metal or cardboard. The order that the assembling panels are cut out by a laser-cutting is the same order that the assembling panels will be joined by the fabricator. This strategy greatly reduces the assembling time and eases the hard labors work of the fabricator.

Using this approach, three Architecture students have constructed a large version of Stanford Bunny in the design and fabrication course in College of Architecture, spring 2011 (see Figure 1). The costs of poster-board papers and fasteners were very minimal, less than \$300. This approach can be used to construct even larger shapes using stronger materials. The structures obtained by these approaches can be used as molds to cast large plastic or cement sculptures.

CHAPTER II

RELATED WORK

My research is in the intersection of geometric modeling and architectural fabrication. In this section I will discuss about these fields and current research.

II.1. Digital Modeling and Fabrication

Digital fabrication started with Computer Numerical Control (CNC) machines in 1940s and 1950. The first Numerical Control machines were simple servomechanisms that are controlled by software provided by punched tape. These early servomechanisms were rapidly augmented with analog and digital computers, creating the modern computer numerical control (CNC) machine tools that have revolutionized the machining processes.

Fig. 2. Examples of machines used in fabrication: (a)A stationary three-axis CNC machine. Photo is from [43]. (b) A laser cutting machine. Photo is from [23]. (c) An 3D printer. Photo is from [60].

In modern CNC systems, end-to-end component design is highly automated using computer-aided design (CAD) and computer-aided manufacturing (CAM) programs.

The programs produce a computer file that is interpreted to extract the commands needed to operate a particular machine via a postprocessor, and then loaded into the CNC machines for production. Since any particular component might require the use of a number of different tools-drills, saws, etc., modern machines often combine multiple tools into a single “cell”. In other cases, a number of different machines are used with an external controller and human or robotic operators that move the component from machine to machine. In either case, the complex series of steps needed to produce any part is highly automated and produces a part that closely matches the original CAD design.

To fabricate larger pieces, industry usually use Computer Numerical Control (CNC) machines. CNC machines are specifically successful in milling materials such as plywood, foam board, and steel at a fast speed. CNC machines can cut pieces using a set of axes.

CNC machines are classified according to the number of axes that they possess, see Figure 2 (a). CNC machine beds are typically large enough to allow 4 x 8 (123 cm x 246 cm) sheets of 3/4” (19mm) thick plywood to be cut.

Another option for cutting materials is to use a laser cutter. The laser cutter is a machine that uses a laser to cut materials such as chip board, matte board, thin sheets of wood, and prexy up to 3/8” (1 cm) thickness, see Figure 2 (b). *AutoCAD* [9] is used in the production of lines on a grid, which would be sent to the laser cutter as a .eps file. Lines can either cut through the material or score it depending on the color of the line drawn. Lines drawn in blue will be cut and lines in yellow will score the material in the cutting bed. Objects cut out of materials can be used in the fabrication of physical models, which will only require piecing together of the parts.

3D printing is a form of additive manufacturing technology where a three dimensional object is created by laying down successive layers of material, see Figure 2 (c). 3D printers are generally faster, more affordable and easier to use than other additive manufacturing technologies. 3D printers offer product developers the ability to print parts and assemblies made of several materials with different mechanical and physical properties in a single build process. Advanced 3D printing technologies yield models that closely emulate the appearance and functionality of product prototypes. A 3D printer works by taking a 3D computer file and using and making a series of cross-sectional slices. Each slice is then printed one on top of the other to create the 3D object. The technology also finds use in the jewelery, footwear, industrial design, architecture, construction, automotive, aerospace, dental and medical industries. The factors that limits its application are the scale of objects that the 3D printer can handle and the production cost. The current largest build size is 21.65 x 15.5 x 11.8 inches (550 x 393 x 300mm) [22].

In our research, our system enables the user use a laser cutter and inexpensive clip board to building large physical structure.

Because of all these developments, now, architecture design and construction are much more interconnected through digital modeling and fabrication. The digital modeling and fabrication tools allow designers to produce digital materiality, which is something greater than an image on screen, and actually tests the accuracy of the software and computer lines. Computer milling and fabrication integrate the computer assisted designs with that of the construction industry. In this process, the sequence of operations becomes the critical characteristic in procedure. Architects can propose complex surfaces, where the properties of materials should push the design. Some of the modeling software programs include Blender (software), Form-Z, Rhinoceros 3D, Cadwork, 3ds Max and SketchUp.

However, there are still a lot to do theoretically and practically.

II.2. Computer Graphics Applied to Architecture

Computer Graphics techniques have been used in mostly virtual design and there have been only a few works on physical structure construction and cost reduction of fabrication and assembling process. I think we should look at the combined process of design, analysis and manufacture. I am going to review relevant previous research works in this chapter. Subsection II.2.1 talks about computer graphics techniques developed to aid architectural design. Subsection II.2.2 talks about works that computer graphics researchers did to address the practical fabrication and construction problems.

II.2.1. Research from the Perspective of Design

Over the last decade, computer aided geometric design area emerged at the borders between discrete differential geometry and architecture. This area of computer graphics research combines applied geometry and architecture. This trend is primarily driven by the increasing demands in designing and modeling freeform surfaces. It lies at the core of architectural design and strongly challenges contemporary practice, the so-called architectural practice of the digital age. The geometric problems which occur when realizing freeform geometries as steel/glass structures can be efficiently dealt with by the concept of paneling. Geometry processing applications of the concept of paneling meshes are geometric modeling with such meshes, and approximating freeform surfaces with them.

Some recent examples are the construction of various models. The research group led by Akleman developed and maintain the mesh modeler called *TopMod3D* [7]. *TopMod3D* is a topologically robust mesh modeler which is free to public that allows users to easily create a wide variety of very high genus structures [61]. *TopMod3D* provides a large set of graphics operations for creating very complicated 3D objects. Since its first announcement, users from all over the world have been using it to design interesting models in many varieties. The users have formed a community where they share their design experiences and ideas. The most important for this dissertation, *TopMod3D* provides a set of operations that can create developable surfaces [29]. These tools are especially useful for designing D-forms [57, 65].

Paul Haeberli developed Lamina design to convert any given 3D surface to a set of developable panels [33]. Lamina makes it easy to fabricate large scale free-form structures from planar (sheet) materials like plastic, metal, or plywood. Lamina builds a precise physical structures in the real world by approximating given 3D surfaces by a number of 2D parts that are numerically cut and attached to fabricate the final structure.

In [48], Mitani and Suzuki propose a new method for producing unfolded paper-craft patterns of rounded toy animal figures from triangulated meshes by means of strip-based approximation. Their approach is to approximate a mesh model by a set of continuous strips, not by other ruled surfaces such as parts of cones or cylinders. Set of strips can be crafted just by bending the paper (without breaking edges) and can represent smooth features of the original mesh models.

Massarwi et al. introduce an algorithm for approximating a 2-manifold 3D mesh by a set of developable surfaces [45]. Each developable surface is a generalized cylinder represented as a strip of triangles not necessarily taken from the original mesh. The approximation quality is controlled by a user-supplied parameter specifying the allowed Hausdorff distance between the input mesh and its piecewise-developable approximation.

In [50], Mori and Igarashi introduced *Plushie*, an interactive system that allows nonprofessional users to design their own original plush toys. The user draws and edits

2D patterns on a sketching interface. Internally, the system constructs a 2D cloth pattern in such a way that the simulation result matches the user's input stroke.

[54] simplifies the process of modeling developable surfaces by introducing an intuitive sketch-based approach for modeling developables. The authors develop an algorithm that given an arbitrary, user specified 3D polyline boundary, constructed using a sketching interface, generates a smooth discrete developable surface that interpolates this boundary. This method utilizes the connection between developable surfaces and the convex hulls of their boundaries. The method explores the space of possible interpolating surfaces searching for a developable surface with desirable shape characteristics such as fairness and predictability.

Fascinating and elegant shapes may be folded from a single planar sheet of material without stretching, tearing or cutting, if one incorporates curved folds into the design. In [37], Kilian et al. present an optimization-based computational framework for design and digital reconstruction of surfaces which can be produced by curved folding. Their work not only contributes to applications in architecture and industrial design, but it also provides a new way to study the complex and largely unexplored phenomena arising in curved folding.

The work of Gal et al. [27] is an example where a model is analyzed and then optimized. They introduce an approach *iWIRES* that analyzes and edits man-made models by learning the inter-relation among the model parts. Prior to editing, they perform a light-weight analysis of the input shape to extract a descriptive set of wires. Analyzing the individual and mutual properties of the wires, and augmenting them with geometric attributes make them intelligent and ready to be manipulated. Editing the object by modifying the intelligent wires leads to a powerful editing framework that retains the original design intent and object characteristics.

Inspired by freeform designs in architecture which involve circles and spheres, Schiftner et al. introduce a new kind of triangle mesh whose faces' incircles form a packing [55]. They cover surfaces with circle patterns, sphere packings, approximate circle packings, hexagonal meshes which carry a torsion-free support structure, hybrid tri-hex meshes, and others. This kind of structure represents the shape of input mesh. Triangle meshes are optimized so as to have the incircle packing property. The examples they give is a rich source of geometric structures relevant to architectural geometry.

A kind of tree-woven structure can be generated from triangular meshes. In [41], the authors describe an interactive editing framework that provides control over the type, location, and number of irregular vertices in a triangular mesh. They first provide a theoretical analysis to identify the simplest possible operations for editing irregular vertices and then introduce a hierarchy of editing operations to control the type, location, and number of irregular vertices. By rearranging irregular vertices on the input triangular mesh, the resulting tree woven structure becomes more regular.

Recently, the research group led by Akleman have been conducting a deep study on different woven structures from the perspective of topology theory. In [6], they show how to create plain-weaving over an arbitrary surface. To create a plain-weaving on a surface, they create cycles that cross other cycles (or themselves) by alternatingly going over and under. 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 method that converts plain-weaving cycles to 3D thread structures. Using this method, it is possible to cover a surface without large gaps between threads by controlling the sizes of the gaps. They have developed a system that converts any manifold mesh to a plain-woven object, by interactively controlling the shapes of the threads with a set of parameters. They have demonstrated that by using this system, a user can create a wide variety of plain-weaving patterns, some of which may not have been seen before.

In [69], Xing et al. show that it is always possible to create a single-cycle plain-weaving starting from a mesh on an arbitrary surface, by selecting an appropriate subset of edges to be twisted. They also demonstrate how, starting from a mesh, to construct a large number of single-cycle plain-woven objects. Interestingly, the single-cycle solutions with a minimal number of edge twists correspond to plain-woven objects that are visually similar to Celtic knots. For converting plain-weaving cycles to 3D thread structures, they extend the original projection method in [6], which previously worked only when all mesh edges are twisted. With the extension described in [69], their projection method can also be used to handle untwisted edges. They have developed a system that converts any manifold mesh into single-cycle plain-woven objects, by interactively controlling the proportion of edges that are twisted. The system also allows users to change the shapes of the threads with a set of parameters, interactively in real-time.

Classical (or biaxial) twill is a textile weaving in which the weft threads pass over and under two or more warp threads, with an offset between adjacent weft threads to give an appearance of diagonal lines. In [2], Akleman et al. introduce a theoretical framework for constructing twill-woven objects, cyclic twill-weavings on arbitrary surfaces, and provide methods to convert polygonal meshes into twill-woven objects. They develop a general technique to obtain exact triaxial-woven objects from an arbitrary polygonal mesh surface.

In [70], Akleman et al. present the concept of *Surface Filling Curves*, which is related to space filling curves. Surface covering curves are based on a theoretical property of manifold mesh surfaces. They show that any mesh surface can be converted to a closed 3D curve structure that follows the shape of the mesh surface. They have developed two methods to construct corresponding 3D ribbon and tread structures from the mesh structure and the connectivity of the curve. The first method constructs equal thickness ribbons (or equal diameter threads). The second method, which is based on the projection method in [69], creates ribbons with changing thickness (or threads with changing diameter) that can densely cover the mesh surface. Since each iteration of any subdivision scheme results in a denser mesh, the procedure outlined above can be used to obtain denser and denser curves, ribbons and treads. These curves densely cover the mesh surface in limit. Therefore, this approach along with a subdivision scheme behaves like fractal algorithms that create space filling curves.

II.2.2. Research from the Perspective of Construction

Recently, there have been significant research results which investigate the interconnection between virtual and physical design and cost reduction of fabrication and construction.

A large body of previous work on the paneling problem deal with planar panels. Initial research in this direction dealt with special surface classes. In [28], the authors discussed using planar quadrilateral glass facets for the Jerusalem Museum of Tolerance project by Gehry Partners, in collaboration with Schlaich Bergermann & Partners, engineers.

In [35] the authors introduce D-Charts, a simple and robust algorithm for mesh segmentation into (nearly) developable charts. They develop a new metric of developability for mesh surfaces. They used the patterns produced by their algorithm to make fabric and paper copies of popular computer graphics models.

Aiming at robust surface structure recovery, Wu and Kobbelt extend the optimization technique of variational shape approximation by allowing for several different primitives to represent the geometric proxy of a surface region [67]. They use their algorithm to segment a given mesh model into characteristic patches and provide a corresponding geometric proxy for each patch. The primitives include planes, spheres, cylinders, and more complex rollingball blend patches.

Yan et al. [72] proposed a variational approach to extract general quadric surfaces from mesh surfaces. Quadric proxies are progressively inserted (or merged) against an error threshold to improve the surface approximation. A method based on graph cut is proposed to smooth irregular boundary curves between segmented regions, which greatly improves the final results.

Covering general freeform surfaces with planar quad panels could be approached with methods of discrete differential geometry. This trend led to new ways of supporting beam layout and the related computation of multi-layer structures. Motivated by practical architectural need, in [42] Liu et al. introduce conical meshes, which have planar faces and yet possess offset meshes at constant face-to-face distance from the base mesh. Conical meshes satisfy the planar quad panels and offset supporting structure requirements. The authors developed an optimization model to convert quad-meshes into canonical meshes and combined Catmull-Clark subdivision [14] with their approach.

Building upon the concept of parallel meshes, in [52] Pottmann et al. develop methods to optimize meshes with offset properties relevant to architectural modeling considering assembly problem: physical realization of beams and nodes. They discuss planar faces, beams of controlled height, node geometry, and multilayer constructions. Beams of constant height are achieved with the new type of edge offset meshes. Mesh parallelism is also the main ingredient in a novel discrete theory of curvatures. These methods are applied to the construction of quadrilateral, pentagonal and hexagonal meshes, discrete minimal surfaces, discrete constant mean curvature surfaces, and their geometric transforms.

More recently, this approach was extended to the covering of freeform surfaces by single-curved panels arranged along surfaces strips. In [53] the authors propose the new concept of semi-discrete surface representation, which constitutes a link between smooth and discrete surfaces. The developable strip model is the semi-discrete equivalent of a quad mesh with planar faces, or a conjugate parametrization of a smooth surface. They present a B-spline based optimization framework for efficient computing with D-strip models. Additional results in this direction, e.g., hexagonal meshes with planar faces, have been presented at [62].

The emergence of large-scale freeform shapes in architecture poses big challenges to the fabrication of such structures. A key problem is the approximation of the design surface by a union of panels, which can be manufactured with a selected technology at reasonable cost, while meeting the design intent and achieving the desired aesthetic quality of panel layout and surface smoothness. Originally introduced by Cohen-Steiner et al. [17] for surface approximation by planes, various extensions have been proposed for additional surface types, e.g., spheres and cylinders [67], quadrics [72], or developable surfaces [35]. An optimization has been proposed to simultaneously partition the input surface, as well as determine the types and number of shape proxies required [40]. These methods optimize for a surface segmentation to reduce the approximation error. Similarly, state-of-the-art methods in surface fitting and local registration (see e.g. [12, 56]), are insufficient to solve large-scale freeform paneling problems.

Recently, there are three similar pieces of independent research work that share the same general motivation [21, 25, 58]. All three aim at optimizing the number of tile shapes required to construct a surface.

In [58], Singh and Schaefer propose a technique that takes a triangulated surface as input and outputs a surface with the same topology but altered geometry such that each polygon falls into a set of discrete equivalence classes. They search for a clustering of triangles that can be optimized to approximate the given model. Starting from a single cluster, they keep adding clusters until the approximation is sufficient. They also describe how to incorporate a fairness criteria into the optimization to avoid oscillations of the surface. They have been able to successfully reduce the number of unique triangles to lie within a small percentage of the total number of triangles in the surface.

The approach of [58] is similar to [25]. In [25], Fu et al. introduce a method for optimizing the tiles of a quadmesh. With an input quad-based surface, their algorithm generates a set of K quads whose instances produce a tiled surface that approximates the shape of the input surface. Optimizing all panels into K-sets leads to an effective cost reduction. They take an optimization approach that consists of iterative clustering and analysis. First, iteratively cluster and analyze: clusters of similar shapes are merged, while edge connections between the K quads on the target surface are analyzed to learn the induced flexibility of the K-set tilable surface. Then, apply a non-linear optimization model with constraints that maintain the K quads connections and shapes. Their optimization tends to capitalize on the inherent symmetry of the given shape [71]. In that sense, a related work is the model symmetrization from Mitra et al. [49].

The work of Eigensatz et al [21] considers molds and panels rather than congruent tiles. Their focus is on the reusability of molds in fabricating curved panels for forming globally coherent surfaces. The production of curved panels is mostly based on molds. Since the cost of mold fabrication often dominates the panel cost, there is strong incentive to use the same mold for multiple panels. They cast the major practical requirements for architectural surface paneling, including mold reuse, into a global optimization framework that interleaves discrete and continuous optimization steps to minimize production cost while meeting user-specified quality constraints. The search space for optimization is mainly generated through controlled deviation from the design surface and tolerances on positional and normal continuity between neighboring panels. They use a 6-dimensional metric space to compute approximate inter-panel distances, which improves the performance of the optimization and enables the handling of complex arrangements with thousands of panels.

CHAPTER III

THEORETICAL BACKGROUND

In this chapter, I will discuss the theoretical framework that this research is built upon. The theoretical background knowledge are two parts, the topology and geometry.

III.1. Topology

III.1.1. Classical Graph Rotation Systems

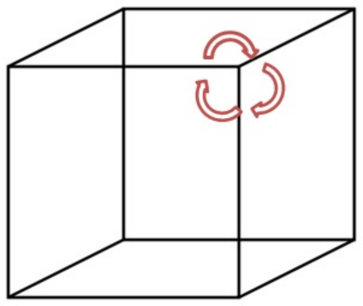
In combinatorial mathematics, graph rotation systems encode the embedding of graphs onto orientable surfaces, by describing the circular ordering of a graph's edges around each vertex [30, 31]. The formal definitions for graph rotation system are quoted from [31] as below:

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.

In [31], the authors explain how a graph embedding induces a rotation system of the graph. Let $h : G \to S_g$ be an imbedding of the graph G on the oriented surface S_g. The *induced rotation* $\rho_h(v)$ at a vertex v of G is the cyclic permutation of the edge-ends incident on v in the order in which they are encountered in a traversal around v in the preferred direction on the surface G_g. The *induced rotation system* of the graph G by the embedding $h : G \to S_g$ is the collection of the induced vertex rotations $\rho_h(v)$ for all vertices v in G.

Let ρ be a rotation system for a graph G. The *rotation table for* ρ, denoted T_ρ has one row for each vertex of G. The content of each row of the table is the name of a vertex, followed by a complete list of the edge-ends incident on that vertex, in an order consistent with $\rho(v)$. (It is a common custom to write each row in lexicographic order.)

Figure 3 (a) illustrates the rotation of edges at a vertex. Faces can be identified with "Face Tracing" algorithm as seen in Figure 3 (b). Graph Rotation System is introduced recently as a new shape modeling method [1]. It can guarantee the 2-mainfold property of a mesh while changing the topology of the mesh, and has the advantage of the simplicity of creating an artistic graph. A new data structure Doubly Linked Face List is proposed, which can implement the operations of graph rotation system efficiently.


(a) The rotation of edges at a vertex.

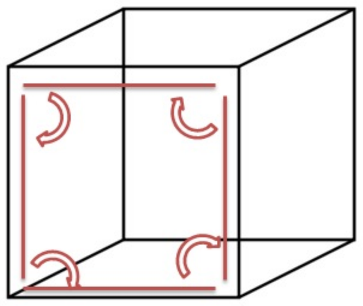

(b) Tracing edges belonging to a face.

Fig. 3. An illustration of the classical graph rotation system.

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 [20]. As a direct consequence of Heffter-Edmunds theorem, to assemble the structure all construction workers have to do is to attach the corresponding edge-ends of vertex components. Once all the components are attached to each other, the whole structure will correctly be assembled.

Graph Rotation Systems needs to be converted to geometry for virtual or physical construction. I propose a method for converting GRS to a geometric shape using 2D-thickening method. The resulting shapes will be called band decomposition structures.

III.1.2. Extended Graph Rotation Systems and Weaving

Formally, a *cyclic plain-weaving* on an orientable surface S is a projection of a link L on S, such that (1) there are no triple intersections at a single point on S, and (2) a traversal of the image on S of each component of L goes over and under alternatingly as it crosses the images of other components or of itself.

Our theoretical framework for cyclic plain-weaving is based on an extension of graph rotation systems, which have been extensively studied in topological graph theory [30]. Some of the concepts related to graph rotation systems have been implicitly [11, 32, 44] and explicitly [1] studied in computer graphics. An important concept in graph rotation systems is edge twisting. In the pre-existing theory, an edge has type 0 if it is untwisted and type 1 if it is twisted. Topologically, a double-twisted edge is the same as an untwisted edge.

Face-Tracing Algorithm.

(A slight revision of the algorithm given by Gross and Tucker)

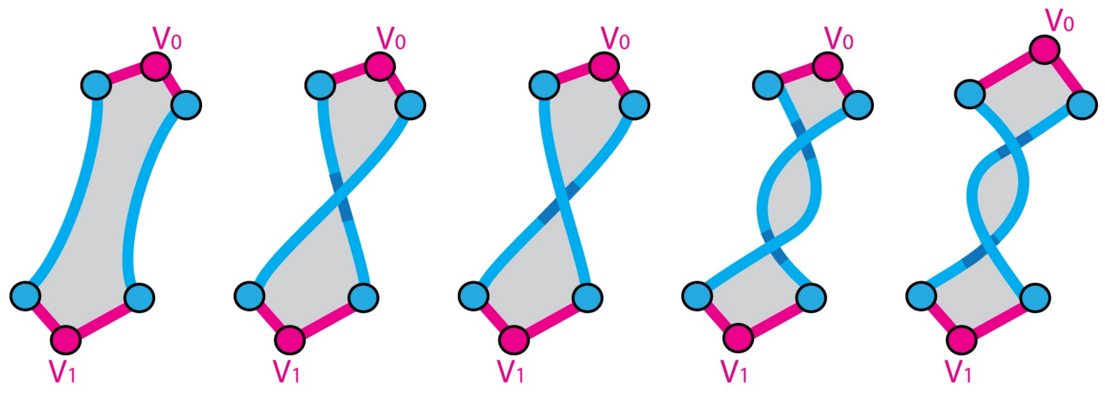
```
Subroutine FaceTrace($(\langle u_0, w_0\rangle, t_0)$)
\\$\langle u_0, w_0\rangle$ is an oriented edge, $t_0 \in \{0,1\}$ is the "trace type".

1.  trace $\langle u_0, w_0\rangle$;
2.  $t = t_0 + type([u_0, w_0])\ (mod\ 2)$;
3.  $\langle u, w\rangle = \text{Next}(\langle w_0, u_0\rangle, t)$;   \\$u = w_0$
4.  while ($(\langle u, w\rangle \neq \langle u_0, w_0\rangle)$ and $(t \neq t_0)$) do
        trace $\langle u, w\rangle$;
        $t = t + type([u, w])\ (mod\ 2)$;
        $\langle u, w\rangle = \text{Next}(\langle w, u\rangle, t)$.

Algorithm Trace-All-Faces ($\rho(G)$)
\\ $\rho(G)$ is a general graph rotation system.
while there is an untraced face corner $(u, e, e')$ in $\rho(G)$ do
        suppose that $e' = \langle u, w\rangle$;
        call FaceTrace($(\langle u, w\rangle, 0)$).
```

A fundamental algorithm on graph rotation systems, known as *face-tracing*, is given immediately above. This algorithm on a graph rotation system $\rho(G)$ returns a collection of graph cycles that are the boundary-walks of the faces in $\rho(G)$. For detailed explanation and discussion of the face-tracing algorithm see [4, 30].

In [4], the concept of edge twisting is extended. Here, the direction in which an edge is twisted and the number of revolutions in the twisting are taken into consideration.

Definition. An edge is k^+-*twisted* (resp. k^--*twisted*) if it is obtained from a flat paper strip by standing on one end of the strip and twisting the other end in clockwise (resp. counterclockwise) direction by $k \times 180^0$.

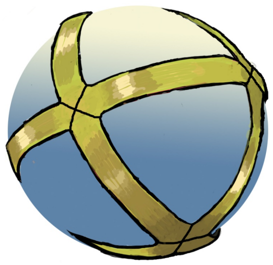
Definition. An *extended graph rotation system* (EGRS) is a graph rotation system with extended edge twists. Note that the face-tracing algorithm can be applied to an EGRS without change if we take the edge type of a k^+-twisted (resp. k^--twisted) edge to be k (resp. $-k$). See Figure 4 for illustrations.

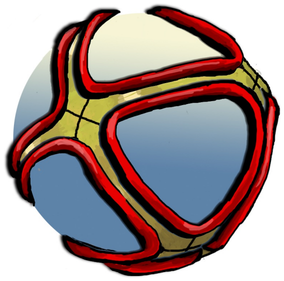


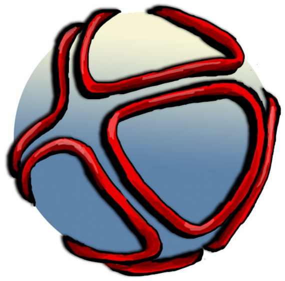
(a) $k=0$ (b) $k=1$ (c) $k=-1$ (d) $k=2$ (e) $k=-2$

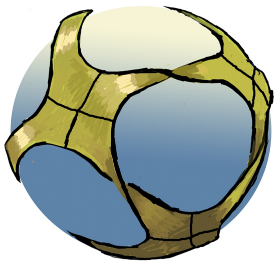
Fig. 4. (a) An untwisted/flat edge. (b) A counterclockwise twist of an edge. (c) A clockwise twist of an edge. (d) A counterclockwise double twist of an edge. (e) A clockwise double twist of an edge.

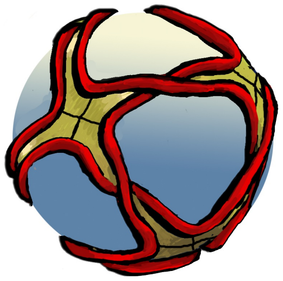
Akleman et al. noticed that the boundary walks induced by a graph rotation system define a link in 3D-space, and they used this property to construct plain-weaving cycles on arbitrary polygonal mesh surfaces [6, 4]. This insight can be realized visually using paper-strip sculptures, as shown in Figure 5. Figure 5(a) gives an octahedron constructed by paper strips, embedded in a sphere. As shown in Figure 5(c), the eight face-boundary walks of the octahedron are unlinked in 3D-space. However, if the edges are twisted as shown in Figure 5(d), then the cycles represented by the boundary walks become linked, as in Figure 5(f).


(a)


(b)


(c)


(d)


(e)

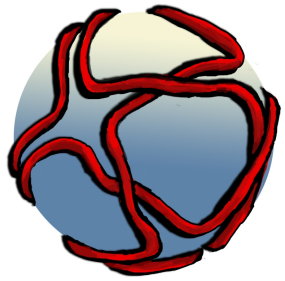
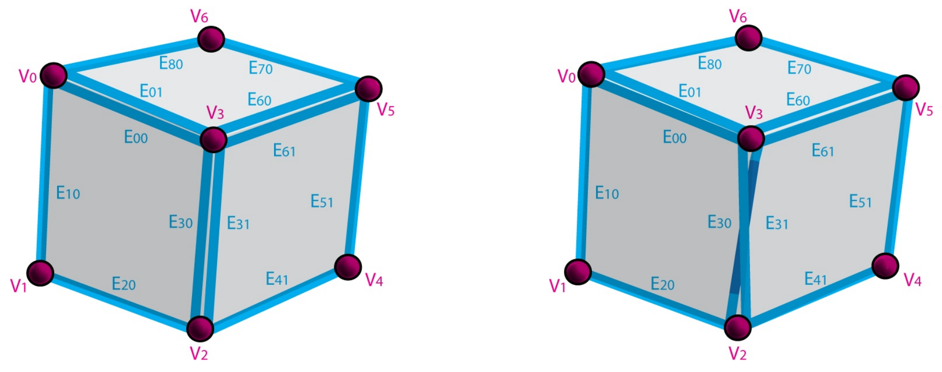

(f)

Fig. 5. The boundary walks of paper-strip sculptures are links in 3D space.

