Complex surface modeling : methods, algorithms and applications

Chen, Wenyu

2011

Chen, W. Y. (2011). Complex surface modeling : methods, algorithms and applications. Doctoral thesis, Nanyang Technological University, Singapore.

https://hdl.handle.net/10356/43840

https://doi.org/10.32657/10356/43840

COMPLEX SURFACE MODELING: METHODS, ALGORITHMS AND APPLICATIONS

NANYANG TECHNOLOGICAL UNIVERSITY

COMPLEX SURFACE MODELING: METHODS, ALGORITHMS AND APPLICATIONS

WENYU CHEN

SCHOOL OF MECHANICAL AND AEROSPACE ENGINEERING

2011

WENYU CHEN

2011

COMPLEX SURFACE MODELING: METHODS, ALGORITHMS AND APPLICATIONS

WENYU CHEN

School of Mechanical and Aerospace Engineering

A thesis submitted to Nanyang Technological University
in partial fulfillment of the requirement for the degree of
Doctor of Philosophy

2011

Abstract

Nowadays, three dimensional (3D) movies, such as "Avatar", "Clash of the Titans" and "How to Train Your Dragon", are leading the film industry into the 3D era. One supporting element behind the amazing scenes of the 3D movies is the complex surface. Apart from the film industry, complex surfaces have intensive applications in CAD/CAM, Biomedicine, and Interactive & Digital Media (IDM). These applications require efficient and robust methods for modeling of complex surfaces.

A complex surface is a surface that is derived from a complex structure of arbitrary topology. A molecular surface is a complex surface geometrically constructed as the boundary of a set of spheres, and a minimal surface can be derived from a single contour by minimizing the surface area. A complex surface can also be built up in free-form interactively. Users can conveniently perform local shape modifications with parameter control. Composite surface can be utilized to locate an area on a complex surface to facilitate the modifications. A hierarchical structure is necessary to represent the final complex surface.

This thesis investigates the complex surface modeling techniques covering four topics: molecular surfaces, minimal surfaces, composite surfaces, and hierarchical NURBS (H-NURBS) surfaces.

Molecular surfaces fall into three types: van der Waals surfaces (vdWS), solvent accessible surfaces (SAS) and solvent excluded surfaces (SES). A molecule can be geometrically represented as a set of spheres, whose topology can be highly complex. Different solutions for different types of molecular surfaces have been proposed. We design a uniform explicit solution to model all three types of molecular surfaces. In our solution, a molecular surface is decomposed and organized as a set of rational Bézier surfaces. The singularities for an SES are specially treated to avoid self-intersections. The study indicates that rational Bézier representation, more specifically, a bi-cubic or 2×4 rational Bézier surface, is sufficient for kernel modeling of molecular surfaces and related applications.

Minimal surface modeling is to obtain a surface given a closed contour, which is equivalent to the computation for a surface of minimal area. This is an old and complex problem. A numerical method for minimal surfaces is studied to construct

an aesthetically pleasing triangular mesh with a closed polygonal contour given as boundary. From all triangular meshes with the prescribed boundary and the number of triangles, a triangular mesh of minimal area is identified as the solution for this problem.

The composition technique is necessary to identify a local area of a surface. We employ the composition technique to derive a triangular sub-patch $\mathbf{S}$ from a triangular Bézier surface $\mathbf{T}$ of degree n. By assigning three boundary curves on the domain of $\mathbf{T}$, a domain surface $\mathbf{P}$ of degree m can first be constructed. The composite surface $\mathbf{S}$, of degree mn, is the composition of $\mathbf{P}$ and $\mathbf{T}$. An explicit formula for control points of $\mathbf{S}$ is obtained by shifting operators. A robust algorithm is developed in this thesis to calculate the control points.

H-NURBS allows a complex surface to be locally modified and globally transplanted. Based on multiresolution and refinement schemes, H-NURBS is investigated to design a mechanism for the purpose of carrying localized geometric information. H-NURBS based Monge mapping is developed for detail and local shape modifications. Monge mapping on H-NURBS patch can be easily performed via simple cut-&-paste operations. Parametric control of the local shapes is developed to facilitate easier and better 3D local modeling. With such a technique, a complex surface can be transplanted to any portion of another surface.

Acknowledgments

I wish to express my deepest appreciation and gratitude to my research supervisors Associate Professor Cai Yiyu and Associate Professor Zheng Jianmin, for their excellent guidance, constructive suggestions, patient encouragement and continuous support during the study of this research topic.

I thank Professor Guozhao Wang, the supervisor for my master's degree, who encouraged and recommended me to pursue my Ph.D degree. I thank my colleagues with whom I collaborated and had useful discussion: Baifang Lu, Chandrasekaran Indhumathi, Yunqing Guan, Patricia Chiang Wei Ying and Rongdong Yu. I also thank those friends who provide lots of helps during my study: Dong Huang, Rui Wang, Yuan Jiang, Zhiming Rao, Yuexiang Huang, Xianfei Jin, Qing Fu, Lishi Jiao, Yang Liu, and many others

I am also grateful to Mr. Chia Y.K., Mr. Soh B.C., and Mr Koh W.H., the technicians in Advanced Design and Modeling (ADAM) laboratory, for the technical supports and for providing a pleasant, productive and academic environment throughout this research.

I would like to express my sincere appreciation to my family members especially my parents who offer permanent supports. Finally, thanks to my wife Qing Li. Success in my research comes from her supports and understanding.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

In this thesis, the term 'complex surface' is short for 'complex-shaped surface', which is a surface that can be derived from a complex structure of arbitrary topology. A complex surface can be a surface of irregular topology or a surface with detail features. Nowadays, complex surfaces play important roles in many applications, among which we can group the complex surface modeling into two types: *beauty modeling* and *precision modeling*.

Beauty modeling is for beautifications. The focus of the *beauty modeling* is to create fine details on the model and improve the realism of the model. Some special effects in movies are impressive examples created by *beauty modeling*. Three dimensional (3D) movies, such as "Avatar", "Clash of the Titans" and "How to Train Your Dragon", are leading the film industry into the 3D era. To better attract audiences, the 3D movie usually provides beautiful scenes, which consist of complex elements such as trees, water, and monsters. To improve the realism of the scenes, each complex element should also provide details, for example, the human faces and the animal furs. Without effective modeling of these complex elements, movie production will become a very difficult job.

Precision modeling, however, requires the modeling to follow exactly their physical appearance. *Precision modeling* can be found in many fields including traditional computer aided design (CAD). Computer aided surgery is a new area of *precision*

modeling application, requiring the modeling of the human organism like the beating heart and the blood vessel network. The surgical procedures necessitate highly reliable models of the human anatomy. Computer aided drug design (CADD) uses computer modeling techniques to design small molecules (ligands) that are complementary to a molecular target. To make a ligand become a safe and efficacious drug, the exactly modeling of the molecular surfaces is important.

1.2 Motivations

This thesis attempts to investigate several techniques involved in complex surface modeling. These techniques relate to the recognition and perception of complex surfaces, which are important issues in the human visual system.

The human visual system can detect and discriminate between an incredibly diverse assortment of stimuli. Stimuli signals are first received by visual cortexes. There is a visual cortex for each hemisphere of the brain. The left hemisphere visual cortex receives signals from the right visual field and the right hemisphere visual cortex receives signals from the left visual field. Visual information is received at the primary visual cortex (V1), and passed through a cortical hierarchy (V2, V3, V4 and V5). Infero temporal cortex (IT) is crucial for visual object recognition and is considered to be the final stage in the visual recognition process (Figure 1.1).

The human visual system represents a complex shape as many simple geometric components and organizes the spatial relationships of these components [1–3]. A complex shape is usually decomposed into simple geometric components such as cylinders, cones, wedges and blocks. It has long been suggested that complex shape representations are organized as hierarchical structures [3–5].

Visual shape recognition depends on a multi-stage pathway from V1 to IT. The mechanisms by which the signals from V1 are transformed into IT are not well understood. The research in [6–8] suggests that, at intermediate stages in the V1-IT transformation, complex objects are represented at least partly in terms of the configurations and positions of their contour components, and the major boundary features could be used to reconstruct (approximately) the original shape.

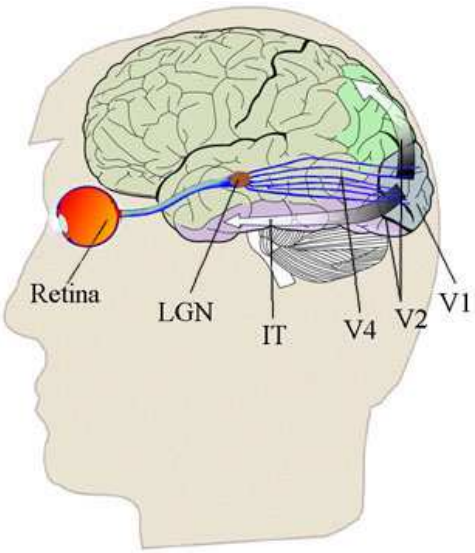


Figure 1.1: Human visual recognition process.
(From Wikipedia [9])

Basing on all these researches on the human visual system, we are interested in investigating three techniques in complex surface modeling:

(1) *Surface decomposition* which divides a complex surface into sub-patches.

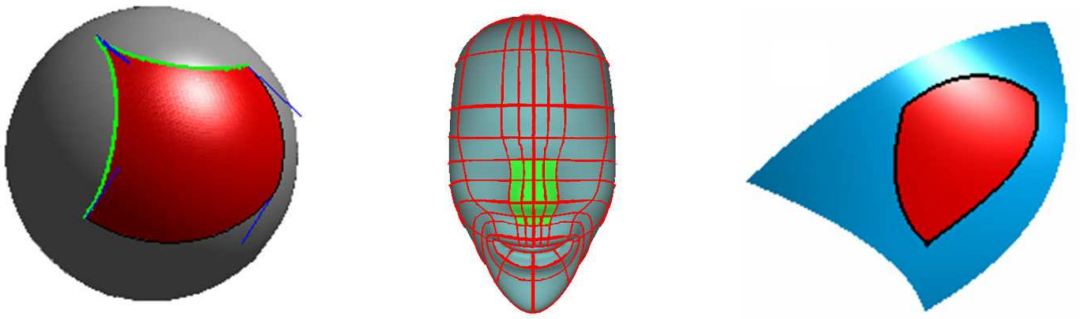
(2) *Surface in hierarchy* which maintains the sub-patches and modify the local features.

(3) *Surface from contours* which reconstructs surface information from boundary contours.

1.2.1 Surface decomposition

Decomposition of a complex model into simple components is an intuitive way to represent the model. Different decompositions will create different representations of the complex model. During decomposition, how to describe simple components and how to organize these simple components are also crucial.

A complex model can be represented in two different ways: *constructive solid geometry* (CSG) or *boundary representation* (B-reps). In CSG, a complex object is represented as a solid, which can be assembled by elementary objects like cubes, tetrahedrons, pyramids, spheres. The solid is finally modeled as a binary tree of Boolean operations [10]. In B-reps, a complex surface is used to represent a complex



(a) (b) (c)

Figure 1.2: Retrieval of a sub-patch from an existing surface.
(a) retrieval of a regular rectangular Bézier surface from the unit sphere via generalized stereographic projection; (b) retrieval of a NURBS sub-patch from a NURBS surface (red curves are isoparametric curves); and (c) retrieval of a triangular Bernstein-Bézier sub-patch from a triangular Bernstein-Bézier surface via function composition.

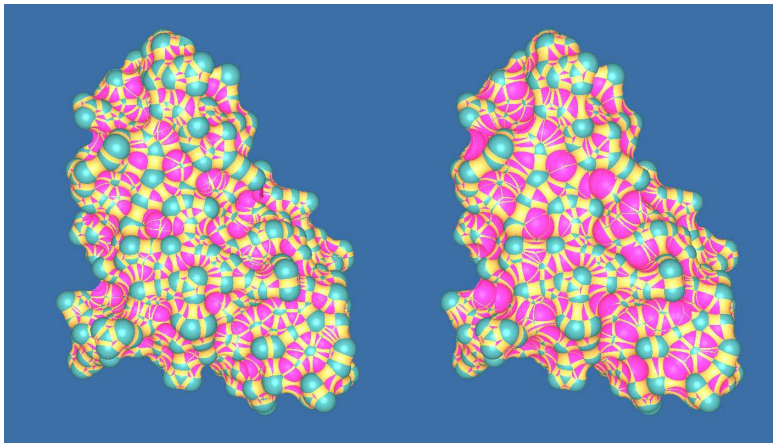
model. Usually, a complex surface is divided into sub-patches, which are boundaries of the elementary objects that form the solid. A well-formed complex surface must satisfy certain conditions [11]: it mush be closed, orientable, non-self-intersecting, bounded, and connected.

In this thesis, we study the B-reps complex surfaces, which can be polynomial parametric surfaces, such as tensor product Bézier surfaces, triangular Bernstein-Bézier surfaces, and NURBS surfaces. In particular, we are interested in decomposing a complex surface into sub-patches which can be achieved via function composition in three different ways:

(1) Rectangular sub-patches: if a rectangular sub-patch of the unit sphere uses circular arcs as its boundaries (Figure 1.2(a)), the sub-patch can be described as a rational tensor product Bézier surface using generalized stereographic projection proposed by Dietz *et al.* [12, 13].

(2) NURBS sub-patches: if a sub-patch of a NURBS surface uses isoparametric curves as its boundaries (Figure 1.2(b)), the sub-patch is also a NURBS surface. The degrees of the sub-patch are same as the degrees of the NURBS surface.

(3) Triangular sub-patches: if a triangular sub-patch of a triangular Bernstein-Bézier surface does not have isoparametric boundary curves (Figure 1.2(c)), the sub-patch will be a triangular Bernstein-Bézier surface with higher degree.



(a) (b)

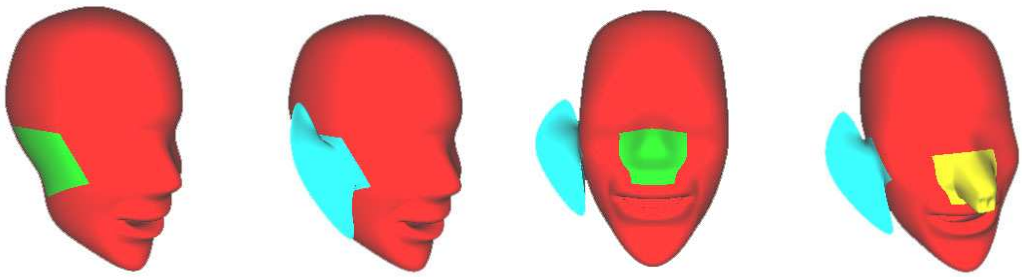
Figure 1.3: Decomposing molecular surfaces.
three types of sub-patches in three different colors: (a) with probe radius 1.5Å; and (b) with probe radius 2Å.

We can take the molecular surface as an example to show how a complex surface is divided into patches. A solvent excluded surface (SES) is a molecular surface defined by rolling a spherical probe along its boundary (Figure 1.3). An SES is a complex surface, which is usually decomposed into concave patches, saddle patches and convex patches [14]. Concave patches and convex patches are all spherical patches, each of them is a sub-patch of a sphere. Function composition is necessary to represent each sub-patch in rational Bézier form. Generalized stereographic projection [12, 13], will be employed to describe a spherical patch (Figure 1.2(a)).

1.2.2 Surface in hierarchy

Typically based on a triangular discrete model [15] or an analytical model [16], the hierarchical structure has been widely used in computer graphics in different applications. Our focus is to develop an analytical model with NURBS representation. Such a hierarchical structure is to provide different mechanisms for local modifications. The proposed model is named as the hierarchical NURBS (H-NURBS). Based on H-NURBS, the function composition can be used not only for surface decompositions but also for selecting an area for local modifications.

Figure 1.4 shows an example of local modifications. The initial surface is a NURBS surface with some simple features (Figure 1.4(a)). After selecting a local area, we can do a local modification to form an ear (Figure 1.4(b)). Similarly, a local modification



(a) (b) (c) (d)

Figure 1.4: Local modifications.

red : original surface; green : local shape; other colors: modified shapes. (a) a local shape; (b) a modification of the ear shape; (c) another local shape; and (d) a modification of the nose shape.

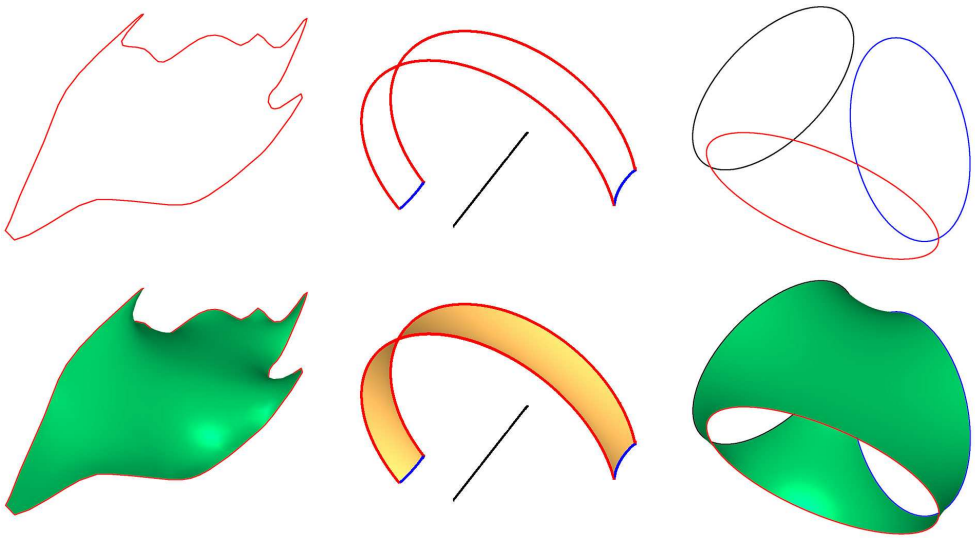
for a nose shape can be created (Figure 1.4(c,d)). After modifications, each part of the surface can be derived in a NURBS representation.

1.2.3 Surface from contours

Surface from contours means to reconstruct a surface using given contours as boundaries fulfilling some conditions.

With different conditions, different surfaces can be reconstructed from a single closed contour. If we want the surface to be of minimal area, the problem becomes the minimal surface problem or the Plateau's problem (Figure 1.5(a)). If a closed contour consists of four arcs (two red arcs and two blue arcs in Figure 1.5(b)), we can create a surface of revolution, which is a saddle patch. In molecular surface modeling, a saddle patch is created when the probe is tangent to two atoms. It is created by rotating probe along the edge connecting the two atom centers (the black line in Figure 1.5(b)).

Other applications of surfaces from contours require obtaining a surface from multiple closed contours. In modeling the human blood vessel network, we need to handle the bifurcations. In Figure 1.5(c), one blood vessel ends at the red circle and two blood vessels start at the black and blue circles, respectively. The bifurcation area can be described as a surface using three circles as contours.



(a) (b) (c)

Figure 1.5: Surface from contours.
(a) a minimal surface from one contour; (b) a saddle surface defined by four arcs; and (c) a surface of minimal area using three contours as boundaries.

1.3 Objectives

The purpose of this research is to develop different complex surface modeling techniques for complex surfaces.

1.3.1 Molecular surfaces

There are three types of molecular surfaces: van der Waals surface (vdWS), solvent accessible surface (SAS) and solvent excluded surface (SES). Different approaches, such as skin surfaces, reduced surfaces, and trimmed NURBS, have been proposed to model these molecular surfaces. Usually, each type of molecular surfaces has one specific surface approach for the modeling job. In this thesis, a new method to model and render all three types of molecular surfaces is proposed with the following objectives:

(1) To use only rational Bézier surfaces for all three types of molecular surfaces.

(2) To derive multiresolution meshes dynamically.

(3) To achieve higher computational and rending efficiency from a compact rational Bézier representation.

1.3.2 Minimal surfaces

A new iterative algorithm to construct a triangular mesh from a given polygonal boundary is presented. The algorithm aims:

(1) To generate a triangular mesh with minimal area while minimizing the mean curvature on each vertex.

(2) To obtain an initial mesh of a pre-defined number of triangles.

(3) To extend the approach for multi-contours problems.

1.3.3 Composite surfaces

A new algorithm of constructing a triangular Bézier sub-patch from a triangular Bézier surface is proposed. The sub-patch is a composite surface of a triangular domain surface and the triangular Bézier surface. Our algorithm aims:

(1) To handle surfaces of any degrees.

(2) To calculate the control points of the composite surface using explicit formulae.

(3) To design a geometric approach to construct control points.

1.3.4 Hierarchical NURBS

Hierarchical NURBS (H-NURBS) aims to describe complex surfaces with a new model:

(1) To deal with NURBS surfaces, and hence to handle all existing models in current CAD/CAM systems.

(2) To support the local shape modifications.

(3) To transplant surface features using cut-&-paste operations.

(4) To control the local shapes parametrically.

1.4 Thesis organization

This is a multi-disciplinary research involving applied mathematics, computer graphics, and bio-medicine. The thesis is organized as follows:

Chapter 1 gives a general introduction to our research.

Chapter 2 proposes a uniform solution for molecular surface modeling after a detail review of the prior art. A kernel modeler is developed for three types of molecular surfaces using a uniform rational Bézier representation with bi-cubic or 2×4 rational Bézier surfaces. Each molecular surface is modeled as piecewise rational Bézier surfaces. Meanwhile, to make a molecular surface compatible with B-reps, the self-intersecting problem is specially addressed.

Chapter 3 describes a new method to model minimal surfaces. An iterative algorithm is presented to construct a triangular mesh from a given polygonal boundary. Experimental results are promising. The proposed algorithm can also be applied to construct a triangular mesh from multi-contours. Literature review on minimal surface is also carried out in this chapter.

Chapter 4 discusses the prior work and then presents an algorithm to obtain the triangular Bézier sub-patches from a triangular Bézier surface. Explicit formulae for the control points of the sub-patch are developed. A robust algorithm is provided to obtain the control points.

Chapter 5 first reviews the relevant research and then investigates the H-NURBS hierarchy for detail and local shape modifications. Based on the multiresolution and refinement schemes, Monge mapping using H-NURBS is developed for easy cut-&-paste operations. The parametric control of the local shapes is developed to facilitate easier and better 3D local modeling.

Chapter 6 concludes the researches and highlights the future work.

Chapter 2

Molecular Surfaces[1]

In this chapter, a rational Bézier surface is proposed as a uniform approach to modeling all three types of molecular surfaces: van der Waals surface (vdWS), solvent accessible surface (SAS) and solvent excluded surface (SES). Each molecular surface can be divided into molecular patches, which can be defined by their boundary arcs. The solution consists of three steps: topology modeling, boundary modeling and surface modeling. Firstly, using a weighted α-shape, topology modeling creates two networks to describe the neighboring relationship of the molecular atoms. Secondly, boundary modeling derives all boundary arcs from the networks. Thirdly, surface modeling constructs all three types of molecular surfaces patch-by-patch, based on the networks and the boundary arcs. For an SES, the singularities are specially treated to avoid self-intersections. Instead of approximation, this proposed solution can produce precise shapes of molecular surfaces. Since rational Bézier representation is much simpler than a trimmed non-uniform rational B-spline surface (NURBS), computational load can be significantly saved when dealing with molecular surfaces now. It is also possible to utilize the hardware acceleration for tessellation and rendering of a rational Bézier surface. CAGD kernel modelers typically use NURBS as a uniform representation to handle different types of free-form surfaces. This research indicates that rational Bézier representation, more specifically, a bi-cubic or 2×4 rational Bézier surface, is sufficient for kernel modeling of molecular surfaces and related applications.

[1]The following publication is based on the results of this chapter:
Chen,W.Y., Zheng,J.M., and Cai,Y.Y. (2010). **Kernel modeling for molecular surfaces using a uniform solution**,*Computer-Aided Design*, 42(4), 267-278.

2.1 Prior art

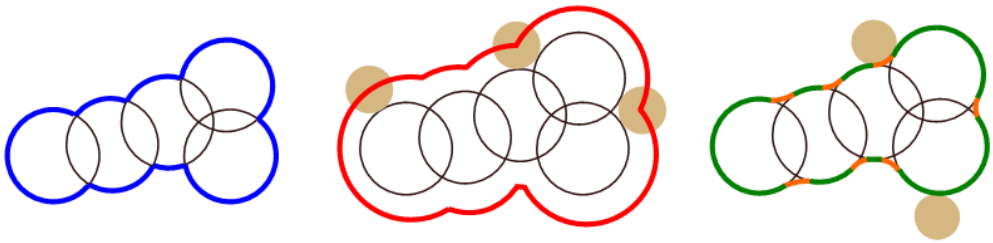
Biological molecules like proteins are important for all biological organisms. In order to understand the molecular functions, molecular structures are intensively studied at different levels. Since molecules interact at their surfaces, an understanding of the molecular surfaces can be useful for studying these interactions. Molecular surfaces can be used in molecular visualization and analysis, function predictions and drug design. Three types of molecular models have been proposed: van der Waals surface (vdWS), solvent accessible surface (SAS), and solvent excluded surface (SES). Different methods have been proposed to model these surfaces.

The vdWS [17, 18] is used to describe a molecule based on its atoms of van der Waals radius. It can be defined as the union of all portions of every atomic sphere surface that is not occluded by neighboring atoms (Figure 2.1(a)). Several methods have been developed to model vdWS [19–21]. A simple application of a vdWS is to compute the molecular volume, which can be described as the volume enclosed by the vdWS. Adams [22] discussed the calculation of the volume and area of vdWS.