To construct a cyclic plain-weaving on an orientable surface S_h, we start with a graph rotation system $\rho_0(G)$ with no twisted edges that determines a graph embedding on S_h. The face boundary walks of $\rho_0(G)$ form a collection of disjoint cycles on S_h, which we regard a projection of a link onto that surface. This is the initial weaving on S_h, in which each edge of G lies between two parallel strands on S_h. When we apply the extended edge-twisting operations on $\rho_0(G)$, we obtain an EGRS $\rho(G)$. Under the face tracing algorithm, this will result in a new collection of cycles. Moreover, if we associate the edge-twists with crossings of the link components on the surface S_h, then the EGRS $\rho(G)$ specifies a link projection on the surface S_h.

The following theorem is a foundation for our development of cyclic plain-weaving (see [4] for a proof of the theorem):

Theorem III.1.1 *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.*



(a) The initial mesh whose edges are not twisted.

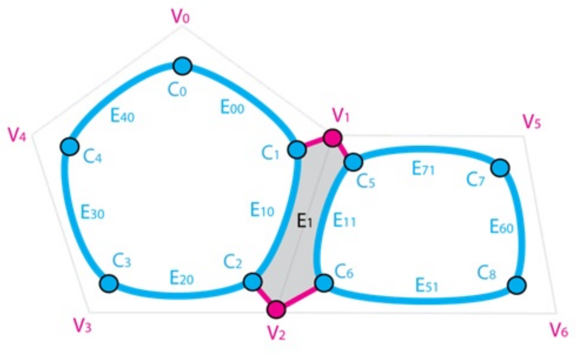
(b) The resulting mesh after twisting one edge which is incident to v_2 and v_3.

Fig. 6. Face boundary walks before and after twisting one edge.

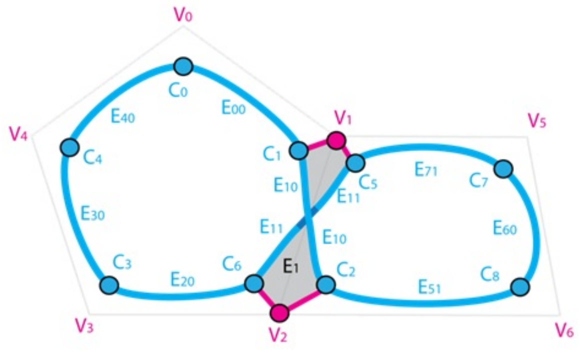
Trace all face boundary walks using the *Face-Tracing Algorithm*. Figure 6 shows the face boundary walks. For instance, for the face with a twisted edge in Figure 6 the computed face boundary walk is the cyclically ordered set

$$K_1 = \{E_{0,0}, E_{1,0}, E_{2,0}, E_{3,0}, E_{6,1}, E_{5,1}, E_{4,1}, E_{3,1}\}$$

From another perspective, we can regard edge twisting as an operation on cycles. Face boundary walk defines an alternating link. Twisting an edge e in a general rotation system $r(G)$ satisfies the following rules. First, suppose that the two trace-pairs induced by e belong to the face boundaries of two different faces. Then twisting e merges the two faces into a single face. Second, suppose that the two trace-pairs induced by e belong to the face boundary of the same face, If the two trace-pairs induced by e use the same oriented edge, then twisting e splits the face F into two faces. Figure 7 demonstrates both the first and second rules. Third, if the two trace-pairs induced by e use different oriented edges, then twisting e converts the face F into a new single face. In other words, we can construct weaving structures that we want from orientable meshes by twisting edges.



(a) The initial mesh



(b) The mesh after twisting one edge

Fig. 7. An illustration of the edge twisting as a process of cycle operation.

We discuss cyclic plain-weaving in more detail in Appendix 1.

The plain-weaving cycles that are created by twisting edges are mathematical links and do not have a solid shape. In order to create geometric forms, these cycles need to be converted to 3D thread structures, such as ribbons (extruded lines) or yarns (extruded polygons). The resulting 3D thread structures must look smooth and must not self-intersect. Plain woven structures defined by Extended Graph Rotation Systems (EGRS) needs to be converted to geometry for virtual or physical construction. I propose converting EGRS to a geometric shape using a projection method. The resulting shapes will be called plain woven structures.

III.2. Geometry

III.2.1. Gaussian Curvature

We will first discuss the curvature of a plane curve and curvature of surfaces. Let C be a plane curve. The curvature of C at a point is a measure of how sensitive its tangent line is to moving the point to other nearby points.

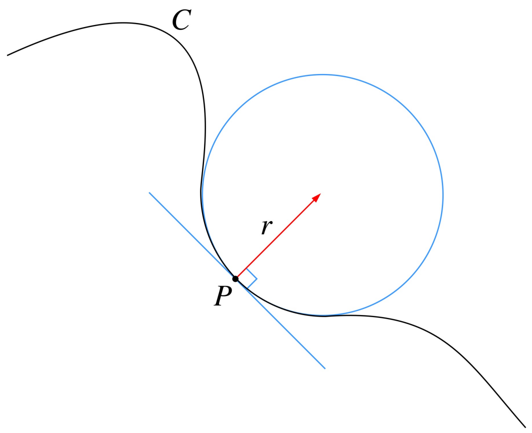


Fig. 8. Curvature of a plane curve. Image is from [15].

Given any curve C and a point P on it as seen in Figure 8, there is a unique circle or line which most closely approximates the curve near P, which is called the osculating circle at P. The curvature of C at P is then defined to be the reciprocal of the radius R:

$$k = \frac{1}{R} \tag{3.1}$$

When a one dimensional curve lies on a two dimensional surface embedded in

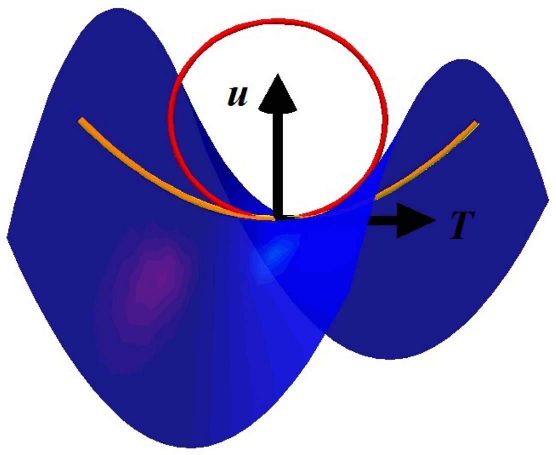


Fig. 9. Normal curvature. Image is from [64].

three dimensions, further measures of curvature are available, which take the surface's unit-normal vector, u into account. As seen in Figure 9, any non-singular curve on a smooth surface will have its tangent vector T lying in the tangent plane of the surface orthogonal to the normal vector. The normal curvature, k_n, is the curvature of the curve projected onto the plane containing the curve's tangent T and the surface normal u.

All curves with the same tangent vector will have the same normal curvature, which is the same as the curvature of the curve obtained by intersecting the surface with the plane containing T and u as seen in Figure 10. Taking all possible tangent vectors then the maximum and minimum values of the normal curvature at a point are called the principal curvatures, k_1 and k_2, and the directions of the corresponding tangent vectors are called principal directions.

In differential geometry, the Gaussian curvature or Gauss curvature of a point on a surface is the product of the principal curvatures, k_1 and k_2, of the given point [38]. It is an intrinsic measure of curvature, i.e., its value depends only on how distances

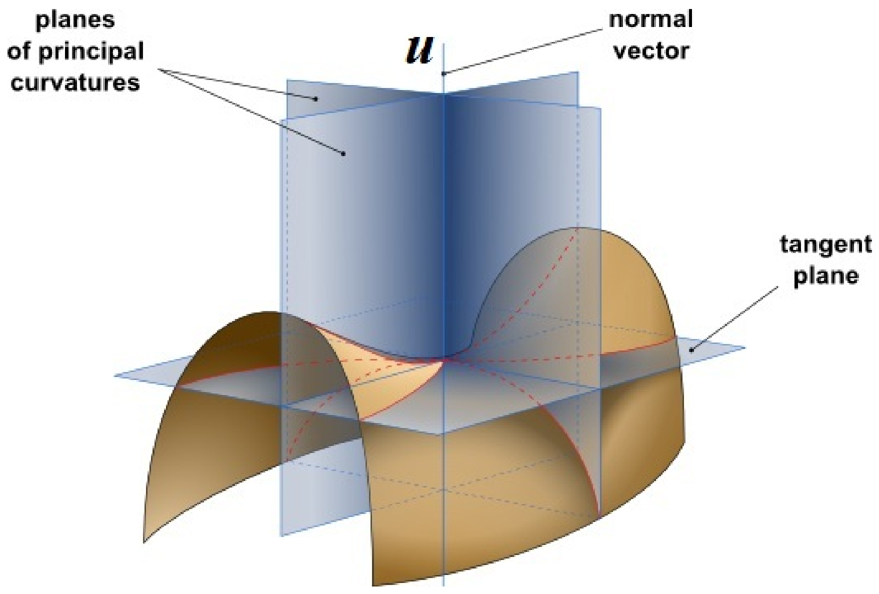


Fig. 10. Principal curvature. Image is from [26].

are measured on the surface, not on the way it is isometrically embedded in space. This result is the content of Gauss's Theorema Egregium. Symbolically, the Gaussian Curvature κ is defined as

$$\kappa = k_1 k_2 \tag{3.2}$$

where k_1 and k_2 are the principal curvatures. This meant that regardless of the direction of the normal, both principal curvatures would change sign simultaneously (thus, both would either be positive or negative).

Figure 11 shows examples of surface of different Gaussian Curvature. Surface with zero Gaussian Curvature is developable surface which means we can flatten the surface on a plane without distortion.

In [3], Akleman et al. proposed the concept of discrete Gaussian curvature and face defect. This is the theocratical foundation which explains why our two structures will stand in 3D space. Both the plain woven structure and band decomposition

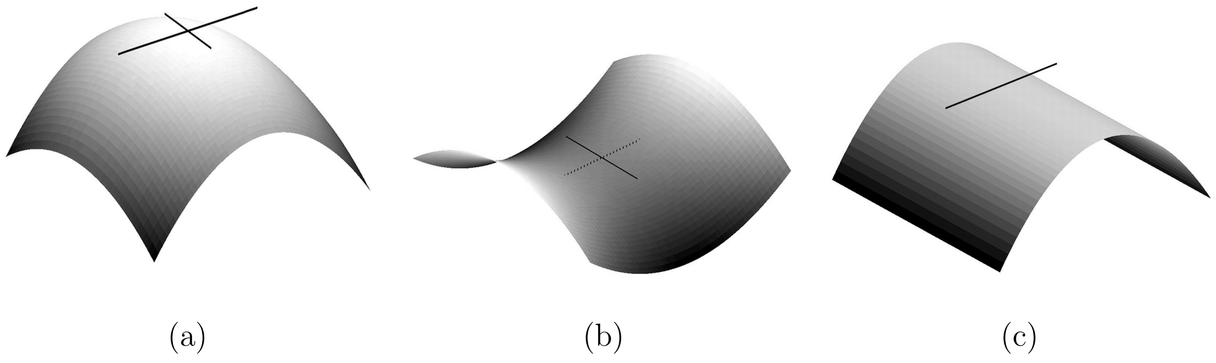


Fig. 11. (a) A surface of positive Gaussian Curvature. (b) A surface of negative Gaussian Curvature. (c) A surface of zero Gaussian Curvature. Images are from [39].

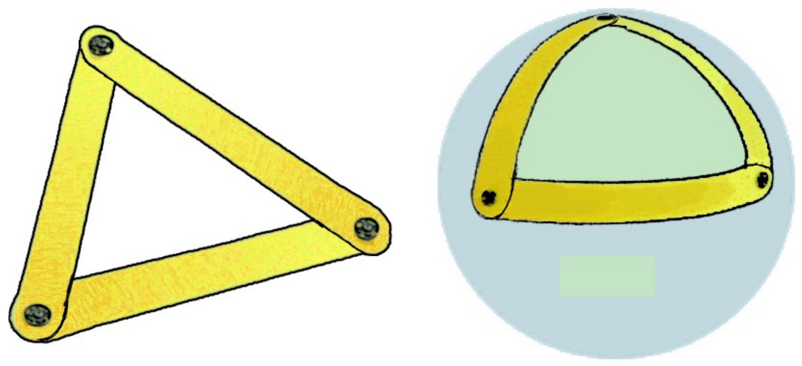
structure use developable panels. Developable surfaces have zero Gaussian curvature. Moreover, since around a vertex of a band decomposition structure, a panel is also locally flat, the Gaussian curvature is also zero around the vertices. Accordingly, all the Gaussian curvature exists in the empty spaces that correspond to faces. We define ϕ, the discrete Gaussian curvature of a face, or *face-defect*, for an n-sided face, as

$$\phi = \sum_{i=0}^{n} \theta_i - (n-2)\pi \tag{3.3}$$

where θ_i is the angle at corner i. Of course, $(n-2)\pi$ is the sum of the angles if the polygon is planar. In other words, the face-defect is a measure of how much a polygon deviates from a flat surface.

As can be seen in Figure 12(a), if the sum of the internal angles of the triangle is exactly π, then the result becomes a flat triangle. However, as soon as we increase the sum of the internal angles, the triangle becomes curved, and it becomes easy to imagine that the triangle is drawn on a surface of a sphere, as seen in Figure 12(b).

Note that we can categorize the shapes of the faces, based on face-defect.



(a) (b)

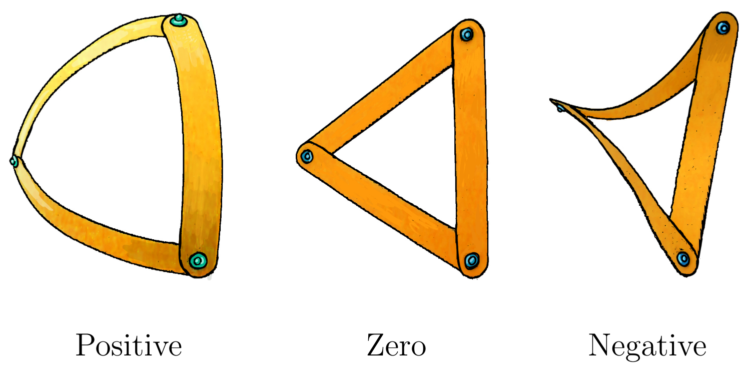
Fig. 12. Triangles with zero face defect and positive face defect.

- If the face-defect is zero, then either the face is flat or the Gaussian curvatures inside of the face cancel each other. This means that the polygon is either drawn on a flat surface or it is a part of a toroidal or cylindrical shape. It is also possible to call such a polygon a *Euclidean polygon.*

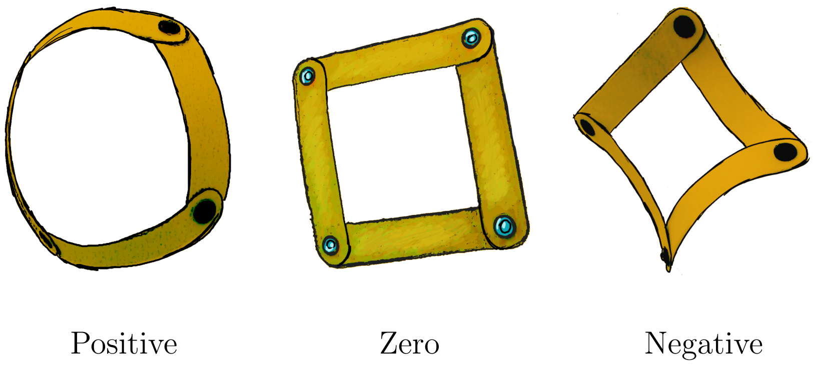
- If the face-defect is positive, then the face has a convex or a concave shape. It is possible to call such polygons *elliptic*. An interesting case is when every angle is π, implying that $\phi = 2\pi$, in which case the polygon forms a circular band.

- If the face-defect is negative, then the face has a saddle shape. We call such polygons *hyperbolic*, and they can be drawn on a surface of negative curvature.

These cases are illustrated in Figures 13(a) and Figures 13(b). Note that these shapes were created from each other by simply changing the sums of the internal angles of the polygons formed by paper strips. Changing the internal angles is easy using simple connectors, since the paper strips can rotate freely around the snaps.



(a) Triangles.



(b) Squares.

Fig. 13. Triangles and Quadrilaterals. (a) shows triangles with positive, zero, and negative Gaussian curvature. (b) shows squares with positive, zero, and negative Gaussian curvature.

III.2.2. Gauss Bonnet Theorem

The surface integral of the Gaussian curvature over some region of a surface is called the total curvature. The Gauss-Bonnet theorem states for any compact, boundaryless two-dimensional Riemannian manifold M, the integral of the Gaussian curvature over the entire manifold with respect to area is two π times the Euler characteristic χ of the manifold:

$$\int_M \kappa dA = 2\pi\chi \tag{3.4}$$

For a polygonal mesh of manifold, χ is defines as:

$$\chi = V - E + F = 2(1 - g) \tag{3.5}$$

where V is the number of vertices; E is the number of edges; F is the number of faces; g is the genus of the manifold.

The Gauss-Bonnet theorem is an important theorem in differential geometry. It is intrinsically beautiful because it relates the curvature of a manifold (a geometrical object) with the its Euler Characteristic (a topological one). In [16], Akleman et al. proved the discrete version of Gauss-Bonnet theorem which states the sum of vertex defects, edge defects, and face defects is two π times the Euler characteristic χ of the discrete mesh. So the total Gaussian curvature of a mesh is independent of number of vertices, edges, or faces.

III.2.3. Developable Surfaces

In mathematics, a developable surface is a surface with zero Gaussian curvature. That is, it is a "surface" that can be flattened onto a plane without distortion (i.e. "stretching" or "compressing"). Conversely, it is a surface which can be made by transforming a plane (i.e. "folding", "bending", "rolling", "cutting" and/or "gluing"). Therefor, in our case, developable surface can be constructed from sheets of metal or paper which can be cut inexpensively using a laser cutter. Ddevelopable surface are useful for inexpensive physical construction. In three dimensions all developable surfaces are ruled surfaces as shown in Figure 14.

Since both the plain woven structures and band decomposition structures are made up with only developable panels, Gaussian curvature is zero everywhere on the solid parts. According to the Gauss-Bonnet theorem, the Gaussian curvature

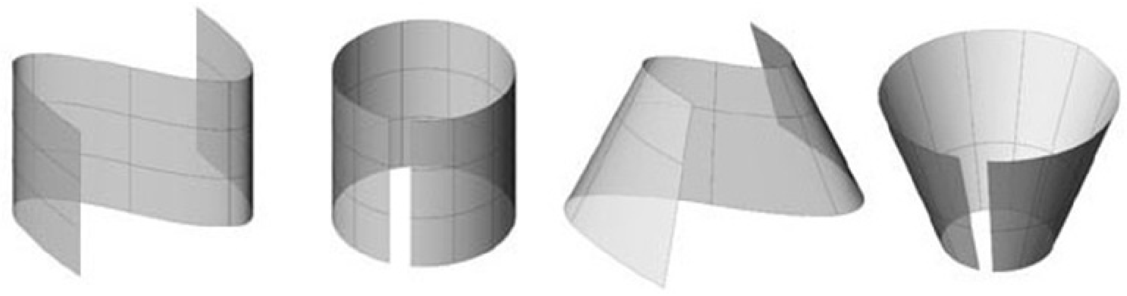


Fig. 14. Examples of developable surface in 3D space. Images are from [51].

happens only in empty regions and that are determined uniquely. Since we correctly form Gaussian curvature of holes, the structures will always be raised and formed in 3-space.

CHAPTER IV

PLAIN WOVEN STRUCTURES

Extended Graph Rotation Systems, as we have discussed in section III.1.2, define cyclic weaving structures, which are mathematical links embedded in surfaces. A mathematical link is 1-dimensional, without solid shape. Therefore these 1-dimensional links need to be converted to 3D geometry for physical construction. I have developed a process to convert these links to developable ribbons that can easy be segmented into pieces. The segments are assembled to construct a plain woven structure. The process consists of four steps.

1. **Geometry Conversion:** This step is used to convert links to ribbons that are represented as series of triangles. I have developed a method called "projection" for geometry conversion, which is discussed in section IV.1.

2. **Panel Creation:** This step is used to generate smooth developable bands in 3D virtual space and segment ribbons into panels. This step is discussed in section IV.2.

3. **Unfolding Panels:** This step is used for unfolding and placement of assembly panels. This step is discussed in section IV.3.

4. **Numbering and Assembling:** In this step, we label connectivity information on panels to simplify the assembly of panels based on the numbers marked on each panel. This step is discussed in section IV.4.

The following sections describe the four steps of the procedure to construct plain woven structures.

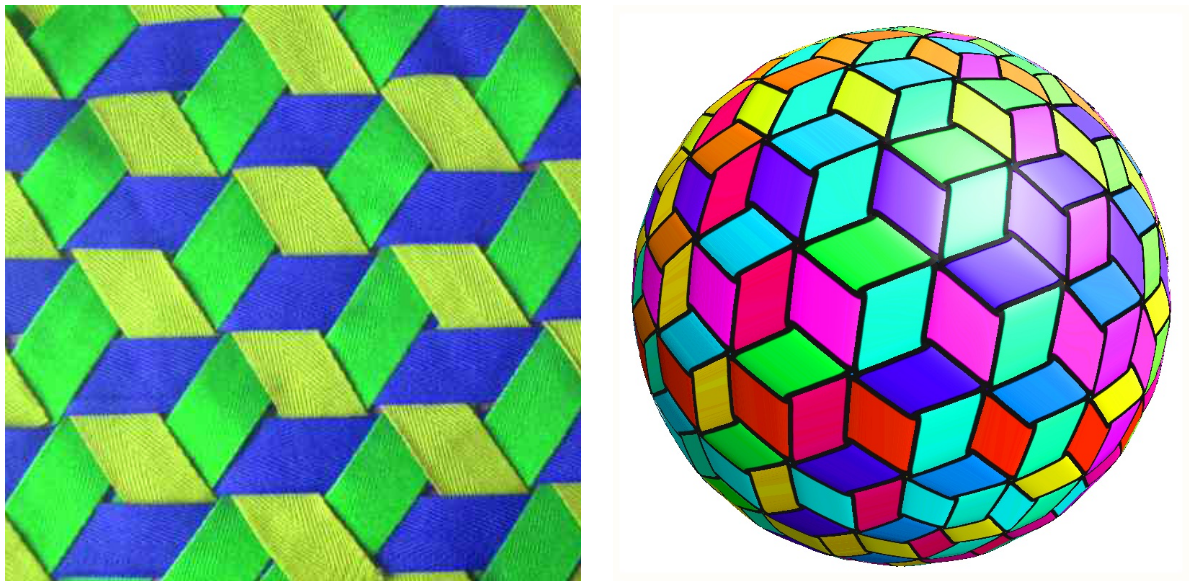
IV.1. Geometry Conversion

In practice, we twist all the edges of graph G, instead of an arbitrary subset of edges, and for our rotation system $\rho_0(G)$, we only consider the most commonly used polygonal mesh surfaces in computer graphics. In our polygonal mesh surfaces, every vertex has valence at least 3, and every face has at least three sides (i.e. triangles). Moreover, every edge has positive length, and every vertex has position information.

Let E_i denote an edge of a manifold mesh, and let $E_{i,0}$ and $E_{i,1}$ denote the over and under half-edges (in the sense of [44]) that lie close beside the twisted edge E_i. We assign and compute a unit normal vector $\vec{n}_i$ for every edge E_i. The faces do not have to be flat, but we assume that for each face we have an approximating planar polygon that is given by a normal vector and a center point. The edge normal vector $\vec{n}_i$ can be computed as the average of the normals of the two approximating planes for the two respective sides of edge E_i. The faces do not have to be convex but if we project a face to its approximate plane from the center point of the face, then all projected edges must be visible. All these conditions eliminate degenerating faces, and they guarantee that we can have a normal vector defined for all the edges of the manifold mesh.

Our goal is to create dense weavings, such as the dense triaxial weaving shown in Figure 15. Extrusion methods are appropriate for drawing Celtic knots in a planar surface [36, 47], but they are not suitable for covering an arbitrary surface since they leave large gaps, and they cannot create dense weaving for all possible weaving patterns.

I have developed the projection algorithm to convert the mathematical links to ribbons that can cover surfaces both densely or sparsely, as seen in Figure 15 and Figure 16. The projection algorithm provides control of the size of the gaps in the weave. Moreover, by using our method, the unusual structure of some weaving patterns becomes more perceptible.



(a) Physical Dense Triaxial Weaving

(b) Our computer model for Dense Triaxial Weaving

Fig. 15. (a) A real "dense" triaxial weaving. This type of weaving is not very common, since it is difficult to manufacture. (b) Using our projection method, we can also create dense triaxial weaving on any regular triangular mesh, as shown. This mesh is the same as that in Figure 16. Photograph in (a) is courtesy of Tim Tyler, see [63].

I will present the projection algorithm in 2D first. The algorithm can be easily extended to 3D cases.

IV.1.1. Projection Method in 2D

The algorithm of the projection method consists of two main steps. The first step is to construct a control polygon for each weaving cycle which we call the *PR control*

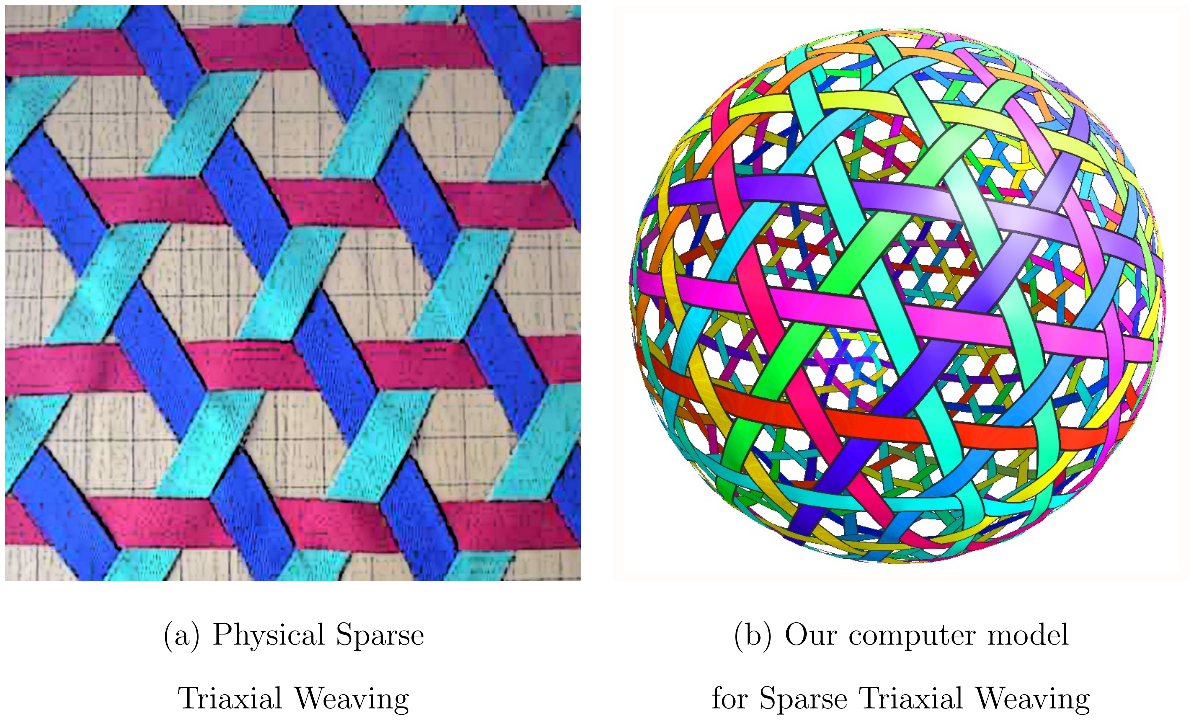
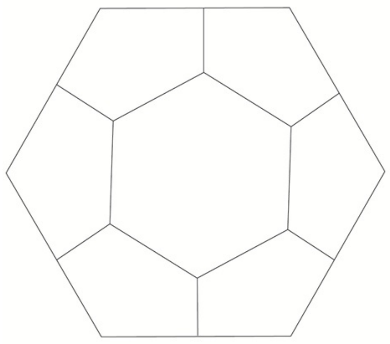


Fig. 16. (a) A photograph of real "sparse" triaxial weaving that leaves gaps (see the large hexagonal-shaped gaps). (b) Our projection method (PR) can create the same type of "sparse" weaving, by leaving gaps as shown. This particular mesh consists of 18 cycles of ribbons. Photograph in (a) is courtesy of Tim Tyler, see [63].

polygon. The second step is to construct a developable ribbon from the PR control polygon. Without loss of generality, we explain the steps of the process using one local area of a planar mesh as the example.

We start with a planar mesh as input mesh and compute the center of mass for each face as seen in Figure 17. Suppose a face f_i has k vertices. The 3D position vector for each vertex in face f_i is denoted as v_j with j is from 1 to k. The equation to compute the center of mass $C(f_i)$ for each face is

$$C(f_i) = \frac{\sum_{j=1}^{k} v_j}{k} \tag{4.1}$$


(A) Initial planer mesh

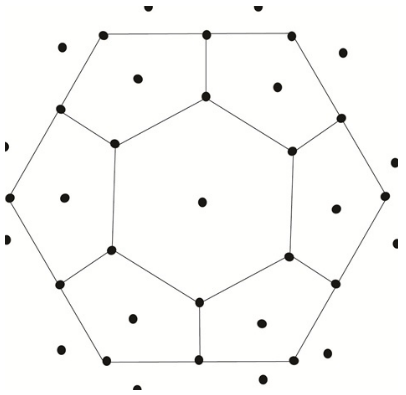
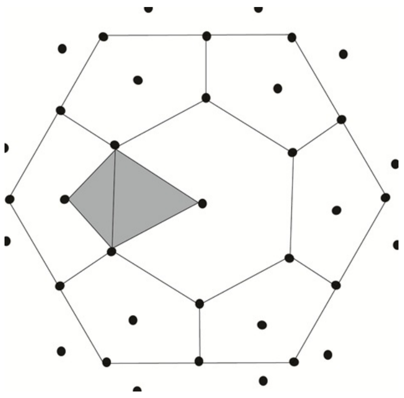

(B) Black dots are the centers of the faces

Fig. 17. Compute center of mess for each face of the input 2D mesh.

After we finish compute center of mass for all faces. We convert each edge to a quadrilateral by connecting two face centers with two vertex points as seen in Figure 18. Each quadrilateral is called an *edge region*.


(A) One edge region (in grey)

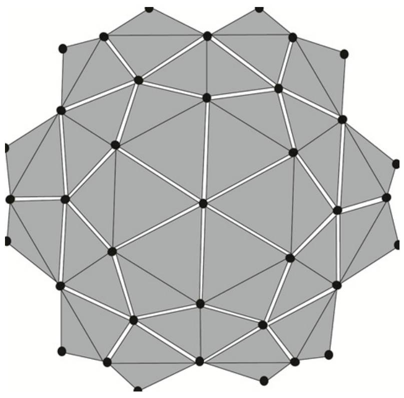

(B)Edges regions for all the edges in the initial mesh

Fig. 18. Construct edge regions on the input mesh.

These edge regions cover the plane as quadrilateral tiles, as seen in Figure 19.

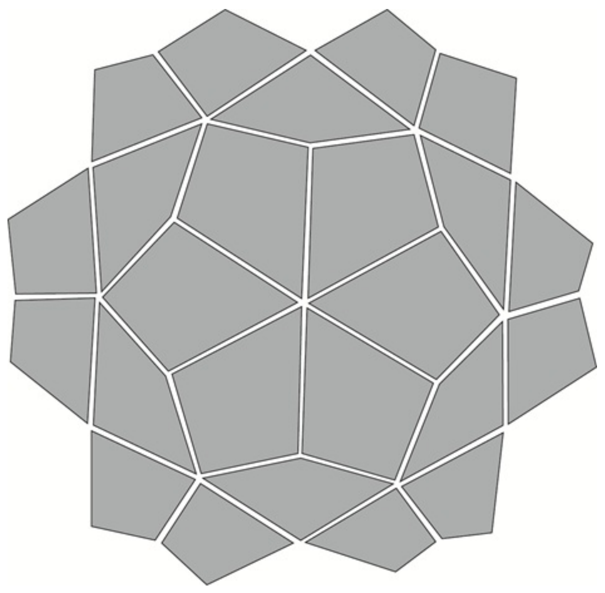
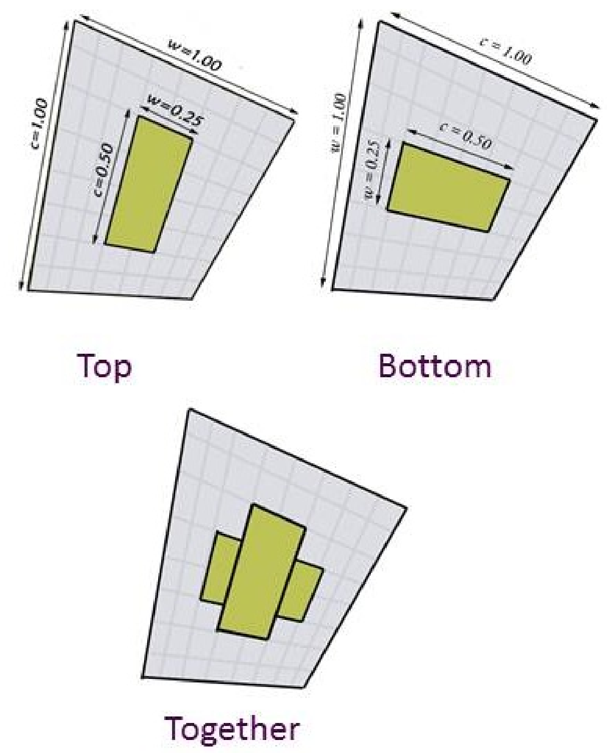
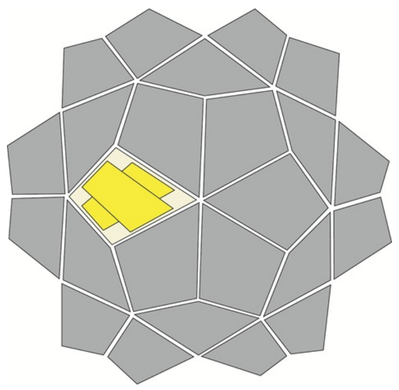


Fig. 19. All edges regions cover the initial mesh.

We now create two planar quadrilaterals by projecting edge regions to the planes that are slightly above and below the original 2D plane. For both the top and bottom quadrilaterals, we compute a fractional quadrilateral from the projected quadrilateral controlled by the two parameters w and c, as seen in Figure 20.

The fractional quadrilateral is computed as a fraction of the projected quadrilateral with user controlled fractional values w and c using bilinear interpolation. Here we will discuss how to compute the fractional quadrilateral for the top projected edge region, as seen in Figure 21. The caudation for the bottom part is symmetric in the sense that parameters c and w are switched in place. We denote the 3D position vectors of the four corners of the top projected quadrilateral as P_{vs}, P_{cr}, P_{vt}, and P_{cl}. The parameters c and w take values between 0.0 and 1.0. c controls how curvy the weaving ribbons will be. w controls the relative widths of the weaving ribbons.





(A) Bilinear interpolation of the projected quadrilaterals

(B) One edge region (in grey) has a cross (in yellow)

Fig. 20. Create a cross for all edge regions.

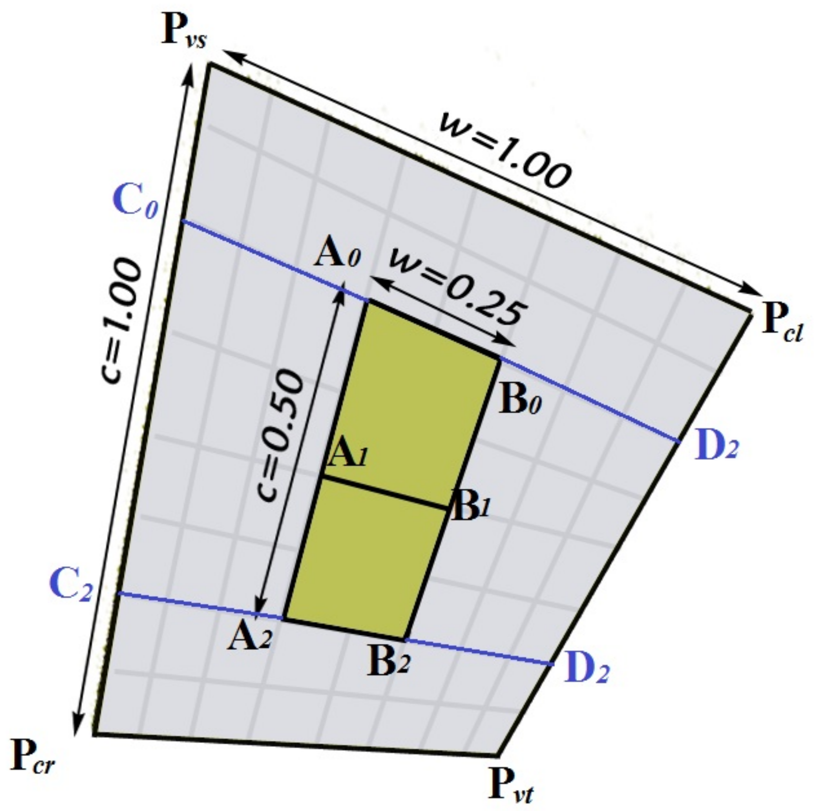


Fig. 21. Bilinear interpolation to compute the cross for each edge region.

The bilinear interpolation is a two-pass linear interpolation process. In the first pass of the linear interpolation, we have the 3D position vector C_0, C_2, D_0, and D_2 as interpolations of the four corners of the top projected quadrilateral as P_{vs}, P_{cr}, P_{vt}, and P_{cl} controlled by the parameter c. The equations are below:

$$C_0 = P_{vs}\frac{1+c}{2} + P_{cr}\frac{1-c}{2} \tag{4.2}$$
$$C_2 = P_{vs}\frac{1-c}{2} + P_{cr}\frac{1+c}{2} \tag{4.3}$$
$$D_0 = P_{cl}\frac{1+c}{2} + P_{vt}\frac{1-c}{2} \tag{4.4}$$
$$D_2 = P_{cl}\frac{1-c}{2} + P_{vt}\frac{1+c}{2} \tag{4.5}$$

In the second pass of the linear interpolation, we have the 3D position vector A_0, A_2, B_0, and B_2 as interpolations of the four 3D position vectors C_0, C_2, D_0, and D_2 controlled by the parameter w. The equations are below:

$$A_0 = C_0\frac{1+c}{2} + D_0\frac{1-c}{2} \tag{4.6}$$
$$B_0 = C_0\frac{1-c}{2} + D_0\frac{1+c}{2} \tag{4.7}$$
$$A_2 = C_2\frac{1+c}{2} + D_2\frac{1-c}{2} \tag{4.8}$$
$$B_2 = C_2\frac{1-c}{2} + D_2\frac{1+c}{2} \tag{4.9}$$

Then we calculation the middle points (A_1 and B_1) of the line segments A_0A_2 and B_0B_2.

$$A_1 = \frac{A_0 + A_2}{2} \tag{4.10}$$
$$B_1 = \frac{B_0 + B_2}{2} \tag{4.11}$$

Next this fractional quadrilateral is subdivided by creating two quadrilaterals along the thread, in the same direction as c. The whole process is illustrated in Figures 21. If the values of c and w are not the same, these shapes form crosses in space.

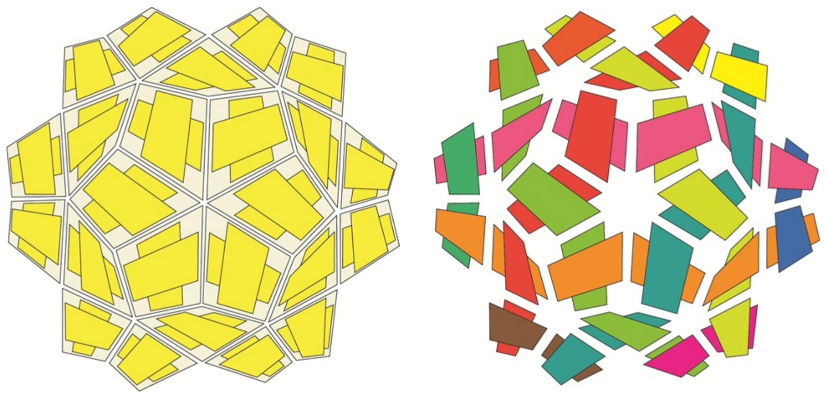


Fig. 22. Weaving thread structure in color.

For each edge, we shape the two projected quadrilaterals, which are slightly above and below of the plane. We use two parameters c and w to control the size of the projected quadrilaterals. New quadrilaterals form an "X" shape since they are on top of each other as seen in Figure 22.

We connect every pair of consecutive fractional quadrilaterals along each weaving thread by connecting the two tail corners of the previous quadrilateral and the two head corners of the next quadrilateral. We call this kind of new quadrilaterals "connectors", as see in Figure 23. In 2D case, these connectors are planar. In 3D case, they do not necessarily have to be planar.

After we connect all quadrilaterals, weaving threads become completely visible. Thread structures can be made even more visible by coloring threads with different colors based on the weaving thread tracing alright as seen in Figure 24.

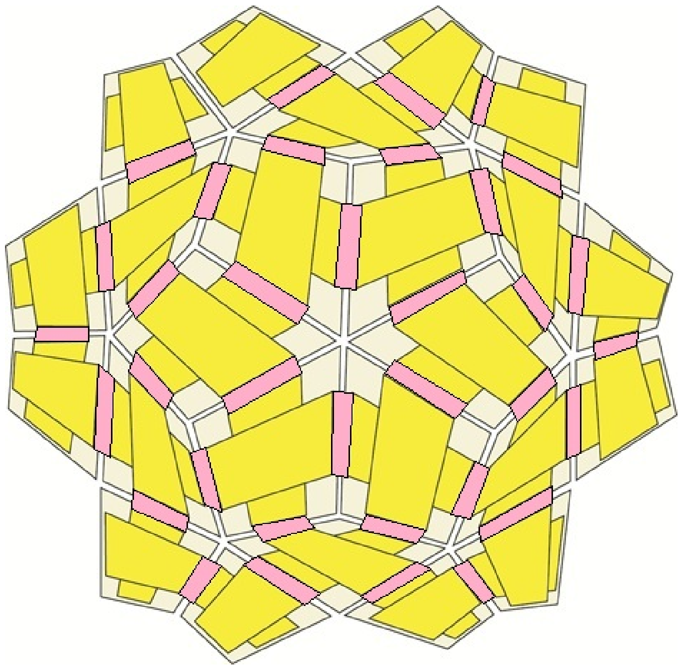


Fig. 23. "Connectors" (in pink) consecutive fractional quadrilaterals along each weaving thread.

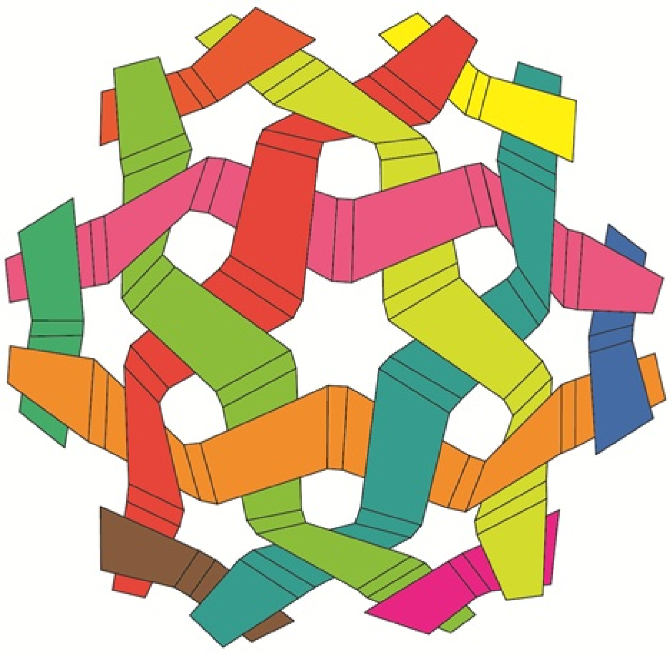


Fig. 24. Different weaving threads are colored differently.

IV.1.2. Projection Method in 3D

The algorithm for polygonal mesh surfaces in 3D is almost the same with the same steps. The only difference is that the edge regions are not guaranteed to be planar anymore. The first step is to construct a control polygon for each weaving cycle which we call the *PR control polygon*. The second step is to construct a developable ribbon from the PR control polygon. Without loss of generality, we explain the steps of the process using the mesh of a cube in 3D space.

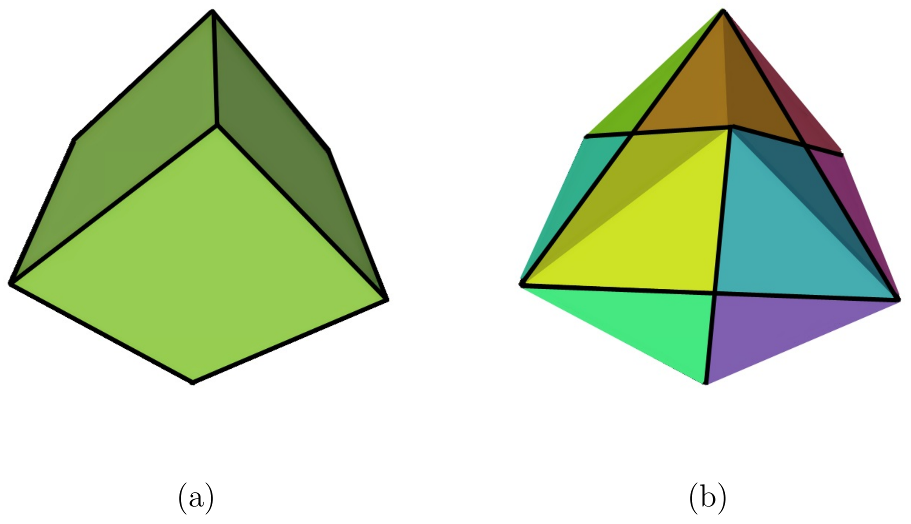


Fig. 25. (a) The initial mesh (a cube with all its edges 1^+-twisted) to create a cyclic plain-weaving. (b) Quadrilateral edge regions that are obtained from the two endpoints of the edge and centers of the two faces on the two sides of the edge.

We start to compute the center of mass for each face of the input mesh. Then we convert each edge to a quadrilateral by connecting the two end vertices of the edge and the two centers of the two faces on the two sides of the edge as seen in Figure 25. Each quadrilateral is still called an *edge region*. As we can see more obviously, that the edge regions in the 3D case are not planar. So we will project these on-planar quadrilaterals (edge regions) to a plane in the next step. We define the projection plane for each edge by the edge middle point and edge normal. We denote the 3D position of the two end vertices of an edge as P_{vs} and P_{vt}. The middle point of the edge is the average of the two end vertices.

$$P_m = \frac{P_{vs} + P_{vt}}{2} \tag{4.12}$$

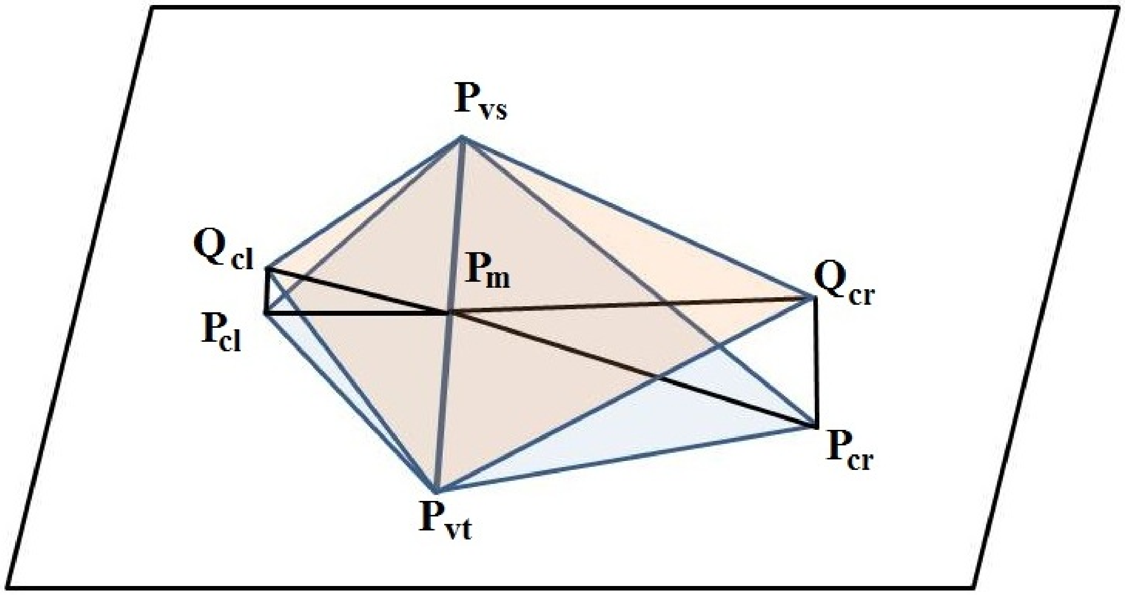


Fig. 26. We project the non-planar edge region to the plane defined by edge middle point and edge normal.

The unit edge normal N_e is computed as the average of the unit normal vectors N_l, N_r of the two faces sharing the edge.

$$N_e = \frac{N_l + N_r}{2} \tag{4.13}$$

The edge middle point P_m and the edge N_e define a unique plane. Let P be any point on the plane, the implicit plane equation is:

$$N_e \cdot (P - P_m) = 0 \tag{4.14}$$

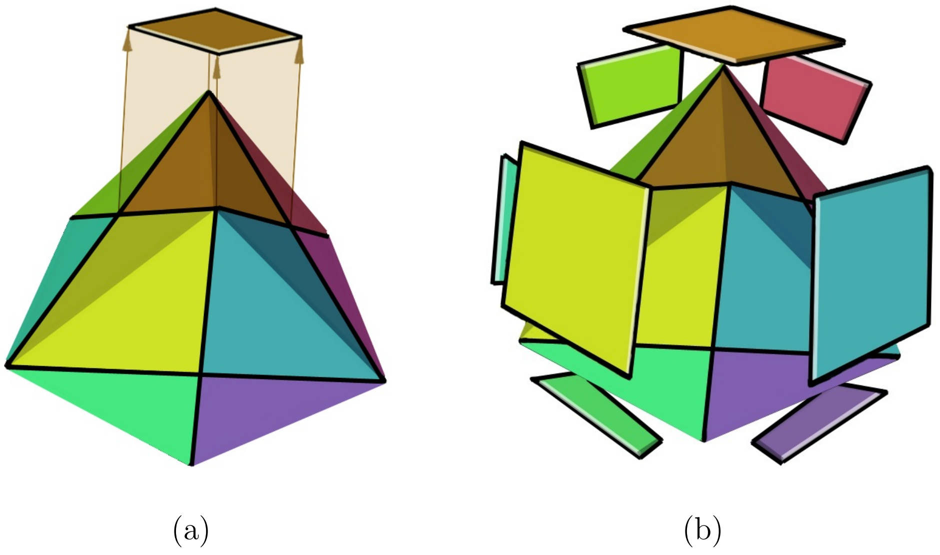


Fig. 27. (a) A projection of an edge region to one of the its corresponding projection planes. (b) All the projected edge-regions.

We project the edge region to this plan. The edge is inside the plane, so the two end vertices P_{vs} and P_{vt} are in the plane. We only need to project the two face centers P_{cl} and P_{cr} to the plane. We denote the two projected face centers as Q_{cl} and Q_{cr}, as seen in Figure 26 Their equations are:

$$Q_{cl} = (P_{cl} - P_m) - [(P_{cl} - P_m) \cdot N_e]\, N_e \tag{4.15}$$

$$Q_{cr} = (P_{cr} - P_m) - [(P_{cr} - P_m) \cdot N_e]\, N_e \tag{4.16}$$

To get the crosses of the weaving, we displace the projected edge region along the edge normal N_e such that one is slightly below the edge, and the other one is slightly above the edge to get to overlapping quadrilaterals, as seen in Figure 27. The user controlled parameter h is a small positive real number that controls the displacement distance.

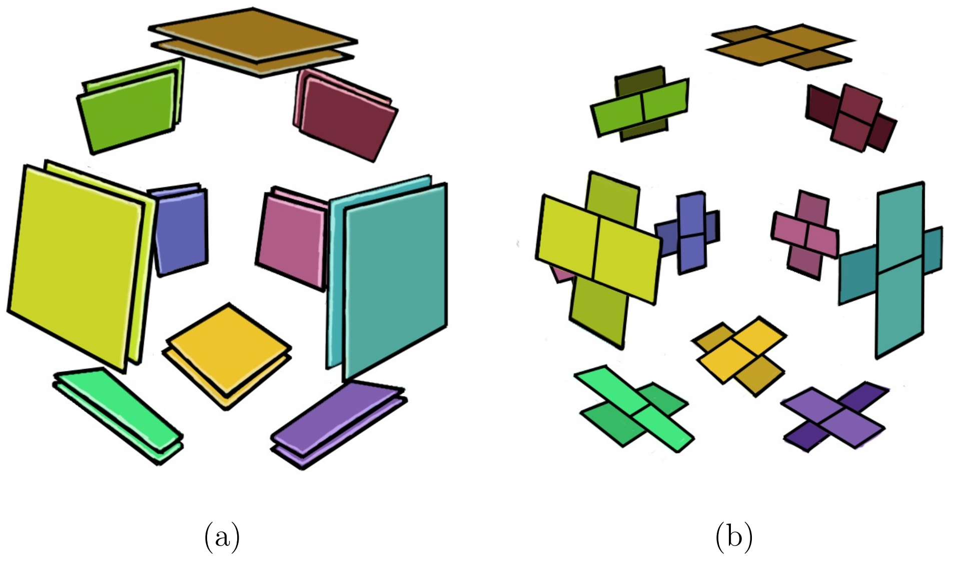


Fig. 28. (a) All the projected edge-regions. (b) The pairs of "X" shaped crosses by bilinearly interplaiting the projected edge regions.

The same as in the 2D case, for both the top and bottom quadrilaterals, we compute a fractional quadrilateral from the projected quadrilateral controlled by the two parameters w and c. We have the pairs of "X" shaped fractional projected quadrilaterals for each edge, as seen in Figure 28.

We connect every pair of consecutive fractional quadrilaterals along each weaving thread by connecting the two tail corners of the previous quadrilateral and the two head corners of the next quadrilateral. The "conector" quadrilaterals are colored with pink as seen in Figure 29.

We use the same technique to create smooth Beziér ribbons as seen in Figure 30. Control polyongs are constructed by smoothing PR control ribbons as cubic Beziér

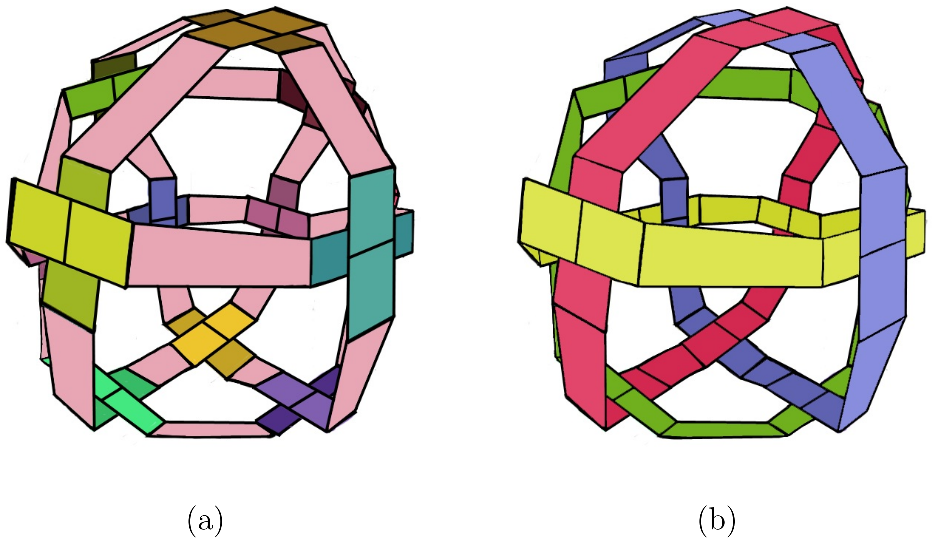


Fig. 29. (a) Pink-colored quadrilateral connecters that connect two corresponding planar pieces. (b) The resulting control meshes with one consistent color for each cycle.