Although the vdWS is a reasonable model for molecules, it does not address the issue of whether or not an atom is accessible to the solvent environment. The SAS [18, 23] was proposed, taking into account the effect of the solvent. It is described as the surface created by the center of a probe rolling over the entire vdWS. This can be considered as a van der Waals surface whose atomic radii have been extended by the probe radius (Figure 2.1(b)). Hence, the modeling of an SAS can be similar to that of a vdWS [24–28]. Calculations of the volumes and areas of SAS were studied in [29–33].

The SES [34] is another description of the molecular surfaces which also takes into account the solvent. It, however, is defined as the surface traced out by the inward-facing surface of a probe (Figure 2.1(c)). The SES consists of contact surfaces and re-entrant surfaces. Contact surfaces are the parts of the vdWS that can be touched by a probe and the re-entrant surfaces are the inward-facing parts of the probe atom when it is in touch with more than one atom. An SES is a smooth molecular surface except at the singular points, which will be elaborated later. As

proposed by Connolly [24], an SES consists of three types of regions: concave spherical patches, convex spherical patches and saddle-shaped toroidal patches (or briefly saddle patches). Various methods have been proposed to model SES [35–38].



(a) (b) (c)

Figure 2.1: Three types of molecular surfaces.
(a) vdWS, in blue; (b) SAS, in red; and (c) SES, with contact surfaces in green and re-entrant surfaces in orange.

Besides the above three types of molecular surfaces, several other surfaces have been used to describe molecules. Blobby molecules [39] used an isosurface as an algebraic representation of a molecule. This isosurface is defined by an implicit function, which can be computed as a distance field for the atoms by superimposing a set of radial basis functions (RBF). Since each atom contributes to the function, the computation cost for large molecules dramatically increases. To solve this problem, the soft objects [40–42] provided several improvements in the formulation and optimization of the implicit functions. The final surfaces for these methods can be extracted by marching cubes [43]. The skin surface [44] is based on a framework of Voronoi, Delaunay and α complexes of a finite set of weighted points. One advantage of the skin surface is that it is smooth and free of self-intersections. Therefore, it needs no handling of the singularities. Several methods have been proposed for the triangulation of the skin surface [45–50]. All these surfaces can be treated as approximations to the three types of molecular surfaces. In this chapter, we only discuss the modeling of the three types of molecular surfaces, i.e. vdWS, SAS, and SES.

2.2 Research aims

The three different types of molecular surfaces are all important for structure and function analysis. Often, different methods are used to model these different types

of molecular surfaces. For example, Richard's method [34] is used to model an SAS and Connolly's method [14] is used to model an SES. As such, often different data structures are required to model different surfaces. To have compatible and uniform molecular surface representation, a simple polygonal representation [25, 51–54] is used through approximation and tessellation. Such an approximation method may produce an inaccurate representation, which may cause problems especially for dynamics surface simulation. Besides, tessellation can easily create millions of triangular meshes. A rational Bézier surface is more cost-effective in terms of computing compared to a trimmed non-uniform rational B-spline surface (trimmed NURBS) [55, 56]. With a rational Bézier surface, there is also an advantage of creating an adaptive and crack-free tessellation model when needed. Furthermore, the hardware support for Bézier surfaces has been developed. For example, NVIDIA's GeForce3 supports the tessellation of Bézier surfaces with OpenGL extensions for Bézier surface evaluation (both rational and non-rational) [57].

Connolly's method [14] modeled an SES as a collection of analytical patches without handling singularities. In CAGD community, there is an increasing interest on molecular surface modeling. Bajaj *et al.* [55, 56] used a trimmed NURBS (not a NURBS) to describe the molecular surfaces with a control mesh and a set of trimmed curves. A rational Bézier surface is more convenient in modeling and rendering with potential hardware acceleration. A piecewise polynomial Bernstein-Bézier spline function was used by Zhao [58] for molecular surface representation. Theoretically, this approach can only produce an approximate representation. For accurate molecular surface representation, a rational Bézier surface is necessary.

The aims of this research are as follows:

(1) To develop a uniform approach to modeling all three types of molecular surfaces.

(2) To have an accurate representation of molecular surfaces instead of approximation.

(3) To design an effective and robust method for molecular surface representation, using a low-degree rational Bézier surface, thus avoiding the computationally expensive NURBS.

The proposed research has several applications:

(1) To design a kernel modeler for molecular surfaces using a uniform rational Bézier representation, and therefore uniform data structure.

(2) To tell exactly how many rational Bézier patches are there in the molecular surfaces, and thus to calculate the surface areas, etc.

(3) To speed up the rendering process using graphics hardware.

(4) To support the level of detail through the subdivision of the control meshes of the rational Bézier represented molecular surfaces.

2.3 An overview of molecular surface modeling

2.3.1 A uniform method for three types of molecular surfaces

The rational Bézier modeling method provides a uniform solution to represent all three types of molecular surfaces. Notice that the SES consists of concave spherical patches, convex spherical patches and saddle-shaped toroidal patches, while the vdWS and SAS only contain convex spherical patches. Therefore, our focus is to model all these patches in rational Bézier form. To do so, we need to set up the topology of the boundaries.

2.3.2 Networks, arcs, and patches

The rational Bézier modeling contains three steps (Figure 2.2): topology modeling, boundary modeling and surface modeling. In topology modeling, two networks can be developed to reveal the neighboring relationship of the boundary atoms: the vdWS network and the solvent network. The vdWS network is used to construct the vdWS while the solvent network is used for both the SES and the SAS. The two networks can be created from the existing methods, for example, regular triangulation [59], β-shape [38, 60] and weighted α-shape [61, 62]. In our system, a weighted α-shape is adopted. From the two networks, boundary modeling creates boundary arcs for

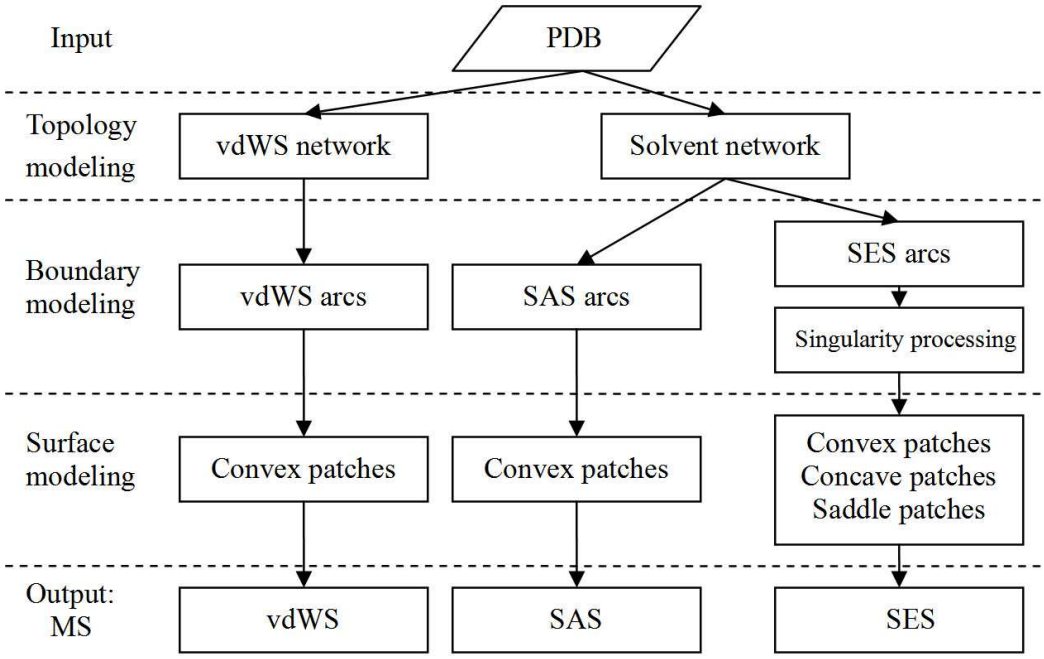
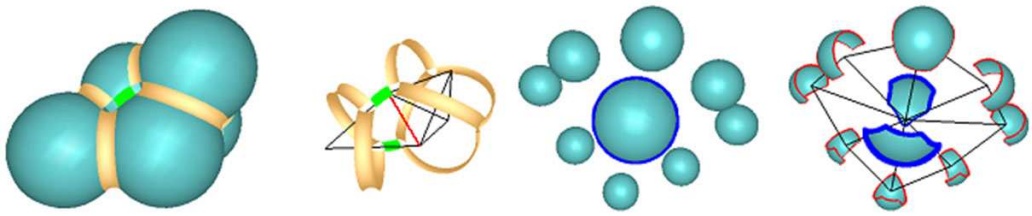


Figure 2.2: A uniform solution for molecular surface modeling.

all the patches. Each patch is formed by its boundary arcs. For all three types of molecular surfaces, three collections of boundary arcs are created: vdWS arcs, SES arcs and SAS arcs. We can first create the intersecting circles on the atoms and then divide the circles into arcs [55, 63]. It is a time-consuming task to calculate the intersections of arcs and circles. We thus develop a more efficient way to create boundary arcs via probe rotation. We identify saddle patches to find the boundary arcs. For the SES, our method is different from Connolly's [14] and Bajaj's [55]. Connolly constructed the torus and divided it into saddle patches. Bajaj used a single trimmed NURBS to model the saddle toroidal patches from each edge of the regular triangulation boundary. There are several saddle patches for one edge (two saddle patches are created from one edge in Figure. 2.3). We, however, handle saddle patches one by one. For the SES, we can eliminate singularities by cutting the existing SES arcs and adding new SES arcs. After boundary modeling, we can find all the patches based on the networks and the boundary arcs. Each patch is N-sided, with N arcs as boundaries. Surface modeling is to create each patch using a rational Bézier surface. From one boundary atom, several N-sided spherical patches can be created (two N-sided patches are created from one atom in Figure 2.3(c, d)). Based on the boundary arcs, Connolly [14] proved that the SES is analytical without discussing



(a) (b) (c) (d)

Figure 2.3: Several patches created from one edge and one atom.
(a) saddle patches are created among atoms; (b) two green saddle patches are created from the red edge; (c) convex patches can be created from atoms; and (d) two N-sided spherical patches with blue boundary arcs are created from one atom.

the singularities. Our method not only provides a mathematical description for each patch but also deals with all the singularities for the SES. In the trimmed NURBS method, corresponding to a vertex of the regular triangulation boundary, a trimmed NURBS was created to form part of the molecular surfaces. However, an atom may contain more than one convex N-sided patch. Hence, a single trimmed NURBS may contain several N-sided patches. It is difficult to directly know how many convex patches a trimmed NURBS may contain. Our method models each separated patch and thus it is straightforward to obtain the information of each patch and the neighboring relationship among all patches. Moreover, the trimmed NURBS approach may possibly produce isolated cavities inside a molecule. We are not interested in the interior part of a molecule (for details see Section 2.4.1). Ryu *et al.* [37, 38] represented a re-entrance patch using a single implicit equation. Our method adopts the explicit rational Bézier representation, which enables easier tessellation and faster rendering.

We use rational Bézier surfaces to model all three types of molecular surfaces. Generally, the shape of a molecule is controlled by its network. The boundary arcs are created from the network and further used to create the molecular surfaces. For the SES, the modeling of the re-entrance surface (the concave patches and the saddle patches) was investigated by [35–38]. The convex patches are replaced by boundary atoms for visualization. Our method, however, also models the convex patches in rational Bézier form.

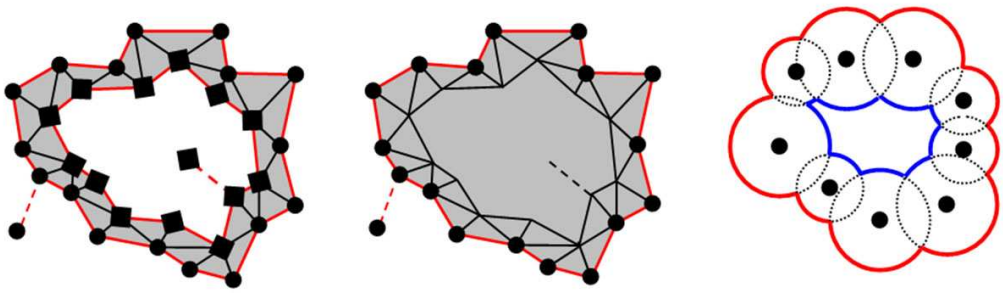
Our uniform solution contains three steps to model the molecular surfaces (Figure 2.2): topology modeling, boundary modeling and surface modeling. In next sections, we will detail the three steps.

2.4 Topology modeling

Topology modeling aims to find the neighboring relationship of the boundary atoms. For any molecule, the topology contains two networks: the vdWS network and the solvent network. The vdWS network contains the neighboring relationship of the molecular atoms and the solvent network reveals the connectivity of atoms when the probe is rolling along the boundary. Each network contains five basic elements: boundary vertices, singular edges, regular edges, singular triangles and regular triangles. In our system, given a probe radius α_0, we create these elements based on a weighted α-shape using CGAL [64].

2.4.1 Elements of a weighted α-shape

A molecule can be viewed as a set of weighted points S. Using a weighted α-shape [61, 62], six sets can be derived: singular edges SE(S), regular edges RE(S), singular triangles ST(S), regular triangles RT(S), interior tetrahedra ITE(S) and all tetrahedra ATE(S). The set ATE(S) contains several infinite tetrahedra which use the infinite points as one of their vertices. The radius of the sphere circumscribing an interior tetrahedron is smaller than the probe radius. A regular triangle is a facet of an interior tetrahedron while a singular triangle is not contained by any interior tetrahedron. A regular edge is an edge of a triangle (both regular and singular) while the singular edge is not contained by any triangle. Figure 2.4(a) shows an example of the weighted α-shape on the plane. The red solid lines are regular edges and the red dotted lines are singular edges. The weighted α-shape is derived from the regular triangulation. From



(a) (b) (c)

Figure 2.4: Topology modeling creates networks from a weighted α-shape. (a) elements of the weighted α-shape; (b) elements after topology modeling; and (c) results of the trimmed NURBS method.

all the tetrahedra of the regular triangulation, a weighted α-shape can be formed by removing the tetrahedra whose circumscribed spheres are larger than the probe. The weighted α-shape allows the probe to move to any point in the space to remove the tetrahedra. Since we want to model the boundary surface of the molecule, we will not allow the probe to move inside the molecule (Figure 2.4(b)). The black squares and black dots in Figure 2.4(a) are boundary vertices of the weighted α-shape. However, only the black dots can be touched by the probe now. Hence, we need to derive another subset from the weighted α-shape.

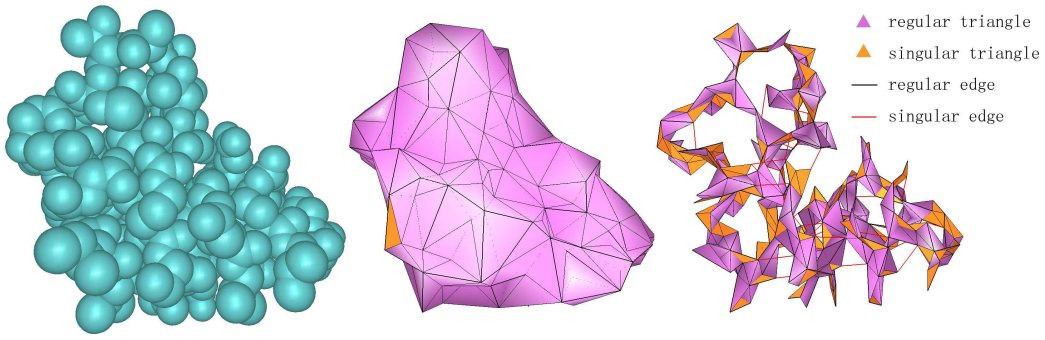
In Figure 2.4(c), all the atoms are boundary atoms. Our focus is to model only the boundary surface (arcs in red). The trimmed NURBS method creates two N-sided spherical patches from one atom (one blue arc and one red arc). A single trimmed NURBS surface is used to describe the two N-sided spherical patches. Therefore, it produces cavities inside the molecule. Of course, one can modify the trimmed NURBS method to model each N-sided spherical patch using one trimmed NURBS. However, it is not straightforward to check whether the N-sided spherical patch is inside the molecule or on the molecular boundary.

2.4.2 Solvent network

As shown in Figure 2.2, the solvent network is used for the SES and the SAS. It reveals the neighboring relationship of the atoms when a probe with radius α_0 rolls along the molecular shape. We move the probe from infinity until it touches the molecule. Five basic elements are created: solvent boundary vertices (SBV), solvent singular edges (SSE), solvent regular edges (SRE), solvent singular triangles (SST) and solvent regular triangles (SRT). The center of the atom that the probe can touch is called a solvent network vertex (black dots in Figure 2.4(b)). A solvent (singular or regular) edge is created connecting two atom centers if both atoms contact the probe at the same time (red line segments in Figure 2.4(b)). If the probe is tangent to three atoms, a solvent (singular or regular) triangle is created connecting the three atom centers. In order to derive these elements, we need to obtain the solid SO(S), which is a set of tetrahedra.

Suppose A(S) is another set of tetrahedra, which initially contains one infinite tetrahedron. Let triangle set C(S)= B(S)-ST(S)-RT(S), where B(S) contains all boundary triangles of A(S). If C(S)≠ ∅, the tetrahedra sharing the triangles in C(S) are inserted into A(S). Repeat checking C(S) and inserting tetrahedra until C(S)= ∅. Then, SO(S)= ATE(S)-A(S).

Element SRT contains all boundary triangles of the solid SO(S) and element SST contains all triangles in ST(S) that are outside the solid SO(S). Element SSE consists of all the edges in SE(S) that are outside the solid SO(S). Element SRE consists of all the edges from both SRT and SST. Element SBV contains all the vertices of the edges and the triangles. A radius of $\alpha_0 = 1.5$Å is typically used to represent the effective radius of a water molecule. Figure 2.5(b) shows an example of the solvent network with probe radius 1.5Å.



(a) (b) (c)

Figure 2.5: Two networks created from molecule 1CRN.
(a) all the atoms; (b) the solvent network of radius 1.5Å and (c) the vdWS network of radius 0Å.

2.4.2.1 The vdWS network

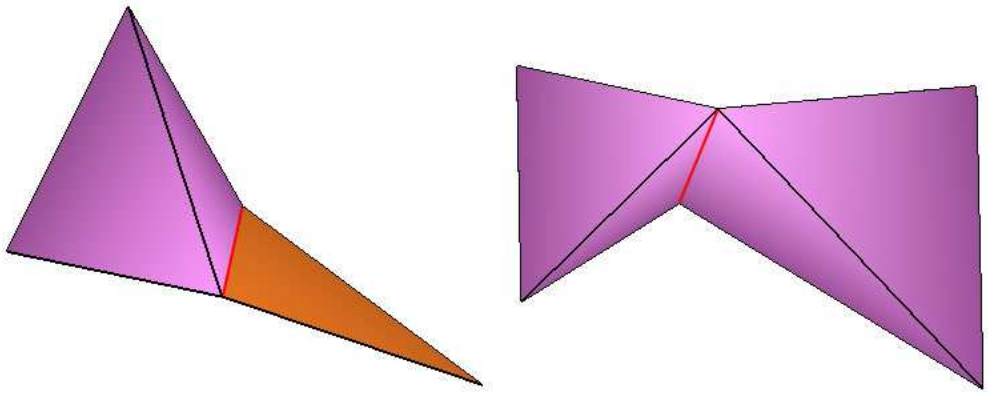
The solvent network elements are created using a probe with radius α_0. Following the same idea as above, the vdWS network elements can be created using the probe with radius 0Å (equivalent to a point). Five basic elements are created: vdWS boundary vertices (VBV), vdWS singular edges (VSE), vdWS regular edge (VRE), vdWS singular triangles (VST) and vdWS regular triangles (VRT). The center of the atom that the probe can touch is a vdWS network vertex. A vdWS (singular or regular) edge is created connecting two atom centers if the two atoms are not separated. If three atoms contact to one another, a vdWS (singular or regular) triangle is created

by connecting the three atom centers. Figure 2.5(c) shows an example of the vdWS network.

2.4.3 Properties of the dual-layers topology

After topology modeling, we obtain the solvent network and the vdWS network. We will use these two networks to model the molecular surfaces. A regular edge is an edge of the triangle (regular or singular). If the regular edge is an edge of a singular triangle, it is possible that several singular triangles share this edge. If the regular edge is an edge of a regular triangle, it will be on the solid SO(S). There must be an even number of regular triangles sharing this edge (Figure 2.6). Hence, we have

Property 1: A regular edge can be shared by $2n$ regular triangles and m singular triangles, where n and m are non-negative integers.



(a) (b)

Figure 2.6: A regular edge and triangles.
a regular edge (in red) shared by $2n$ regular triangles and m singular triangles. (a) an example with $n = m = 1$; and (b) another example with $n = 2, m = 1$.

2.5 Boundary modeling

All boundaries of each molecular patch are circular arcs [14, 55, 56]. Boundary modeling is thus to create all the boundary arcs. A boundary arc can be denoted as $[p_{start}, p_{end}, d]$, with start point p_{start}, end point p_{end}, and the start tangent direction d. For all three types of molecular surfaces, three collections of boundary arcs will be created: vdWS arcs, SES arcs and SAS arcs. The solvent network is used to create the SES arcs and SAS arcs. The vdWS network is used to create the vdWS arcs.

Clearly, when the probe rotates along an edge while keeping in touch with two atoms of the edge, boundary arcs are created. During the boundary arc formation, the probe has two special positions: the rotation starts and the rotation ends. We name these positions as stations, which are spheres of the same probe radius.

2.5.1 Stations

For a topology triangle T, there is a station where the probe is tangent to the three atoms (Figure 2.7(a)). Suppose the three atoms from the triangle are located at P_1, P_2, and P_3 with radii r_1, r_2, and r_3, respectively. If the center of the station is P, it must satisfy

$$|PP_i| = \alpha + r_i, i = 1, 2, 3.$$

There are two possibilities: $P = P_{normal}$ and $P = P_{symmetry}$ (Figure 2.7(c)). P_{normal} lies along the normal of the triangle while $P_{symmetry}$ lies on the opposite side. The volume of the tetrahedron $PP_1P_2P_3$ is

$$V = \frac{1}{12}\sqrt{4DEF - FXX - DYY - EZZ + XYZ} \qquad \text{(Eq.2.1)}$$

where

$$\begin{aligned} &A = |P_1P_2|^2, \quad B = |P_2P_3|^2, \quad C = |P_1P_3|^2, \\ &D = (\alpha + r_1)^2, \quad E = (\alpha + r_2)^2, \quad F = (\alpha + r_3)^2, \\ &X = D + E - A, \quad Y = E + F - B, \quad Z = D + F - C. \end{aligned} \qquad \text{(Eq.2.2)}$$

We can set

$$\begin{aligned} &P_{ij} = P_j - P_i = P_iP_j, \quad n = P_{12} \otimes P_{13}, \\ &G = P_{12} \cdot P_{13}, \quad H = n \cdot n, \quad I = P_2 \cdot P_2 - P_1 \cdot P_1, \\ &J = P_3 \cdot P_3 - P_1 \cdot P_1, \quad K = n \cdot P_1, \quad L = I - E + D, \\ &M = J - F + D, \quad S = CL - GM, \quad T = AM - GL. \end{aligned} \qquad \text{(Eq.2.3)}$$

where "$\otimes$" is the cross product and "$\cdot$" is the dot product.

The point P (P_{normal} or $P_{symmetry}$) satisfies

$$\begin{aligned} |P_2P|^2 - |P_1P|^2 &= E - D, \\ |P_3P|^2 - |P_1P|^2 &= F - D, \\ n \cdot P_1P &= \pm 6V. \end{aligned}$$

That is

$$\begin{aligned} P_2 \cdot P_2 - P_1 \cdot P_1 - (E - D) &= 2P_{12} \cdot P, \\ P_3 \cdot P_3 - P_1 \cdot P_1 - (F - D) &= 2P_{13} \cdot p, \\ n \cdot P &= \pm 6V + n \cdot P_1. \end{aligned}$$

In matrix form, we have

$$\begin{pmatrix} P_{12} \\ P_{13} \\ n \end{pmatrix} P = \begin{pmatrix} (P_2 \cdot P_2 - P_1 \cdot P_1 - E + D)/2 \\ (P_3 \cdot P_3 - P_1 \cdot P_1 - F + D)/2 \\ \pm 6V + n \cdot P_1 \end{pmatrix}$$

Equivalently, we have $M_1P = V_1$ or $M_1P = V_2$ where

$$M_1 = \begin{pmatrix} P_{12} \\ P_{13} \\ n \end{pmatrix}, V_1 = \begin{pmatrix} L/2 \\ M/2 \\ 6V + K \end{pmatrix}, V_2 = \begin{pmatrix} L/2 \\ M/2 \\ -6V + K \end{pmatrix}.$$

We thus have

$$|M_1| = (P_{12} \otimes P_{13}) \cdot n = |n|^2 = H, \qquad M_1^{-1} = \frac{1}{H} \begin{pmatrix} P_{13} \otimes n \\ n \otimes P_{12} \\ n \end{pmatrix}^{\perp}.$$

where "$\perp$" is a transpose operator of a matrix and "-1" is an inverse operator of a matrix.