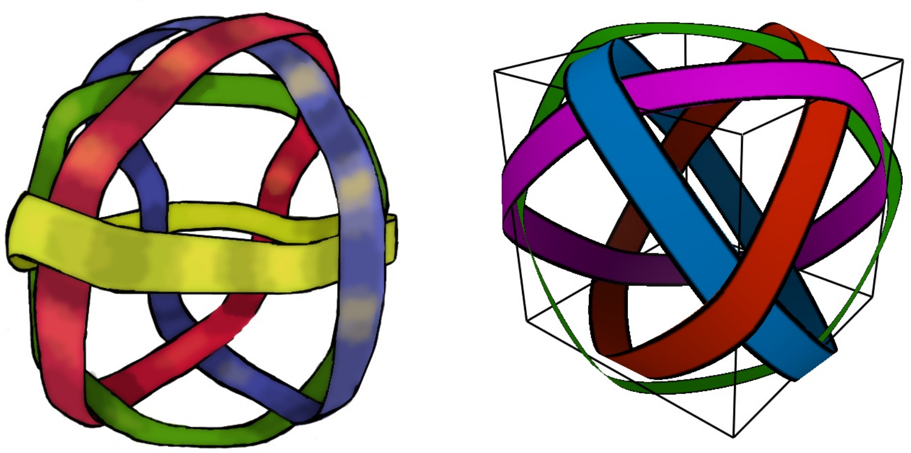


Fig. 30. The final smooth ribbon that is created as Beziér ribbons. The image on the left is an illustration. The image on the right is rendered by my program.

ribbons that use one connector and two side quadrilaterals as a control mesh. Smooth ribbons are constructed by connecting the sample points along the two cubic Beziér curves. We get a series of skinny triangles. If the sampling is dense, the weaving ribbons are smooth. The ribbons can cover planar tiling regions well if there is no star-shaped polygon. We will unfold these weaving ribbons and them cut them out by a laser cutter for physical construction. Unfolded ribbons are usually wavy.

IV.2. Panel Creation

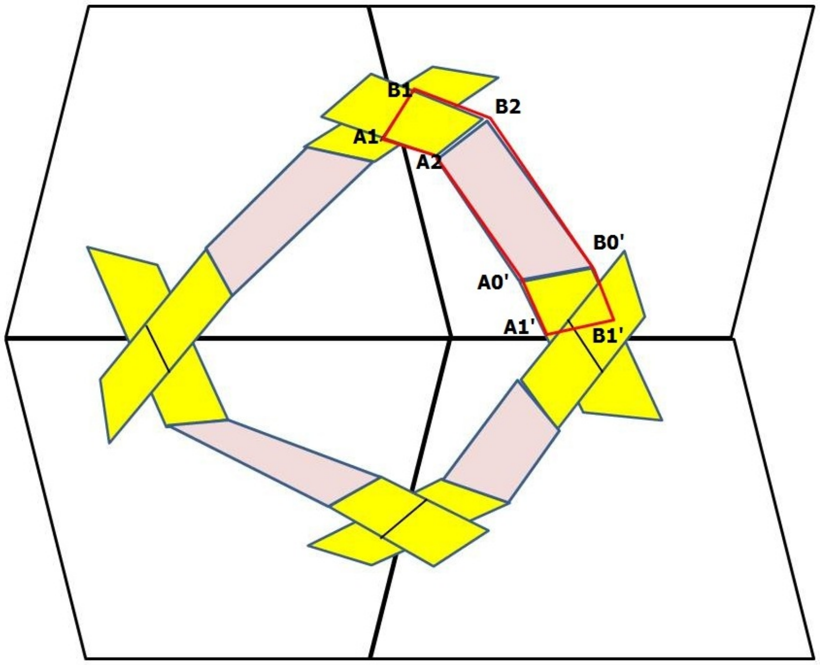


Fig. 31. Control polygon for one segment of the weaving ribbon consists of three parts: two half of fractional quadrennials and one connector (outlined in red).

Until now each weaving thread has a geometry form as a series of fractional quadrilaterals and connectors inbetween. We want each weaving thread as a smooth ribbon. We use the piecewise ribbon approach. Control polygons for each segment of the weaving ribbon are obtained by adding one connecting quadrilaterals inbetween two half of the projected quadrennials along each of the weaving thread, as seen in Figure 31.

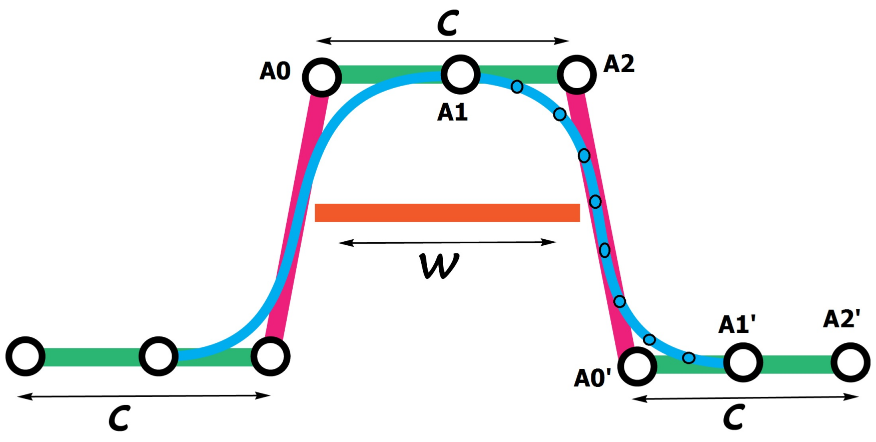


Fig. 32. The two Beziér curves (in blue) have same tangent at A_1.

Control polygons are smoothed to obtain smooth ribbons. We use each of the long sides (poly-line) as control line to compute two cubic Beziér curves on both sides as seen in Figure 32. The four control points for one piece of the cubic Beziér curve are A_1, A_2 from one fractional quadrantal and A_0', A_1' from the following fractional quadrantal. The curve starts at A_1 going toward A_2 and arrives at A_1' coming from the direction of A_0'. Usually, it will not pass through A_1 or A_0'. These points are only there to provide directional information. The distance between A_1 and A_2 determines "how long" the curve moves into direction A_0' before turning towards A_1'. The parameter equation of the cubic Beziér curve is:

$$P(t) = (1-t)^3 A_1 + 3(1-t)^2 t A_2 + 3(1-t)t^2 A_0' + t^3 A_1' \tag{4.17}$$

where t is a parameter which takes value from 0.0 to 1.0. $P(t)$ is the 3D position vector of a point corresponding to parameter t on the cubic Beziér curve. When t is equal to 0.0, $P(0.0)$ is A_1. When t is equal to 1.0, $P(1.0)$ is A'_1. When t takes a value inbetween, $P(t)$ is a point along the cubic curve.

The cubic Beziér curve has vector $V_{12} = A_1 - A_2$ and $V'_{10} = A'_1 - A'_0$ at the two end as tangent vectors. Since A_1 is computed as the middle point of A_0 and A_2 in Figure 32, the two 3D vectors V_{12} and V_{10} are in the same line and have same length. This approach guarantees that the resulting piecewise smooth curves have C^1 continuity at their touching point A_1.

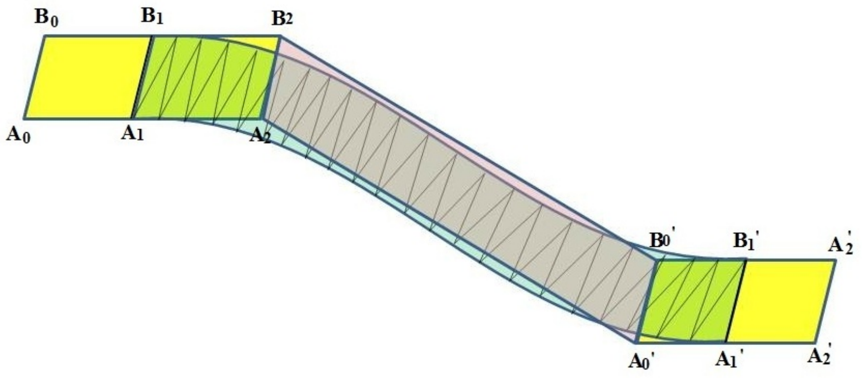


Fig. 33. An illustration of the smooth ribbon creation process.

Smooth ribbons are constructed by connecting the sample points along the two cubic Beziér curves, as seen in Figure 33. We get a series of skinny triangles. If the sampling is dense, the weaving ribbons are smooth. The ribbons can cover planar tiling regions well if there is no star-shaped polygon. We will unfold these weaving ribbons and them cut them out by a laser cutter for physical construction. Unfolded ribbons are usually wavy.

The weaving geometry obtained from the projection method results in ribbons that are mostly long and wavy. If unfolded, these ribbons can intersect with themselves. Therefore, we need to decompose the ribbons into short panel segments for fabrication and assembling. I will discuss how we actually decompose these ribbons into panels later in section IV.4. In the following section, I will assume that ribbons are already decomposed into a set of shorter panels. After we decompose the long ribbons into short panels, we need to unfold each panel into a plane.

IV.3. Unfolding Panels

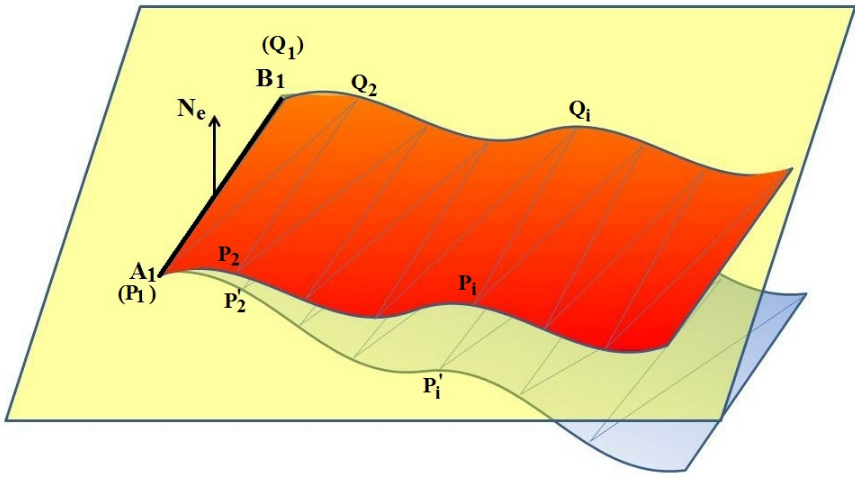


Fig. 34. Unfold (flatten) one weaving panel to the projection plane.

We unfold each weaving element to the plane defined by the edge normal N_e and the two 3D points A_1 and B_1 with respect to the middle "up" edge as shown in Figure 34. The plane contains A_1 and B_1 and is perpendicular to N_e. We unfold half of the series of triangles starting from A_1 and B_1. The other half of the weaving panel can be unfolded in the same way. We start from the first triangle. We first translate it so that its one side lies in the unfolding plane. Then we rotate the triangle along that edge so that the third vertex is in the unfolding plane. This is an iterative algorithm.

We denote the points on one side of the weaving elements as $P_1, P_2, ..., P_n$ and points on the other side of the weaving elements as $Q_1, Q_2, ..., Q_n$, while n is the number of sample points on one side of the weaving elements. We denote the points on one side of the unfolded weaving elements as $P'_1, P'_2, ..., P'_n$ and points on the other side of the unfolded weaving elements as $Q'_1, Q'_2, ..., Q'_n$ correspondingly.

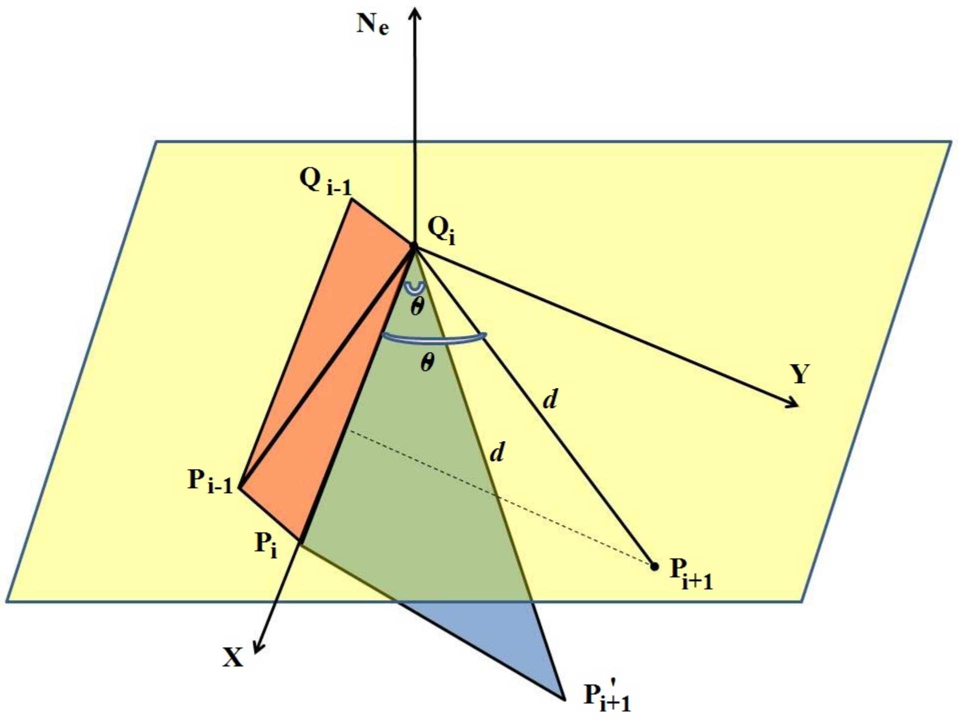


Fig. 35. Rotate one triangle (in blue) of the weaving panel to the projection plane.

For the first triangle, its side P_1Q_1 is the same as A_1B_1. So we only need to rotate it along P_1Q_1 so that the third point Q_2 lies on the unfolding plane. Next, we translate the second triangle so that its side P_1Q_2 overlaps with the side of the first unfolded triangle $P'_1Q'_2$. Suppose the previous triangles are already transformed to the unfolding plane, now we are going to place the triangle $P'_iP'_{i+1}Q'_i$ on the unfolding plane, as seen in Figure 35.

We first establish the local frame at point Q_i. Edge normal vector N_e is already a unit vector. It serves as one axis. The two unit vectors X and Y are the other two axes. The equations to compute X and Y are below:

$$X = P_i - Q_i \tag{4.18}$$
$$X = \frac{X}{\|X\|} \tag{4.19}$$
$$Y = N_e \times X \tag{4.20}$$
$$Y = \frac{Y}{\|Y\|} \tag{4.21}$$

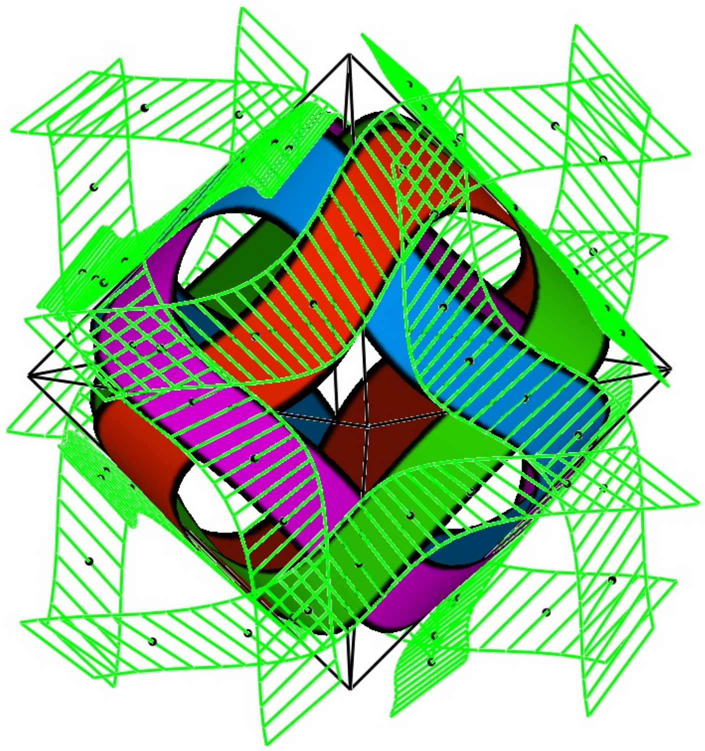
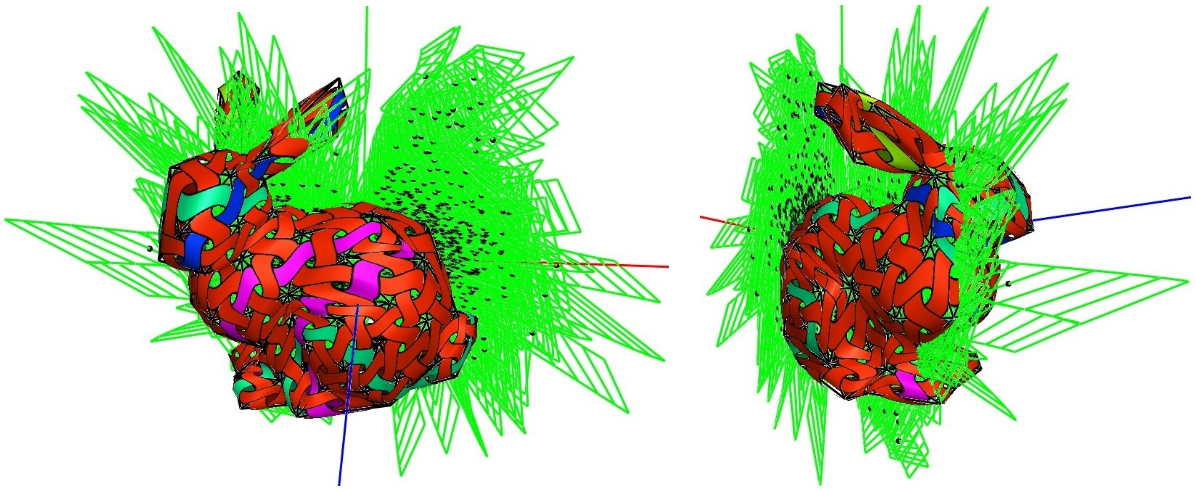


Fig. 36. Unfolding panels of a plain woven structure created from an octahedron.

Then we compute the angle θ between the two sides $P'_iQ'_i$ and $P'_{i+1}Q'_i$ of the triangle $P'_iP'_{i+1}Q'_i$. We also calculate the length d of the side $P'_{i+1}Q'_i$.

$$
\begin{aligned}
V &= P'_{i+1} - Q'_i && (4.22)\\
d &= \|V\| && (4.23)\\
V &= \frac{V}{d} && (4.24)\\
\theta &= \arccos(X \cdot V) && (4.25)\\
P_{i+1} &= Q_i + \cos(\theta)dX + \sin(\theta)dY && (4.26)
\end{aligned}
$$



(a) Front view

(b) Back view

Fig. 37. Transform the unfold weaving elements to the XY plane.

Using these equations, we unfold each triangle one-by-one. An example of unfolding of each panel of a plain woven structure is shown in Figure 36. The flattened weaving elements are then transformed and arranged on 2D plane. The Figure 37 shows the flattened weaving panels for the Bunny model are transformed onto a 2D plane.

The weaving elements are later arranged within the bounding box of the output paper size. Our program outputs the weaving elements in .eps file format. This file format can be read by the laser cutter directly. Figure 38 shows the weaving panels for the bunny input mesh are output in a .eps file.

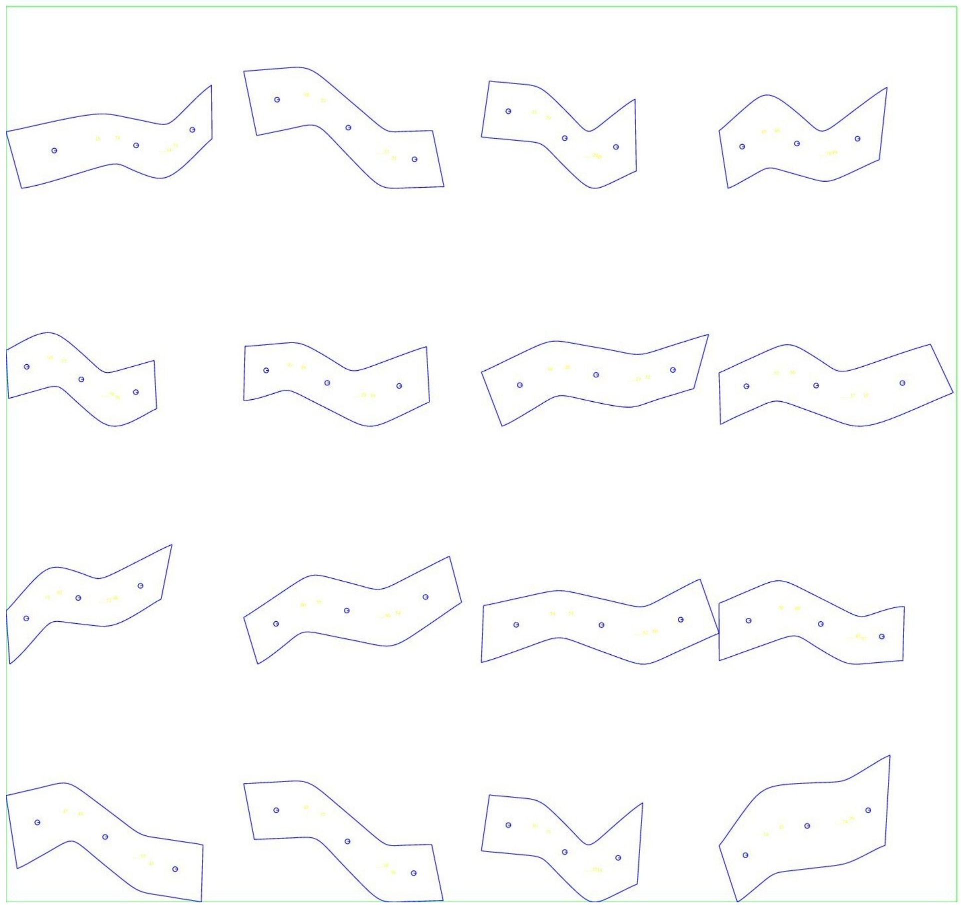


Fig. 38. The weaving elements are written into .eps files.

We print the flattened weaving panels of each cycle one by one. Also, within each cycle the weaving panels are ordered by their relative position on the weaving cycle. Therefore, the order that weaving panels are output is the same as the order that they should be assembled. This strategy greatly reduces the construction time because workers do not have to search matching pieces among the sea of thousands of weaving panels.

IV.4. Numbering and Assembling

As we can see from Figure 39, the weaving cycles interlace with each other. If we trace along one weaving cycle, we encounter edges of the base mesh. The weaving cycle goes up the edge, then under the next edge and so on.

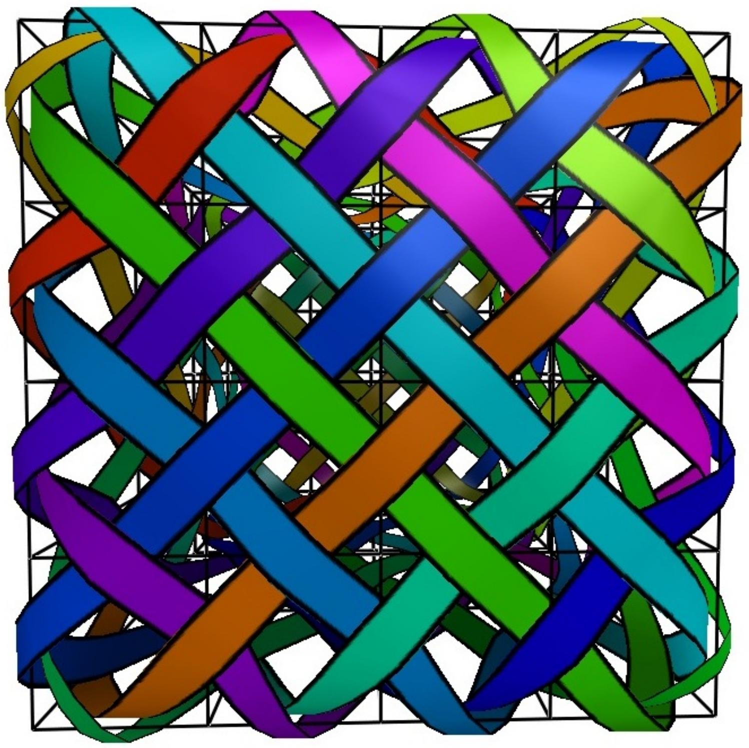


Fig. 39. Example of weaving geometry on a mesh.

So we segment all the weaving cycles by a down-up-down pattern, as seen in Figure 40. We burn two holes at the down/up position for assembling purpose. If the position is a “down”, we arrange the two holes vertically. If the position is a “up”, we arrange the two holes horizontally. The directions of holes match at the same edge for the two overlaying weaving cycles. We also mark the edge indexes near the holes to help the builder identify the connectivity information. If two weaving panels have same edge indices at their ends, these two weaving panels are two consecutive segments along the same weaving cycle. If one weaving panel has an edge index in its middle the same with other two weaving panels, this weaving panel goes above the underling weaving cycle. So together the three matching weaving panels make up a weaving cross for the edge.

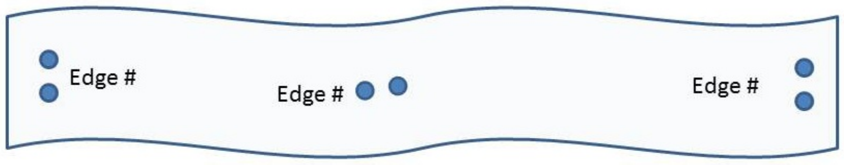


Fig. 40. One segmentation piece of a weaving cycle.

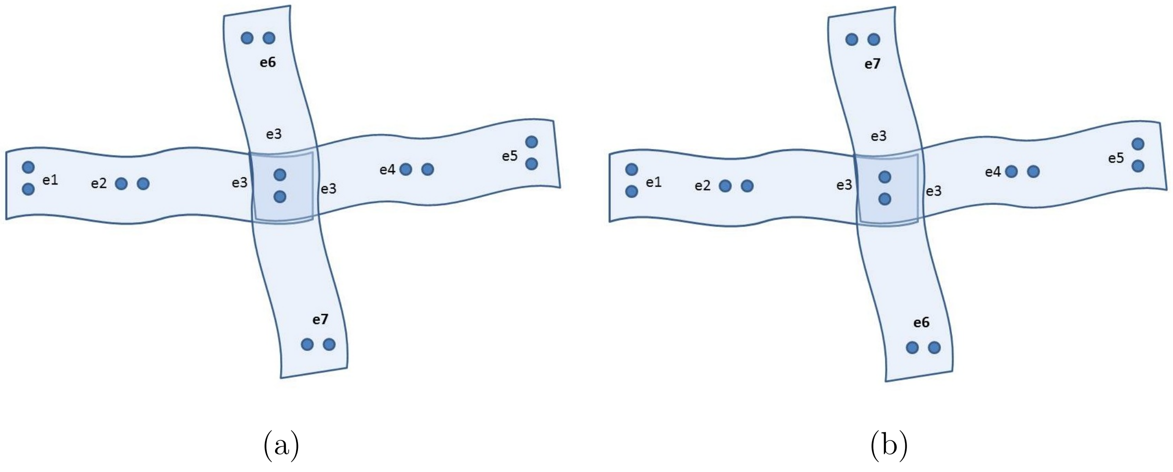


Fig. 41. There are two possible ways for assembling the three matching weaving panels.

At every edge, we join the crossing of two weaving cycles using two braids through the holes on the matching pieces, as seen in Figure 41 (a). The ambiguity arises. For the same three matching weaving panels, there are two possible orientation to join them.

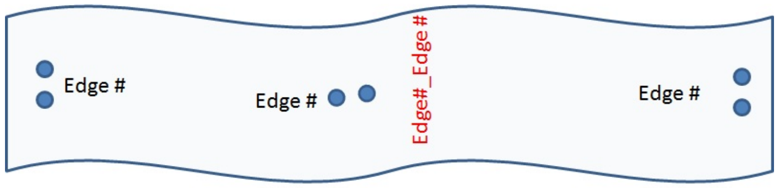


Fig. 42. Extra neighboring edge index (in red).

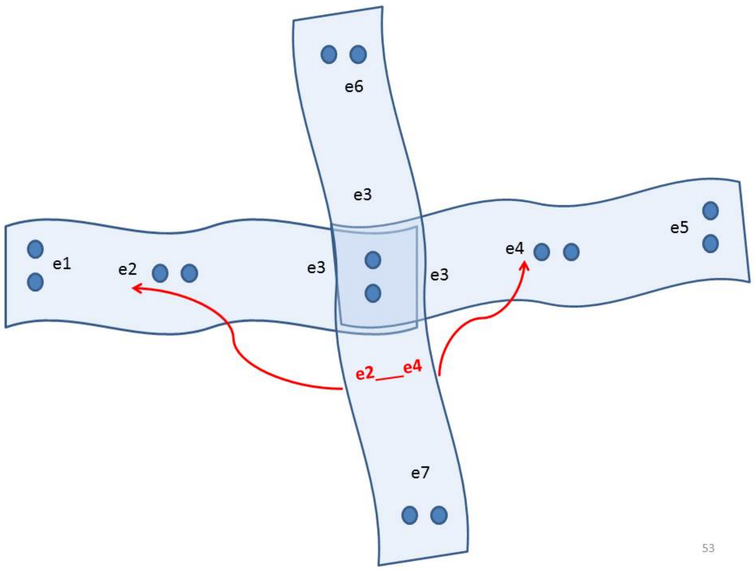


Fig. 43. New weaving panels with additional information.

To eliminate the ambiguity, we add additional information on the weaving panels. We differentiate the neighboring edge index with other edge indices marked on the weaving panels with underline bars. In Figure 42, we use red color to emphasize it. The laser cutter either cuts through the material or etches on the material. So we use a special underline bar to indicate the difference.

In Figure 43, the red double-headed arrow points to the two corresponding edge indices. When one panel serves as the “up” weaving cycle, its neighboring edge index must be the same as the edge index for the “up” position as we rotate counterclockwise.

CHAPTER V

BAND DECOMPOSITION STRUCTURES

Band decomposition is another conceptual idea coming from Graph Rotation Systems. It represents the shape of the corresponding surface with a set of bands. I have developed a method to automatically convert a graph rotation system to a band decomposition. Graph Rotation Systems are graphs embedded on surfaces, as we have discussed in section III.1.1. A graph is 1-complex that does not have thickness, i.e. graphs do not have a solid shape. For practical applications, the graph need to be converted to 3D geometry for physical construction. I have developed a process to convert the graph rotation systems to star-shaped panels that can be easily assembled to construct a band decomposition structure. The process consists five steps.

1. **Geometry Conversion:** This step is used to convert graphs to 2-D thickened band decomposition. I have developed a method called ”2-D thickening” for geometry conversion, which is discussed in Section V.1.

2. **Panel Creation:** This step is used to segment band decomposition to obtain assembly panels, that are star-shaped vertex components. This step is discussed in Section V.2.

3. **Developable Panel Conversion:** This step is used generate smooth developable bands in 3D virtual space. This step is discussed in Section V.3.

4. **Unfolding Panels:** This step is used for unfolding and placement of assembly panels. This step is discussed in Section V.4.

5. **Numbering and Assembling:** In this step, we label connectivity information on panels to simplify the assembly process based on the numbers marked on

each panel. This step is discussed in Section V.5.

The following sections describe the five steps of the procedure to construct band decomposition structures.

V.1. Geometry Conversion

In this section, I present geometry conversion using 2D-thickening method. I first introduce the theoretical background on 2D-rotations and 2D-thickenings and mesh operations to obtain dynamically changing progressive surfaces.

The theory of surfaces (also called 2-manifolds) is well-established in mathematics and shape modeling. In reviewing some details of rotation systems for graphs, we emphasize the viewpoint of band-decompositions, since that is the approach we will use to obtain occlusions, which are a key to visualization of large networks.

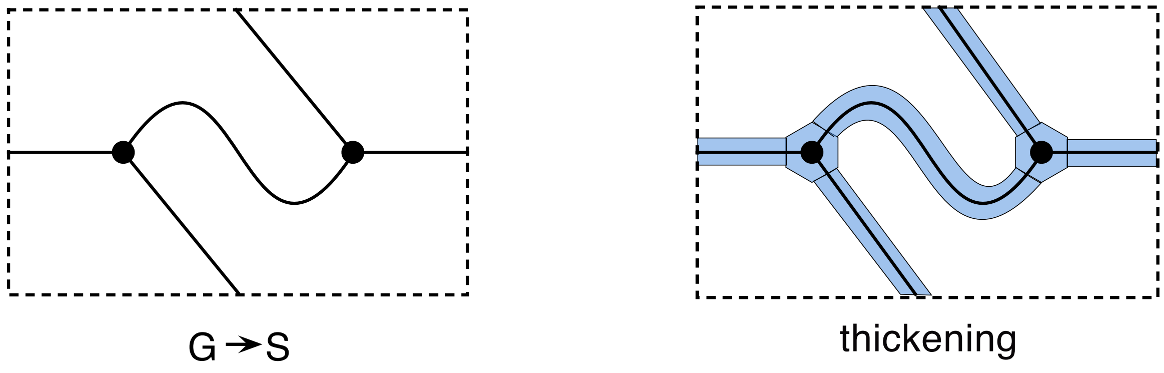


Fig. 44. A graph G in a torus and the associated 2D-thickening.

Imagine a finite graph G embedded in a closed oriented surface S. One usually requires that the embedding be *cellular*, so that each component of $S - G$ is homeomorphic to an open disc. The graph G can be “thickened” in the surface S by enlarging each vertex of G to a small polygonal disk called a *vertex-band* and widening each edge to a narrow rectangular called *edge-band* that joins the vertex-bands at its endpoints. The union of the vertex-bands and the edge-bands is topologically a surface T with boundary, called a *thickening of the graph* G (see Figure 44), and also called a *band decomposition.*

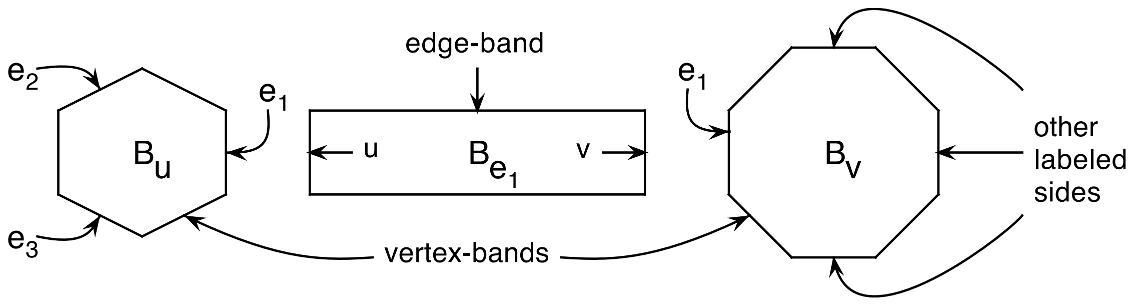
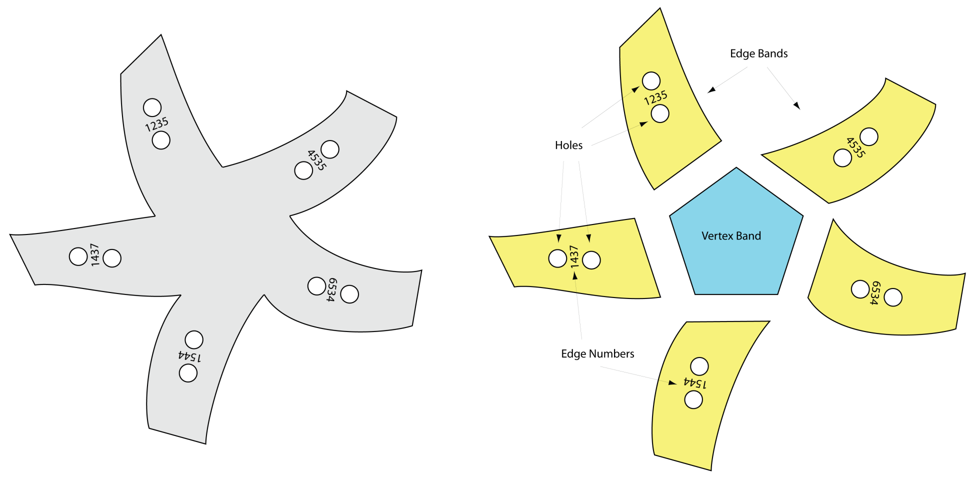


Fig. 45. Identifying an edge-band B_{e_1} with vertex-bands B_u and B_v.

A rotation system ρ can serve as instructions for assembling the vertex-bands and the edge-bands into a thickening of G (see [30]). If at vertex u the cyclic edge order is $e_1, \cdots, e_k$, then the vertex-band B_u for vertex u is a $2k$-gon with k sides that are consecutively labeled (in cyclic order) $e_1, \cdots, e_k$. If edge e is incident to vertices u and v of the graph G, then the edge-band B_e is a "rectangular" strip with its two opposite end labeled u and v, as shown in Figure 45. The vertex-bands and the edge-bands each have a prescribed orientation. We paste the sides labeled u and v on the edge-band B_e to the sided labeled e on the respective vertex-bands B_u and B_v, so that the orientations agree. We make two observations about this construction.

1. The graph G is a *spine* of the thickening T: that is, the surface T has a deformation retraction onto the embedded graph G (i.e., a homotopy $T \to T$ that is everywhere the identity mapping on G and that ends with a mapping of T onto G). Intuitively, the thickening T contains the graph G, together with a local product structure that can be used to shrink T back to G.

2. If we construct a closed surface associated with the thickening, then we can fill each hole with a disk. The resulting cellular embedding for G is uniquely determined, up to homeomorphism type, by the rotation system ρ.



(a) (b)

Fig. 46. Initial vertex component.

A vertex component consists of vertex-band and its edge-bends, as seen in Figure 46. In the next section, we describe how to obtain vertex and edge bands.

V.1.1. Initial Vertex and Edge Bands

We observe that the initial versions of vertex and edge bands can directly be obtained by using a corner-cutting scheme such as the Doo-Sabin subdivision [5, 19]. The result of the mesh refinement is shown in Figure 47, where the black vertices and edges constitute the new mesh, and the gray vertices and edges are in the old mesh (and removed at the end of the process). The refinement rules for a corner-cutting scheme are as follows:

Step 1: For each corner vertex v_n of an N-sided face $f = \{v_0, v_1, \ldots, v_n, \ldots, v_{N-1}\}$ of the old mesh (as in Figure 47a), create a new vertex v'_n. Compute its position as

$$v'_n = \sum_{m=0}^{N-1} a_{n,m} v_m \tag{5.1}$$

where $a_{n,m}$ are real coefficients described by the corner-cutting scheme.

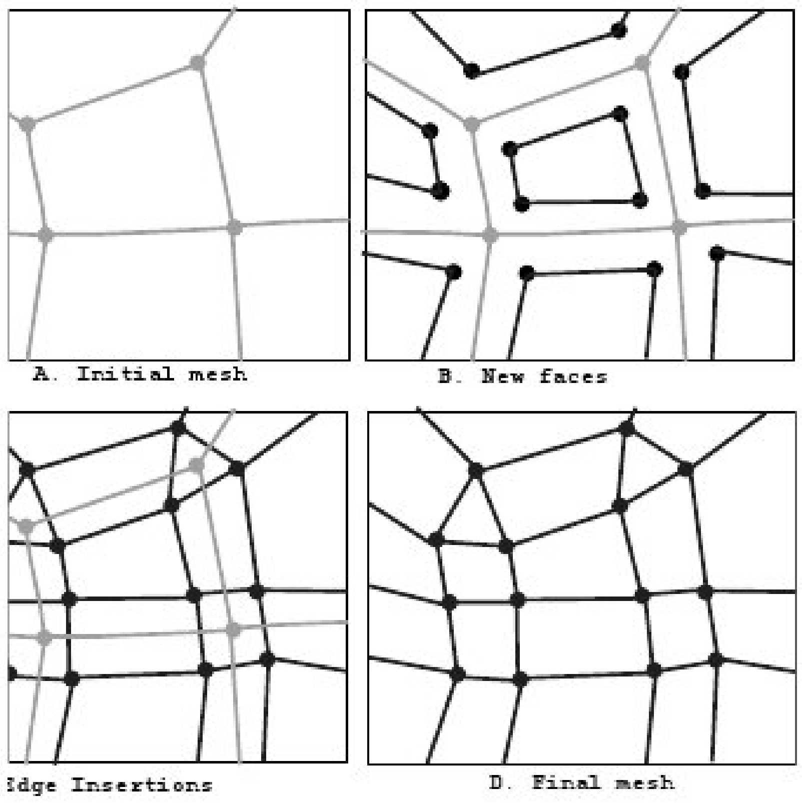


Fig. 47. Illustration of corner cutting remeshing scheme.

Step 2: For each face of the old mesh, create a new face (as in Figure 47 (b)) by joining each pair of new vertices lies near the endpoints of an old edge.

Step 3: For each old vertex v, create a new face (as in Figure 47 (c)) by joining each pair of new vertices that lie in a pair of old faces that meet at an edge incident at v.

Step 4: Remove all the old vertices and old edges.

With the corner-cutting scheme, each vertex becomes a face, and each edge becomes a quadrilateral. Thus, the corner-cutting scheme yields a band decomposition of a 2-manifold surface, as if by 2D-thickening its 1-skeleton. To obtain aesthetic results the choice of coefficients $a_{n,m}$ is important since it uniquely defines the geometry. For any subdivision scheme using approximation, the coefficients $a_{n,m}$ in Step 1 of the above algorithm must satisfy the following conditions (we assume that the vertex indices are given in the order of a face-boundary traversal):

1. $a_{n,m} \geq 0$, for all n and m, and
2. $\sum_{m=0}^{N-1} a_{n,m} = 1$, for all n.

These conditions guarantee convergence of the algorithm and provide C^0 continuity and affine invariance properties [34, 73]. To obtain a good geometric quality, we use a generalized corner-cutting scheme that also includes Doo-Sabin subdivision. For generalized corner cutting scheme the coefficients $a_{n,m}$ in equation (5.1) are computed using the following formulas:

$$\begin{aligned} a_{n,m} &= a && \text{if } n = m \\ a_{n,m} &= M\frac{1-a}{3N-5} && \text{otherwise} \end{aligned} \tag{5.2}$$

where

$$M = 3 + 2\cos\left(\frac{2(n-m)\pi}{N}\right).$$

In these equations, parameter a in equation (5.2) is provided by users and it is used as a tension parameter [10]. If a is chosen as

$$a = \frac{1}{4} + \frac{5}{4N},$$

the result is Doo-Sabin subdivision. This gives subdivision rules for generating quadratic B-spline tensor-product surfaces for $N = 4$ and $a = 9/16$. Note that, the coefficients add up to 1 and if $0 < a < 1$ each coefficient is greater than zero. By using Fourier analysis, we can show that the new scheme also has three distinct eigenvalues

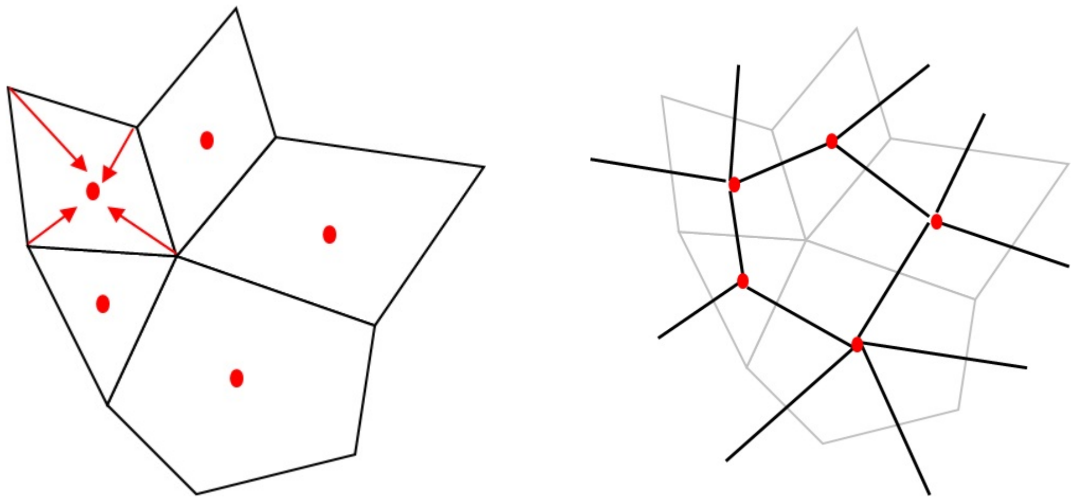
$$1, \frac{(2a+1)N-5}{3N-5}, \frac{3aN-5}{3N-5}$$

and if $a < 1$ then regardless of the value of N the sequence is strictly decreasing. For Doo-Sabin case these three distinct eigenvalues are $1, 1/2, 1/4$, regardless of face-size. Similar to Doo-Sabin scheme, only one eigenvector corresponds to eigenvalue 1, two eigenvectors correspond to the second largest eigenvalue and the rest of the eigenvectors correspond to the smallest eigenvalue. This structure of eigenvalues guarantee that the scheme provides tangent plane continuity. In other words, under the this scheme, each face eventually approaches a plane. It was shown in [5] that the Doo-Sabin algorithm is superior to other algorithms used to obtain reasonably planar and convex faces from any given mesh structure, even only with one application of the scheme.

There exists a lower limit over the value of a. To have positive eigenvalues, for a triangular face (i.e., $N = 3$) a must be larger than $1/3$ (Note that in triangle case, the third eigenvalue will not exist.) For a quadrilateral face (i.e., $N = 4$) a must be larger than $5/12$. Otherwise, the smallest eigenvalue can become negative. It is safer to choose $5/12$ as a lower bound for a since in corner cutting schemes we cannot avoid quadrilaterals (each old edge become a quadrilateral after the application of corner cutting scheme once.)

V.2. Panel Creation

The assembling elements of the band decomposition structure are panels. A initial panel consists of one initial vertex-band and its corresponding initial edge-bends. I create initial panels by applying a sequence of processes to the input polygonal mesh. I will use a region of the input mesh to demonstrate the process of constructing initial panels.



(a) compute center of mass for each face (b) input mesh (grey) and its dual mesh

Fig. 48. The process to compute its dual mesh from the input mesh.

Figure 48 illustrates the process of construct the dual mesh from an input mesh. The dual of a 2-manifold polygonal mesh without boundary is commonly defined as another mesh in which faces and vertices occupy complementary locations and the position of each dual vertex is computed as the center of mass of the vertices that support the corresponding face. Once we have the dual mesh, we apply the corner-cutting subdivision algorithm discussed in the previous section to the dual mesh.

Figure 49 illustrates the process of applying the corner-cutting subdivision algo-

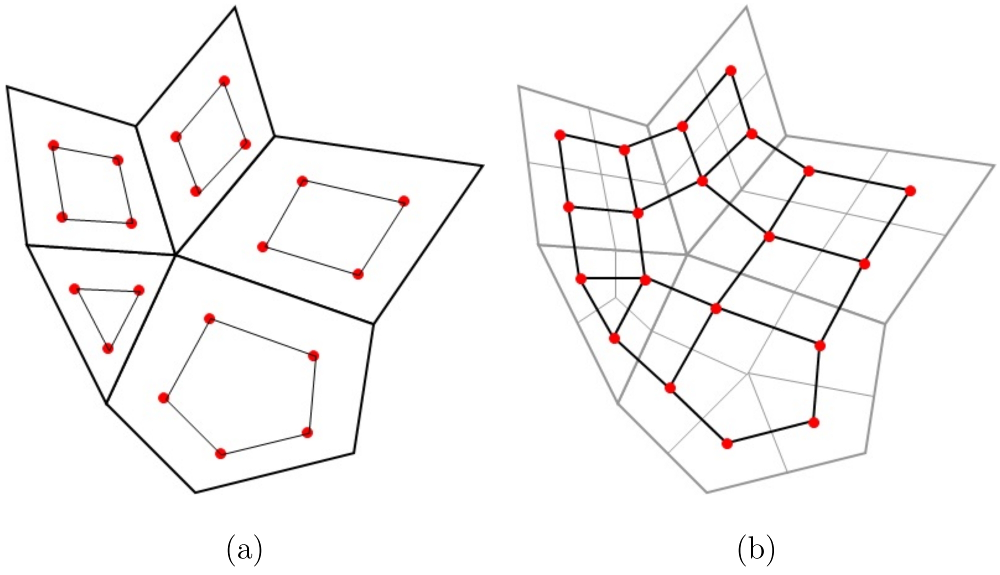


Fig. 49. The corner-cutting subdivision algorithm.

rithm to the dual mesh. We first compute similar polygon within each face and then connect new polygons from neighboring faces. So each original face becomes a similar face, each vertex becomes a polygon whose size is the original vertex's valance, and each edge becomes a quadrilateral which connects the two corresponding faces of the original faces that share the original edge.

This improves the quality of final geometry making the vertices of each face almost planar. Then we project the vertices of each red face to its approximating plan to make sure all the red faces are planar.

As we discussed in the previous section, each vertex in the original mesh corresponds to a vertex band which are the yellow faces in Figure 50, each edge in the original mesh corresponds to an edge band which are the blue faces, and each face correspond to a gap in the sculpture. Each penal consists of one vertex band and all edge bands of edges incident to it (in the red outline). The panels are the building elements for the band decomposition sculpture.

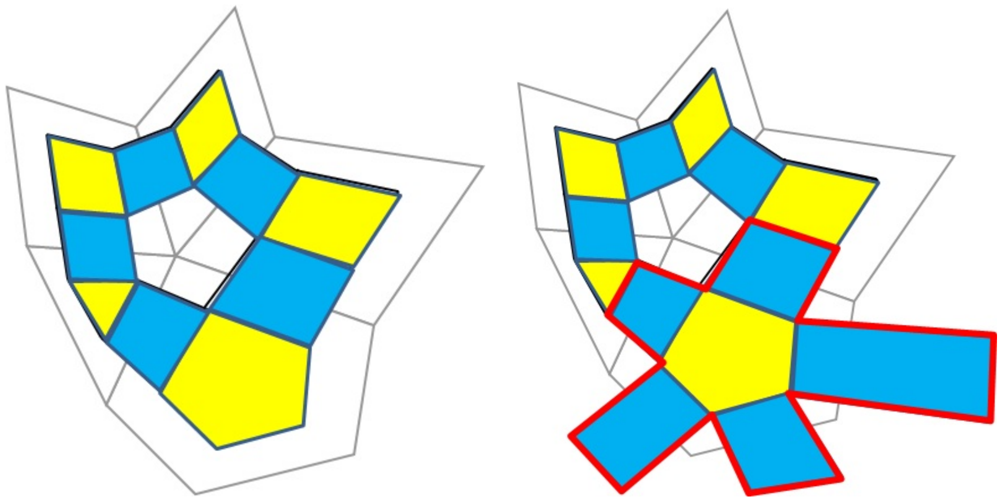


Fig. 50. A initial panel (red) consists of one initial vertex-band (yellow) and its corresponding initial edge-bends (blue).

V.3. Conversion to Developable Surfaces

We will cut the panels out of sheets of cardboard or steel. So we need to convert each panel to a developable component so that we can unfold them on a plane in the later stage. The geometry we obtained from described processes consists of pieces that are mostly developable surfaces. The initial vertex bands are already planar. But the initial edge components are quadrilaterals in 3D space. They are not necessarily planar. Therefore, the first step is to convert non-developable pieces to developable pieces. There are three cases that we need to deal with which are illustrated in Figures 51, 52, and 53 respectively.

- **Case 1:** As see in Figure 51, the two planes containing the two vertex bands connected by an edge band are parallel. In this case, we add the two corresponding edges from the two neighboring vertex band and divide the sum by two. We swipe the resulting line segment along a Hermit curve to get the band of a swiping surface. Figure 51 shows a Hermit curve which is tangent to the two planes at the two end points. We want the swiping surface to have a smooth transition with the vertex bands.

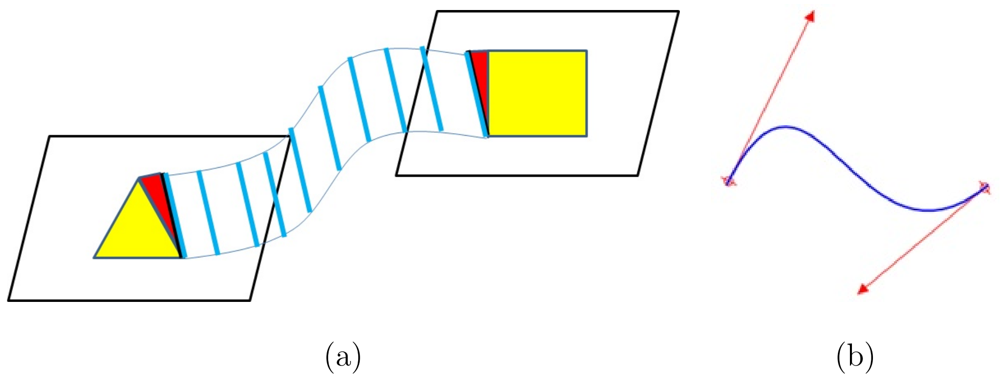


Fig. 51. Case 1: (a) An illustration of the smooth band creation process in case 1. (b) An Hermit curve defined by two end points and two tangent vectors at end points.

- **Case 2:** The second case is when the two planes containing the two vertex bands intersect. The intersection line has a small angle with the average of the two corresponding edges. Similarly, we can first define a Hermit curve. We calculate a line segment which is in the same direction as the intersecting line and have the average length of the two corresponding edges. And then we swipe the line segment along the curve to get a swiping plane as seen in Figure 52.

- **Case 3:** The third case is that the two planes containing the two vertex bands intersect. But the intersection line has a large angle with the average of the two corresponding edges as seen in Figure 53. In this case, we can not use the previous method to swipe a line segment along the Hermit curve that is tangent to both planes. Because we can not find the line segment which is parallel to both planes. We will not have smooth transition between the edge band and

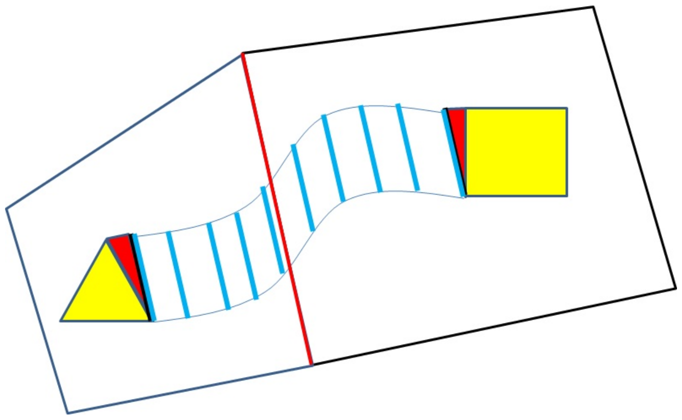


Fig. 52. Case 2: An illustration of the smooth band creation process in case 2.

the two vertex bands. We use bilinear interpolation of the two corresponding edges. We get a sequence of inbetween line segments which morph from one edge to the other. We connect every pair of neighboring line segments to get a sequence of thin triangles. The resulting developable surface is not smooth. The two triangles which share a common edge will have an angle. In practice, this will not cause problems. Since the two vertex bands are computed from the neighboring faces in the original input mesh, the angle between the two planes is very small. This angle is further distributed into the sequence of triangle. So even if there is an angle between two consecutive triangles, the angle is very small.

V.4. Unfolding Panels

If every piece is a swiping surface then it is easy to unfold them to planar pieces. Otherwise, we can always unfold a triangle strip on a plane. Figure 54 illustrates the unfolding process of an edge band which is a swiping surface. The edge band is a series of parallelograms whose sharing edges are parallel. We first triangulate

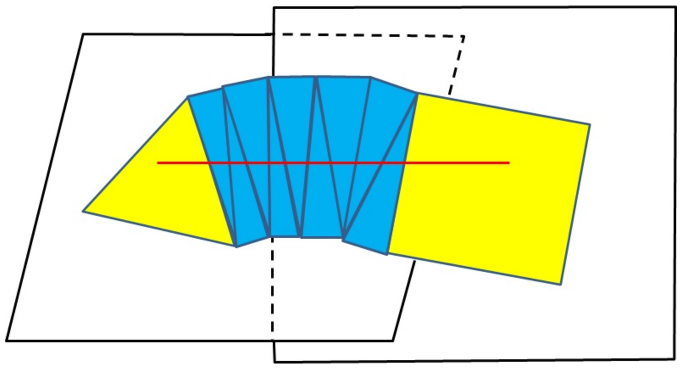


Fig. 53. Case 3: An illustration of the smooth band creation process in case 3.

them into series of triangles. Then we use the unfolding method presented in Section IV.3. Since the sharing edges of the parallelograms are parallel, the unfolding band is guaranteed not to self-intersect.