Since

$$\begin{aligned}
a \otimes (b \otimes c) &= b\,(a \cdot c) - c\,(a \cdot b)\,, \\
P_{13} \otimes n &= P_{13} \otimes (P_{12} \otimes P_{13}) \\
&= |P_{13}|^2 P_{12} - P_{13}\,[P_{13} \cdot P_{12}] = CP_{12} - GP_{13}, \\
n \otimes P_{12} &= (P_{12} \otimes P_{13}) \otimes P_{12} = P_{12} \otimes (P_{13} \otimes P_{12}) \\
&= |P_{12}|^2 P_{13} - P_{12}\,[P_{13} \cdot P_{12}] = AP_{13} - GP_{12}, \\
M_1^{-1} &= \tfrac{1}{H} \begin{pmatrix} CP_{12} - GP_{13} \\ AP_{13} - GP_{12} \\ n \end{pmatrix}^{\perp}.
\end{aligned}$$

Hence, the two solutions are

$$P_{normal} = M_1^{-1} V_1, \quad P_{symmetry} = M_1^{-1} V_2.$$

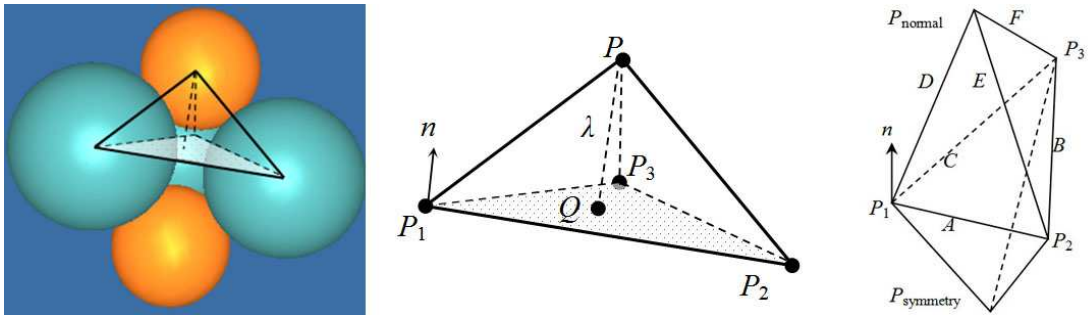
Therefore, the station can be determined by the following items:

$$P_{\text{normal}} = Q + \lambda n, \quad P_{\text{symmetry}} = Q - \lambda n,$$

where $Q = \frac{K}{H} n + \frac{S}{2H} P_{12} + \frac{T}{2H} P_{13}$ is the projection of P on the triangle, and $\lambda = \frac{6V}{H}$ is the distance from P to the triangle (Figure 2.7(b)).

Stations fall into two groups: negative stations ($\lambda < \alpha$) and positive stations ($\lambda \geq \alpha$). For negative stations, the two stations, $P = P_{normal}$ and $P = P_{symmetry}$, intersect.

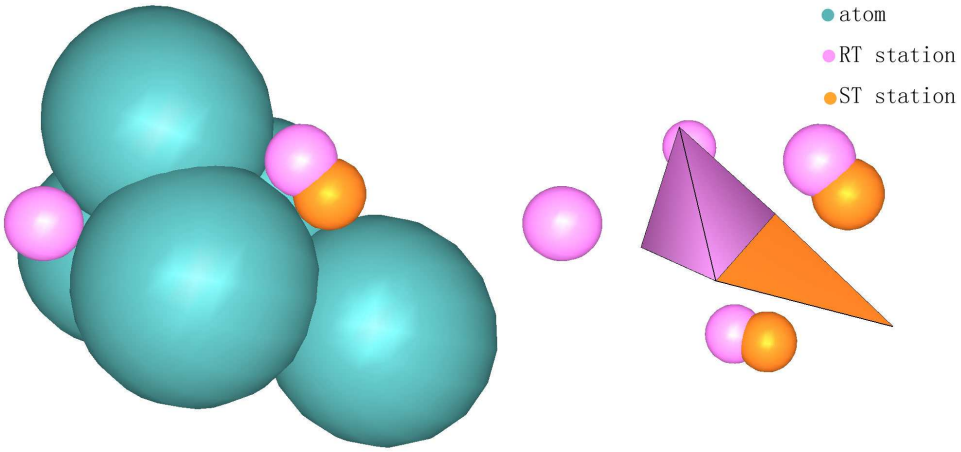
If T is a regular triangle, a station $P = P_{normal}$ is obtained. If T is a singular triangle, two stations, $P = P_{normal}$ and $P = P_{symmetry}$, are derived (Figure 2.8). From the solvent network, solvent stations can be created. When the probe moves from one station to another, an SES saddle can be created. We will find the SES arcs from all the SES saddles in the next section. The SAS arcs and vdWS arcs can be derived similarly.



(a) (b) (c)

Figure 2.7: Stations.

(a) one station is tangent to three atoms; (b) Q is the projection of P on the triangle and λ is the distance from P to the triangle; and (c) two stations, $P = P_{normal}$ and $P = P_{symmetry}$, can be derived from one triangle.



(a) (b)

Figure 2.8: Stations derived from triangles.

(a) one station is tangent to three atoms; and (b) one station $P = P_{normal}$ is derived from one regular triangle (pink) and two stations $P = P_{normal}$ and $P = P_{symmetry}$ are obtained from one singular triangle (yellow).

2.5.2 SES saddle patches

An SES saddle can be created by rotating the probe along one network edge from one solvent station to another. Along a singular edge, the probe will rotate for 360^o. A saddle patch can be created with two circles as the two boundaries.

From **Property 1**, a regular edge (P_s, P_e) can be shared by $2n$ regular triangles and m singular triangles. Take Φ as a plane perpendicular to the edge P_sP_e and project different elements onto this plane: the regular edge becomes a point O (the green dot in Figure 2.9), regular triangles are regular line segments (red line segments in Figure 2.9) and singular triangles are singular line segments (green line segments in Figure 2.9). There are $(2n + 2m)$ stations (black dots in Figure 2.9). All these

stations are tangent to the two atoms. Hence, they are located at a same circle (red dots in Figure 2.9). Consider the disk surrounded by the circle with center at O. All line segments cut the disk into $(2n + m)$ sectors. Some sectors are outside the solid $SO(S)$ while others are inside. Obviously, there are n sectors inside the solid if $n > 0$. The boundaries of an inside sector are two regular line segments. Hence, only $(n+m)$ sectors lie outside the network. Within an outside sector, there are two stations. A saddle patch can be formed rotating between the two stations. Therefore, $(n + m)$ saddle patches can be obtained (blue arrows in Figure 2.9). We denote each saddle patch as $[P_s, P_e, Q_1, Q_2, d_1, d_2]$, where P_sP_e is the edge, Q_1 is the start station, Q_2 is the end station, d_1 is the tangent direction at Q_1 and d_2 is the tangent direction at Q_2. The saddle patches are created clockwise if we view the plane Φ from the point P_s to P_e.

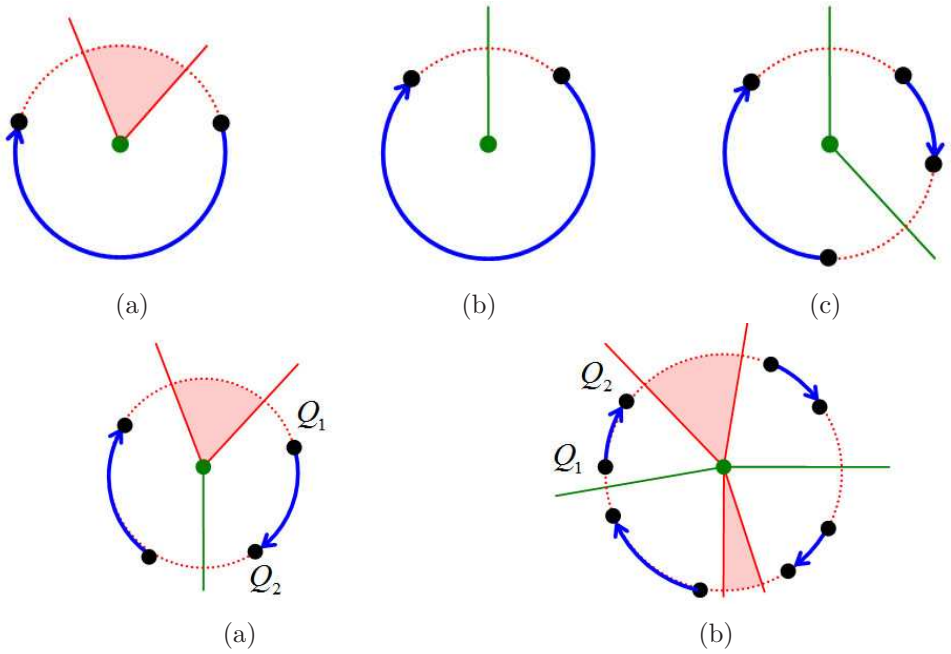


Figure 2.9: A regular edge and its neighboring triangles projected onto the plane Φ. pink area: the solid; black dots: stations; red line segments: regular triangles; green line segments: singular triangles; red dots: the circle; blue arrows: saddle patches; green dots: regular edges. (a) first example with $n = 1, m = 0$; (b) second example with $n = 0, m = 1$; (c) third example with $n = 0, m = 2$; (d) forth example with $n = 1, m = 1$; and (e) fifth example with $n = 2, m = 2$.

2.5.3 Boundary arcs

For the SES, different arcs will be created as boundaries of different patches. These arcs are named as SES arcs; they can be divided into probe-arcs and atom-arcs. An atom-arc is an arc on an atom sphere and a probe-arc is an arc on a station. For a regular solvent edge, we can obtain different SES saddle patches. From each saddle patch $[P_s, P_e, Q_1, Q_2, d_1, d_2]$, we have

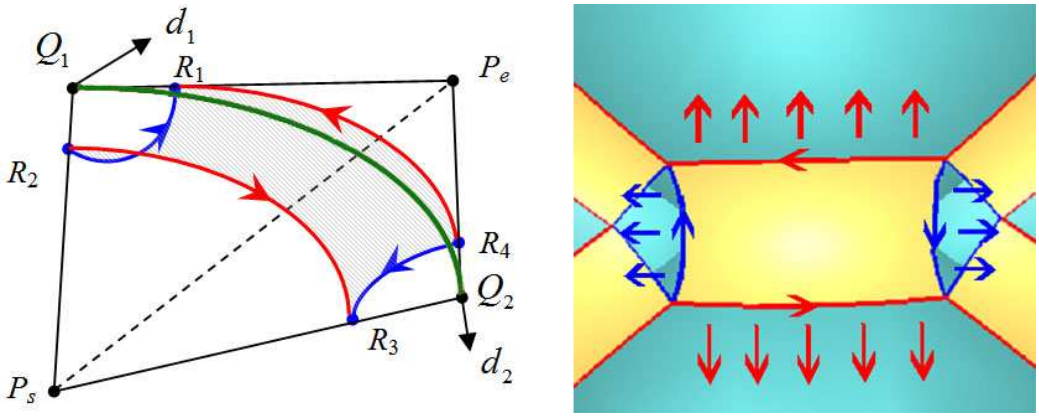
$$R_1 = Q_1 + \frac{Q_1P_e}{|Q_1P_e|}\alpha, \quad R_2 = Q_1 + \frac{Q_1P_s}{|Q_1P_s|}\alpha, \quad R_3 = Q_2 + \frac{Q_2P_s}{|Q_2P_s|}\alpha, \quad R_4 = Q_2 + \frac{Q_2P_e}{|Q_2P_e|}\alpha,$$
$$D_1 = R_2Q_1 \otimes d_1, \qquad D_2 = d_1, \qquad D_3 = d_2 \otimes R_4Q_2, \qquad D_4 = -d_2.$$

Hence, we have four boundary arcs, where $[R_2, R_1, D_1]$ and $[R_4, R_3, D_3]$ are two atom-arcs (red arcs in Figure 2.10(a)), $[R_2, R_3, D_2]$ and $[R_4, R_1, D_4]$ are two probe-arcs (blue arcs in Figure 2.10(a)). We assign each atom-arc to the corresponding solvent network vertex ($[R_2, R_3, D_2]$ for P_s and $[R_4, R_1, D_4]$ for P_e) and assign each probe arc to the corresponding station ($[R_2, R_1, D_1]$ for Q_1 and $[R_4, R_3, D_3]$ for Q_2). For convenience, the two atom-arcs are counter-clockwise and the two probe-arcs are clockwise when one faces the saddle patch outside the molecule (Figure 2.10(b) and Figure 2.11(b)). A convex patch is part of an atom sphere. When we face the convex spherical patch outside the atom, the patch lies on the right-hand side of the atom-arcs (red arrows in Figure 2.11(c)). A concave patch is part of a station, which is also a sphere. When we face the concave spherical patch outside the station, the patch lies on the right-hand side of the probe-arcs (blue arrows in Figure 2.11(c)). For a singular solvent edge, the probe will rotate along the edge for 360^o. Two atom-arcs (circles) are created. A probe-arc is the shared boundary between a saddle patch and a concave patch while an atom-arc is the shared boundary between a saddle patch and a convex patch (Figure 2.12(a)).

The SAS contains only convex patches, whose boundaries are arcs, named as SAS arcs (Figure 2.12(b)). Corresponding to each saddle patch $[P_s, P_e, Q_1, Q_2, d_1, d_2]$, we can create two SAS arcs as $[Q_1, Q_2, d_1]$ and $[Q_2, Q_1, -d_2]$, which are the loci of the probe center when the probe contacts the two atoms. The two arcs are superimposed but with different directions (green arcs in Figure 2.10(a)). Each arc is assigned to

the corresponding network vertex: $[Q_1, Q_2, d_1]$ for P_s and $[Q_2, Q_1, -d_2]$ for P_e.

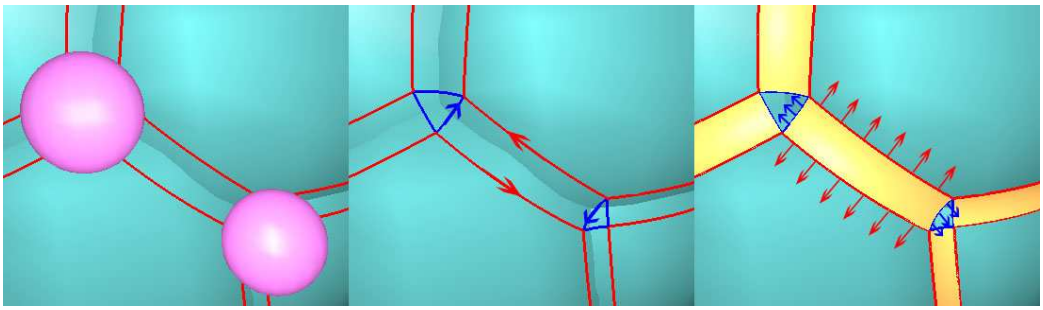
SES arcs and SAS arcs are created from solvent network edges. Similarly, vdWS arcs can be created from vdWS network edges. The radius of the probe becomes 0Å and a saddle patch is degenerated to be an arc while the two probe-arcs become points (Figure 2.12(c)). These vdWS arcs are assigned to its corresponding vdWS network vertex. A vdWS arc is the boundary between two convex spherical patches on the vdWS. It is the locus of the probe center when it contacts two atoms.



(a) (b)

Figure 2.10: Four arcs identified from a saddle.

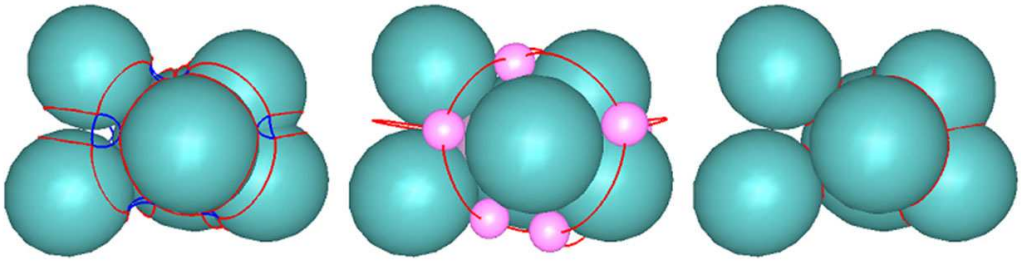
(a) two SES atom-arcs (red), two SES probe-arcs (blue), and two SAS arcs (green); and (b) if we face the convex patch outside the sphere (outside SES), the convex patch is on the right hand side of atom-arcs; if we face the concave patch outside the station (inside SES), the concave patch is on the right hand side of the probe-arcs.



(a) (b) (c)

Figure 2.11: A saddle patch created between two stations.

(a) a saddle patch is created by rotating the probe between two stations; (b) four SES arcs are created with atom-arcs (red) counter-clockwise and probe-arcs (blue) clockwise; and (c) the convex patch is on the right-hand side of the atom-arcs and the concave patch is on the right-hand side of the probe-arcs.



(a) (b) (c)

Figure 2.12: Three types of boundary arcs.
(a) SES arcs (probe-arcs in blue and atom-arcs in red); (b) SAS arcs (red); and (c) vdWS arcs (red).

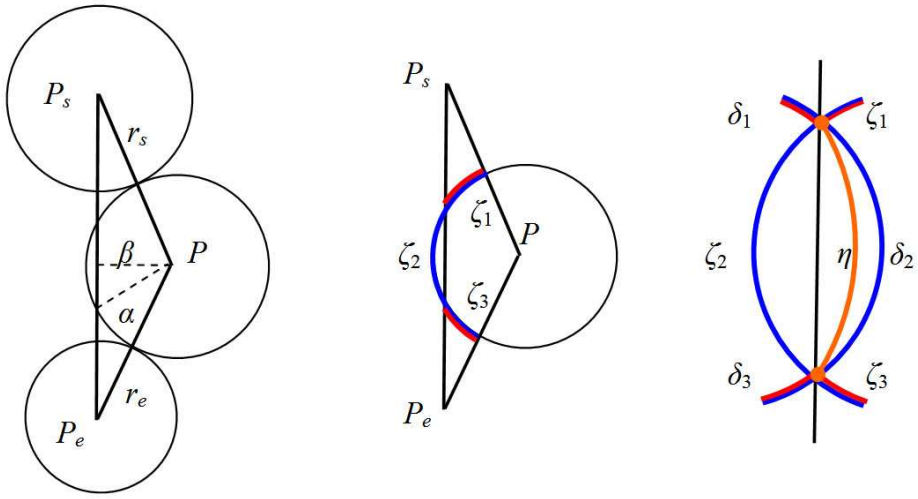
2.5.4 Singularity processing

Only SES requires singularity processing. This may be due to the self-intersections of saddle patches and concave patches (no self-intersection for convex patches). There exist two types of singularities: station-edge singularities and station-triangle singularities. Singularity processing will remove the arc portion causing self-intersections. New SES arcs will be added to eliminate the self-intersections. The re-entrance surfaces created before and after singularity processing will be illustrated below.

Suppose $[P, \alpha]$ is a station with radius α, same radius size with the probe. $[P, \alpha]$ is created from a triangle $\Delta P_1 P_2 P_3$. The distance from P to $\Delta P_1 P_2 P_3$ is λ (Figure 2.7(b)). If the station $[P, \alpha]$ is tangent to two atoms $[P_s, r_s]$ and $[P_e, r_e]$, we denote the distance between P and $P_s P_e$ as β (Figure 2.13(a)), then

Case 1: Station-edge singularities ($\beta < \alpha$). A probe-arc (the blue arc in Figure 2.13(b)) intersects with the edge. A saddle patch will be created by rotating the probe-arc. There will be singularities (Figure 2.14(c), Figure 2.15(b), and Figure 2.16(b)). Two concave patches sharing this probe-arc will intersect (Figure 2.16(c) and Figure 2.17(e,g)). The intersection is an arc.

Case 2: Station-triangle singularities ($\beta \geq \alpha, \lambda < \alpha$). No self-intersection for the saddle patch. However, two concave patches may intersect with each other (Figure 2.17).



(a) (b) (c)

Figure 2.13: A station intersects with an edge.
(a) the intersection exists only when $\beta < \alpha$; (b) the edge cut a probe-arc (blue) into three arcs; and (c) each probe-arc is replaced with three new probe-arcs.

2.5.4.1 Station-edge singularities removal

A saddle patch $[P_s, P_e, Q_1, Q_2, d_1, d_2]$ is bounded by two atom-arcs and two probe-arcs. Suppose ζ is the probe-arc on the station $[Q_1, \alpha]$ and δ is the probe-arc on the station $[Q_2, \alpha]$. Based on the definition, ζ is the intersection between the sphere $[Q_1, \alpha]$ and $\Delta Q_1 P_s P_e$, while δ is the intersection between the sphere $[Q_2, \alpha]$ and $\Delta Q_2 P_s P_e$. If $[Q_1, \alpha]$ and $[Q_2, \alpha]$ intersect with $P_s P_e$, then

(1) ζ and δ intersect with $P_s P_e$. $P_s P_e$ cuts ζ into three arcs ζ_1, ζ_2 and ζ_3 (Figure 2.13 (b)) and cuts δ into three arcs δ_1, δ_2 and δ_3 (Figure 2.13 (c)).

(2) $[Q_1, \alpha]$ and $[Q_2, \alpha]$ intersect at a circle which is divided by $P_s P_e$ into two arcs. Denote the one with smaller arc angle as η (the yellow arc in Figure 2.13 (c)).

(3) ζ is replaced with ζ_1, η and ζ_3 while δ is replaced with δ_1, η and δ_3.

2.5.4.2 Station-triangle singularities removal

If φ is a concave patch on a negative station $[P, \alpha]$ satisfying $\lambda < \alpha$ (Figure 2.7(b)), φ may intersect with other concave patches. For any station $[Q, \alpha]$, according to the definition, the portion of φ that inside $[Q, \alpha]$ should be removed.

Suppose a concave patch T is on one station S_1. Another station S_2 can intersect with S_1. Suppose C is the part of S_1 inside S_2. Then, the new concave patch should

be $T_n = T - C$. There are three possible relationships between T and C:

Case 1: T and C separate. No singularity is to be processed.

Case 2: C is inside T (Figure 2.17(a,b)). The circle will be added as a new boundary arc (Figure 2.17(c,d)).

Case 3: T and C intersect (Figure 2.17(e, g)). Part of T will be deleted by C. Several arcs on the circle are added as new boundary arcs. The new concave T_n can be a single patch (Figure 2.17(f)) or more than one patch (Figure 2.17(h)).

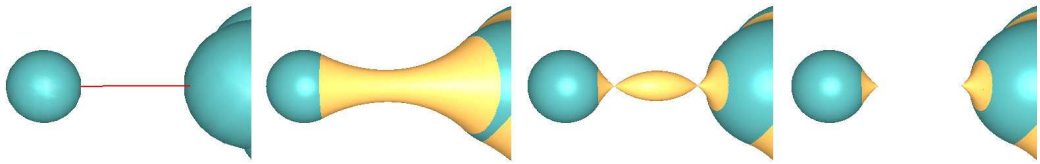
The station-triangle singularities removal only needs to be checked between any two negative stations, after the station-edge singularities removal.

2.5.4.3 On saddle patches

The saddle patch can contain self-intersections, for patches created from singular edges and regular edges. A self-intersecting saddle patch is created by rotating an arc ζ along an edge P_sP_e. Part of the saddle patch will be deleted, if we rotate only ζ_1 and ζ_3.

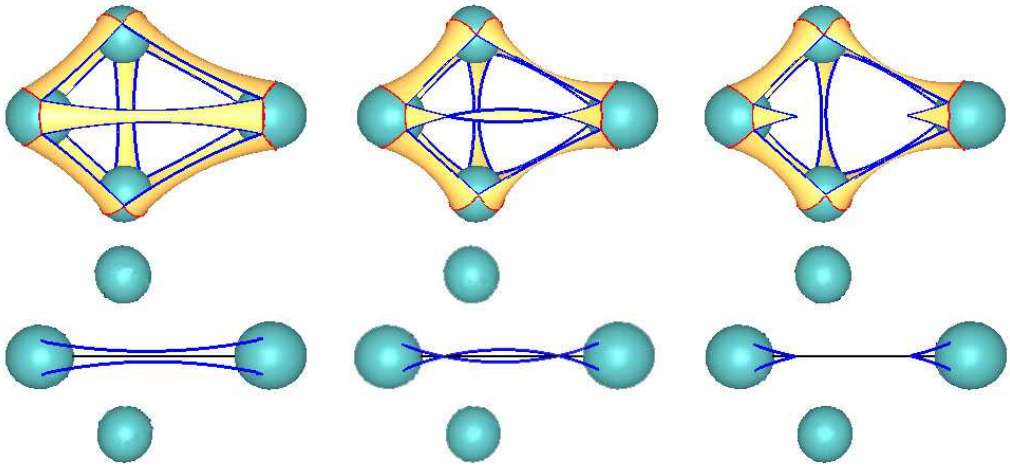
Along a singular edge, the arc ζ rotates for 360^o (Figure 2.14(b)). If ζ intersects with the edge, the saddle patch will self-intersect (Figure 2.14(c)). By rotating only ζ_1 and ζ_3, we delete the self-intersection to form the final surface. This processing will generate two cusped points on the surface (Figure 2.14(d)).

Along a regular edge, if ζ is not intersect with P_sP_e (equivalently $\beta \geq \alpha$), there is no self-intersection within this patch (Figure 2.15(a)). Otherwise, the saddle patch self-intersects (Figure 2.15(b) and Figure 2.16(a,b)). Rotating only ζ_1 and ζ_3 will remove the self-intersection (Figure 2.15(c) and Figure 2.16(d,e)).



(a) (b) (c) (d)

Figure 2.14: A station-edge singularity along a singular edge.
(a) along a singular edge (red); (b) a saddle is created without self-intersection; (c) self-intersection; and (d) the self-intersection will be deleted.



(a) (b) (c)

Figure 2.15: A station-edge singularity along a regular edge.
(a) if two probe-arcs are separated, there is no self-intersection; (b) if two probe-arcs intersect, the saddle patch is self-intersecting; and (c) the self-intersection will be deleted.

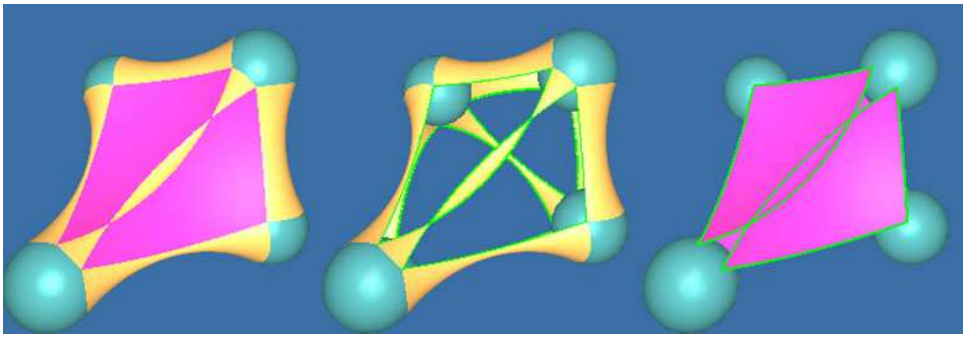
2.5.4.4 On concave patches

Two concave patches neighboring with a same saddle patch may intersect. In Figure 2.16(a), the two probe-arcs, ζ and δ of the saddle patch, intersect with each other. Suppose one concave patch A uses ζ as a boundary while the other concave patch B uses δ as a boundary. Consequently, the two concave patches intersect (Figure 2.16(c)). After the station-edge singularities removal, A uses ζ_1, η and ζ_3 as three boundaries, and B uses δ_1, η and δ_3 as three boundaries. There is no intersection between A and B (Figure 2.16(d,f)). A and B, sharing a same boundary η, become two neighbors.

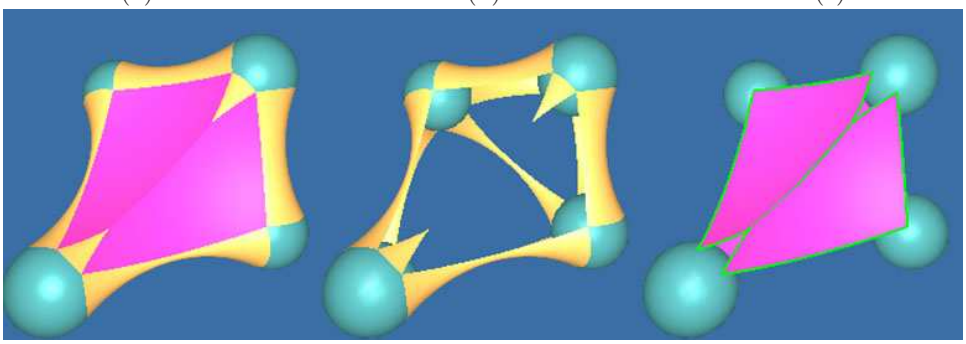
Two concave patches from a singular triangle can also intersect with each other (green concave patches in Figure 2.17(a-d)). The molecular surface in Figure 2.17(d) has genus one.

2.6 Surface modeling

The SES and the SAS will be created based on the solvent network and the vdWS will be created based on the vdWS network. For the SES, using the SES arcs as boundaries, convex spherical patches, saddle patches and concave spherical patches



(a) (b) (c)



(d) (e) (f)

Figure 2.16: Singularities along a regular edge.

(a) the molecular surface before singularity processing; (b) a saddle patch with self-intersection; (c) intersection between two concave patches; (d) the molecular surface before singularity processing; (e) saddle patches without self-intersections; and (f) two adjacent concave patches.

can be created according to the vertices, edges and triangles on the solvent network. For the SAS, a convex spherical patch can be created from a solvent network vertex using the SAS arcs as its boundaries. For the vdWS, a convex spherical patch can be created for a vdWS network vertex using the vdWS arcs as its boundaries. In this section, we will formulate all these patches in rational Bézier form. Generally, all these patches are saddle patches or spherical patches.

2.6.1 Saddle patches

Only the SES contains saddle patches. For a regular solvent network edge, each saddle patch is bounded by two atom-arcs and two probe-arcs. The patch will be created by rotating a probe-arc along the atom-arcs to the other probe-arc with respect to the edge. For a singular solvent network edge, the saddle patch is created by rotating an arc for 360^o.

We follow Wang [65] to construct an arc as a cubic rational Bézier curve with all

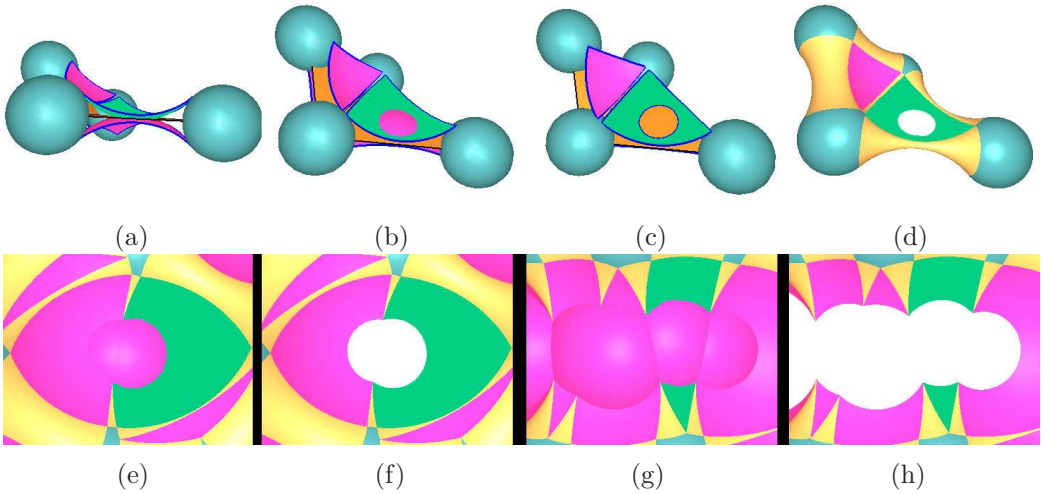


Figure 2.17: Station-triangle singularities.
(a) and (b) if two nearby concave patches intersect at a circle; (c) a new boundary circle will be added as boundary arc; (d) a hole will be created in the concave patch; (e) two nearby concave patches intersect at an arc; (f) the arc will be added as boundary arcs; (g) two nearby concave patches intersect at two arcs; and (h) the arc will be added as boundary arcs and the concave patch becomes two.

positive weights. If an arc starts from P_s and ends at P_e, spanning an angle of 2θ, it can be modeled as a cubic rational Bézier curve $c(t) = \sum_{i=0}^{3} B_i^3(t)c_i$, where $c_i = (p_i, w_i)$ are the control points with position p_i and weight w_i. The control points are on the same plane satisfying

$$\begin{aligned} &w_1 = w_2 = \tfrac{2\cos\theta+1}{3}, w_0 = w_3 = 1, \\ &p_0 = R_1, p_3 = R_2, |p_0p_1| = |p_2p_3| = \tfrac{|p_0p_3|}{2\cos\theta+1}. \end{aligned} \tag{Eq.2.4}$$

All the weights are positive if and only if the arc angle $2\theta < 4\pi/3$. Eq.20 in [65] provides a method to calculate the control points when rotating a cubic rational Bézier curve on ZY-plane along the Z-axis. Owing to the characteristic of affine invariance, we can develop a geometric algorithm to model a saddle patch $[P_s, P_e, Q_1, Q_2, d_1, d_2]$ using bi-cubic rational Bézier surfaces

$$C(u, v) = \sum_{i=0}^{3}\sum_{j=0}^{3} B_i^3(u)B_j^3(v)C_{ij},$$

with control points $C_{ij} = (P_{ij}, w_{ij})$.

Suppose the arc angle of the probe-arc $[R_3, R_4, D_3]$ is $2\theta_1$. We can formulate it as a cubic rational Bézier curve to obtain control points (Figure 2.18(a)) $C_{0j}, j = 0, 1, 2, 3$

with

$$w_{01} = w_{02} = \frac{2\cos\theta_1+1}{3},$$
$$w_{00} = w_{03} = 1.$$

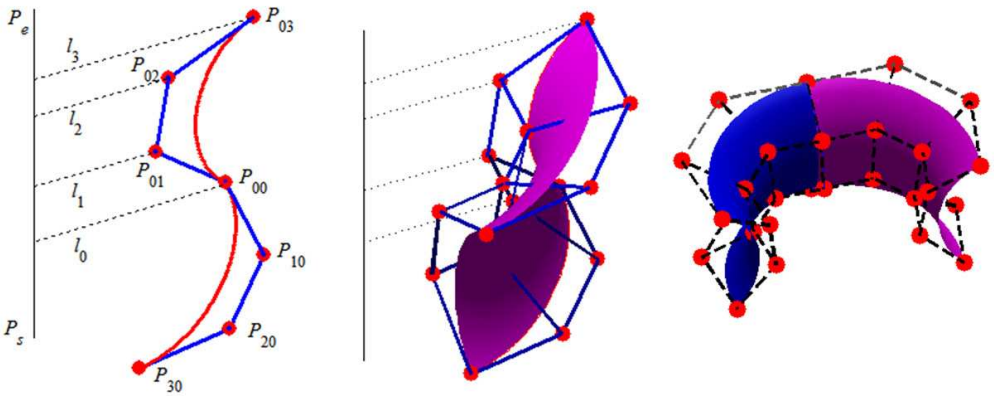
Similarly, if the sphere-arc $[R_2, R_3, D_2]$ spans an angle of $2\theta_2$, we can obtain four control points (Figure 2.18(a)) $C_{i0}, i = 0, 1, 2, 3$ with

$$w_{10} = w_{20} = \frac{2\cos\theta_2+1}{3},$$
$$w_{00} = w_{30} = 1.$$

Suppose the distance of the point P_{0j} to the edge P_eP_s is l_j; then the remaining control points can be derived from

$$P_{i+1,j} = P_{i,j} + \frac{l_j}{l_0}\left(P_{i+1,0} - P_{i,0}\right),$$
$$w_{ij} = w_{i0}w_{0j}.$$

In Eq.2.4, w_1 and w_2 ($w_1 = w_2$) will be positive if the arc angle is smaller than

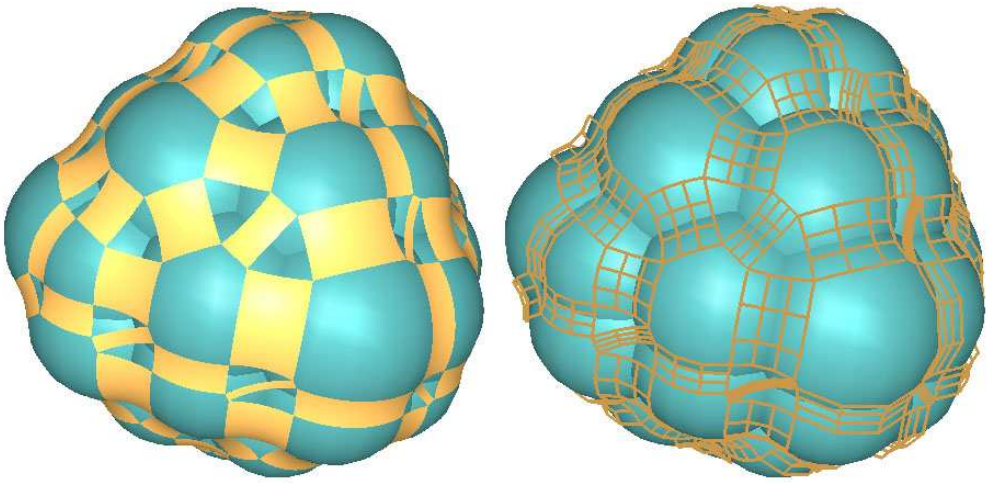


(a) (b) (c)

Figure 2.18: A saddle patch in rational Bézier form.
(a) boundary arcs; (b) the saddle patch has one sub-patch; and (c) the saddle patch has two sub-patches.

$4\pi/3$. For any probe-arc, $2\theta_1 < \pi$, we have $w_{01} = w_{02} > 0$. However, for an atom-arc, there are two possibilities: $2\theta_2 < 4\pi/3$ and $2\theta_2 \geq 4\pi/3$. If $2\theta_2 < 4\pi/3$, which leads to $w_{10} = w_{20} > 0$, all weights w_{ij} will be positive. Figure 2.18(b) shows an example of the saddle patch and its control points. Otherwise, If $2\theta_2 \geq 4\pi/3$, we equally divide the atom-arc into two and use two rational Bézier surfaces to describe the saddle patch. Figure 2.18(c) shows a saddle patch represented by two rational

Bézier surfaces. Figure 2.19 shows an example of the saddle patches and their control meshes.



(a) (b)

Figure 2.19: Saddle patches as rational Bézier surfaces. (a) patches; and (b) control meshes.

2.6.2 Spherical patches

For the SAS and the vdWS, all the patches are spherical. For the SES, the concave patches and the convex patches are spherical. In boundary modeling, boundary arcs are created on the spheres (atoms and stations). On a sphere, the spherical patch, which lies on the right-hand side of its boundary arcs, is the part of the sphere contributing to the molecular surfaces. By arranging the boundary arcs end to end, we obtain several boundary curves of piecewise arcs. There are two types of spherical patches: single boundary patches and multi-boundary patches. A spherical patch bounded by one closed curve is a single boundary patch. A multi-boundary spherical patch is a spherical patch bounded by several closed curves.

2.6.2.1 Single boundary patches

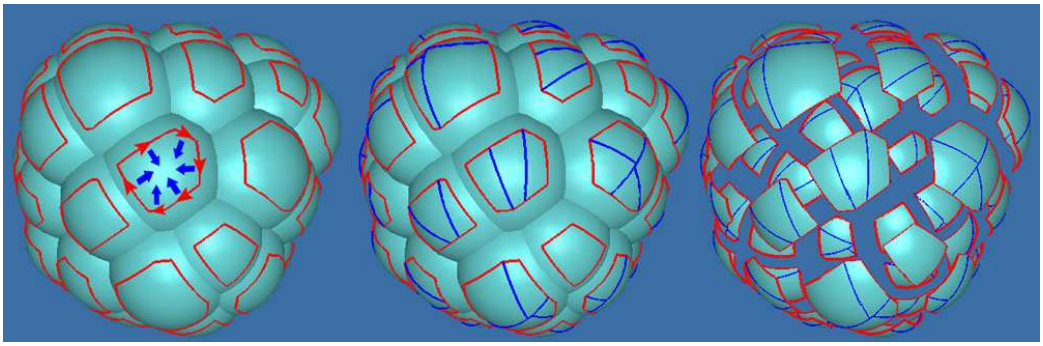
Suppose Ω is a sphere. A closed curve $\Upsilon(N)$ on Ω is formed by N direct arcs (the red arrow in Figure 2.20(a)). A single boundary spherical patch $\bar{\Upsilon}(N)$ is a patch using $\Upsilon(N)$ as its boundary. If we face each arc from outside the sphere, $\bar{\Upsilon}(N)$ is the portion of Ω that lies on the right hand side of the arcs (the blue arrow in Figure 2.20(a)).

$\bar{\Upsilon}(N)$ is an N-sided spherical patch and each side is an arc. As a special case, a sub-patch is defined as a rectangular spherical patch, equivalently $\bar{\Upsilon}(4)$.

We need to subdivide $\bar{\Upsilon}(N)$ into several sub-patches before using rational Bézier representation. If $N < 4$, we can treat $\bar{\Upsilon}(N)$ as a degenerated sub-patch. Otherwise, if $N > 4$, subdivision is needed. A trapezoidal decomposition [66, 67] has been developed for subdivision. Unfortunately, this may result in too many small sub-patches. In our system, we create an arc starting from a corner point of the N-sided patch $\bar{\Upsilon}(N)$. The arc should cut $\bar{\Upsilon}(N)$ into two N-sided patches, $\bar{\Upsilon}_1(N_1)$ and $\bar{\Upsilon}_2(N_2)$. Rotate the cutting arc until $N_1 < N, N_2 < N$ (blue arcs in Figure 2.20(b). In this way, the N-sided spherical patch is divided into sub-patches:

$$\bar{\Upsilon}(N) = \bigcup_i \bar{\Upsilon}_i(N_i), N_i \leq 4.$$

Finally, we describe each rectangular sub-patch as a rational Bézier surface following Dietz *et al.* [12, 13]. A rational Bézier surface of degree $(2, 4)$ can be used to represent a spherical sub-patch bounded by four arcs (for details see Section 2.8). However, since a rational Bézier curve of degree 2 cannot represent an arc of angle π, a sub-patch will be further subdivided if it has two adjacent arcs of angle π.



(a) (b) (c)

Figure 2.20: Dividing each N-sided spherical patch into sub-patches

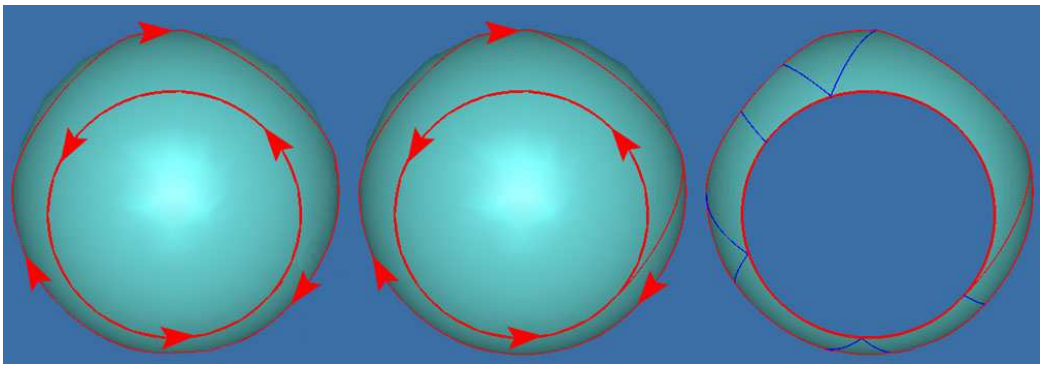
(a) 7 arcs form a closed curve $\Upsilon(7)$; (b) cutting arcs (blue) cut the patch into sub-patches; and (c) each sub-patch is modeled as 2×4 rational Bézier surface.

2.6.2.2 Multi-boundary patches

A multi-boundary spherical patch can be viewed as the portion of the sphere lies at the right hand side of all its boundary curves (Figure 2.21(a)). We can convert the

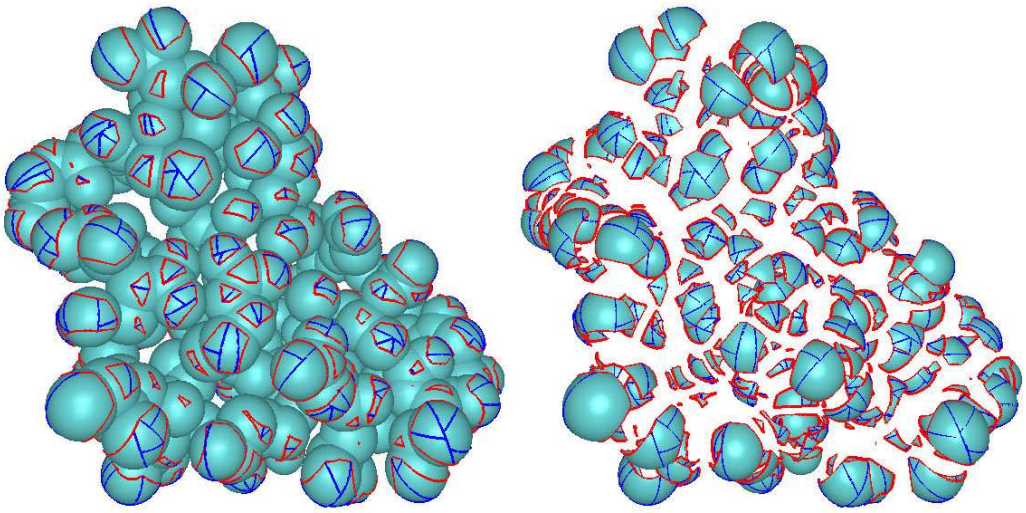
patch into single boundary patch by to connecting two boundary curves using a new arc (Figure 2.21(b)). Then, a multi-boundary patch can be modeled as several 2×4 rational Bézier surfaces (Figure 2.21(c)).

Figure 2.22 is an example for molecule 1CRN [68]. Each SES convex patch is divided into several sub-patches.



(a) (b) (c)

Figure 2.21: Dividing a multi-boundary spherical patch into sub-patches.
(a) the spherical patch is bounded by two boundary curves; (b) the patch can be converted into single boundary patch; and (c) the patch can be represented as several 2×4 rational Bézier surfaces.

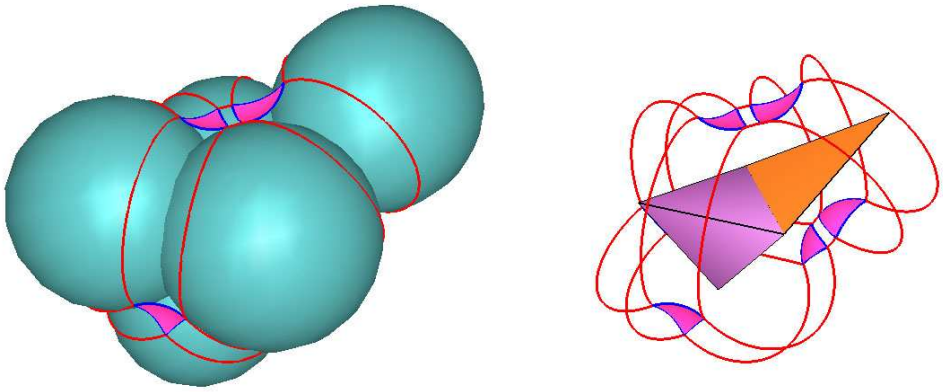


(a) (b)

Figure 2.22: Subdividing the SES convex patches of the molecule 1CRN.
(a) the atom-arcs (red) and the cutting arcs (blue) are on atoms; and (b) each rectangular spherical sub-patch can be formulated as a rational Bézier patch.

2.6.3 Concave patches

For a solvent network station, a set of SES arcs are obtained. Figure 2.23 gives the examples for stations from singular triangles and regular triangles.



(a) (b)

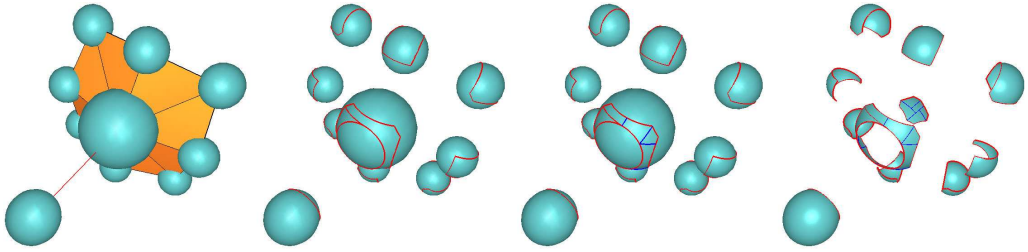
Figure 2.23: Concave patches are formulated in rational Bézier form.
(a) a concave patch is on a station; and (b) a concave patch can be created from one regular triangle (pink) and two concave patches can be created from one singular triangle (yellow).

2.6.4 Convex patches

For a solvent network vertex, several SES arcs and several SAS arcs can be obtained. For a vdWS network vertex, several vdWS arcs are created. The SES arcs form the boundaries of a SES convex patch, the SAS arcs form the boundaries of a SAS convex patch, and the vdWS arcs form the boundaries of a vdWS convex patch.

The arcs can be connected to form several boundary curves. For the central atom in Figure 2.24, there are three boundary curves and two spherical patches associated with the atom, and one of the curves is a circle. One of the patches is N-sided ($\bar{\Upsilon}_1$, the one behind), and the other one contains two boundary curves (the one in front). The latter will first be transferred into a new N-sided spherical patch ($\bar{\Upsilon}_2$) (Figure 2.24(b,d)). Each N-sided spherical patch can be further divided into sub-patches (Figure 2.24(c,d)). All of them can finally be represented by using a rational Bézier surfaces.

If one uses trimmed NURBS method [55, 56] to model the example in Figure 2.24, a single trimmed NURBS is provided to describe the two patches $\bar{\Upsilon}_1$ and $\bar{\Upsilon}_2$. In our method, $\bar{\Upsilon}_1$ and $\bar{\Upsilon}_2$ are modeled separately. It is easy to find out the neighboring saddle patches for $\bar{\Upsilon}_1$ and $\bar{\Upsilon}_2$.