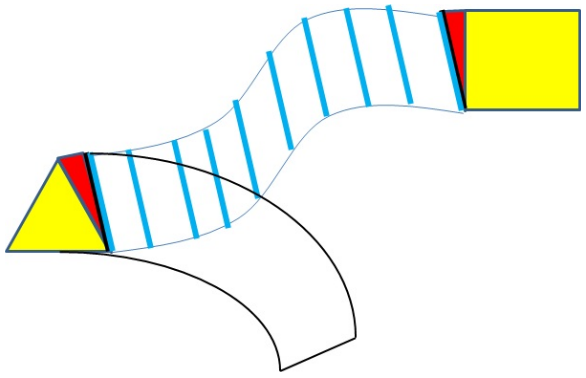


Fig. 54. An illustration of unfolding a swiping surface edge band.

Figure 55 illustrates the unfolding process of an edge band which is a series of triangles from the previous bilinear interpolation process. We use the same unfolding method presented in Section IV.3. Since the every other edges of the triangles are not parallel, the unfolding band might self-intersect. Fortunately, in our specific situation the edge band connecting two vertex bands is short and the intersection angle of the two planes containing the two vertex bands is small. Therefore, in practice, we do not see any case where an edge band overlaps with itself.

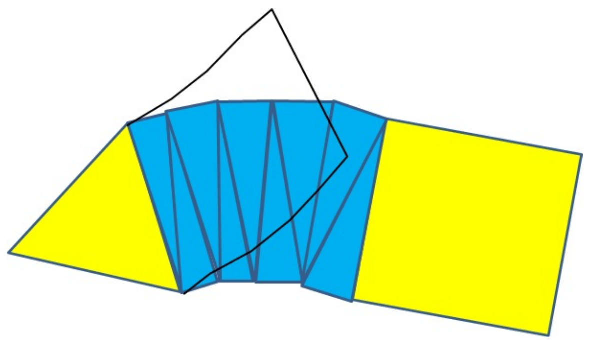


Fig. 55. An illustration of unfolding a series of triangles.

Figure 56 shows the result of unfolding of the panels of a band decomposition structure that is resulted from an octahedron model. The flattened band decomposition panels are transformed and arranged on 2D plane. In Figure 57, the flattened band decomposition panels for the Bunny model are transformed onto the a 2D plane.

The band decomposition panels are later arranged within the bounding box of the output paper size. Our program outputs the band decomposition panels in .eps file format. This file format can be read by the laser cutter directly. Figure 58 shows the band decomposition panels for the bunny input mesh which are output in a .eps file.

V.5. Numbering and Assembling

Once all pieces are unfolded, the problem is how to easily assemble them. For this problem, there is a need for solving two sub-problems:

1. **Ease in Assembly:** The lay people should be able to join the pieces easily.

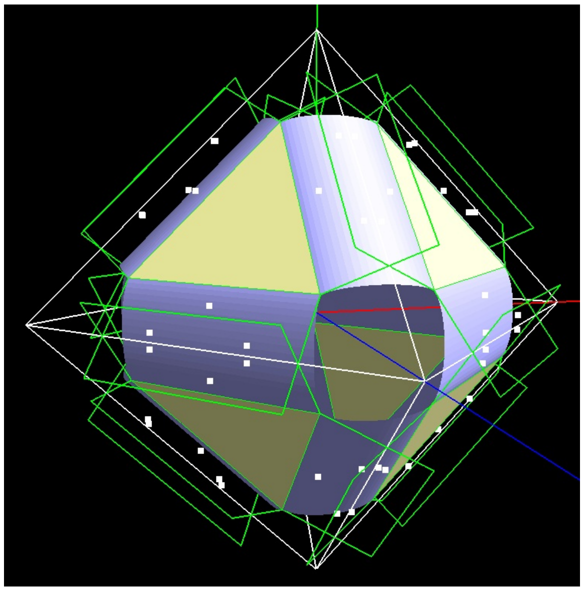
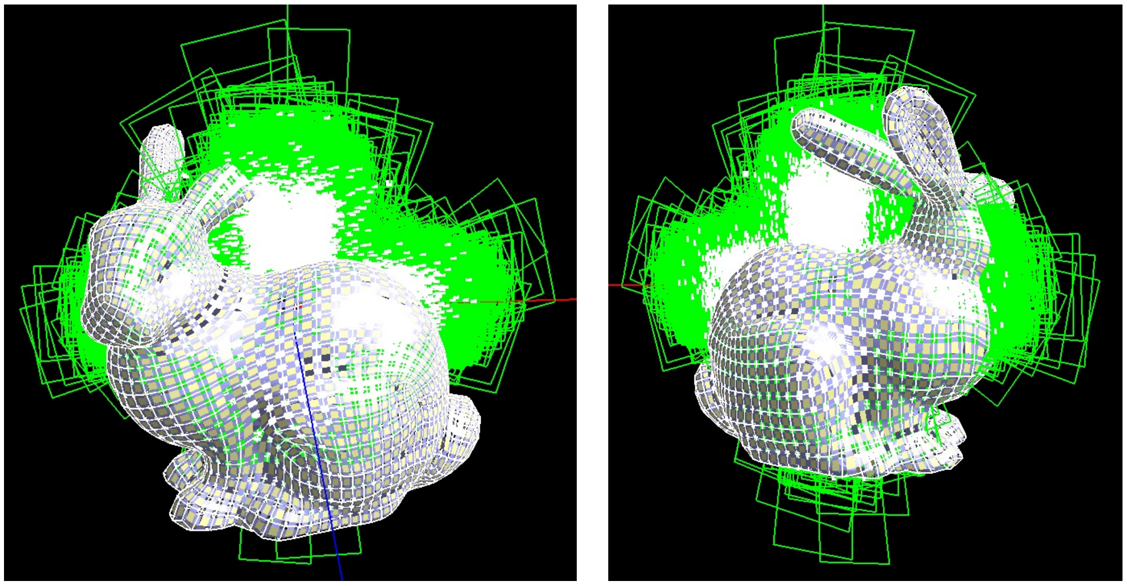


Fig. 56. Unfold the band decomposition panels on an octahedron model.



(a) Front view

(b) Back view

Fig. 57. Transform the unfolded band decomposition panels to the XY plane.

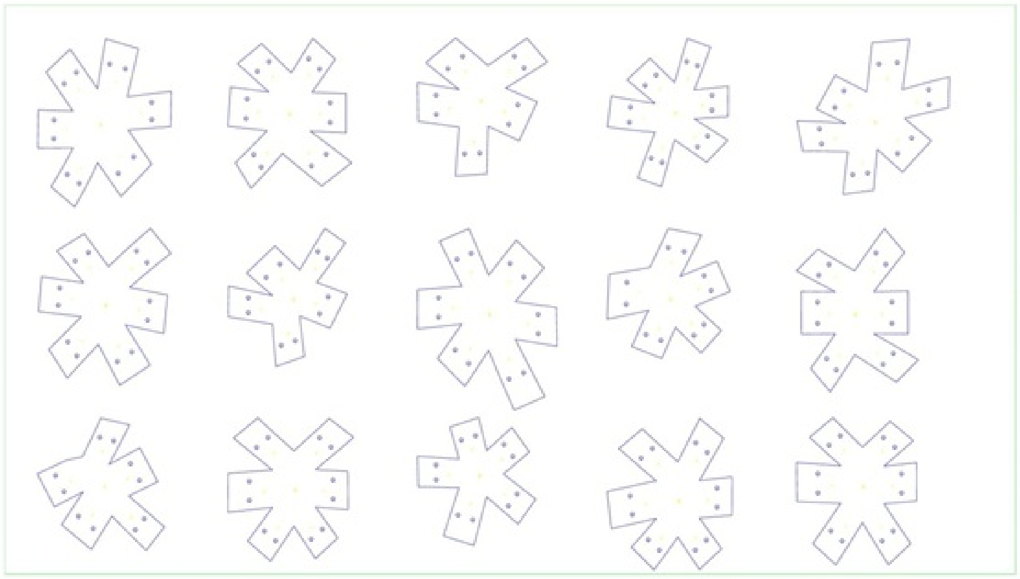


Fig. 58. The band decomposition panels are written into .eps files.

2. **Ease in Finding:** Lay people should also be able to find pieces quickly.

To provide ease in assembly, we provide a simple approach. Two neighboring band decomposition panels are joined together at their common edge bands. The two edge bands corresponding to the same edge are overlapped and fastened by fasteners.

Figure 59 shows an example of band decomposition panel. Edge numbers are marked on edge bands. Workers find the two edge bands which have the same edge number. We burn holes on edge bands for fastening purpose. Two edge bands for the same edge are snapped at holes. The diameter of the holes is fixed, which is the diameter of the fasteners. We adaptively determine the number of holes and arrangement of holes on edge bands. For example, on a thin long edge band, we place two holes vertically along the edge band; on a wide short edge band, we place two holes horizontally along the edge band; on a tiny edge band where there is not enough

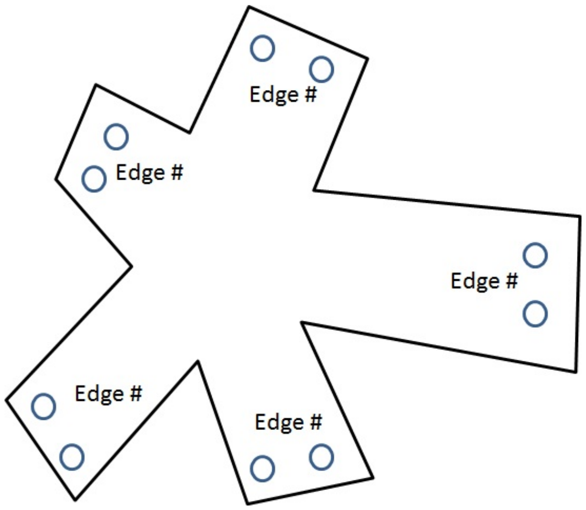


Fig. 59. An illustration of numbers and holes on a band decomposition panel.

space for two holes, we place one hole.

Edge numbers on edge bands are sufficient to assemble all the panels to build the whole structure. However, they are not efficient. There are usually many thousands of panels. Searching two edge bands with the same edge number consumes the largest part of the overall construction time. We developed a strategy to help workers find matching panels quickly. We use a flood-fill algorithm to number faces. Then we mark the extra information (face number) on each panel as see in Figure 60. Numbering using flood-fill helps the lay people find panels that are next to each other easily.

The flood-fill algorithm simulates the process of one way to assemble the panel. We start from an arbitrary panel. We join its neighboring panels to it. Then we join matching panels to the boundary panels. Figure 61 illustrates the process of growing the assembled region gradually. Panels are output in breath-first-traverse order. This guarantees that two neighboring panels have associated numbers that are close to each other. Therefor, the order that panels are output is the same as the order that they should be assembled. This strategy greatly improves the constructions time

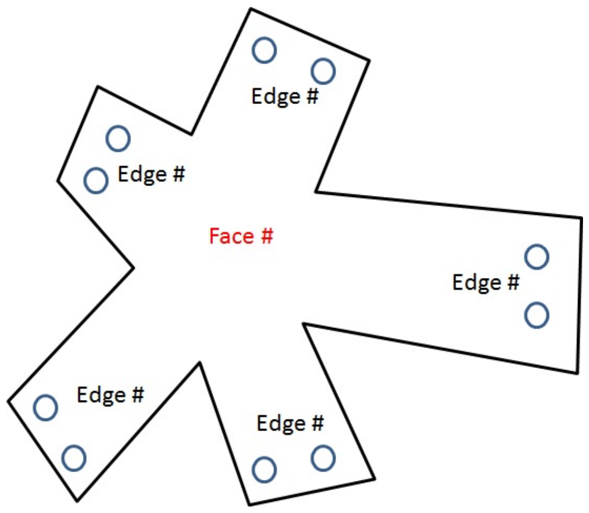


Fig. 60. An illustration of panel with extra face number.

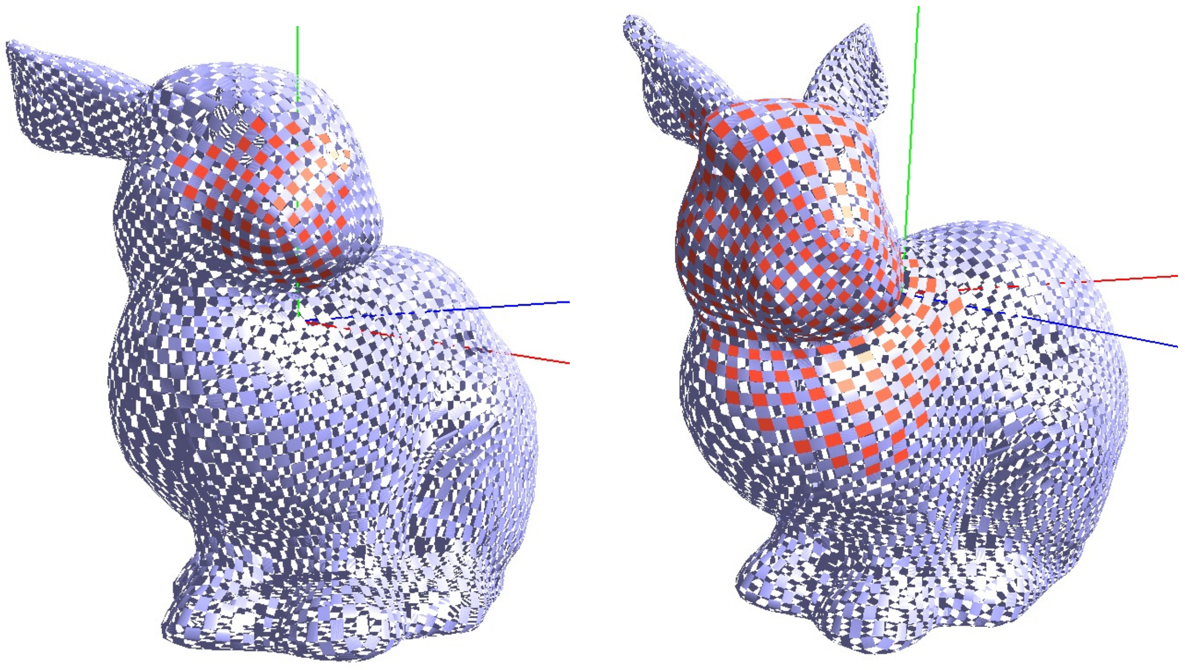


Fig. 61. The flood-fill assembling process of bunny model.

because workers do not have to search matching pieces among the sea of thousands of band decomposition panels. They can simply join panels in the same order as they are cut out by a laser cutter.

CHAPTER VI

IMPLEMENTATION AND RESULTS

VI.1. Implementation

I use C++ programming language and Standard Template Library as the programming language for the developments of the weaving structure software and band decomposition structure software.

The C++ programming language is an extension of C that was developed by Bjarne Stroustrup in the early 1980s at Bell Laboratories. C++ provides a number of features that "spruce up" the C language, but more importantly, it provides capabilities for object-oriented programming. There are several significant language features of C++. C++ is a hybrid language, it is possible to program in either a C-like style, an object-oriented style, or both. C++ programs consist of pieces called classes and functions. The user can program each piece she may need to form a C++ program. The advantage of creating her own functions and classes is that she will know exactly how they work. The user will be able to examine the C++ code. C++ provides a collection of predefined classes, along with the capability of user-defined classes. The classes of C++ are data types, which can be instantiated any number of times. Class definitions specify data objects (called data members) and functions (called member function). Classes can name one or more parent classes, providing inheritance and multiple inheritance, respectively. Classes inherit the data members and member functions of the parent class that are specified to be inheritable. Therefore it is mainly used for software engineering and computer graphics.

C++ has certain characteristics over other programming languages. Object-oriented programs are easier to understand, correct and modify. Many other object-oriented languages have been developed. The possibility to orientate programming to objects allows the programmer to design applications from a point of view more like a communication between objects rather than on a structured sequence of code. In addition it allows a greater reusability of code in a more logical and productive way.

Portability is another advantage of C++. The user can practically compile the same C++ code in almost any type of computer and operating system without making any changes. C++ is the most used and ported programming language in the world. Also, code written in C++ is very short in comparison with other languages, since the use of special characters is preferred to key words, saving some effort to the programmer. An application's body in C++ can be made up of several source code files that are compiled separately and then linked together. This save compiling time since it is not necessary to recompile the complete application when making a single change but only the file that contains it. In addition, this characteristic allows to link C++ code with code produced in other languages, such as Assembler or C. C++ has the C Compatibility. C++ is backwards compatible with the C language. Any code written in C can easily be included in a C++ program without making any change. The resulting code from a C++ compilation is very efficient, due indeed to its duality as high-level and low-level language and to the reduced size of the language itself.

I use the Standard Template Library (STL) in my software coding for the standard data structures like binary tree, graph, set and so on. I also use the standard algorithms provided by STL. There are many benefits to using the Standard Template Library for C++ development, including the ability to use generic data structures and algorithms. The C++ Standard Template Library is a powerful and versatile collection of classes and functions that provides an efficient, lightweight, and extensible framework for application development. The Standard Template Library also offers a sophisticated level of abstraction that promotes the use of generic data structures and algorithms without the overhead of a generic solution. With the exception of requirements that are very specific to an application domain and perhaps some other rare situations, STL is sufficiently flexible to address the development needs of all kinds of applications. Contrary to popular belief, applications do not have to sacrifice performance in order to use STL constructs, since the STL makes specific performance guarantees for the algorithms it supplies and the user can select the one that best meets her needs.

For the realtime rendering modules, I use the Open Graphics Library (OpenGL). OpenGL is an application programming interfaces (API). It has many advantages over other APIs for realtime rendering, e.g. Direct3D by Microsoft. OpenGL has lower CPU overhead for drawing calls and state changes than Direct3D. It has more detailed documentation than Direct3D. The learning curve scales from totally simple to most complex realtime 3D graphics. OpenGL is cross-platform. It is supported under Windows, Linux, Mac, even some handheld devices. It has stable interface for extensible, new hardware features which are exposed quickly. The third party toolkits help managing first window management hurdles and the wealth of add on features. OpenGL is supported on many hardware platforms whereas Direct3D is Windows only. The can't do cross-platform using Direct3D only, but she can do cross-platform using OpenGL for other platforms and OpenGL or Direct3D for Windows. One downside is there may not be a good OpenGL driver automatically installed on Windows, requiring a user download. Macs come with good OpenGL drivers right out of the box. Linux has OpenGL too but the user have to go find it and install it.

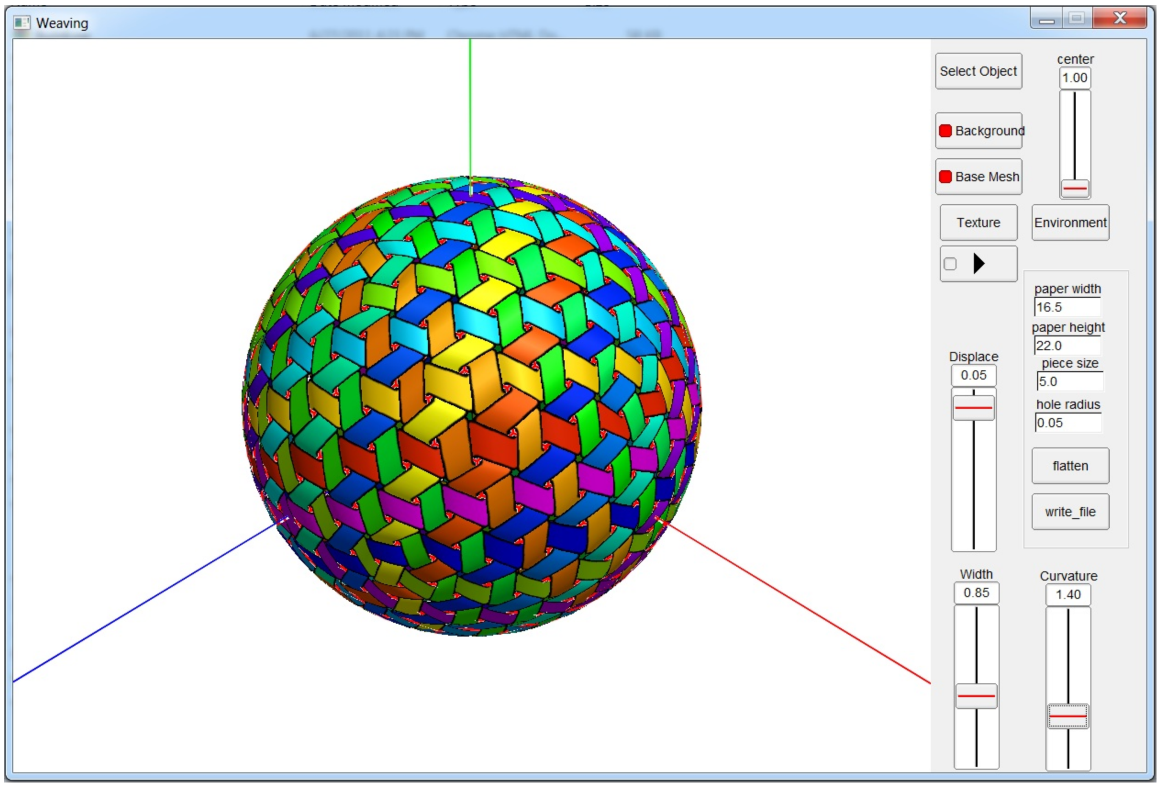
For the special shading efforts like Phong shading, environment map, and so on, I use OpenGL Shading Language. OpenGL Shading Language(GLSL), also known as GLslang, is a high level shading language based on the C programming language. It was created by the OpenGL ARB to give developers more direct control of the graphics pipeline without having to use assembly language or hardware-specific languages. This is the current standard, as far as OpenGL is concerned. It is the only shading language that is a part of the OpenGL specification. Because of this, it is kept up-to-date with current OpenGL features. Each new version of the base standard usually means a new version number of GLSL as well.

There are some disadvantages about using GLSL. A fully compiled and linked GLSL program must implement all of the stages within itself. This means that any mixing and matching of vertex and fragment shaders can only happen before link time. So every possible combination of shaders that the user intends to use must be explicitly linked. Combined with the fact that linking is not a fast operation in GLSL, and one finds that generating all of the programs that a user might want to use can take a long time. In highly degenerate cases, this can be tens of minutes. Resource binding, like which attribute index a particular input variable uses and so forth, must be handed in the OpenGL API rather than in GLSL itself. The complexity of having a full C-style language in a graphics driver causes quite a few driver bugs to show themselves, particularly in ATI compilers.

An NVIDIA graphics hardware is installed in my desktop PC. This is one more reason that I choose to use GLSL. NVIDIA's GLSL compiler is really a slight modification of the Cg compiler. Because of that, some of the differences between GLSL and Cg will occasionally show their heads. Cg is more permissive syntactically than GLSL, so a GLSL program that compiles on an NVIDIA driver may not compile on an ATI driver. It can also give unusual error messages, with references to a "profile" (a concept that exists in Cg but not GLSL).

The Graphics User Interface (GUI) of the software was developed using FLTK. FLTK (the Fast, Light Toolkit, pronounced "fulltick") is a cross-platform GUI library developed by Bill Spitzak and others. Made with 3D graphics programming in mind, it has an interface to OpenGL, but it is also suitable for general GUI programming. Using its own widget, drawing and event systems (though FLTK2 has gained experimental support for optionally using the cairo graphics library) abstracted from the underlying system-dependent code, it allows for writing programs which look the same on all supported operating systems. FLTK is free software, licensed under LGPL with an additional clause permitting static linking from applications with incompatible licenses. It includes FLUID (FLTK User Interface Designer), a graphical GUI designer that generates C++ source and header files. In contrast to libraries like Qt and wxWidgets, FLTK uses a more lightweight design and restricts itself to GUI functionality. Because of this, the library is very small (the FLTK "Hello World" program is around 100 KB), and is usually statically linked. It also avoids complicated macros and separate code preprocessors, and does not use the following advanced C++ features: templates, exceptions, RTTI or, for FLTK 1.x, namespaces. Combined with the modest size of the package, this leads to a relatively short learning curve for new users. These advantages come with corresponding disadvantages. FLTK offers fewer widgets than most GUI toolkits and, because of its use of non-native widgets, does not have native look-and-feel on any platform. The Fast Light Tool Kit is indeed a multiplatform UI library. For a simple application, you can create source that will compile, unmodified, for Win, Nix, OSX among others. This is accomplished by some non-standard implementations of things to handle the differences. Good examples of this can be seen in the clock demo in the test directory. Other good (cross-platform) examples include the browser for reading directories. The code itself is C++ with a C'ish flavor. The standards for the FLTK library don't use parts of C++ that are not well supported by all compilers/platforms. Things like the STL and C++ strings are not used by the library. The user can of course use them for her FLTK programs, they are just not used by the core.

Figure 62 shows the graphics user interface of the weaving structure software. Figure 63 shows the graphics user interface of the band decomposition structure software. Both softwares are cross-platform. The code can be complied without modification under Windows, Macs, and Linux.

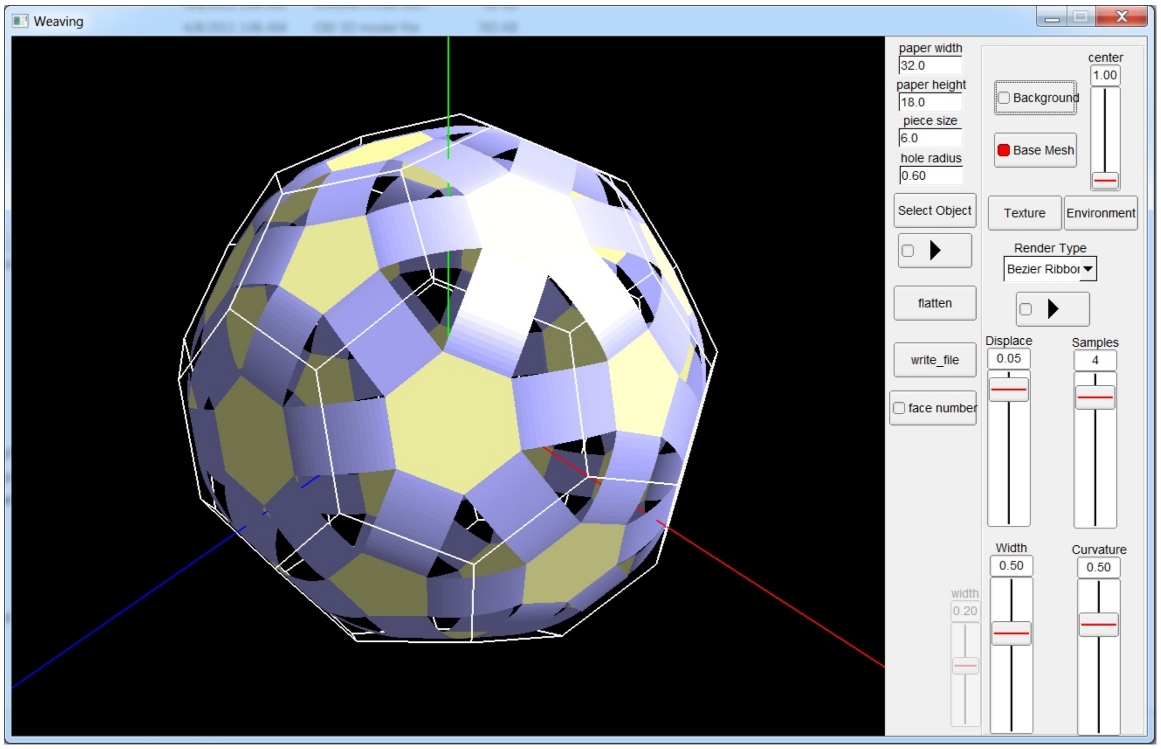


center 1.00
paper width 16.5
paper height 22.0
piece size 5.0
hole radius 0.05
Displace 0.05
Width 0.85
Curvature 1.40

Fig. 62. The program interface for the plain woven structures.

The assembling elements for both the weaving structure and band decomposition structure are output by the softwares in .eps file format for laser cutter. The laser cutter distinguishes different color for either cut through or etching.

Figure 64 shows the different color codes of different interpretations by the laser cutter. Yellow lines are etched by the laser cutter. Blue lines are cut through by the laser cutter. The green lines indicate the material outline.



paper width 32.0
paper height 18.0
piece size 6.0
hole radius 0.60
center 1.00
Render Type Bezier Ribbon
Displace 0.05
Samples 4
width 0.20
Width 0.50
Curvature 0.50

Fig. 63. The program interface for the band decomposition structures.