(a) (b) (c) (d)

Figure 2.24: Two separate patches created on the central atom.
(a) one singular edge and several triangles connect to the atom; (b) one spherical patch contains two boundary curves; (c) N-sided spherical patches are divided into several rectangular sub-patches; and (d) modeling the sub-patches using rational Bézier surfaces.

2.7 The design of a modeling kernel for molecular surfaces

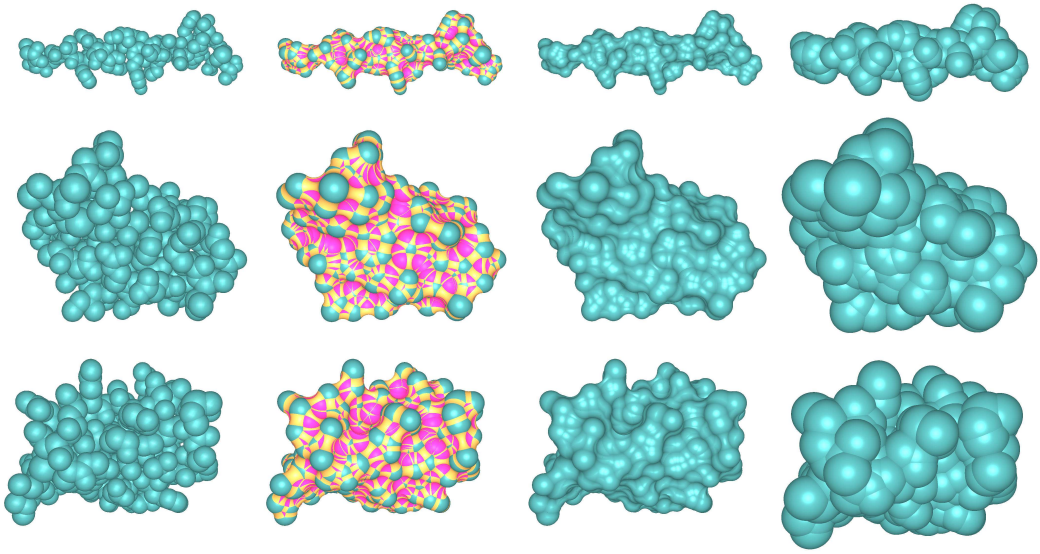
2.7.1 Kernel structure design

A rational Bézier patch is the basic element of the molecular kernel modeler. A standard data structure can be designed for rational Bézier patches in either degree 2×4 or degree 3×3.

In contrast to triangle meshes as the basic element in a typical tessellation of molecular surfaces, the rational Bézier patch takes advantage of accurate representation of the molecular surfaces. No approximation is involved. The basic element design of the rational Bézier patch will use less computational memory for molecular surfaces representation compared to NURBS-based representation.

The basic element design of the kernel modeler makes it possible to have a uniform solution to modeling all three molecular surfaces: the vdWS, the SAS and the SES. Each type of molecular surfaces will have a list of molecular patches that can be represented by the basic element in the rational Bézier form.

Figure 2.25 shows the result of the uniform solution. Each molecular patch is divided into sub-patches in rational Bézier form: degree 2×4 for spherical sub-patches and degree 3×3 for saddle sub-patches. Figure 2.26 shows the molecular surfaces with the rational Bézier surfaces rendered in different colors. The rational Bézier surface is rendered using OpenGL Shading Language (GLSL) [69].



(a) (b) (c) (d)

Figure 2.25: Molecular surfaces.

1BH0 with 242 atoms (the first row), 1CRN with 327 atoms (the second row), and 1PTQ with 402 atoms (the third row). (a) vdWS; (b) three types of patches for the SES; (c) the SES; and (d) the SAS.

2.7.2 The vdWS

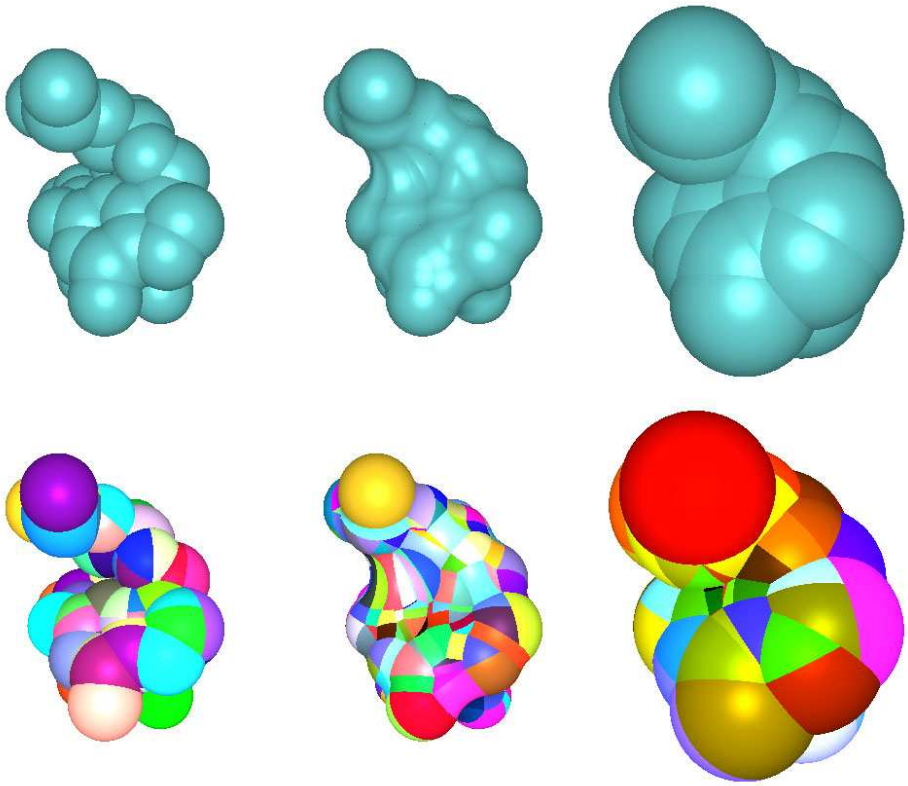
Figure 2.27 is the flowchart to model a vdWS. From the input PDB [68] file, the vdWS network is built. The vdWS arcs are then created. The vdWS is modeled as a set of convex spherical patches, each of which is modeled as a set of rational Bézier surfaces. Figure 2.25(a) and Figure 2.26(b) show examples of vdWS.

2.7.3 The SAS

Figure 2.28 is the flowchart to model an SAS. The input for SAS modeling is a PDB file and the probe radius. Typically, a probe radius of 1.50Åis used. The solvent network is then obtained from the weighted α-shape. After that, the SAS arcs are created. Similar to the vdWS, the SAS consists of convex spherical patches, which are represented in rational Bézier form. Figure 2.25(d) and Figure 2.26(c) show examples of SAS.

2.7.4 The SES

Figure 2.29 is the flowchart to model an SES. From the solvent network, the solution firstly derives the SES arcs: probe-arcs and atom-arcs. Singularities are then



(a) (b) (c)

Figure 2.26: Molecular surfaces for protein 279D.
the first row shows the molecular surfaces and the second row shows the surfaces in Bézier form. (a) the vdWS; (b) the SES; and (c) the SAS.

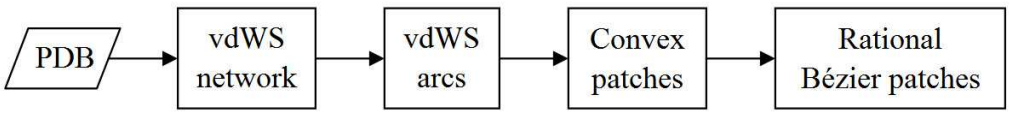


Figure 2.27: The modeling of a vdWS.

processed to eliminate the self-intersections by removing some portions of probe-arcs and adding new probe-arcs. After that, we can derive three types of patches: convex spherical patches, concave spherical patches and saddle patches. A concave spherical patch is modeled from probe-arcs while a convex spherical patch is modeled from atom-arcs. A saddle patch is obtained from two probe-arcs and two atom-arcs. Finally, each patch can be model using rational Bézier surfaces. Table 2.1 shows the results from the rational Bézier modeling. The first column shows testing molecules

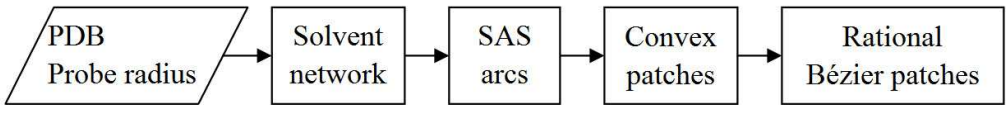


Figure 2.28: The modeling of an SAS.

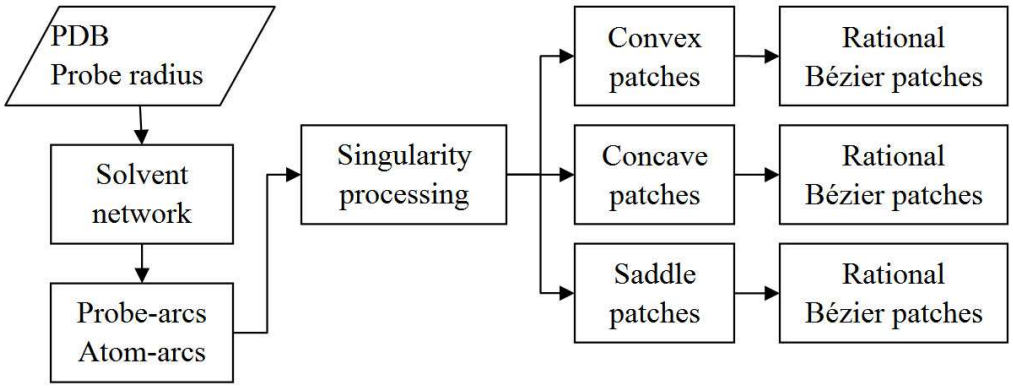


Figure 2.29: The modeling of an SES.

with PDB IDs. The second column is the number of atoms. The third column indicates the type of molecular surfaces. The fourth-sixth columns show the network information. The seventh-ninth columns show the number of different patches. The final column shows the total number of rational Bézier surfaces.

Table 2.1: The number of the elements for selected PDB samples.

PDB	Atom	Surface	Vertex	Edge		Triangle		Saddle	Convex	Concave	Bézier
				Singular	Regular	Singular	Regular				
		vdWS	26	1	64	19	26	0	32	0	116
2AWU	26	SES	26	0	76	2	50	81	29	54	235
		SAS	26	0	76	2	50	0	29	0	80
		vdWS	92	8	255	57	128	0	111	0	479
1AL1	92	SES	82	0	247	7	160	261	89	174	767
		SAS	82	0	247	7	160	0	89	0	276
		vdWS	127	6	367	72	204	0	159	0	707
3BKY	127	SES	109	0	330	11	212	351	119	234	1002
		SAS	109	0	330	11	212	0	119	0	361
		vdWS	202	23	498	155	188	0	254	0	1044
243D	202	SES	170	0	529	28	336	588	197	392	1607
		SAS	170	0	529	28	336	0	197	0	609
		vdWS	240	32	636	209	222	0	308	0	1354
180D	240	SES	180	2	545	10	356	572	189	378	1614
		SAS	180	2	545	10	356	0	189	0	604
		vdWS	1071	121	2991	656	1538	0	1330	0	5449
1PKP	1071	SES	623	1	1901	26	1254	1963	654	1306	5558
		SAS	623	1	1901	26	1254	0	654	0	2028

2.8 Bézier representations for sub-patches

An arc $[q_s, q_e, d]$ can be reformulated as (q_s, q_c, q_e), where q_c is the arc center. A sub-patch is bounded by four arcs and lies at the right hand sides of the arcs if we face the patch from outside the sphere. Denote the four arcs as (Figure 2.30)

$$a_1 = (q_1, c_1, q_2)\,, \quad a_2 = (q_2, c_2, q_3)\,, \quad a_3 = (q_3, c_3, q_4)\,, \quad a_4 = (q_4, c_4, q_1)\,. \qquad \text{(Eq.2.5)}$$

The four arcs cuts a sphere into two patches. So there are two spherical patches using the four arcs as boundaries. In [12, 13], a circle C_i is calculated using a_i as its segment. From four circles C_i and four corner points q_i, a spherical sub-patch is derived as a rational Bézier surface of degree (2,4) using *generalized stereographic projection*. The algorithm needs to calculate the other intersection points between circles except q_i, for example the two blue points in Figure 2.30. The arc a_i is formulated in Bézier form taking into account that it does not pass the intersection points. Such conditions uniquely define a spherical patch (the shade area in Figure 2.30). However, the algorithm in [12, 13] needs to calculate the intersection points between circles. As stated in Section 2.3.2, during boundary modeling, we avoid the calculations for intersection points. And even worse, if two arcs lie on a same plane, no intersection points can be derived.

Set $q_5 = q_1$. The plane passing a_i has the normal vector $n_i = q_iq_{i+1} \times q_ic_i$. According to the definition of the convex patches and concave patch, the point q on the sphere is a point of the sub-patch if and only if $q_i \cdot n_i \geq 0$ [55, 56]. Under such a definition, the patch using four arcs as boundaries is unique. For convenience, we apply a rotation such that the point $(0, 0, 1)$ is outside the patch.

We want to formulate the arc a_i in Bézier form while it passes the center points c_i. Therefore, the formulae in [12, 13] can not be directly used. In this section, new formulae to calculate the control points for each sub-patch will be provided.

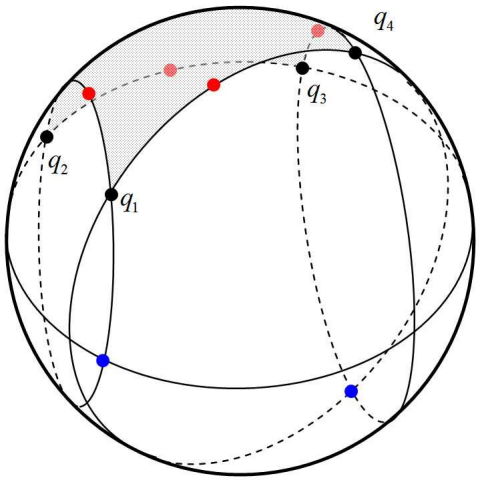


Figure 2.30: A sub-patch on the unit sphere.
black dots : corners of the patch; red dots : centers of boundary arcs; blue dots : intersect points between circles.

2.8.1 Generalized stereographic projection

Suppose R^3 is the Cartesian space. The projective space $\bar{E}^3$ and its subsets P and Q are

$$\begin{aligned}
\bar{E}^3 &= \{p|p = (w, x, y, z)\,, \quad w, x, y, z \in R\}\,, \\
P &= \left\{p|p \in \bar{E}^3, \quad \mathrm{z} = 0\right\}, \\
Q &= \left\{p|p \in \bar{E}^3, \quad x^2 + y^2 + z^2 = w^2\right\}.
\end{aligned}$$

An element in $\bar{E}^3$ with homogeneous coordinates $p = (w, x, y, z)$ is corresponding to a point in R^3. The plane P corresponds to the plane in $z = 0$ in R^3. The sphere Q corresponds to the unit sphere in R^3.

The *hyperbolic projection* $\vartheta : \bar{E}^3 \to P$ is defined as

$$\vartheta : (w, x, y, z) \to \left(w^2 + z^2, wx - yz, wy + xz, 0\right).$$

The *stereographic projection* $\sigma : P \to Q$ is defined as

$$\sigma : (w, x, y, 0) \to \left(w^2 + x^2 + y^2, 2wx, 2wy, x^2 + y^2 - w^2\right).$$

The *generalized stereographic projection* (GSP) $\delta : \bar{E}^3 \to Q$ is

$$\delta = \sigma\vartheta : (w, x, y, z) \to \left(w^2 + x^2 + y^2 + z^2, 2wx - 2yz, 2wy + 2xz, x^2 + y^2 - z^2 - w^2\right).$$

For any point $p = (w, x, y, z)$, we define

$$\begin{aligned}
\bar{p} &= (y, z, -w, -x)\,, \\
p^{\perp} &= (-z, y, -x, w)\,, \\
\bar{p}^{\perp} &= (\bar{p})^{\perp} = (x, -w, -z, y)\,, \\
p^{*} &= \begin{cases} (w - z, x, y, 0), & p \neq (1, 0, 0, 1), \\ (0, 1, 0, 0), & p = (1, 0, 0, 1), \end{cases} \\
p^{*\perp} &= \begin{cases} (x, z - w, 0, y), & p \neq (1, 0, 0, 1), \\ (0, 0, -1, 0), & p = (1, 0, 0, 1). \end{cases}
\end{aligned}$$

Then

$$\bar{E}^3 = span\{p, \bar{p}, p^{\perp}, \bar{p}^{\perp}\},$$
$$\delta(\lambda p + \mu p^{\perp}) = (\lambda^2 + \mu^2)\delta(p),$$
$$\delta(p^*) = \begin{cases} 2(w-z)p, & p \neq (1,0,0,1), \\ p, & p = (1,0,0,1), \end{cases} \quad p \in Q.$$

A line with the homogeneous coordinate vector p is

$$L(p) = \left\{\lambda p + \mu p^{\perp} | \lambda, \mu \in R\right\}, p \in E.$$

For any point $p \in Q$, all its pre-image is $\delta^{-1}(p) = L(p^*)$, which is named as a *projecting line*.

A plane with the homogeneous vector $v \in \bar{E}^3$ is

$$E(v) = \{p | p \cdot v = 0\} = span\{\bar{v}, v^{\perp}, \bar{v}^{\perp}\}.$$

Notice that $\overline{(\bar{p})} = -p$, we have $p^* = -\overline{\left(\overline{(p^*)}\right)}$. Given a point $p \in Q$, $E(\overline{(p^*)}) = span\{-p^*, \overline{(p^*)}^{\perp}, -p^{*\perp}\}$. Hence, the plane $E(\overline{(p^*)})$ contains the *projecting line* $L(p^*)$. From [12, 13], we have:

Lemma 2.1 For any point $p = (w, x, y, 0) \in P$, $\vartheta^{-1}(p)$ forms a *projecting line*: $\lambda p + \mu p^{\perp}$.

Lemma 2.2 The image of given *non-projecting line* L_1 under δ is a circle $\delta(L_1)$ on the unit sphere Q.

Lemma 2.3 The pre-image of a circle on Q not passing through (1,0,0,1) under δ is a doubly ruled quadric surface.

Lemma 2.4 The plane $E \subset \bar{E}^3$ contains exactly one *projecting line* L_E. Suppose L_1 and L_2 are two distinct *non-projecting lines* on E. The images $\delta(E)$ contain $\delta(L_1 \cap L_2)$ and $\delta(L_E)$. If L_1 and L_2 intersect on L_E, the tangents of $\delta(L_1)$ and $\delta(L_2)$ at $\delta(L_E)$ coincide.

Lemma 2.5 If a line $L(p)$ is not on a plane $E(v)$. The intersection point of $L(p)$ and $E(v)$ is

$$s = (v \cdot p)\, p^{\perp} - \left(v \cdot p^{\perp}\right) p. \tag{Eq.2.6}$$

Lemma 2.6 The plane $E \subset \bar{E}^3$ contains exactly one *projecting line* L_E and $q = \delta(L_E)$ is a point on Q. For a line L_1 in $E(v)$, the image $\delta\left(L_1\right)$ is a circle passing the fix point q.

Lemma 2.7 If a plane $E(v)$ contains a point p and the line $L(q)$, which does not contain p. Then

$$v = (p \cdot \bar{q})\bar{q}^{\perp} - (p \cdot \bar{q}^{\perp})\bar{q}.$$

2.8.2 An arc as a Bézier curve

An arc on Q can be formulated as a Bézier curve

$$q(t) = (y_0(t), y_1(t), y_2(t), y_3(t)) = \sum_{k=0}^{2n} r_k^{2n} B_k^{2n}(t), r_i \in Q. \tag{Eq.2.7}$$

Correspondingly, there is a Bézier curve in $\bar{E}^3$

$$\begin{aligned} &p(t) = (x_0(t), x_1(t), x_2(t), x_3(t)) = \sum_{i=0}^{n} p_i B_i^n(t), \\ &p_i = (p_{i,0}, p_{i,1}, p_{i,2}, p_{i,3}) \in \bar{E}^3, \end{aligned} \tag{Eq.2.8}$$

such that $q(t) = \delta\left(p(t)\right)$. The product of two Bézier $x_s(t) \cdot x_k(t)$ with $s, k = 0, 1, 2, 3$ can be formulated as

$$x_s \cdot x_k = \left(\sum_{i=0}^{n} p_{i,s} B_i^n(t)\right)\left(\sum_{j=0}^{n} p_{j,k} B_j^n(t)\right) = \sum_{k=0}^{2n} \left[\frac{\sum\limits_{i+j=k} p_{i,s} p_{j,t} \begin{pmatrix} n \\ i \end{pmatrix} \begin{pmatrix} n \\ j \end{pmatrix}}{\begin{pmatrix} 2n \\ k \end{pmatrix}} \right] B_k^{2n}(t).$$

Taking

$$r_{ij} = \begin{pmatrix} p_{i,0}p_{j,0} + p_{i,1}p_{j,1} + p_{i,2}p_{j,2} + p_{i,3}p_{j,3} \\ 2\left(p_{i,0}p_{j,1} - p_{i,2}p_{j,3}\right) \\ 2\left(p_{i,0}p_{j,2} + p_{i,1}p_{j,3}\right) \\ p_{i,1}p_{j,1} + p_{i,2}p_{j,2} - p_{i,3}p_{j,3} - p_{i,0}p_{j,0} \end{pmatrix}, \tag{Eq.2.9}$$

we can formulate Eq.2.7 into

$$\begin{aligned} q(t) &= \delta\left(p(t)\right) \\ &= (x_0^2 + x_1^2 + x_2^2 + x_3^2, 2x_0x_1 - 2x_2x_3, 2x_0x_2 + 2x_1x_3, x_1^2 + x_2^2 - x_3^2 - x_0^2) \\ &= \sum_{k=0}^{2n} B_k^{2n}(t) \frac{1}{\begin{pmatrix} 2n \\ k \end{pmatrix}} \sum_{i+j=k} \begin{pmatrix} n \\ i \end{pmatrix} \begin{pmatrix} n \\ j \end{pmatrix} r_{ij}. \end{aligned}$$

Therefore

$$r_k^{2n} = \frac{1}{\begin{pmatrix} 2n \\ k \end{pmatrix}} \sum_{i+j=k} \begin{pmatrix} n \\ i \end{pmatrix} \begin{pmatrix} n \\ j \end{pmatrix} r_{ij}. \tag{Eq.2.10}$$

2.8.2.1 An arc as a quadratic Bézier curve

A circular arc (q_0, q_1, q_2) can be formulated as a quadratic Bézier curve. Suppose $v = \overline{(q_1^*)}$, $r_0 \in L(q_0^*)$. From Lemma 2.7, the plane $E(u_1)$ passing $L(q_1^*)$ and r_0 is

$$u_1 = (r_0 \cdot v)v^{\perp} - (r_0 \cdot v^{\perp})v. \tag{Eq.2.11}$$

From Lemma 2.5, the plane $E(u_1)$ intersects $L(q_2^*)$ at

$$r_2 = (u_1 \cdot q_2^*)\, q_2^{*\perp} - \left(u_1 \cdot q_2^{*\perp}\right) q_2^*. \tag{Eq.2.12}$$

Find a plane $E(u_2)$ passing $L(q_0^*)$ and r_2 as

$$u_2 = (r_2 \cdot \overline{q_0^*})\overline{q_0^*}^{\perp} - (r_2 \cdot \overline{q_0^*}^{\perp})\overline{q_0^*}.$$

The plane $E(u_2)$ will intersect with $L(q_1^*)$ at

$$r_1 = (u_2 \cdot q_1^*)\, q_1^{*\perp} - \left(u_2 \cdot q_1^{*\perp}\right) q_1^*.$$

So both $E(u_1)$ and $E(u_2)$ contain p_0, r_1 and r_2. Therefore, p_0, r_1 and r_2 are collinear:

$$\frac{\frac{r_{11}}{r_{10}} - \frac{r_{01}}{r_{00}}}{\frac{r_{11}}{r_{10}} - \frac{r_{21}}{r_{20}}} = \frac{\frac{r_{12}}{r_{10}} - \frac{r_{02}}{r_{00}}}{\frac{r_{12}}{r_{10}} - \frac{r_{22}}{r_{20}}} = \frac{\frac{r_{13}}{r_{10}} - \frac{r_{03}}{r_{00}}}{\frac{r_{13}}{r_{10}} - \frac{r_{23}}{r_{20}}} = \lambda_0,$$

where $r_i = (r_{i0}, r_{i1}, r_{i2}, r_{i3}), i = 0, 1, 2$. Then

$$(1 - \lambda_0)\frac{r_{1i}}{r_{10}} = \frac{r_{0i}}{r_{00}} - \lambda_0 \frac{r_{2i}}{r_{20}}, \quad i = 0, 1, 2, 3.$$

The above four equations equal to

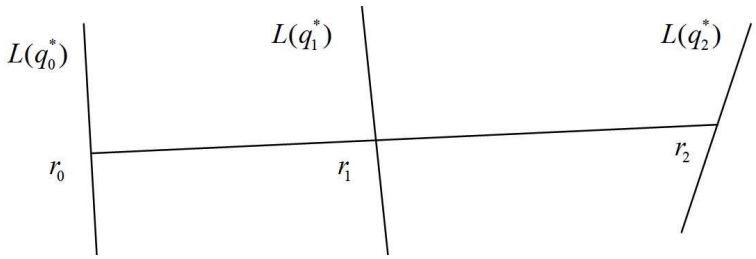


Figure 2.31: A line passing three projecting lines.

$$(1 - \lambda_0)\frac{r_{00}}{r_{10}} r_1 = r_0 - \frac{\lambda_0 r_{00}}{r_{20}} r_2.$$

Select $\eta \in (0, 1)$ and set

$$p_0 = r_0, \qquad p_1 = \frac{(1 - \eta)\lambda_0 p_{00}}{-\eta r_{20}} r_2.$$

The line segment $p_0 p_1 \in \bar{E}^3$ can be formulated as a Bézier curve

$$p(t) = (x_0(t), x_1(t), x_2(t), x_3(t)) = p_0 B_0^1(t) + p_1 B_1^1(t). \qquad \text{(Eq.2.13)}$$

The line segment $p(t)$ starts at $p_0 \in L(q_0^*)$, intersects with $L(q_1^*)$, and ends at $p_1 = L(q_2^*)$ (Figure 2.31):

$$\begin{aligned} p(0) &= p_0, \\ p(1) &= p_1, \\ p(\eta) &= (1-\eta)r_0 + \eta\frac{(1-\eta)\lambda_0 p_{00}}{-\eta r_{20}} r_2 \\ &= (1-\eta)\left(r_0 - \frac{\lambda_0 p_{00}}{r_{20}} r_2\right) \\ &= (1-\eta)(1-\lambda_0)\frac{r_{00}}{r_{10}} r_1 \in L(q_1^*). \end{aligned}$$

According to Lemma 2.2, the image of the line segment Eq.2.13 under δ is a circular arc on Q:

$$q(t) = \delta\left(p(t)\right) = r_0^2 B_0^2(t) + r_1^2 B_1^2(t) + r_2^2 B_2^2(t), r_i^2 \in Q. \tag{Eq.2.14}$$

The control points can be obtained using Eq.2.9 and Eq.2.10:

$$\begin{aligned} r_0^2 &= r_{00}, \\ r_1^2 &= \tfrac{1}{2}(r_{01} + r_{10}), \\ r_2^2 &= r_{11}. \end{aligned} \tag{Eq.2.15}$$

The circular arc Eq.2.14 satisfies

$$\begin{aligned} q(0) &= \delta(p(0)) = \delta(p_0) = q_0, \\ q(\eta) &= \delta(p(\eta)) = q_1, \\ q(1) &= \delta(p(1)) = \delta(p_1) = q_2. \end{aligned}$$

Hence the circular arc Eq.2.14 is the arc (q_0, q_1, q_2).

2.8.2.2 An arc as a quartic Bézier curve

A circular arc (q_0, q_1, q_2) can also be formulated as a quartic Bézier curve. Select any two points $p_0 \in L(q_0^*), p_2 \in L(q_2^*)$ (Figure 2.32). From Lemma 2.7, the plane $E(v_1)$

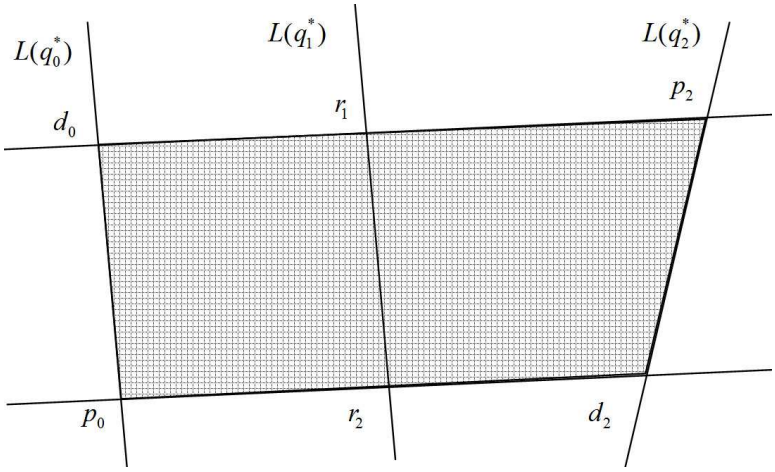


Figure 2.32: The doubly ruled surface passing p_0 and p_2.

passing $L(q_1^*)$ and p_2, and the plane $E(v_2)$ passing $L(q_1^*)$ and p_0 are

$$v_1 = (p_2 \cdot \overline{q_1^*})\overline{q_1^*}^{\perp} - (p_2 \cdot \overline{q_1^*}^{\perp})\overline{q_1^*},$$
$$v_2 = (p_0 \cdot \overline{q_1^*})\overline{q_1^*}^{\perp} - (p_0 \cdot \overline{q_1^*}^{\perp})\overline{q_1^*}.$$

From Lemma 2.5, $E(v_1)$ intersects $L(q_0^*)$ at d_0 and $E(v_2)$ intersects $L(q_2^*)$ at d_2 with

$$d_0 = (v_1 \cdot q_0^*)\, q_0^{*\perp} - \left(v_1 \cdot q_0^{*\perp}\right) q_0^*,$$
$$d_2 = (v_2 \cdot q_2^*)\, q_2^{*\perp} - \left(v_2 \cdot q_2^{*\perp}\right) q_2^*.$$

The line segments p_0d_2 and p_2d_0 are mapped to the arc. The plane $E(v_3)$ passing $L(q_2^*)$ and d_0, and the plane $E(v_4)$ passing $L(q_0^*)$ and d_2 are

$$v_3 = (d_0 \cdot \overline{q_2^*})\overline{q_2^*}^{\perp} - (d_0 \cdot \overline{q_2^*}^{\perp})\overline{q_2^*},$$
$$v_4 = (d_2 \cdot \overline{q_0^*})\overline{q_0^*}^{\perp} - (d_2 \cdot \overline{q_0^*}^{\perp})\overline{q_0^*}.$$

$E(v_3)$ intersects $L(q_1^*)$ at r_1 and $E(v_4)$ intersects $L(q_1^*)$ at r_2 with

$$r_1 = (v_3 \cdot q_1^*)\, q_1^{*\perp} - \left(v_3 \cdot q_1^{*\perp}\right) q_1^*,$$
$$r_2 = (v_4 \cdot q_1^*)\, q_1^{*\perp} - \left(v_4 \cdot q_1^{*\perp}\right) q_1^*.$$

Obviously, each plane contains one *projecting line* and four points

$$\begin{aligned} E(v_1): &\quad L(q_1^*), \quad p_2, d_0, r_1, r_2, \\ E(v_2): &\quad L(q_1^*), \quad p_0, d_2, r_1, r_2, \\ E(v_3): &\quad L(q_2^*), \quad d_0, r_1, p_2, d_2, \\ E(v_4): &\quad L(q_0^*), \quad d_2, r_2, p_0, d_0. \end{aligned}$$

Therefore, $E(v_1)$ and $E(v_3)$ intersect at $d_0 r_1 p_2$ while $E(v_2)$ and $E(v_4)$ intersect at $p_0 r_2 d_2$. Suppose

$$\begin{aligned} d_0 &= (d_{00}, d_{01}, d_{02}, d_{03}), \qquad r_1 = (r_{10}, r_{11}, r_{12}, r_{13}), \qquad p_2 = (p_{20}, p_{21}, p_{22}, p_{23}), \\ d_2 &= (d_{20}, d_{21}, d_{22}, d_{23}), \qquad r_2 = (r_{20}, r_{21}, r_{22}, r_{23}), \qquad p_0 = (p_{00}, p_{01}, p_{02}, p_{03}). \end{aligned}$$

Since d_0, r_1 and p_2 are collinear, and p_0, r_2 and d_2 are collinear, we have

$$\frac{\frac{r_{1i}}{r_{10}} - \frac{p_{2i}}{p_{20}}}{\frac{r_{1i}}{r_{10}} - \frac{d_{0i}}{d_{00}}} = \lambda_2, \qquad \frac{\frac{r_{2i}}{r_{20}} - \frac{p_{0i}}{p_{00}}}{\frac{r_{2i}}{r_{20}} - \frac{d_{2i}}{d_{20}}} = \lambda_0, \quad i = 1, 2, 3. \tag{Eq.2.16}$$

Taking $\eta \in (0, 1)$ and

$$c_0 = \frac{\eta p_{20} \lambda_2}{(\eta - 1) d_{00}} d_0, \qquad c_2 = \frac{(\eta - 1) p_{00} \lambda_0}{\eta d_{20}} d_2, \tag{Eq.2.17}$$

we have a doubly ruled quadric surface as

$$f(u, v) = B_0^1(v) B_0^1(u) p_0 + B_0^1(v) B_1^1(u) c_2 + B_1^1(v) B_0^1(u) c_0 + B_1^1(v) B_1^1(u) p_2.$$

With $u, v \in R$, $f(u, v)$ contains four lines

$$\begin{aligned} f(0, v) &= B_0^1(v) p_0 + B_1^1(v) c_0 \in L(q_0^*), \\ f(1, v) &= B_0^1(v) c_2 + B_1^1(v) p_2 \in L(q_2^*), \\ f(u, 0) &= B_0^1(u) p_0 + B_1^1(u) c_2, \\ f(u, 1) &= B_0^1(u) c_0 + B_1^1(u) p_2. \end{aligned}$$

Suppose the arc (q_0, q_1, q_2) is a segment of the circle $\xi \in Q$. According to Lemma 2.3, the image, $\delta(f(u, v)), u, v \in R$, equals the circle ξ. With $u = t, v = t$, we have a

Bézier curve as

$$p(t) = f(t,t) = B_0^2(t)p_0 + B_1^2(t)\frac{c_0 + c_2}{2} + B_2^2(t)p_2, t \in R. \quad \text{(Eq.2.18)}$$

Obviously, the image, $\delta(p(t)), t \in R$, is also equivalent to the circle ξ, which can be formulated as

$$q(t) = \delta\left(p(t)\right) = r_0^4 B_0^4(t) + r_1^4 B_1^4(t) + r_2^4 B_2^4(t) + r_3^4 B_3^4(t) + r_4^4 B_4^4(t), t \in R. \quad \text{(Eq.2.19)}$$

The control points can be calculated from Eq.2.9 and Eq.2.10:

$$\begin{aligned} r_0^4 &= r_{00}, \\ r_1^4 &= \tfrac{1}{2}(r_{01} + r_{10}), \\ r_2^4 &= \tfrac{1}{6}\left(r_{02} + 4r_{11} + r_{20}\right), \\ r_3^4 &= \tfrac{1}{2}(r_{21} + r_{12}), \\ r_4^4 &= r_{22}. \end{aligned} \quad \text{(Eq.2.20)}$$

Eq.2.16 leads to

$$(1-\lambda_2)\frac{r_{1i}}{r_{10}} - \frac{p_{2i}}{p_{20}} + \lambda_2\frac{d_{0i}}{d_{00}} = 0, \qquad (1-\lambda_0)\,\frac{r_{2i}}{r_{20}} - \frac{p_{0i}}{p_{00}} + \lambda_0\frac{d_{2i}}{d_{20}} = 0, \quad i = 0,1,2,3.$$

Equivalently,

$$(1-\lambda_2)\frac{p_{20}}{r_{10}}r_1 = p_2 - \frac{p_{20}\lambda_2}{d_{00}}d_0, \qquad (1-\lambda_0)\,\frac{p_{00}}{r_{20}}r_2 = -\frac{p_{00}\lambda_0}{d_{20}}d_2 + p_0. \quad \text{(Eq.2.21)}$$

Coupling with Eq.2.17, we have

$$\eta(1-\lambda_2)\frac{p_{20}}{r_{10}}r_1 = \eta p_2 + (1-\eta)c_0, \qquad (1-\eta)(1-\lambda_0)\,\frac{p_{00}}{r_{20}}r_2 = \eta c_2 + (1-\eta)p_0.$$

Therefore, Eq.2.18 satisfies

$$\begin{aligned} p(0) &= p_0, \\ p(1) &= p_2, \\ p(\eta) &= (1-\eta)^2 p_0 + \eta(1-\eta)(c_0+c_2) + \eta^2 p_2 \\ &= (1-\eta)((1-\eta)p_0 + \eta c_2) + \eta((1-\eta)c_0 + \eta p_2) \\ &= (1-\eta)(1-\eta)(1-\lambda_0)\tfrac{p_{00}}{r_{20}} r_2 + \eta\eta(1-\lambda_2)\tfrac{p_{20}}{r_{10}} r_1 \in L(q_1^*). \end{aligned}$$

This means that the curve $p(t), t \in [0,1]$ uses $p_0 \in L(q_0^*), p_1 \in L(q_2^*)$ as two ends while passing $L(q_1^*)$. Hence, $q(t), t \in [0,1]$ is the arc (q_0, q_1, q_2).

2.8.3 A sub-patch in Bézier form

In $\bar{E}^3$, a Bézier surface is

$$\begin{aligned} p(u,v) &= (x_0(u,v), x_1(u,v), x_2(u,v), x_3(u,v)) \\ &= \sum_{i=0}^{n}\sum_{j=0}^{m} p_{ij} B_i^n(u) B_j^m(v), \qquad \text{(Eq.2.22)} \\ p_{ij} &= (p_{ij,0}, p_{ij,1}, p_{ij,2}, p_{ij,3}) \in \bar{E}^3. \end{aligned}$$

On Q, we have the corresponding Bézier surface as

$$q(u,v) = \delta(p(u,v)) = \sum_{k=0}^{2n}\sum_{l=0}^{2m} r_{kl} B_k^{2n}(u) B_l^{2m}(v), r_{kl} \in Q. \qquad \text{(Eq.2.23)}$$

The product of two Bézier can be formulated as

$$\begin{aligned} x_s x_t &= \left(\sum_{a=0}^{n}\sum_{b=0}^{m} p_{ab,s} B_a^n(u) B_b^m(v)\right)\left(\sum_{c=0}^{n}\sum_{d=0}^{m} p_{cd,t} B_c^n(u) B_d^m(v)\right), \\ &= \sum_{k=0}^{2n}\sum_{l=0}^{2m}\left[\frac{\sum_{a+c=k}\sum_{b+d=l} p_{ab,s}p_{cd,t}\binom{m}{b}\binom{m}{d}\binom{n}{a}\binom{n}{c}}{\binom{2n}{k}\binom{2m}{l}}\right] B_k^{2n}(u) B_l^{2m}(v). \end{aligned}$$

Setting

$$r_{cd}^{ab} = \begin{pmatrix} p_{ab,0}p_{cd,0} + p_{ab,1}p_{cd,1} + p_{ab,2}p_{cd,2} + p_{ab,3}p_{cd,3} \\ 2\left(p_{ab,0}p_{cd,1} - p_{ab,2}p_{cd,3}\right) \\ 2\left(p_{ab,0}p_{cd,2} + p_{ab,1}p_{cd,3}\right) \\ p_{ab,1}p_{cd,1} + p_{ab,2}p_{cd,2} + p_{ab,3}p_{cd,3} - p_{ab,0}p_{cd,0} \end{pmatrix},$$

we have

$$\begin{aligned} q(u,v) &= \delta\left(p(u,v)\right) \\ &= (x_0^2 + x_1^2 + x_2^2 + x_3^2, 2x_0x_1 - 2x_2x_3, 2x_0x_2 + 2x_1x_3, x_1^2 + x_2^2 + x_3^2 - x_0^2) \\ &= \sum_{k=0}^{2n}\sum_{l=0}^{2m} B_k^{2n}(u)B_l^{2m}(v) \left[\frac{\sum\limits_{a+c=k}\sum\limits_{b+d=l} \binom{m}{b}\binom{m}{d}\binom{n}{a}\binom{n}{c} r_{cd}^{ab}}{\binom{2n}{k}\binom{2m}{l}} \right]. \end{aligned}$$

Then

$$r_{kl} = \frac{\sum\limits_{a+c=k}\sum\limits_{b+d=l} r_{cd}^{ab} \binom{m}{b}\binom{m}{d}\binom{n}{a}\binom{n}{c}}{\binom{2n}{k}\binom{2m}{l}}. \qquad \text{(Eq.2.24)}$$

Four arcs are given in Eq.2.5. Following Section 2.8.2.1, we formulate a_1 in Bézier form with control points p_1, p_2 and formulate a_3 in Bézier form with control points p_3, p_4. Take

$$p_{00} = p_1, \qquad p_{02} = p_4,$$
$$p_{10} = p_2, \qquad p_{12} = p_3,$$

Following Section 2.8.2.2, we formulate a_2 in Bézier form with control points p_2, p_5, p_3 and formulate a_4 in Bézier form with control points p_4, p_6, p_1. Take

$$p_{00} = p_1, \quad p_{01} = p_6, \quad p_{02} = p_4,$$
$$p_{10} = p_2, \quad p_{11} = p_5, \quad p_{12} = p_3,$$

A Bézier surface in $\bar{E}^3$ is

$$\begin{aligned} p(u,v) &= (x_0(u,v), x_1(u,v), x_2(u,v), x_3(u,v)) \\ &= \sum_{i=0}^{1}\sum_{j=0}^{2} p_{ij}B_i^1(u)B_j^2(v), p_{ij} = (p_{ij,0}, p_{ij,1}, p_{ij,2}, p_{ij,3}) \in \bar{E}^3. \end{aligned} \quad \text{(Eq.2.25)}$$

On Q, we have the corresponding Bézier surface as

$$q(u,v) = \delta\left(p(u,v)\right) = \sum_{k=0}^{2}\sum_{l=0}^{4} r_{kl}B_k^{2n}(u)B_l^{2m}(v), r_{kl} \in Q. \quad \text{(Eq.2.26)}$$

Combining Eq.2.24 and Eq.2.26, we formulate a sub-patch, whose boundary arcs are given in Eq.2.5, as a Bézier surface of degree (2, 4).

2.9 Discussion and applications

A SES is a rolling ball surface (RBS) of a sphere set. Sphere set is widely used in different applications. In mathematics, a typical sphere packing problem is to fill in the maximum proportion of a space using a sphere set. In computer graphics, the sphere tree approximating to an object can be used for collision detections and shadow calculations.

Our uniform solution can be easily extended to RBS modeling. There are two choices for RBS: hollow model and non-hollow model. As shown in Figure 2.33, RBS has two steps: network modeling and surface modeling. Hollow model will allow hollow cavities (Figure 2.34(a)) and non-hollow model will only model outside surface (Figure 2.34(b)). The non-hollow model is equivalent to SES model.

The trimmed NURBS method can also model the hollow RBS. We take the model in Figure 2.34(c) as an example to illuminate how trimmed NURBS method modeled a hollow cavity. From a vertex (a blue dot), trimmed NURBS method created a single trimmed NURBS surface to describe the two separate spherical patches (one is a green arc and the other is a red arc). It is difficult to check which patch is inside. Therefore, trimmed NURBS method can not provide non-hollow model directly.

If an object is described by a sphere set, we derive the non-hollow RBS model. Figure 2.35 shows the different results by changing the ball radius. Figure 2.35(a,b) are

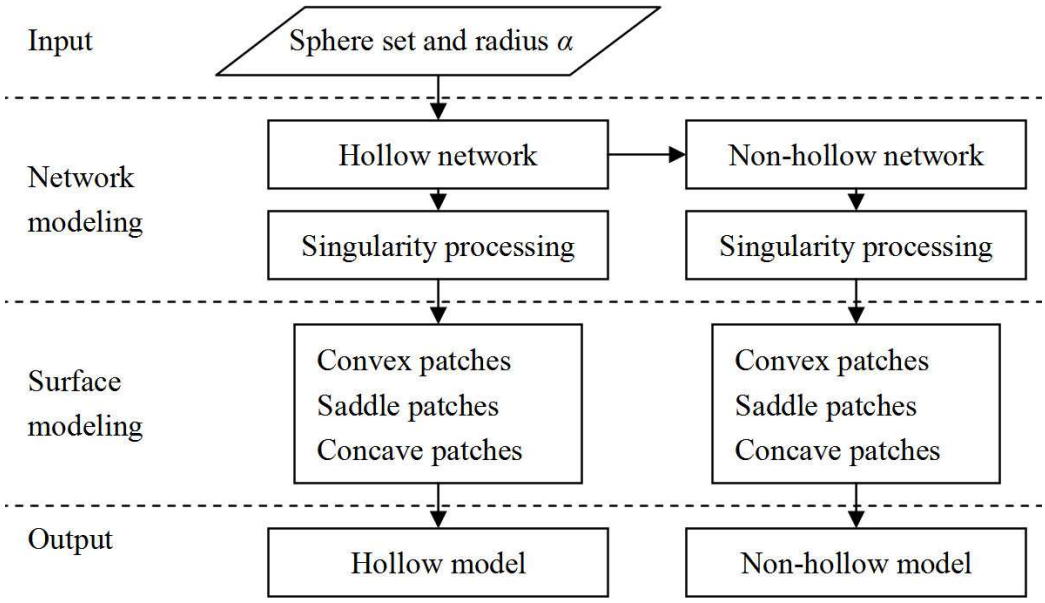
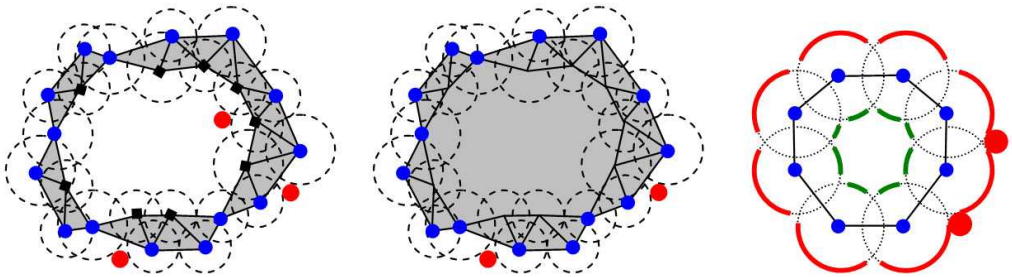


Figure 2.33: Flowchart for rolling ball surfaces.



(a) (b) (c)

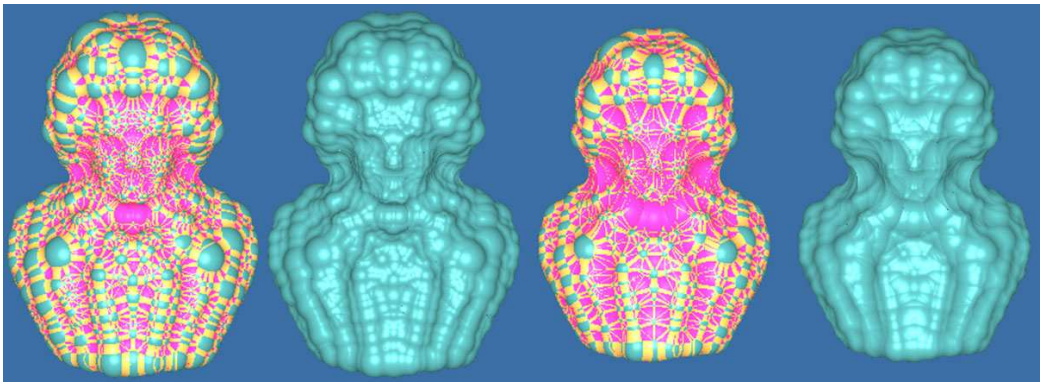
Figure 2.34: Two types of networks for sphere set.

red disk: the rolling ball; blue dots: vertices for both hollow networks and non-hollow networks; black square: vertices for hollow networks; red arcs and green arcs: contact surfaces. (a) hollow networks; (b) non-hollow networks; and (c) contact surfaces from hollow networks.

the boundary surfaces using a selected α value and Figure 2.35(c,d) are the boundary surfaces using the value 2α. Figure 2.36 shows the hollow model and the non-hollow model created from the same sphere set. In order to view the hollow inside the solid, we cut out part of the boundary. As we can see, the non-hollow model (Figure 2.36(d)) is a sub-set of the hollow model (Figure 2.36(b)). Table 2.2 lists the element numbers for these examples.

2.10 Summary

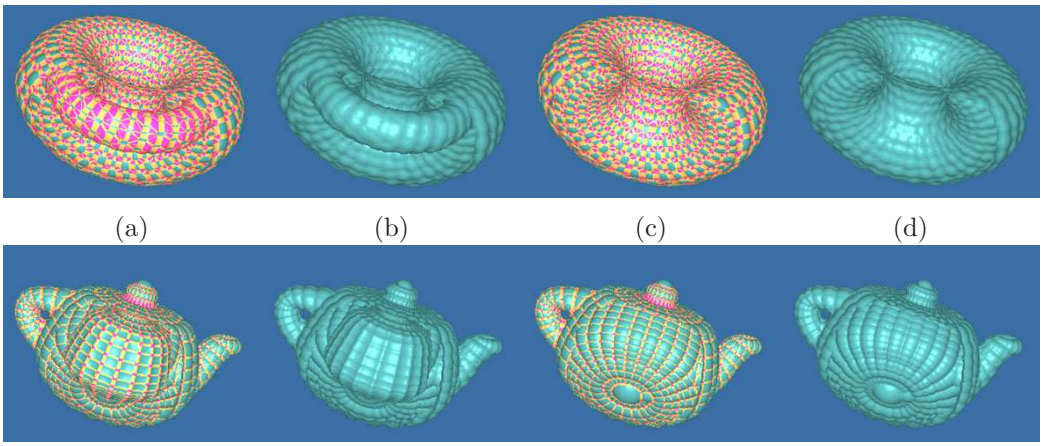
We have proposed a uniform solution to modeling three types of molecular surfaces. Each molecular surface (vdWS, SES and SAS) can be created through topology mod-



(a) (b) (c) (d)

Figure 2.35: Beethoven model.