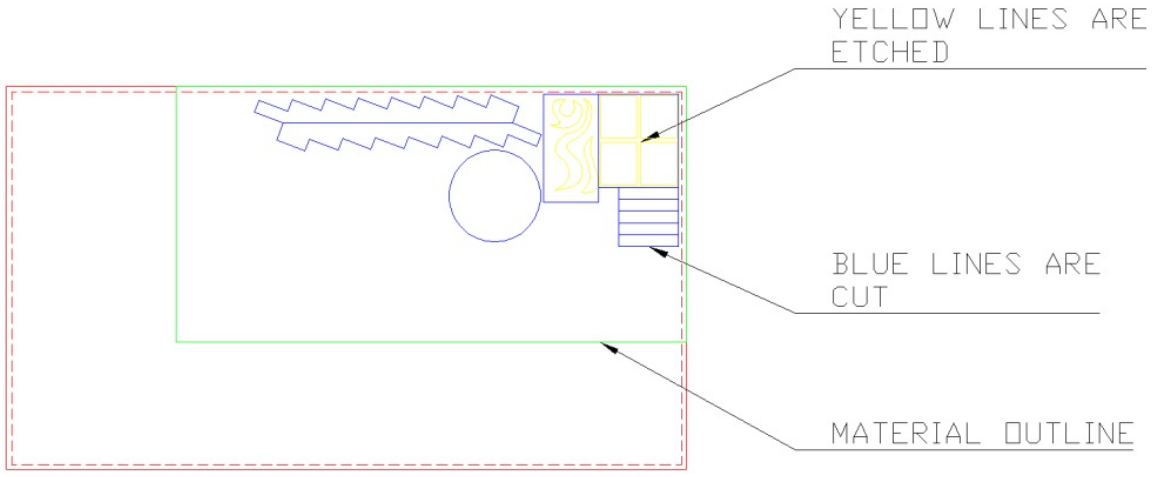
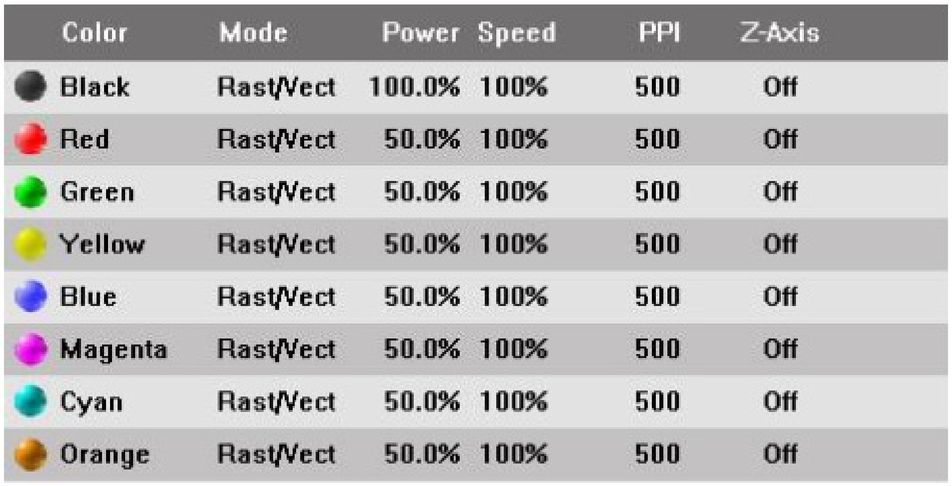


Fig. 64. An illustration of the color coding for the laser cutter. Figure is from [66].



Color	Mode	Power	Speed	PPI	Z-Axis
● Black	Rast/Vect	100.0%	100%	500	Off
● Red	Rast/Vect	50.0%	100%	500	Off
● Green	Rast/Vect	50.0%	100%	500	Off
● Yellow	Rast/Vect	50.0%	100%	500	Off
● Blue	Rast/Vect	50.0%	100%	500	Off
● Magenta	Rast/Vect	50.0%	100%	500	Off
● Cyan	Rast/Vect	50.0%	100%	500	Off
● Orange	Rast/Vect	50.0%	100%	500	Off

Fig. 65. Different color codes of different power and speed. Figure is from [66].

Figure 65 shows different color codes which guide the laser cutter for different power and speed.

We used the laser cutter in the wood workshop at the College of Architecture in Texas A&M University. It has a 32 inch by 18 inch bed and a margin of one fourth inch around all margins. Just like printing with paper, we must include the one fourth inch margin in the .eps output file. The layout can dramatically improve the efficiency of laser cutting and saves building materials. Currently, we first output the initial layout computed automatically by the software. Then we open the output .eps file in AutoCad to optimize the layout for final fabrications by the laser cutter.

VI.2. Results

Most of the images in this document are direct screen-shots from the programs. Some images were rendered using the 3D modeling, animation and rendering package, Maya. For this the models are exported as Wavefront object files from the program and imported into Maya. Post-processing on the images was done using Adobe Photoshop and GIMP. Photographs of the sculptures and other objects were taken by the authors.

VI.2.1. Results of Plain Woven Structures

Since the early development of the weaving structure software, it has been receiving attention of interest and helpful comments and suggestion from different architects.

The *Spulenkorb* project by Ryan Collier, Michael Tomaso, and Gabriel Esquivel was awarded an honorable mention in the REPEAT competition organized by TEX-FAB November 2010 [18]. Figure 66 shows the weaving geometry rendered by Maya. The group plan to use our weaving structure software to construct a physical structure of the Spulenkorb model as their next phase of projet.

The design group use Maya and TopMod to design the initial sculpture shape, as seen in Figure 67.

Then the initial mesh is imported into our weaving structure software. The weaving geometry for the input mesh is computed and rendered in our weaving structure software, as seen in Figure 68 (a). Our weaving structure software saves the weaving geometry in .obj file format for later modification and/or rendering by other modeling softare, as seen in Figure 68 (b).

We also built a physical structure of a carefully chosen model to justify the robustness of our weaving structure software. As seen in Figure 69 (a), we deform the perfect soccer ball mesh severely. There are irregular faces in the mesh. And many of the faces are distorted and not planar. Our weaving structure software still generates weaving geometry for the extreme input mesh as seen in Figure 69 (b).

Figure 70 shows the photos of the physical structure of the deformed soccer ball mesh. Notice the bulging part of the sculpture, it truly conveys the underlying shape

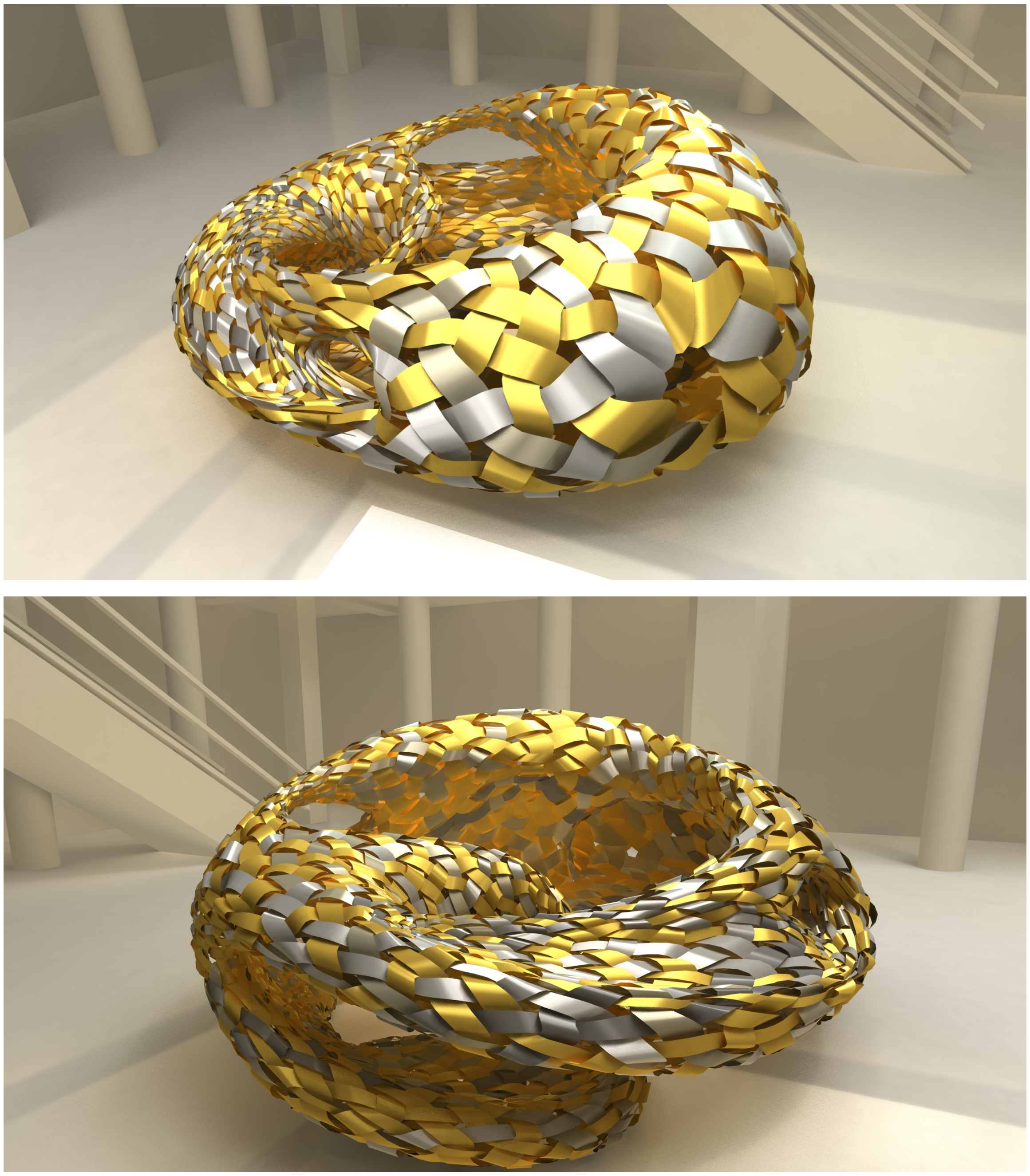
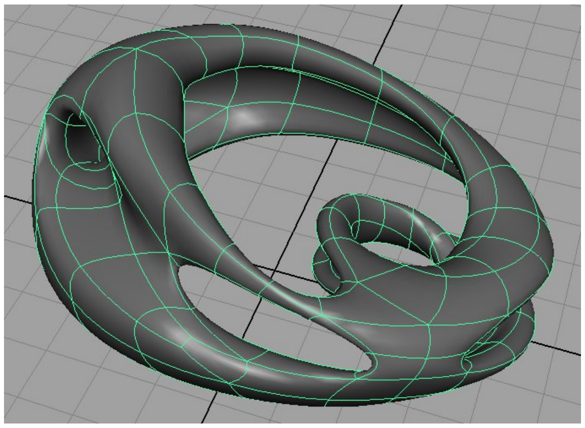
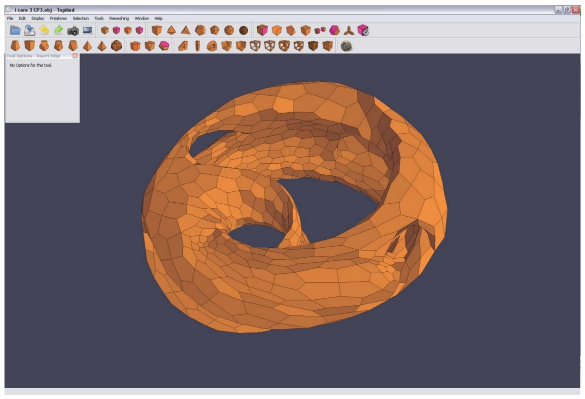


Fig. 66. Weaving design example Spulenkorbproject.

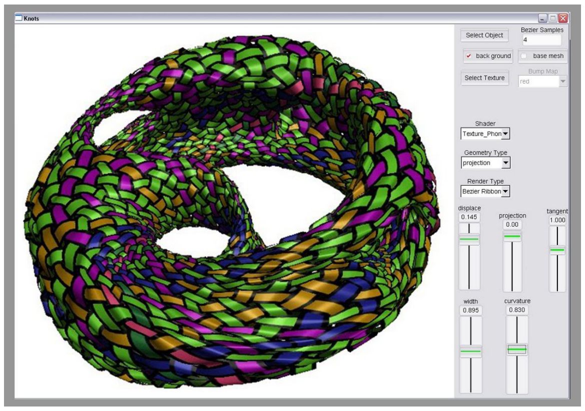


(a) initial design in TopMod



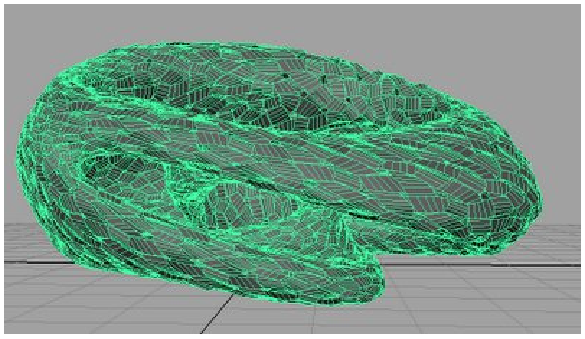
(b) initial design in TopMod

Fig. 67. Design the initial sculpture shape.



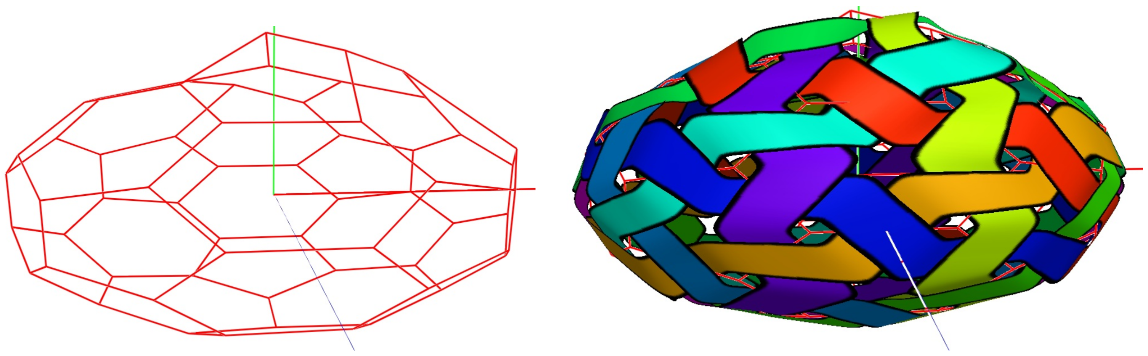
Bezier Samples: 4
back ground ✓
base mesh
Bump Map: red
Shader: Texture_Phon
Geometry Type: projection
Render Type: Bezier Ribbon
displace: 0.145
projection: 0.00
tangent: 1.000
width: 0.895
curvature: 0.830

(a)



(b)

Fig. 68. (a) The weaving geometry is generated by our software. (b) The weaving geometry is imported into Maya for better rendering.



(a) Input mesh (b) Weaving geometry

Fig. 69. An illustration of the input deformed soccer ball mesh and the weaving geometry.

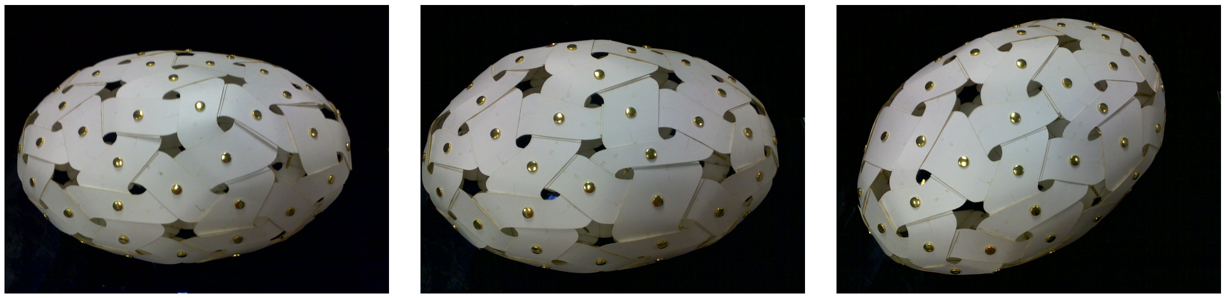


Fig. 70. This sculpture of deformed soccer ball is constructed with laser-cut poster-board papers assembled with brass fasteners.

of the mesh, as we can see the upper pointing part of the input mesh in Figure 69 (a).

This physical weaving structure used four pages of the paper cardboard. It has 90 pieces of the assembling panels which is the number of edges in the input deformed soccer ball mesh. 90 fasteners are used. The laser cutting time for each page is 4 minutes. Two people without assembling experiences used 2 hours and 10 minutes to build this physical structure.

VI.2.2. Results of Band Decomposition Structures

As a proof of concept project, we first constructed a simple band decomposition structure for a perfect soccer ball input mesh.

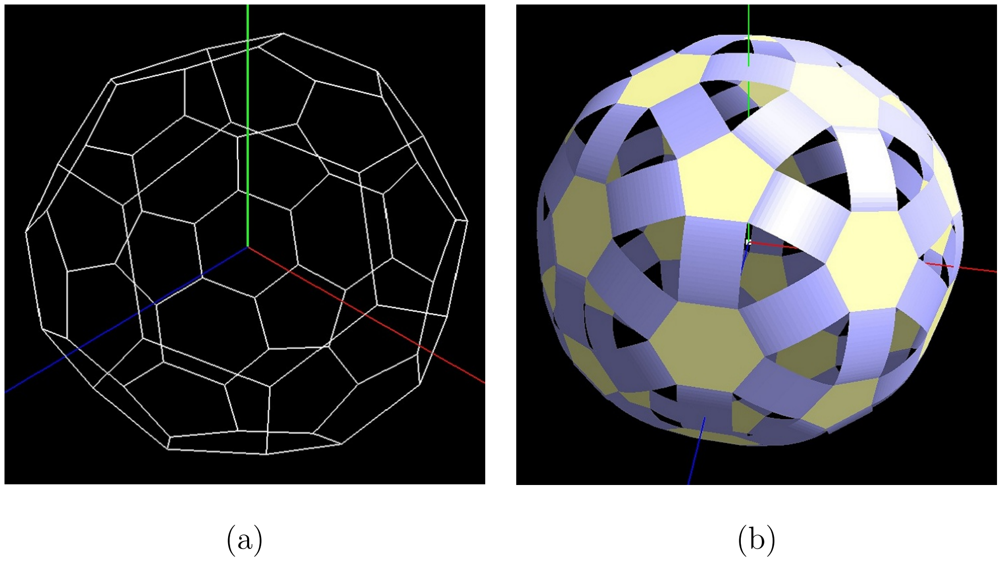


Fig. 71. (a) Input mesh of the soccer ball model. (b) The resulting band decomposition structure rendered by our software.

Figure 71 (a) shows the input mesh of the soccer ball model. Our band decomposition software automatically generates the geometry of the band decomposition for the input mesh. Figure 71 (b) shows the resulting band decomposition structure rendered by our software.

The input soccer ball mesh has 2104 faces and 4208 edges. Therefor, our program produced 2104 assembling elements which is the same number of faces. The physical band decomposition structure used 8416 fasteners which are twice the number of edges. Figure 72 shows the photographs of the physical structure. Two people without previous assembling experience spent 1 hour 40 minutes to build this sculpture.

Our research received interest of Professor Esquivel in the Architecture Depart-

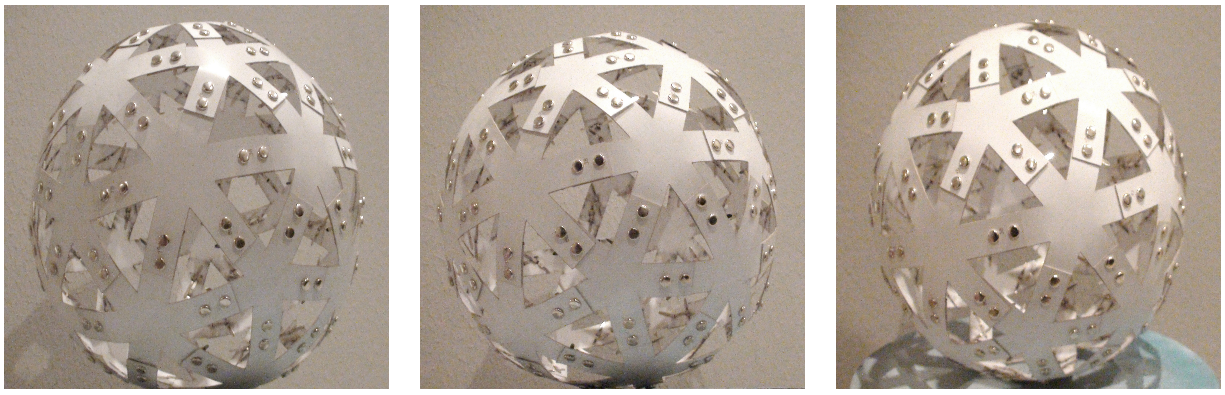


Fig. 72. The band decomposition structure for the soccer ball model.

ment of Texas A&M University. A group of students chose the band decomposition structure as their final project for the course Digital Geometry Workshop in spring 2011 [24].

The group of students started with the construction of the torus model. They experimented with different materials and fastening techniques. They also obtained assembling experience. Figure 73 (a) shows the input mesh of the torus model. Our band decomposition software automatically generates the geometry of the band decomposition for the input mesh. Figure 73 (b) shows the resulting band decomposition structure rendered by our software.

The input torus mesh has 336 faces and 672 edges. Therefor, our program produced 336 assembling elements which is the same number of faces. The physical band decomposition structure used 1342 fasteners which are twice the number of edges. Figure 74 shows the photograph of the physical structure. Three students without previous assembling experience spent 3 hour 20 minutes to build this sculpture.

The next stage is to build the final bunny sculpture. The students chose to use the paper cardboard as the building material because of the cost and bending

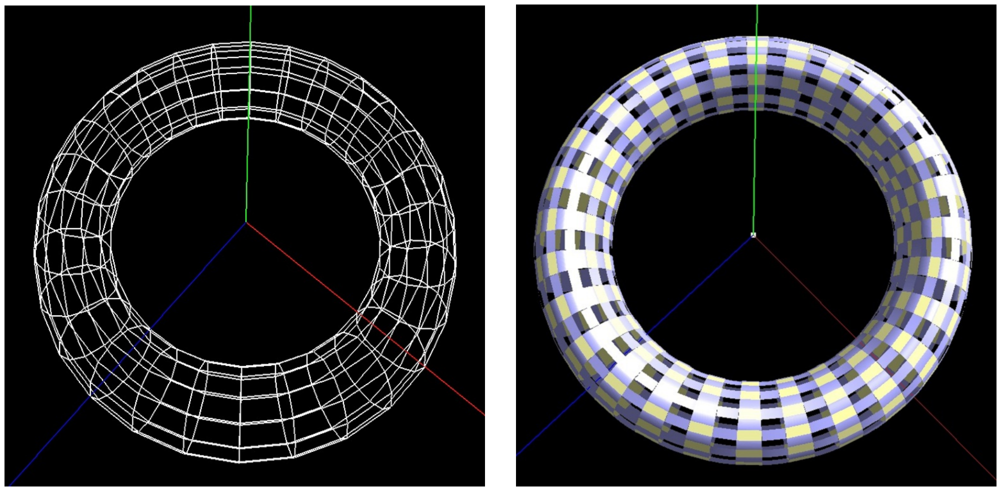

(a) (b)

Fig. 73. (a) Input mesh of the torus model. (b) The resulting band decomposition structure rendered by our software.

property factors.

Figure 75 (a) shows the input mesh of the bunny model. Our band decomposition software automatically generates the geometry of the band decomposition for the input mesh. Figure 75 (b) shows the resulting band decomposition structure rendered by our software.

The input bunny mesh has 2104 faces and 4208 edges. Therefor, our program produced 2104 assembling elements which is the same number of faces. The physical band decomposition structure used 8416 fasteners which are twice the number of edges. Figure 76 show the assembling process for the band decomposition structure of the bunny model. The construction follows the flood-fill strategy as we discussed in Section V.5. During the process of construction, we discovered a hidden programming bug. The edge band at some saddle points were flipped. This caused mismatching with neighboring assembling panels. However, this programming bug was discovered

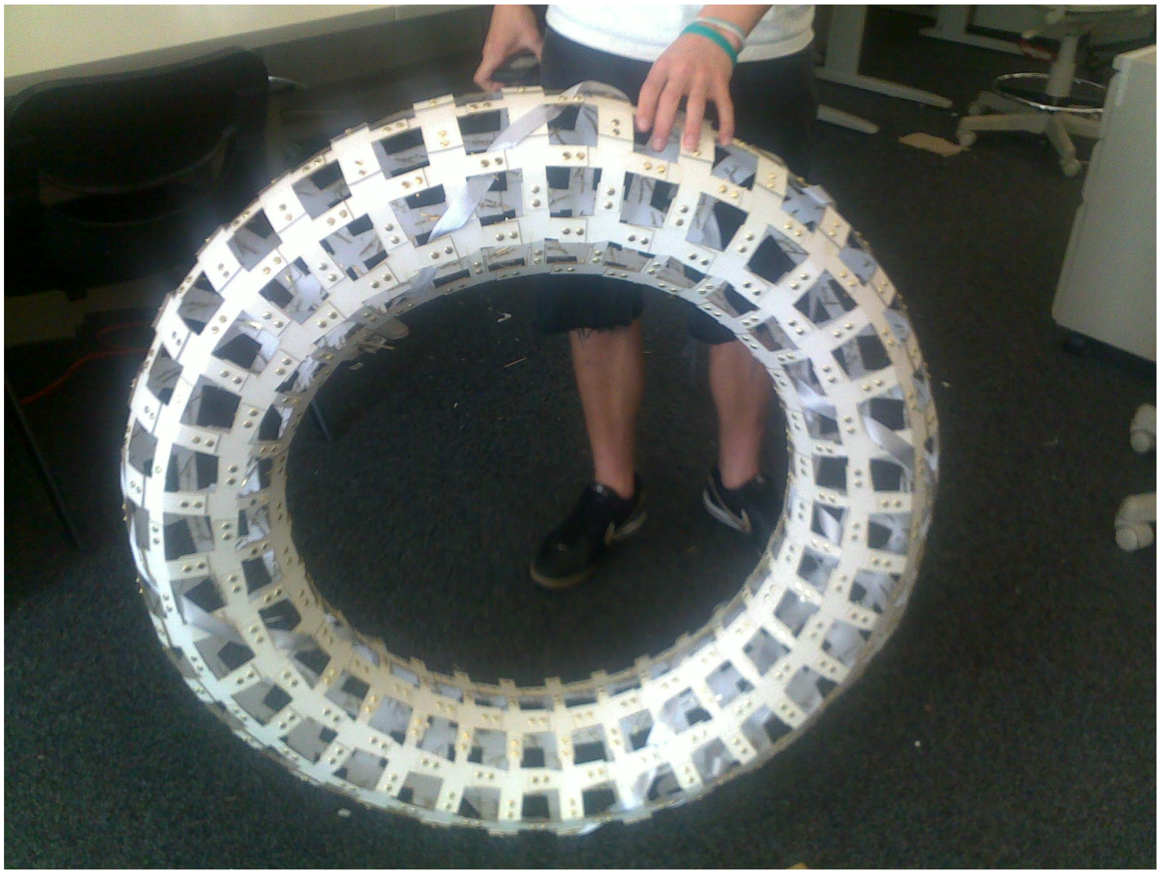


Fig. 74. The band decomposition for the torus model.