(a) and (b) rolling ball surface with ball radius α; and (c) and (d) rolling ball surface with ball radius 2α.



(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 2.36: The hollow model and the non-hollow model.

(a) and (e) three types of patches for the hollow model; (b) and (f) the hollow RBS; (c) and (g) three types of patches for the non-hollow model; (d) and (h) the non-hollow RBS.

eling, boundary modeling and surface modeling. In the first step, topology modeling creates two topological networks using a weighted α-shape. The networks are used to maintain the neighboring relationship of the atoms. The vdWS network is used for the vdWS. The solvent network is used for the SES and the SAS. For each network, there are boundary vertices, edges and triangles. In the second step, three types of

Table 2.2: The number of the elements for the examples.

	Sphere		Saddle		Convex		Concave		Bézier
	All	Boundary	Patch	Bézier	Patch	Bézier	Patch	Bézier	
Figure 2.35(a,b)	1393	1005	3459	3490	1155	3553	2307	2397	9440
Figure 2.35(c,d)	1393	668	2193	2229	733	2248	1463	1570	6047
Figure 2.36(a,b)	800	780	3780	3780	1235	3946	2520	2520	10246
Figure 2.36(c,d)	800	780	2340	2340	780	2380	1560	1560	6280
Figure 2.36(e,f)	1002	811	3612	3612	1206	3615	2408	2408	9635
Figure 2.36(g,h)	1002	723	2169	2169	723	2184	1446	1446	5799

boundary arcs are created: SES arcs, SAS arcs and vdWS arcs. SES arcs and SAS arcs are created from the solvent network. From the vdWS network, vdWS arcs are created. Singularity processing is used to further modify the SES arcs by removing the self-intersections. In the third step, all the arc-bounded patches are modeled in rational Bézier form. Each molecular surface is modeled as a collection of rational Bézier surfaces.

There are several applications that can be developed using the proposed method. We can calculate the area of a single rational Bézier patch. By summing up the area of all the rational Bézier patches, we can obtain the molecular surfaces area. Though the area so calculated is an estimation here, the uniform solution is able to achieve high accuracy as the rational Bézier surface can be tessellated at any desired approximation. A dynamic molecular modeling technique can also be studied. Based on the topological networks, we can construct a dynamic molecular surfaces from the motion of atoms with the same molecule. The chemical properties (hydrophobic/hydrophilic, electrostatic, etc.) can be studied based on the atom property and patch geometry. For a rectangular sub-patch, we have modeled it in rational Bézier form using a *generalized stereographic projection*. Currently the weights of the control point may be negative and we are keen to model them with positive weights.

Chapter 3

Minimal Surfaces[2]

This chapter is concerned with the problem of constructing an aesthetically pleasing triangular mesh with a given closed polygonal contour in three dimensional space as boundary. Triangular meshes of minimal area from all triangular meshes with the prescribed boundary are suggested as the candidates for this problem. An iterative algorithm of constructing such a triangular mesh from a given polygonal boundary is presented. Experimental examples show that the proposed algorithm is reliable and effective. Some related theoretical issues, possible extensions and applications are also discussed.

3.1 Background

One of the main tasks in geometric modeling is the generation of aesthetically pleasing surfaces [70]. The surfaces are usually created from some inputs such as a set of 3D points or curves that serve as constraints or guidance for the surfaces. However, the selection of the most appealing surface or the most reasonable surface from the given constraints is subjective. There is no best answer for all situations. In practice, many energy functionals have been used, which are defined in terms of elastic membranes or thin plates or geometric invariants like curvatures. The surface with the minimal energy is selected.

[2]The following publication is based on the results of this chapter:
Chen,W.Y., Cai,Y.Y., and Zheng,J.M. (2008). **Constructing Triangular Meshes of Minimal Area**, *Computer-Aided Design and Applications*, 5(1-4), 508-518.

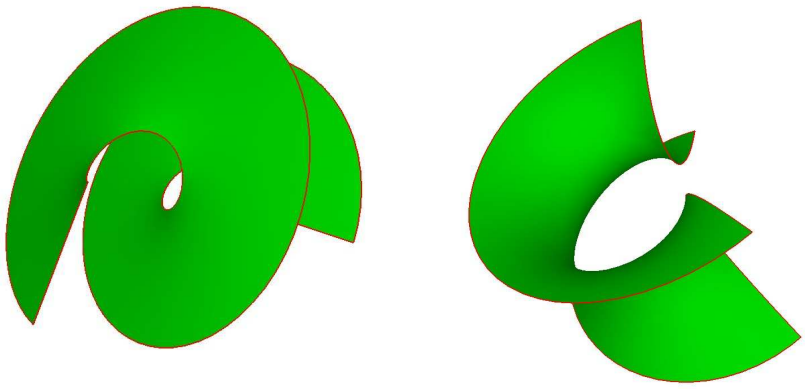


Figure 3.1: The Helicoid and Catenoid surfaces.

This chapter studies the problem of how to construct the visually pleasing shapes from prescribed boundary and the minimal surfaces are considered to be the natural solutions. Figure 3.1 shows two famous minimal surfaces: the Helicoid and Catenoid surfaces. Minimal surfaces refer to the surfaces that minimize surface area. The physical models of minimal area surfaces can be made by dipping a closed, curved wire frame into a solution of soap and water and withdrawing it. A soap film is formed, which is a minimal surface whose boundary is the wire frame. The naturalness of minimal surfaces may be partially explained by this fact from physics: the surface tension that governs the film's shape is proportional to the area, the film tries to minimize the tension everywhere subject to the fixed boundary constraint, and thus the shape tends to form the surface of minimal area among nearby surfaces with the same boundary. Due to their special properties, minimal surfaces often become the candidates of ideal models in many applications. For example, minimal surfaces were used in architecture for light roof constructions, form-finding models for tents, nets and air halls. In computer graphics, polyhedra of minimal surface area were suggested as natural candidates for object models [71] and the triangular tiles of minimal surface area were used to interpolate parallel slices [72].

Mathematically, minimal surfaces are characterized as surfaces whose mean curvature vanishes everywhere, reflecting the fact that there is no pressure differential across the surface. Finding a surface that minimizes the area is actually a problem of calculus of variations [73]. In particular, the problem of finding the minimal surface for a given boundary curve is known as the Plateau problem after the Belgian Physicist

Plateau who carried out extensive experiments with soap films in the mid-nineteenth century. Out of his investigations there developed a conjecture that every closed, non-self-intersecting curve can be spanned by a minimal surface. The conjecture was mathematically proved in 1930 by Rado [74] and in 1931 by Douglas [75] independently. In general, exact solutions are usually complicated and difficult to find. Many numerical methods have been developed to approximate the exact minimal surfaces. For example, Douglas used the finite element method to find a numerical solution of the Plateau problem [76] and Wilson used the boundary element method to produce an approximate minimal surface [77]. Wang *et al.* [78] combined the Trefftz finite element formulation with the radial basis functions and the analogue equation method to analyze minimal surface problems. Different functional energies have been used in developing numerical methods. Area functional, mean curvature flow and the Dirichlet energy are the typical energies. Tsuchiya [79–81] proposed two numerical methods: one minimizes the surface area and the other minimizes the Dirichlet energy. Both solutions converge to the minimal surface in a suitable function space. Dziuk [82] used the mean curvature flow to compute stable minimal surfaces by a semi implicit finite element scheme. The minimal surfaces spanned by a polygon were studied by Hinze and a numerical method was proposed based on a theoretical result that the minimal surfaces spanning the polygon correspond in a one to one manner to the critical points of Shiffman's function [83].

Polynomial approximation to minimal surfaces is also of interest. Monterde *et al.* studied the Plateau-Bézier problem that finds the surface of minimal surface area among all the Bézier surfaces with prescribed border [84–86]. Given three or four Bézier curves, the triangular Bézier patch or tensor-product Bézier patch which minimizes the Dirichlet energy can be found. It is shown that the resulting Bézier surface patch does not minimize the area in general but has the area close to the minimum.

We present a new method for constructing a triangular mesh of minimal area from a given polygonal boundary. Unlike previous numerical approaches that are based on the sophisticated mathematics, the new approach is in the fashion of digital

geometry processing [87–89], which is conceptually simple and easy to implement. The work by Pinkall and Polthier presents a numerical minimization procedure to find discrete minimal surfaces bounded by a number of boundary curves [90]. The method begins with an initial mesh and iteratively updates the mesh by minimizing the Dirichlet integral. Beginning with only the boundary polygon, our method has three basic processes: area minimizing, Laplacian fairing and edge swapping. The first two processes are used to optimize the geometry of the mesh and the third one is used to adjust the connectivity of the triangular mesh. We also gives a simple initialization step which can control the level of detail of the resulting mesh. The experimental results have demonstrated that the new approach performs very stably and effectively. Since nowadays triangular meshes are widely used in computer aided design and computer graphics because of their simplicity and powerful capability to model complicated shapes, the new method can find applications where smooth, visual appealing shapes are required. In addition, the method can also be used as a visualization tool in minimal surface study [91].

3.2 Preliminaries and notations

Let Ω, a closed subset of R^2, be the parameter domain of the surfaces, with boundary $\partial\Omega$. Let Γ be the given 3D curve defined over $\partial\Omega$. The Plateau problem is to find a parametric surface $r(u,v), (u,v) \in \Omega$, which is the solution of the minimization problem:

$$\min_r \iint_\Omega \|r_u(u,v) \times r_v(u,v)\| dudv = \min_r \iint_\Omega \sqrt{EG - F^2} dudv \qquad \text{(Eq.3.1)}$$

with $r(u,v)|_{\partial\Omega} = \Gamma$, where E, F and G are the coefficients of the first fundamental form of $r(u,v)$.

As can been seen, the area expression in (Eq.3.1) is in general complicated. A good number of numerical approximation approaches of minimal surfaces do not minimize the area functional directly. Instead, they try to minimize the following functional called the Dirichlet functional (or thin-plate energy of surface $r(u,v)$ in geometric

modeling):

$$\iint_{\Omega}(E+G)dudv = \iint_{\Omega}(r_u^2(u,v)+r_v^2(u,v))dudv. \qquad \text{(Eq.3.2)}$$

The area functional and the Dirichlet functional have the following relation:

$$\iint_{\Omega}\sqrt{EG-F^2}dudv \leq \iint_{\Omega}\sqrt{EG}dudv \leq \frac{1}{2}\iint_{\Omega}(E+G)dudv.$$

Obviously, both functionals are the same if and only if $E = G$ and $F = 0$, which implies that the surface $r(u, v)$ is a conformal mapping. In addition, while the area functional is independent of the parameterization of the surface, the Dirichlet one depends on the parameterization. However, the Dirichlet functional is easier to manage and there holds an important result: both the area and Dirichlet functional have the same extremals in the unrestricted case [92]. In the Bézier case (i.e., the minimal surfaces are restricted to polynomial surfaces), the Dirichlet extremals are an approximation to the extremals of the areal functional [85].

The variational derivative of the Dirichlet functional corresponds to the Laplacian and can be expressed as

$$\Delta r(u,v) = r_{uu} + r_{vv},$$

where Δ is the Laplacian operator. Therefore, if a surface $r(u, v)$ is harmonic, i.e., $\Delta r(u, v) = 0$, it minimizes the Dirichlet energy. Furthermore, if $r(u, v)$ is also conformal, then it is a minimal surface.

3.3 Construction of optimal triangular meshes

We now describe our approach dealing with the discrete version of the Plateau problem. Suppose that we are given a simple polygon Γ with n vertices and an integer $m(\geq n)$ that has the same parity as n. There are an infinite number of triangular meshes with m triangles spanning Γ. Our task is to find one triangular mesh from the set of such triangular meshes, which has the minimal area. Note that the requirement of the same parity of m and n is due to the Euler-Poincaré formula. In addition, though we can construct a triangular mesh with $n-2$ triangles for the given bound-

ary Γ, we should in general have sufficient number of triangles to make the mesh look smooth. Therefore in this chapter we ignore the case of $m = n-2$ and always assume that $m \geq n$. When $m \geq n$, the triangular mesh will contain some vertices other than the given ones of the boundary. The coordinates of the new vertices provide degrees of freedom for optimizing the shape of the triangular mesh. Number m can also be viewed as a control for levels of detail in approximation of a continuous minimal surface.

A triangular mesh contains two aspects of information: geometry and connectivity. Geometry is defined by the coordinates of vertices of the mesh and it tells the location of the mesh in 3D space. Connectivity defines how the vertices are joined to form the mesh. Our aim is to create a triangular mesh which is optimal in both geometry and connectivity. Optimizing geometry and connectivity simultaneously is a very difficult problem. Our strategy is to separate geometry and connectivity. For the given connectivity of vertices of the mesh, we optimize geometry (i.e., the coordinates of vertices) and for fixed vertices of the mesh, we find an optimal triangulation. In this chapter the former will be implemented by the processes of area minimizing and Laplacian fairing and the latter by the process of edge swapping. These three processes are the main ingredients of our proposed algorithm. Below they will be explained first and then the algorithm is presented.

3.3.1 Area minimizing

Let a triangular mesh M be represented as a triple $< I, P, T >$, where $I = \{1, 2, ..., N\}$ is its vertex index set, $P : I \rightarrow R^3$ is a mapping from the vertex indices to their locations in 3D space, T is its triangle set, and each triangle $t \in T$ is represented as an ordered vertex index triple $t =< i, j, k >$ meaning that the triangle is defined by vertices $P(i), P(j)$ and $P(k)$. Without causing ambiguity, we use P_i to replace $P(i)$ for simplicity. In triple $< I, P, T >$, we assume that the first n vertices $P_i, i = 1, \cdots, n$ are the given vertices on the boundary Γ.

The area A of mesh M is just the sum of all triangles' areas:

$$A = \sum_{t=<i,j,k>\in T} \frac{1}{2}|P_jP_k \times P_jP_i| = \sum_{t=<i,j,k>\in T} \frac{1}{2}\sqrt{(P_jP_k \times P_jP_i)^2}. \qquad \text{(Eq.3.3)}$$

With the boundary vertices fixed, area A is a function of vertices $P_{n+1}, \cdots, P_N$. The process of *area minimizing* is to find appropriate positions for $P_{n+1}, \cdots, P_N$ such that the area functional A will be minimized.

Let $NT(i) \subseteq T$ be the set of all the triangles that contain vertex P_i and $\frac{\partial A}{\partial P_h} = \left(\frac{\partial A}{\partial (P_h)_x}, \frac{\partial A}{\partial (P_h)_y}, \frac{\partial A}{\partial (P_h)_z}\right)^T$. By some simplification, we have for $h = n+1, \cdots, N$

$$\begin{aligned} \frac{\partial A}{\partial P_h} &= \frac{1}{2} \sum_{t=<i,j,k>\in T} \frac{\partial}{\partial P_h}\sqrt{(P_jP_k \times P_jP_i)^2} \\ &= \frac{1}{2} \sum_{t=<h,j,k>\in NT(h)} \frac{1}{2}\frac{\frac{\partial}{\partial P_h}(P_jP_k \times P_jP_h)^2}{\sqrt{(P_jP_k \times P_jP_h)^2}} \\ &= \frac{1}{2} \sum_{t=<h,j,k>\in NT(h)} \frac{(P_jP_k)^2 P_jP_h - (P_jP_k \cdot P_jP_h)P_jP_k}{\sqrt{(P_jP_k \times P_jP_h)^2}}. \end{aligned} \qquad \text{(Eq.3.4)}$$

Setting all these derivatives to zero leads to $(N-n)$ equations with $(N-n)$ variables. The solution renders us an optimal mesh. However, the equations are non-linear. It is difficult to solve such a non-linear system. For a mesh with a large data set, the situation even becomes worse.

Here we propose a local mechanism and iteratively approximate the solution. Rewrite (Eq.3.4) as

$$\frac{\partial A}{\partial P_h} = \frac{1}{2}CP_h + \frac{1}{2} \sum_{t=<h,j,k>\in NT(h)} \frac{(P_jP_k \cdot P_j)P_jP_k - (P_jP_k)^2 P_j}{\sqrt{(P_jP_k \times P_jP_h)^2}},$$

where

$$C = \sum_{t=<h,j,k>\in NT(h)} \frac{(P_jP_k)^2 \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} - (P_jP_k)(P_jP_k)^T}{\sqrt{(P_jP_k \times P_jP_h)^2}}$$

is a 3×3 matrix. Letting $\dfrac{\partial A}{\partial P_h} = 0$, we have

$$P_h = -C^{-1} \sum_{t=<h,j,k>\in NT(h)} \frac{(P_jP_k \cdot P_j)P_jP_k - (P_jP_k)^2 P_j}{\sqrt{(P_jP_k \times P_jP_h)^2}}.$$

The above equation cannot be considered to be an explicit solution for P_h because the right-hand side of the equation also contains P_h. However, it gives us a way to update vertex P_h in the fashion of signal processing [87]. That is, we compute the new vertex $\bar{P}_h$ from the old P_h and its 1-ring neighboring vertices by

$$\bar{P}_h = -C^{-1} \sum_{t=<h,j,k>\in NT(h)} \frac{(P_jP_k \cdot P_j)P_jP_k - (P_jP_k)^2 P_j}{\sqrt{(P_jP_k \times P_jP_h)^2}}. \qquad \text{(Eq.3.5)}$$

When formula (Eq.3.5) applies to all the interior vertices $P_{n+1}, \cdots, P_N$ once, this completes one iteration. The process continues until the area change by one iteration is smaller than a prescribed tolerance.

3.3.2 Laplacian fairing

We also examine the Dirichlet approach. Since the Dirichlet functional (Eq.3.2) depends on the parameterization of the surface, it cannot be used directly for a mesh model. However, we consider its variational derivative—the Laplacian. For a triangular mesh, a discrete Laplacian should be used. Let $N(i) \subseteq I$ be the index set of the 1-ring neighboring vertices of vertex i. The Laplacian operator $\Delta()$ can be approximated at each vertex by the umbrella operator:

$$\Delta(P_i) = \sum_{j\in N(i)} w_{ij}(P_j - P_i)$$

where the weights w_{ij} are positive numbers that sum to one for each i. There are many ways to choose the weights based on the neighborhood structures. In this chapter, we choose

$$w_{ij} = \frac{S_{ij}}{\sum\limits_{k\in N(i)} S_{ik}}$$

where S_{ij} is the area of the two triangles that share the edge connecting P_i and P_j.

Now for all interior vertices $P_i, i = n+1, \cdots, N$ of mesh M, let their Laplacian $\Delta(P_i)$ equal zero. This results in $N-n$ equations with $N-n$ unknowns. Unfortunately, the equations are non-linear due to the fact that our chosen Laplacian operator is non-linear. Then we take a similar approach that we used in Section 3.3.1. We update P_i to $\bar{P}_i$ by averaging its old neighboring vertices:

$$\bar{P}_i = \sum_{j \in N(i)} w_{ij} P_j,$$

where w_{ij} are computed from old P_i and P_j. When all the interior vertices are updated, this completes one iteration. Similarly, the iteration continues until the area change is smaller than the prescribed tolerance. Since the above updating is similar to Laplacian filtering, the process is called *Laplacian fairing*.

3.3.3 Edge swapping

Given a set of points in 3D space, there are many ways to connect them to form a triangular mesh. Obviously, the number of possible triangulations is huge. Not all of them possess equally pleasing shapes and for a particular application some will be much more acceptable than others. This suggests that we need to find an optimal triangulation in some sense.

Here we intend to improve a given mesh by changing its connectivity via a simple local transformation, called the *edge swapping*. Refer to Figure 3.2. The edge swapping transforms two triangles sharing one edge shown on the left into another two triangles shown on the right. Note that during this transformation, the four vertices remain unchanged. What has changed is the connectivity among the vertices. The edge swapping should only be performed if it improves the mesh. For our application, the improvement is measured by the area reduction.

For the given mesh M, our edge swapping algorithm is performed using Lawson's local optimization approach [93]. It visits each interior edge of M and checks whether the edge swapping reduces the area of the mesh. If the area can be reduced, the algorithm swaps the edge. After the algorithm visits all the edges in this way, it

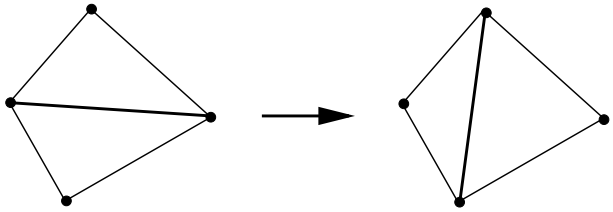


Figure 3.2: Edge swapping.

completes one iteration. If the algorithm makes any swaps during the first iteration, it conducts a second iteration of edge visits. This process continues until no swap is made in one iteration.

This process is simple and fast. It always terminates in a finite number of iterations. However, it is essentially a "best-first" algorithm, which usually has the drawback that it may return a local optimum. To get globally optimal solutions, simulated annealing technique may be considered [94], but the computational cost is much higher.

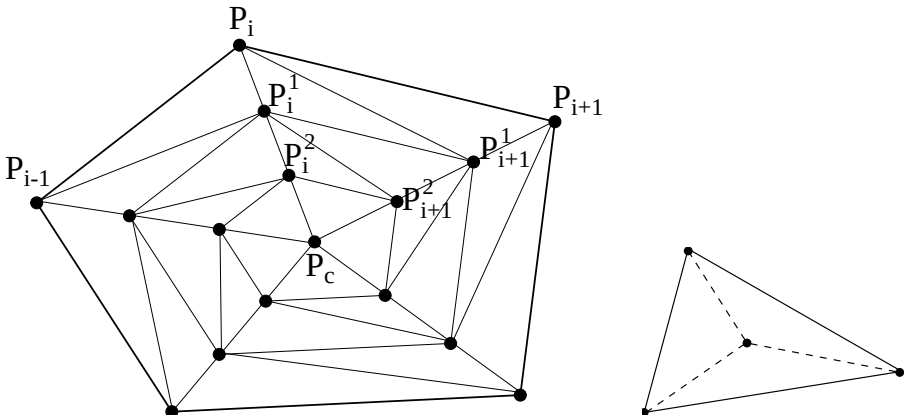
3.3.4 Algorithm

Note that area minimizing, Laplacian fairing and edge swapping cannot be applied to boundary condition directly because the three processes all assume that there has already existed a triangular mesh. Therefore we need an initialization step to create an initial triangular mesh from input.

Here we give a simple way to create our initial triangular mesh. Given the boundary $P_1 \cdots P_n$ and m as the number of triangles, we first compute the central point of the polygon by averaging the vertices: $P_c = \sum_{i=1}^{n} P_i/n$. Then for each vertex P_i, we connect it to P_c and also add s interior vertices P_i^j along line segment P_iP_c:

$$P_i^j = P_i + \frac{j}{s+1}(P_c - P_i), j = 1, 2, \cdots, s,$$

where $s = \lfloor \frac{m-n}{2n} \rfloor$ stands for how many rings will be added. Next, for each j, we join $P_1^j, P_2^j, \cdots, P_n^j$ by the order to form a polygon. This results in a mesh which contains triangles and possibly quadrilaterals. For the quadrilateral facets such as $P_i^j P_{i+1}^j P_{i+1}^{j+1} P_i^{j+1}$, we split them into two triangles by inserting a diagonal edge such as $P_i^j P_{i+1}^{j+1}$. So far we have created a triangular mesh which consists of $(2s+1)n$



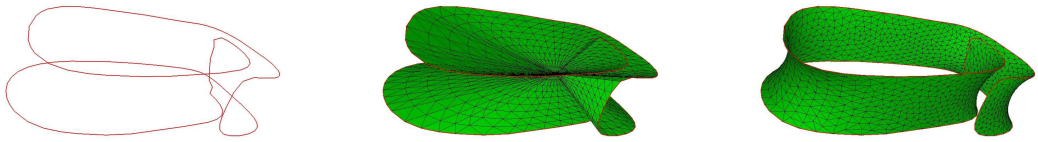
(a) (b)

Figure 3.3: Initial mesh.
(a) creating an initial mesh with $s = 2$; and (b) refining an individual triangle.

triangles (refer to Figure 3.3(a)). If $(2s+1)n$ does not equal to m, we need to further insert $m-(2s+1)n$ triangles to match m. This can be done by arbitrarily choosing $(m-(2s+1)n)/2$ triangles (for example, those on the outer ring) and split each of them into 3 sub-triangles with a new vertex at the center of the triangle (see Figure 3.3(b)).

It should be pointed out that this initialization method is simple and easy to create a mesh with required number of triangles, but it is not optimized and can be improved by taking the shape of the boundary into consideration and/or employing some optimization criteria such as those used in Delaunay triangulation. The current initialization method may generate an initial mesh which is far from satisfactory, but our subsequent techniques are able to correct it. The experimental examples have demonstrated that our proposed algorithm can always return the triangular meshes of minimal area. Figure 3.4 shows one of such examples, in which the initial mesh contains self-intersection. In addition, the initialization step also sets the topology of the final minimal surface besides providing an initial mesh. This is because the subsequent processes do not change the topology of the mesh as a surface. Therefore if a special topology is expected, a different initialization is needed.