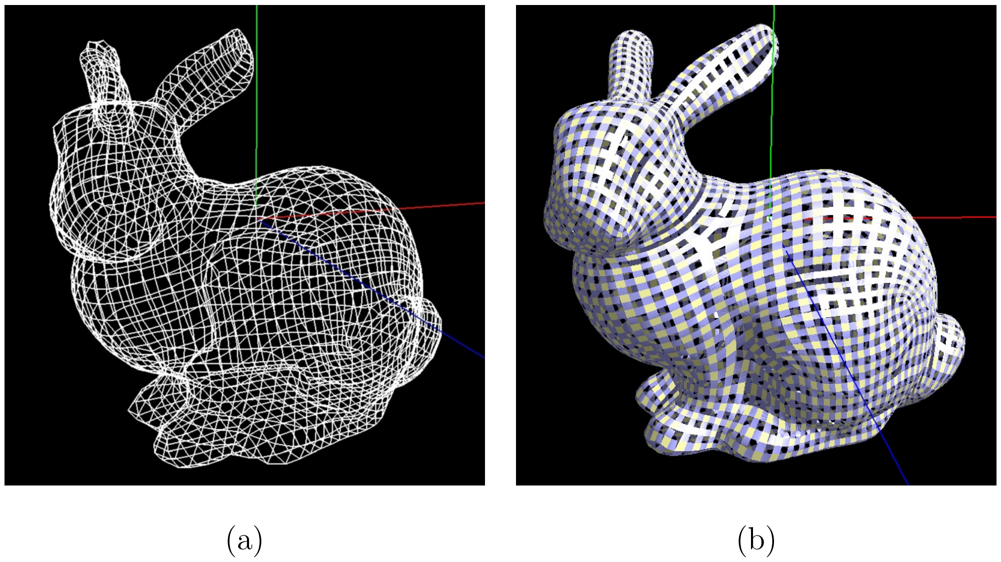


Fig. 75. (a) Input mesh of the bunny model. (b) The resulting band decomposition structure rendered by our software.

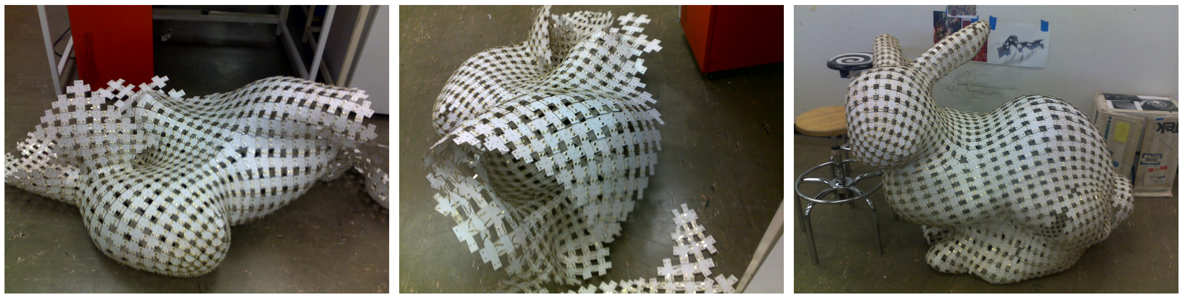


Fig. 76. The assembling progress for the bunny model.

in very rare cases. The whole bunny sculpture was not affected seriously. I have corrected the bug in the source code. Figure 77 shows the photograph of the physical structure. Three students spent 8 hour 20 minutes to build this sculpture. The research result will be presented in Siggraph 11 as a late breaking research talk [8].

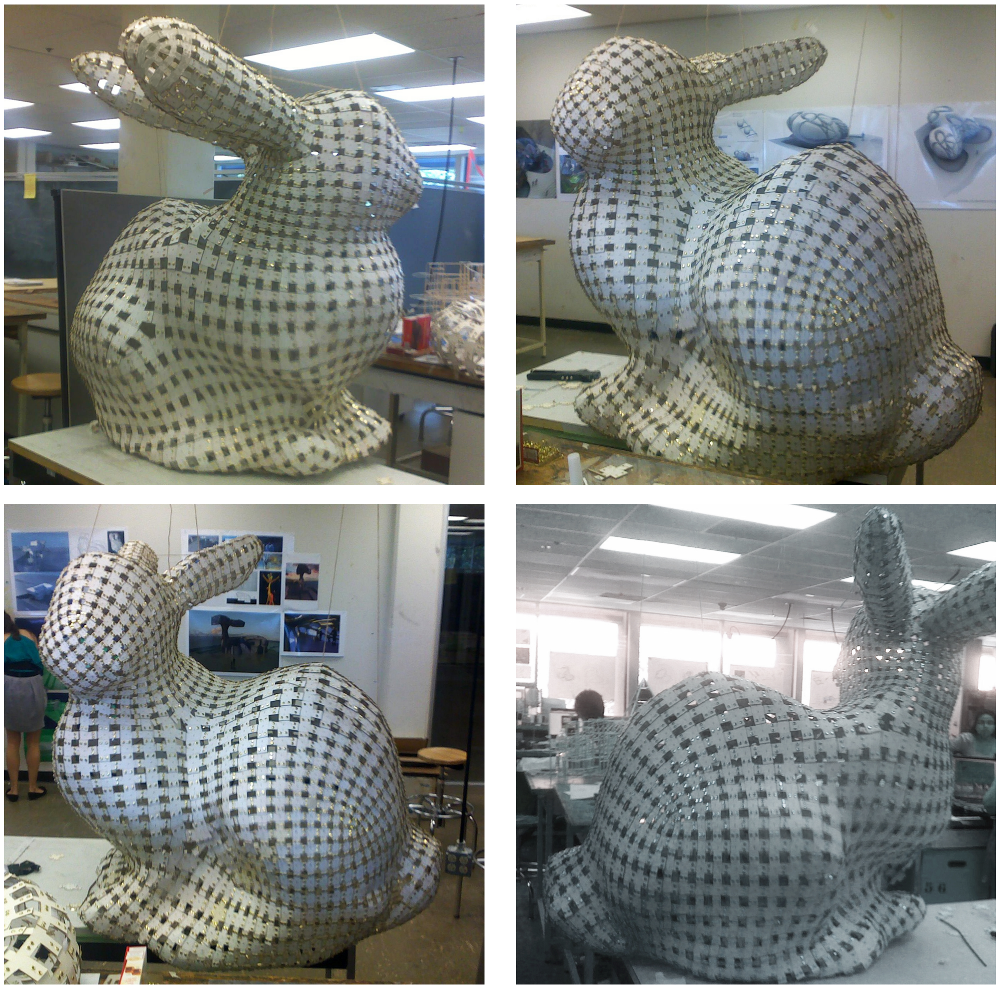


Fig. 77. This large sculpture of Bunny is constructed with laser-cut card board assembled with brass fasteners.

CHAPTER VII

CONCLUSION AND FUTURE WORK

In this dissertation, I propose an approach for physical construction of large structures. This approach is based on the graph rotation system framework. I have developed two techniques to generate developable panels from any input polygonal mesh that can be easily assembled to get the shape of the input polygonal mesh.

The first technique is used to create plain woven structures. I have developed the “projection method” to convert mathematical weaving cycles on any given polygonal mesh to developable strip panels. The widths of weaving strips vary so that the surface of the input model can be covered almost completely. When these strip panels are assembled together, resulting shape resembles to a weaving in 3-space.

The second technique is used to create band decomposition structures. I have developed a method to convert any given polygonal mesh into star-like developable panels, which we call vertex components. Assembling of vertex components results in band decomposition structures. These band decomposition structures correspond to 2D-thickening of graphs embedded on surfaces. In a 2D-thickening, each vertex thickens to a polygon and each edge thickens to a band. Within the resulting band decomposition, each polygon corresponds to a vertex and each band corresponds to an edge that connects two vertex polygons.

The two techniques do not have restrictions on design models. The input mesh can be of any genus. The faces in the input mesh can be triangle, quadrilateral, and any polygon. The advantages of this kind of large physical structure construction are low-cost material and prefabrication, easy assemble. One of the issues with laser cutting and the fabrication of objects is that the objects tend to be flat, simplistic and angular due to the output technology. Our techniques take the digital fabrication in a new direction and create complex and organic 3D forms. Along the theme of architecture this research has great implication for structure design and makes the more difficult task of construction techniques easier to understand for the fabricator. It has implications to the sculpture world as well as architecture.

Our work opens a door to the new research direction in digital fabrication. There are many topics to study in the future. After the physical bunny structure was finished, we realized that the center of mass was not falling on its standing area. So the bunny sculpture can not stand by itself. It falls forward. In the future, we would like to add the analysis model to the software. The program will compute the center of mess based on the input mesh shape, material density, and mass of fasteners. Because of the effort of gravity, the sculpture deforms to some extend. So we hang the bunny sculpture from two points, one on the head, the other on the back. Another quick solution will be to use lighter and stronger materials. A more delicate solution will be to simulate the stress and tension that each assembling panel takes. Then use different materials based on the tension. The upper part of the sculpture has less stress, therefor we use lighter material for the assembling panels. The lower part or parts with high curvatures bear more stress, therefor we use stronger material for the assembling panels.

When the weaving width parameter is set to be very high, the weaving ribbons will intersect with each other. In the current woven structure software, there is not a model to check the intersections. The assembling panels can still be output. But they will push against each other during the construction process. The next research topic will be to detect the collision among weaving ribbons. The user will be notified about the intersection. The parameter can be adjusted accordingly.

Our ultimate goal is to build very large physical structures, probably in the building scale. The weaving and band decomposition structures are both hollow shells which convey the shape of the initial input model. In the future, I would like to collaborate with material and mechanical experts. Researches will be on building materials or fastening techniques to make the building-scale sculpture stand by itself. Or we may need to build inner support structure to support the outer shell.

REFERENCES

[1] E. Akleman and J. Chen, “Guaranteeing 2-manifold property for meshes,” in *Proc. Shape Modeling International*, vol. 5, no. 2, 1999, pp. 18–25.

[2] E. Akleman, J. Chen, Y. Chen, Q. Xing, and J. L. Gross, “Cyclic twill-woven objects,” *Computers & Graphics*, vol. 35, no. 3, pp. 623–631, 2011.

[3] E. Akleman, J. Chen, and J. L. Gross, “Paper-strip sculptures,” in *Proc. Shape Modeling Int. Conf.*, 2010, pp. 236–240.

[4] E. Akleman, J. Chen, J. L. Gross, and Q. Xing, “Extended graph rotation systems as a model for cyclic weaving on orientable surfaces,” Technical Report: TR09-203, Dept. Comput. Sci., Texas A&M Univ., College Station, TX, 2009.

[5] E. Akleman, J. Chen, V. Srinivasan, and F. Eryoldas, “A new corner cutting scheme with tension and handle-face reconstruction,” *Int. J. Shape Modeling*, vol. 7, no. 2, pp. 111–118, 2001.

[6] E. Akleman, J. Chen, Q. Xing, and J. Gross, “Cyclic plain-weaving on polygonal mesh surfaces with graph rotation systems,” *ACM Trans. Graph.*, vol. 28, no. 3, pp. 78:1–78:8, 2009.

[7] E. Akleman, V. Srinivasan, J. Chen, D. Morris, and S. Tett, “Topmod3d: an interactive topological mesh modeler,” in *Proc. Computer Graphics International*, 2008, pp. 171–181.

[8] E. Akleman, Q. Xing, G. Esquivel, J. Chen, and J. L. Gross, “Band decomposition of 2-manifold meshes for physical construction of large structures,” in *ACM SIGGRAPH 2011 Talks*, 2011, pp. 3:1–3:1.

[9] AutoCAD, http://usa.autodesk.com/autocad, 2011.

[10] R. Bartels, J. Beatty, and B. Barsky, *An Introduction to Splines for Use in Computer Graphics & Geometric Modeling.* San Francisco, CA: Morgan Kaufmann Publishers Inc., 1987.

[11] B. G. Baumgart, "A polyhedron representation for computer vision," in *Proc. National Comput. Conf. & Expos.*, 1975, pp. 589–596.

[12] P. Benko, G. Kos, P. Benko, L. Andor, G. Ks, T. Varady, L. Andor, and R. Martin, "Constrained fitting in reverse engineering," *Comput. Aided Geom. Des.*, vol. 19, no. 3, pp. 173–205, 2002.

[13] A. Blanc, M. McEvoy, and R. Plank, *Architecture and Construction in Steel.* New York: Taylor & Francis, 1993.

[14] E. Catmull and J. Clark, "Recursively generated b-spline surfaces on arbitrary topological meshes," *Computer Aided Design*, vol. 10, no. 6, pp. 350–355, 1978.

[15] Cepheus, http://en.wikipedia.org/wiki/Osculating_circle, 2006.

[16] J. Chen and E. Akleman, "Insight for practical subdivision modeling with discrete gauss-bonnet theorem," *Geom. Modeling & Processing*, vol. 4077, no. 1, pp. 287–298, 2006.

[17] D. Cohen-Steiner, P. Alliez, and M. Desbrun, "Variational shape approximation," *ACM Trans. Graph.*, vol. 23, no. 3, pp. 905–914, 2004.

[18] R. Collier, M. Tomaso, and G. Esquivel, "Spulenkorb project page," http://theoremas-gabe00fab.blogspot.com/2010/11/spulenkorb.html, 2011.

[19] D. Doo and M. Sabin, "Behaviour of recursive division surfaces near extraordinary points," *Computer Aided Design*, vol. 10, no. 6, pp. 356–360, 1978.

[20] J. Edmonds, "A combinatorial representation for polyhedral surfaces," Master's thesis, Dept. Fine Arts, Univ. Maryland, College Park, MD, 1960.

[21] M. Eigensatz, M. Kilian, A. Schiftner, N. Mitra, H. Pottmann, and M. Pauly, "Paneling architectural freeform surfaces," *ACM Trans. Graph.*, vol. 29, no. 4, pp. 44:1–44:10, 2010.

[22] EmberSurge, http://www.embersurge.com/pj5000.html, 2011.

[23] R. L. L. Equipment, http://www.ecvv.com/product/1578624.html, 2011.

[24] G. Esquivel, "Bunny project page," http://theoremas-gabe00fab.blogspot.com/2011/05/bunny.html, 2011.

[25] C.-W. Fu, C.-F. Lai, Y. He, and D. Cohen-Or, "K-set tilable surfaces," *ACM Trans. Graph.*, vol. 29, no. 4, pp. 44:1–44:6, 2010.

[26] E. Gaba, http://en.wikipedia.org/wiki/Curvature, 2006.

[27] R. Gal, O. Sorkine, N. J. Mitra, and D. Cohen-Or, "iwires: an analyze-and-edit approach to shape manipulation," *ACM Trans. Graph.*, vol. 28, no. 3, pp. 33:1–33:10, 2009.

[28] 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.

[29] O. Gonen, E. Akleman, and V. Srinivasan, "Modeling D-forms," in *Proc. Bridges: Mathematical Connections in Art, Music and Science*, 2007, pp. 209–216.

[30] J. L. Gross and T. W. Tucker, *Topological Graph Theory.* New York: Wiley Interscience, 1987.

[31] J. L. Gross and J. Yellen, *Graph Theory and Its Applications, Second Edition (Discrete Mathematics and Its Applications).* London, UK: Chapman & Hall/CRC, 2006.

[32] L. Guibas and J. Stolfi, “Primitives for the manipulation of general subdivisions and computation of voronoi diagrams,” *ACM Trans. Graph.*, vol. 4, no. 2, pp. 74–123, 1985.

[33] P. Haeberli, http://laminadesign.com/index.html, 2011.

[34] T. Ju, F. Losasso, S. Schaefer, and J. Warren, “Dual contouring of hermite data,” *ACM Trans. Graph.*, vol. 21, no. 3, pp. 339–346, 2002.

[35] D. Julius, V. Kraevoy, and A. Sheffer, “D-charts: Quasi-developable mesh segmentation,” *Comput. Graph. Forum*, vol. 24, no. 3, pp. 581–590, 2005.

[36] M. Kaplan and E. Cohen, “Computer generated celtic design,” in *Proc. Eurographics Symp. Rendering*, 2003, pp. 9–16.

[37] M. Kilian, S. Flöry, Z. Chen, N. J. Mitra, A. Sheffer, and H. Pottmann, “Curved folding,” *ACM Trans. Graph.*, vol. 27, no. 3, pp. 75:1–75:9, 2008.

[38] E. Kreyszig, *Introduction to Differential Geometry and Riemannian Geometry.* Ontario, Canada: Univ. Toronto Press, 1968.

[39] D. Kwiatkowska, “Structural integration at the shoot apical meristem: models, measurements, and experiments,” *Amer. J. Botany*, vol. 91, no. 9, pp. 1277–1293, 2004.

[40] B. Li, R. Schnabel, S. Jin, and R. Klein, “Variational surface approximation and model selection,” *Comput. Graph. Forum*, vol. 28, no. 7, pp. 1985–1994, 2009.

[41] Y. Li, E. Zhang, Y. Kobayashi, and P. Wonka, “Editing operations for irregular vertices in triangle meshes,” *ACM Trans. Graph.*, vol. 29, no. 6, pp. 153:1–153:12, 2010.

[42] Y. Liu, H. Pottmann, J. Wallner, Y.-L. Yang, and W. Wang, “Geometric modeling with conical meshes and developable surfaces,” *ACM Trans. Graph.*, vol. 25, no. 3, pp. 681–689, 2006.

[43] T. Machine, http://www.thalermachine.com, 2011.

[44] M. Mantyla, *Introduction to Solid Modeling.* New York: W. H. Freeman & Co., 1988.

[45] F. Massarwi, C. Gotsman, and G. Elber, “Papercraft models using generalized cylinders,” in *Comput. Graph. Appl.*, 2007, pp. 148–157.

[46] Maya, http://usa.autodesk.com/maya, 2011.

[47] C. Mercat, “Les entrelacs des enluminure celtes,” *Dossier Pour La Science*, vol. 15, no. 47, pp. 7–11.

[48] J. Mitani and H. Suzuki, “Making papercraft toys from meshes using strip-based approximate unfolding,” *ACM Trans. Graph.*, vol. 23, no. 3, pp. 259–263, 2004.

[49] N. J. Mitra, L. Guibas, and M. Pauly, “Symmetrization,” *ACM Transactions on Graphics (SIGGRAPH)*, vol. 26, no. 3, pp. 63:1–63:8, 2007.

[50] Y. Mori and T. Igarashi, “Plushie: an interactive design system for plush toys,” *ACM Trans. Graph.*, vol. 26, no. 3, pp. 45:1–45:8, 2007.

[51] M. Nettelbladt, http://www.omkrets.se/mnexjobb/english.htm\#e, 2007.

[52] H. Pottmann, Y. Liu, J. Wallner, A. Bobenko, and W. Wang, "Geometry of multi-layer freeform structures for architecture," *ACM Trans. Graph.*, vol. 26, no. 3, pp. 65:1–65:11, 2007.

[53] H. Pottmann, A. Schiftner, P. Bo, H. Schmiedhofer, W. Wang, N. Baldassini, and J. Wallner, "Freeform surfaces from single curved panels," *ACM Trans. Graph.*, vol. 27, no. 3, pp. 76:1–76:10, 2008.

[54] K. Rose, A. Sheffer, J. Wither, M.-P. Cani, and B. Thibert, "Developable surfaces from arbitrary sketched boundaries," in *Proc. 5th Eurographics Symp. Geom. Processing*, 2007, pp. 163–172.

[55] A. Schiftner, M. Höbinger, J. Wallner, and H. Pottmann, "Packing circles and spheres on surfaces," *ACM Trans. Graph.*, vol. 28, no. 5, pp. 139:1–139:8, 2009.

[56] A. Shamir, "A survey on mesh segmentation techniques," *Computer Graphics Forum*, vol. 27, no. 6, pp. 1539–1556, 2008.

[57] J. Sharp, "D-forms and developable surfaces," in *Proc. Bridges: Mathematical Connections in Art, Music and Science*, 2005, pp. 121–128.

[58] M. Singh and S. Schaefer, "Triangle surfaces with discrete equivalence classes," *ACM Trans. Graph.*, vol. 29, no. 4, pp. 46:1–46:7, 2010.

[59] L. Spuybroek, *NOX: machining architecture.* New York: Thames & Hudson, 2004.

[60] D. Systems, http://printin3d.com, 2011.

[61] TopMod, http://www.topmod3d.org, 2008.

[62] C. Troche, “Planar hexagonal meshes by tangent plane intersection,” in *Proc. Adv. Arch. Geom.*, 2008, pp. 57–60.

[63] T. Tyler, http://hexdome.com, 2009.

[64] S. Willerton, http://golem.ph.utexas.edu/category/2010/03/intrinsic_volumes_for_riemanni.html, 2010.

[65] T. Wills, “D-forms,” in *Proc. Bridges: Mathematical Connections in Art, Music and Science*, 2006, pp. 163–172.

[66] Woodshop, http://www.arch.tamu.edu/downloads/woodshop/Using_the_Laser_Cutter.pdf, 2011.

[67] J. Wu and L. Kobbelt, “Structure recovery via hybrid variational surface approximation,” *Comput. Graph. Forum*, vol. 24, no. 3, pp. 277–284, 2005.

[68] Q. Xing, G. Esquivel, R. Collier, M. Tomaso, and E. Akleman, “Weaving methods in architectural design,” in *Proc. Bridges: Mathematical Connections in Art, Music and Science*, 2011, pp. 216–224.

[69] Q. Xing, E. Akleman, J. Chen, and J. L. Gross, “Single-cycle plain-woven objects,” in *Proc. Shape Modeling Int. Conf.*, 2010, pp. 90–99.

[70] Q. Xing, E. Akleman, G. Taubin, and J. Chen, “Surface covering curves,” in *CAe 2011 Posters*, 2011, pp. 4:1–4:1.

[71] K. Xu, H. Zhang, A. Tagliasacchi, L. Liu, G. Li, M. Meng, and Y. Xiong, “Partial intrinsic reflectional symmetry of 3d shapes,” *ACM Trans. Graph.*, vol. 28, no. 5, pp. 138:1–138:10, 2009.

[72] D.-M. Yan, Y. Liu, and W. Wang, “Quadric surface extraction by variational shape approximation,” *Geom. Modeling & Processing*, pp. 73–86, 2006.

[73] D. Zorin and P. Schröder, *Subdivision for Modeling and Animation.* New York: The Association for Computing Machinery, Inc., 2000, ACM SIGGRAPH 2000 Course Notes.

APPENDIX 1

PLAIN WEAVING

A plain-weaving structure consists of threads that are interlaced so that a traversal of each thread alternately goes over and under the other threads (or itself) as it crosses them. To model a plain-weaving structure on a surface, we construct an alternating projection of a link. We prove that it is possible to create such a plain-weaving structure for any given manifold mesh by twisting all of the edges of a related orientable manifold mesh [2]. We regard edge twisting as an operation on cycles. Face boundary walk defines an alternating link. In other words, we can construct weaving structures that we want from orientable meshes by twisting edges.

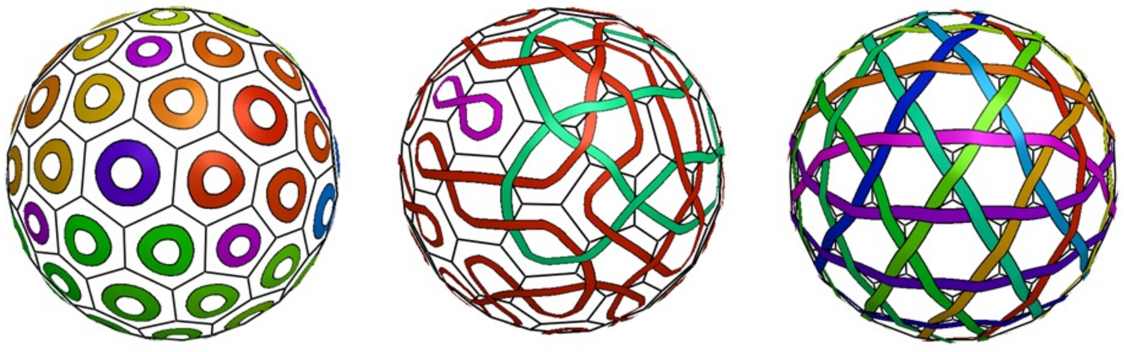
We first give the definition of a cyclic weaving.

Definition Let $C = c_1, ..., c_k$ be a finite set of disjoint cycles. A cyclic weaving from C to an orientable 2-manifold S is an immersion of the union of the cycles of C on the surface S.

Then we can give the definition of a cyclic plain-weaving.

Definition A cyclic weaving w from a cycle collection C to an orientable surface S is a cyclic plain-weaving if it satisfies the following conditions:

- each point on S has at most 2 pre-images in C;
- for each cycle c in C, suppose that $(p1, p2, ..., pm)$ is the ordered sequence of crossing points we encounter when traversing $w(c)$ on S, then $w(c)$ assigns these points alternatively as over and under (i.e., it weaves) when it traverses and crosses other circles or itself.



(a) no twill (b) some edges are twisted (c) all edges are twisted

Fig. 78. Twisting all edges in a mesh gives a cyclic plain-weaving.

Based on the definition, we can construct a cyclic plain-weaving on any input mesh by +1-twisting all the edges in the mesh. Figure 78 illustrates this process.

Theorem VII.0.1 *Let $r(G)$ be a graph rotation system with no twisted edges, which corresponds to an embedding of the graph G on an orientable 2-manifold S_h. Let A be an arbitrary subset of edges of G. If we twist all edges in A (arbitrarily but in the same orientation), then the resulting graph rotation system induces a cyclic plain-weaving on S_h.*

By changing the parameters which control the width and wavyness of the weaving ribbons, we can get both sparse and defense versions of the weaving patterns from the same input mesh. Figure 79 show examples of sparse and dense woven objects.

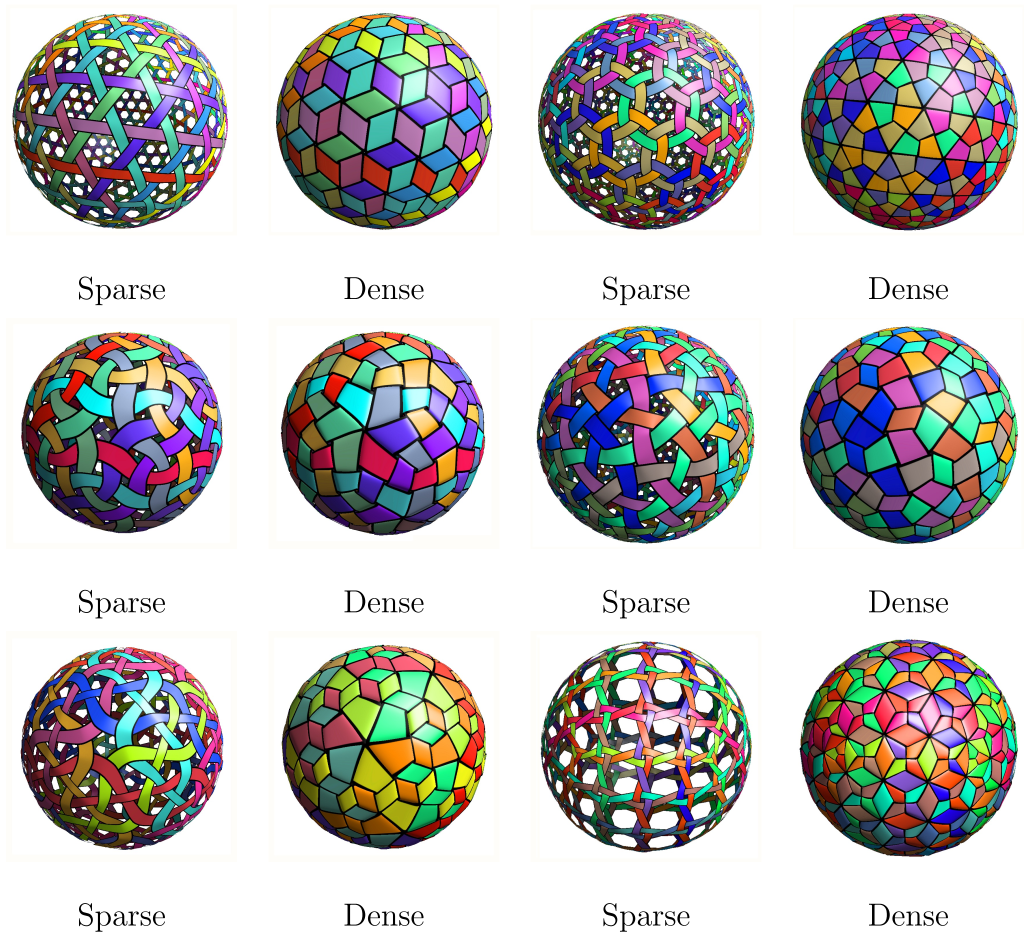


Fig. 79. Examples of sparse and dense versions for plain-weaving patterns on same input meshes.

VITA

Qing Xing
C108 Langford Center,
3137 TAMU
College Station, TX 77843-3137
qingxing@viz.tamu.edu

Education

Ph.D. in Architecture	Texas A&M University, December 2011
M.S. in Computer Science	Korea Advanced Institute of Science and Technology, August 2006
B.S. in Computer Science	Taiyuan University of Science and Technology, August 2004

Employment

Graduate Assistant, Non-Teaching	Texas A&M University, January 2007 - May 2011