Once an initial triangular mesh is created, the edge swapping, Laplacian fairing and area minimizing are used to improve the connectivity and positions of vertices. It can be observed that Laplacian fairing works as a filtering and its numerical performance is much more stable than that of area minimizing. This suggests that we should apply Laplacian fairing first and then use area minimizing. Therefore we propose our



(a) (b) (c)

Figure 3.4: Removal of self-intersections.
(a) the input boundary; (b) the generated initial mesh; and (c) the final minimal area surface.

algorithm as follows:

Step 1: Initialization
Edge swapping

Step 2: DO{
Laplacian fairing
Edge swapping
}WHILE (area change $> \epsilon_1$)

Step 3: DO{
Laplacian fairing
Area minimizing
Edge swapping
}WHILE (area change $> \epsilon_2$)

Step 4: DO{
Area minimizing
Edge swapping
}WHILE (area change $> \epsilon_3$)

Step 5: Output

Algorithm 3.1: Minimal area mesh

In the above algorithm, ϵ_1, ϵ_2 and ϵ_3 are three prescribed tolerances to control when the iterations should stop. Step 1 creates an initial mesh with the given boundary polygon. Step 2 is kind of discrete Dirichlet approach which gives a good approximation to the solution. Step 3 combines Laplacian fairing and area minimizing together as an intermedius step. Step 4 further refines the approximation by minimizing the area functional. In all these steps, the edge swapping is added to improve the connectivity of the mesh.

3.4 Experimental examples

This section provides some examples to demonstrate the algorithm. In particular, the first two examples are designed to check the validity of the algorithm. Their input boundary polygons were generated from the classic Helicoid and Catenoid minimal surfaces. In addition, we also examine the mean curvature of the resulting triangular meshes. The concept of mean curvature for a mesh surface and its computational formula are from *discrete* differential geometry [95]. The absolute value of mean curvature k_i at vertex P_i on a mesh is computed by

$$|k_i| = \frac{1}{4A_i} \left| \sum_{j \in N(i)} (cot\alpha_{ij} + cot\beta_{ij})(P_j - P_i) \right|,$$

where A_i is the sum of areas of triangles that contain vertex P_i, and α_{ij}, β_{ij} are two angles corresponding to edge P_iP_j (refer to Figure 3.5 for illustration).

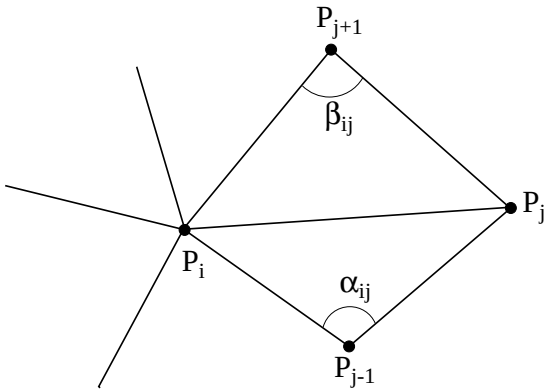
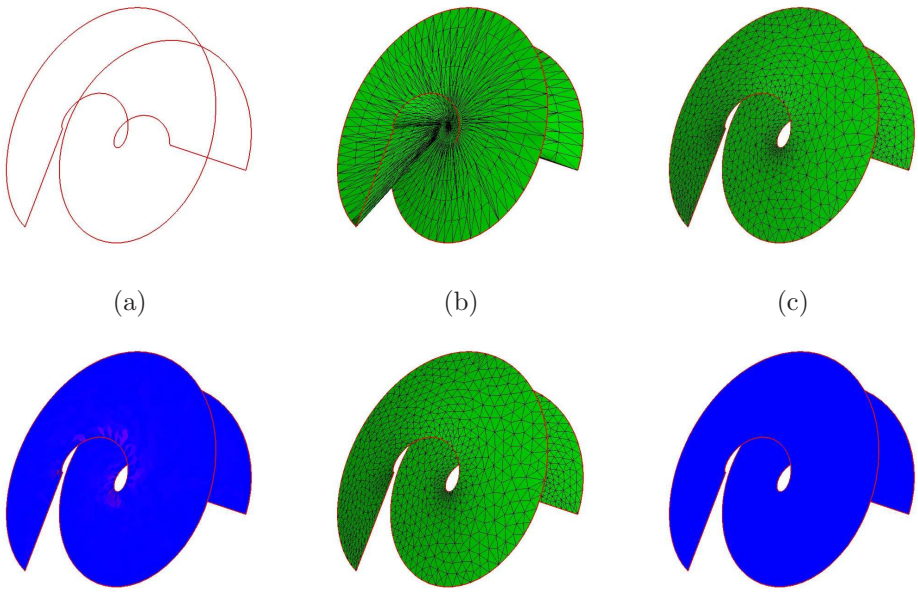


Figure 3.5: Illustration of symbols in the mean curvature formula.

The input polygon in the first example is shown in Figure 3.6(a), which consists of 186 points. There points were obtained by sampling the boundary of a patch of the Helicoid surface. The patch is defined by parametric equations

$$x = u\cos v, \quad y = u\sin v, \quad z = v, \qquad (u, v) \in [2, 10] \times [-1, 10]$$

and has an area of 536.7. Now we use our algorithm to find an open triangular mesh with the input polygon as boundary. The triangular mesh should contain a required number of triangles and have the minimal area. In this example, we let the number



(a) (b) (c)

(d) (e) (f)

Figure 3.6: Reconstructing the Helicoid surface from a polygon.
(a) the input boundary; (b) the output of initialization; (c) the output of Laplacian fairing; (d) the mean curvature of the mesh in (c); (e) the output of area minimizing; and (f) the mean curvature of the mesh in (e).

of triangles be 2050 and three tolerance $\epsilon_1, \epsilon_2, \epsilon_3$ be 0.01. The results are shown in Figure 3.6, where (b), (c) and (e) are the outputs of Step 1 (initialization), Step 2 (Laplacian fairing) and Step 3 (area minimizing), respectively. The areas of the triangular meshes in Figure 3.6(c) and (e) are 536.17 and 536.07, which are even smaller than the actual area of the Helicoid patch. This can be explained by the fact that our input polygon is only an approximation to the exact patch boundary. The maximum absolute values of mean curvature of the meshes in Figure 3.6(c) and (e) are 0.1 and 0.0007. Figure 3.6(d) and (f) are the mean curvature images of (c) and (e), in which color blue stands for low mean curvature and color red for high mean curvature. It is clearly seen from the curvature images that the mesh shown in Figure 3.6(c) is further improved by area minimizing which outputs Figure 3.6(e). Both the area and the mean curvature indicate that the triangular mesh in Figure 3.6(e) is a good solution.

The second example is to reconstruct the Catenoid surface from a given polygon as boundary. The polygon contains 156 points that were originally sampled from the

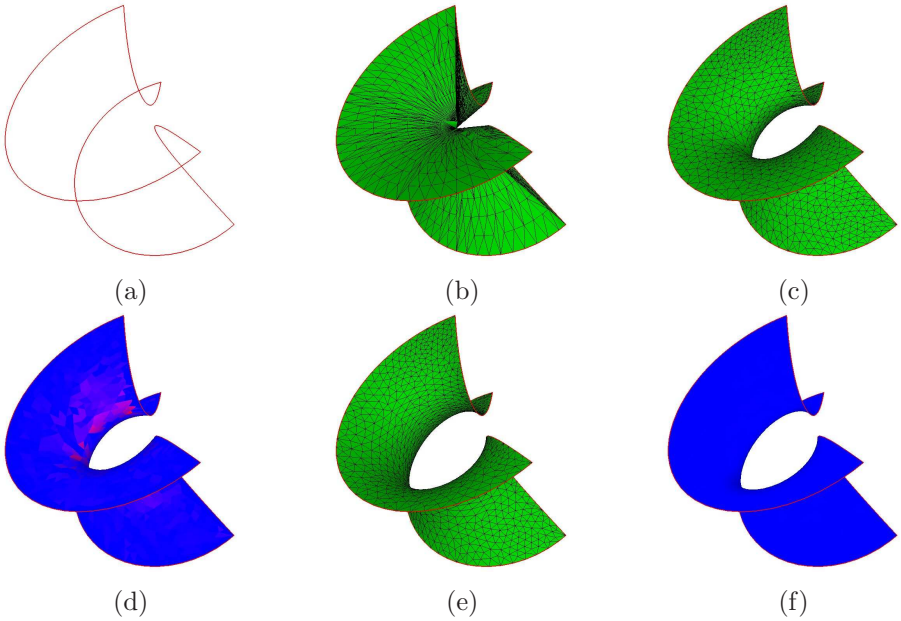


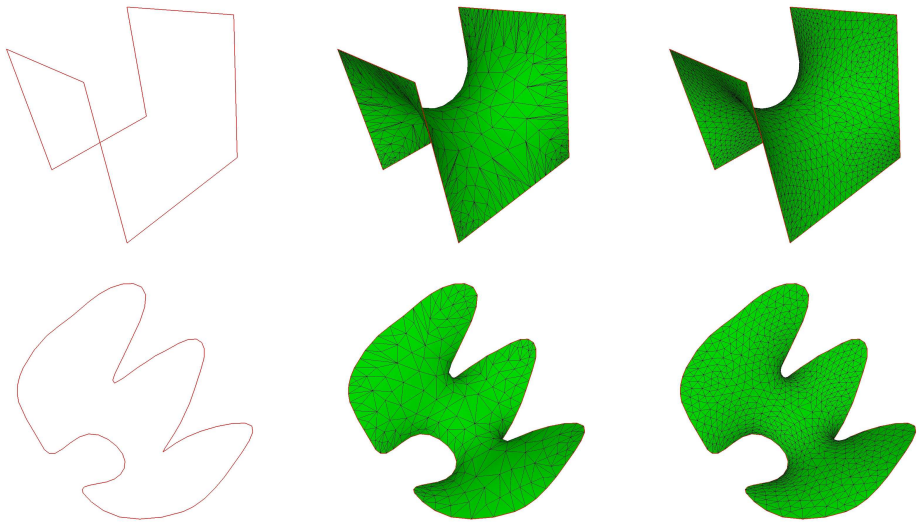
Figure 3.7: Reconstructing the Catenoid surface from a polygon.
(a) the input boundary; (b) the output of initialization; (c) the output of Laplacian fairing; (d) the mean curvature of the mesh in (c); (e) the output of area minimizing; and (f) the mean curvature of the mesh in (e).

boundary of a Catenoid patch defined by

$$x = \cosh v \cos u, y = \cosh v \sin u, z = v,\ (u, v) \in \left[-\frac{3\pi}{5}, \frac{3\pi}{5}\right] \times [-2, 2].$$

The area of the patch is 58.98. We apply our algorithm to the polygon with 2040 triangles and $\epsilon_1 = \epsilon_2 = \epsilon_3 = 0.01$. The results are shown in Figure 3.7, where (a) is the input polygon, (b) is the mesh outputted from Step 1, (c) is the result from Step 2, which has an area of 58.95 and the maximal mean curvature of 0.23, (e) is the final mesh outputted from Step 3, which has an area of 58.2 and the maximal mean curvature of 0.004, (d) and (f) are the mean curvature images of (c) and (e), respectively. These statistics demonstrate the effectiveness of the algorithm.

Two more examples are shown in Figure 3.8, where the left column shows two input polygons consisting of 160 and 97 vertices, respectively, the middle and right columns show the final minimal area meshes of different numbers of triangles. The statistics are shown in Table 3.1 and Table 3.2. It can been seen that the area of output triangular meshes decreases when the level of detail increases.



(a) (b) (c)

Figure 3.8: Construction with different levels of detail.
(a) the input polygons; (b) a minimal area mesh; and (c) another minimal area mesh.

Table 3.1: Statistics for the example shown in the top row of Figure 3.8.

	resulting mesh on the middle	resulting mesh on the right
number of triangles	800	2400
area	9.9	9.87
maximal mean curvature	0.004	0.001

3.5 Extensions and applications

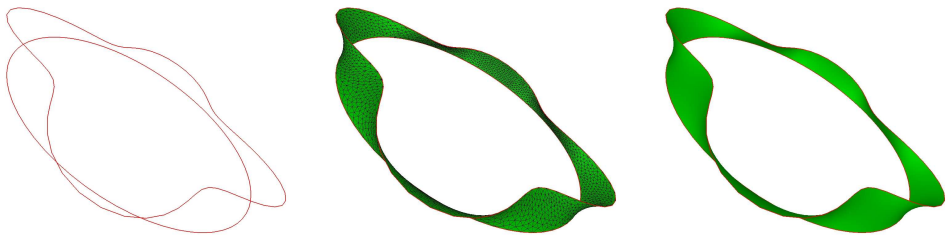
Although the chapter focuses on the construction of triangular meshes from a single contour, it is possible to extend the idea and the three processes to finding the minimal surfaces from multi-contours or with other constraints.

Figure 3.9 shows one example, where the input consists of two contours and the minimal surface interpolating the contours is constructed using our algorithm. Here we only need one extra process. That is to generate an initial triangular mesh from the given contours. Figure 3.10 gives another example with two contours. Different output surfaces after each step are viewed at different angle to check the differences.

Figure 3.11 gives an example with three contours. Table 3.3 lists the statistics for

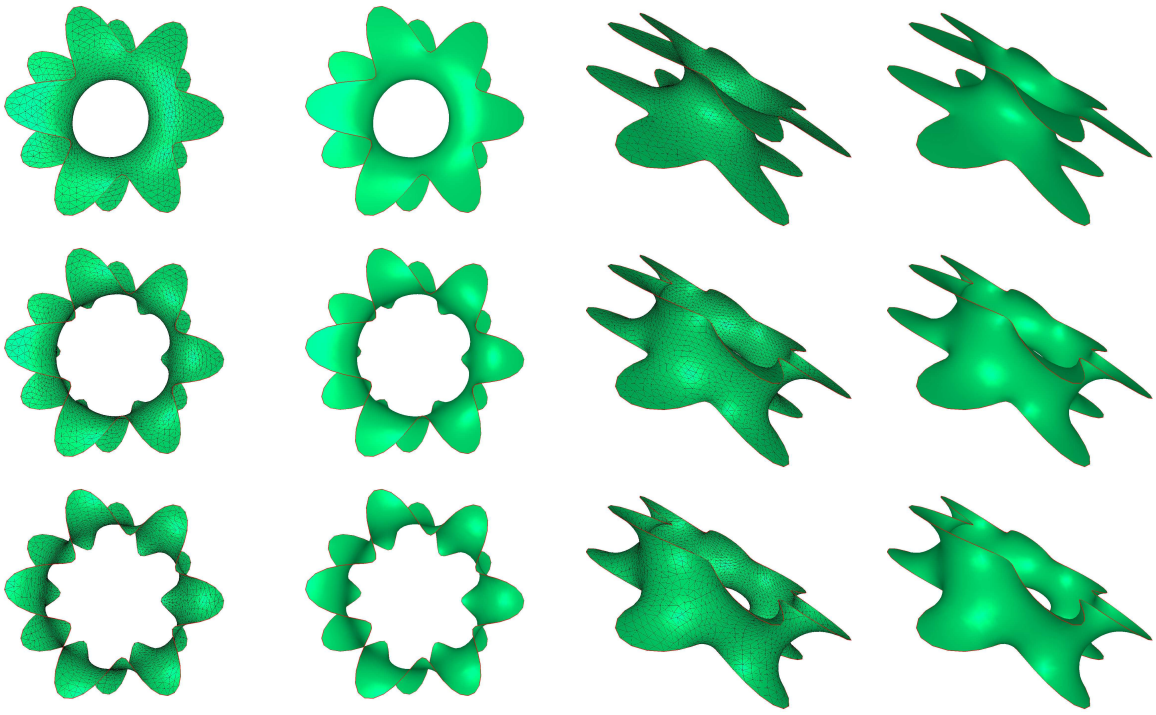
Table 3.2: Statistics for the example shown in the bottom row of Figure 3.8.

	resulting mesh on the middle	resulting mesh on the right
number of triangles	485	1455
area	1191.3	1187.8
maximal mean curvature	4.8×10^{-5}	1.1×10^{-5}



(a) (b) (c)

Figure 3.9: Minimal area mesh with two contours.
(a) two input contours; (b) the resulting mesh; (c) the shading image.



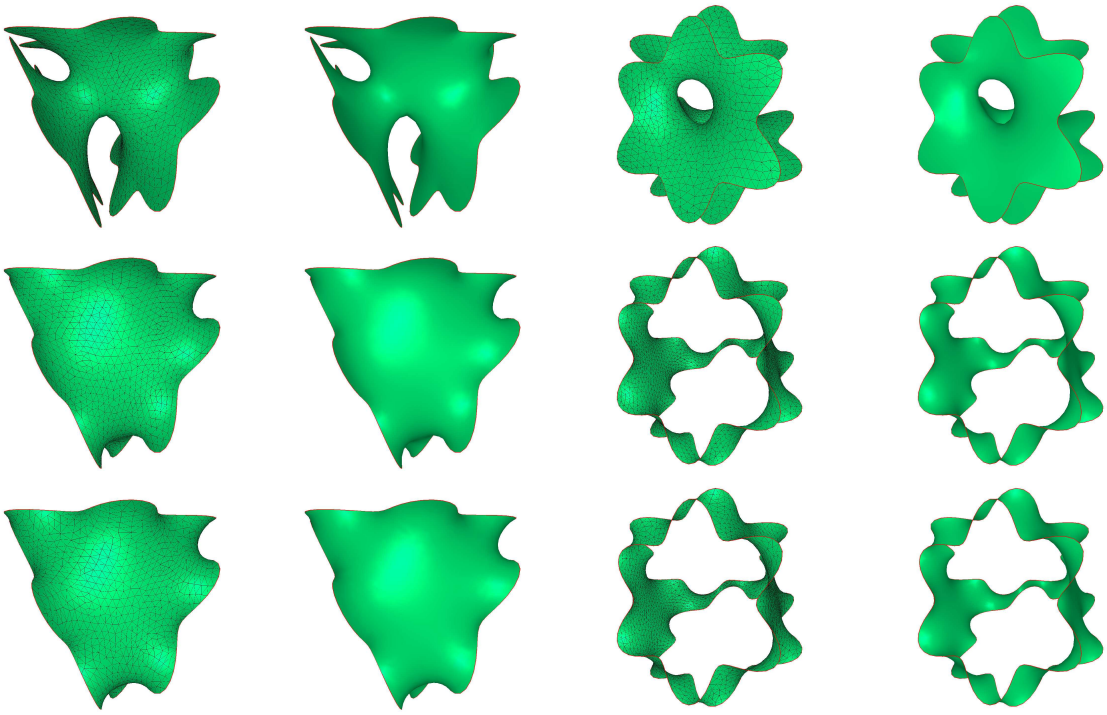
(a) (b) (c) (d)

Figure 3.10: Different views of the algorithm results with two contours.
the first row lists the results after Step 2; the second row provides the results after Step 3; results after Step 4 are given in the third row. (a) the resulting mesh; (b) the shading image; (c) the resulting mesh from another view; (d) the shading image from another view.

the above two examples. In the two examples (Figure 3.10 and Figure 3.11), a curve (piecewise line segments) is created by connecting the contours one by one to form a single closed contour. Then a mesh is initialized after Step 1 of the Algorithm 3.1. After Step 2 of the Algorithm 3.1, the area of the mesh is not small enough. After Step 3 of the Algorithm 3.1, the area of the mesh is smaller but the maximal mean curvature may increase or decrease. The maximal mean curvature after Step 3 is smaller than the maximal mean curvature after Step 2 in Figure 3.10 (1.013 vs 1.286), while the maximal mean curvature after Step 3 is bigger than the maximal mean curvature

after Step 2 in Figure 3.11 (5.012 vs 3.534). However, both the area and the maximal mean curvature will decrease after Step 4.

If contours are far away from each other, the minimal area mesh is degenerated to special cases, as shown in Figure 3.12.



(a) (b) (c) (d)

Figure 3.11: Different views of the algorithm results with three contours. the first row lists the results after Step 2; the second row provides the results after Step 3; results after Step 4 are given in the third row. (a) the resulting mesh; (b) the shading image; (c) the resulting mesh from another view; (d) the shading image from another view.

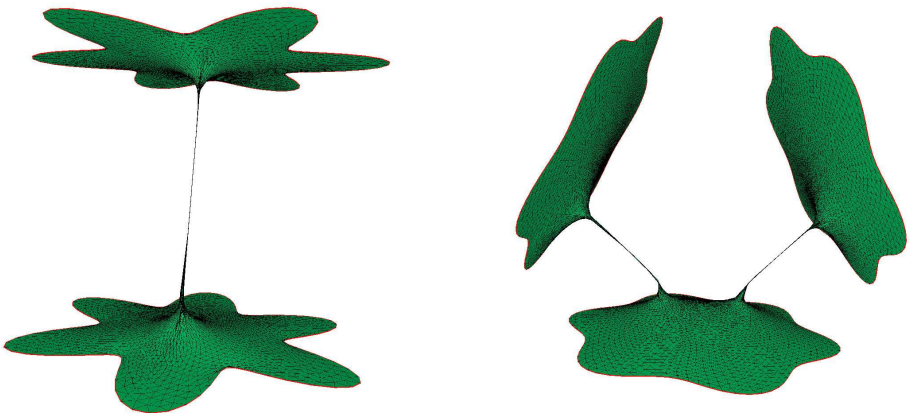


Figure 3.12: Results while contours are far from each other.

Table 3.3: Statistics for the examples.

	triangle number	after Step 2		after Step 3		after Step 4	
		Area	Maximal mean curvature	Area	Maximal mean curvature	Area	Maximal mean curvature
Figure 3.10	2102	6.223	1.286	5.195	1.013	4.99	0.049
Figure 3.11	3214	9.903	3.534	6.506	5.012	6.414	0.015

3.6 Summary

In this chapter, we have proposed an algorithm to construct triangular meshes of minimal area from all the possible triangular meshes with the prescribed boundary and number of triangles. The core techniques of the algorithm are three processes: area minimizing, Laplacian fairing, and edge swapping. They are combined to provide an automatic approach. The algorithm has been shown by the examples to be reliable and effective. On the other hand, since these three processes can be done very fast, they can be used in interactive environments. Especially in digital geometry modeling, after users sketch the boundary and specify the level of detail, the three processes can then be interactively performed in various orders to achieve users' specific requirements.

Chapter 4

Composite Surfaces[3]

An explicit formula is given to derive a triangular sub-patch **S** from a triangular Bézier surface **T**. Based on de Casteljau recursions and shifting operators, the control points for the triangular sub-patch **S** are the convex combinations of the construction points, which are the convex combinations of control points of the triangular Bézier surface **T**. Further more, the control points for **S** can be formulated as the linear combinations of the power points, which can be calculated using pyramid algorithms. Although the formulae are complex, a robust algorithm is provided to calculate the control points of the sub-patch **S**.

4.1 Background

Many applications in CAD/CAM or computer graphics industry require creating geometric entities such as curves or patches on surfaces. Isoparametric curves on a surface are easy to derive, however in many cases, the curves needs to be in a general position such as the intersection curve of two surfaces, the boundary for surface trimming. DeRose [96] examined the curves on triangular Bézier surfaces via functional composition. Jüttler and Wang [97] analyzed the curves on a sphere. Both approaches generate curves on surfaces by parameter space representation.

Beside curves, sub-patches on surfaces are also important. Two types of surfaces

[3]The following paper submitted for possible publication is based on the results of this chapter: Chen,W.Y., Yu,R.D., Zheng,J.M., Cai,Y.Y., and Au,C. **Triangular Bézier sub-surfaces on a triangular Bézier surface**, *Journal of Computational and Applied Mathematics*.

are wildly used: triangular Bernstein-Bézier surface (TB or TBB surface) and tensor product Bézier surface (PB or TPB surface). For instance, the subdivision of a Bézier surface [98] falls into this category and so are the conversions between TB surfaces and PB surface. Brueckner [99] represented a TB surface as a trimmed PB surface. Waggenspack and Anderson [100] transformed a PB surface to a TB representation. Jie [101] extended the equations to rational cases. However, in most cases, the subpatches do not have isoparametric boundary curves. For example, the explicit formula of Goldman and Filip [102] converted a PB surface of degree (m, n) into two TB surfaces of degree $m + n$. Hu [103] developed a method to divide a TB surface into three PB surfaces. In another way, Sheng and Hirsch [104] divided a trimmed surface into many TB surfaces.

As pointed out by DeRose [96], subdivision, reparametrization and continuity constructions are possible applications of composition. Lasser [105] extracted a PB surface from a TB surface using de Casteljau algorithm. Feng and Peng [106] considered a simpler case using shifting operator to derive the composition of a linear triangle with a TB surface.

An explicit formula and an algorithm for deriving a TB patch from a TB surface using the shifting operators and the de Casteljau algorithm are presented. The geometric algorithm is analyzed and some examples are provided.

4.2 Preliminaries and notations

A triangular Bernstein-Bézier surface [107] $\mathbf{T}(u, v, w)$ denoted as TB surface, of degree n can be defined by

$$\mathbf{T}(u, v, w) = \sum_{i+j+k=n} B^n_{ijk}(u, v, w)\mathbf{T}_{ijk}, \quad u, v, w \in D_T. \tag{Eq.4.1}$$

where $\mathbf{T}_{ijk} \in R^3$ are control points and $B^n_{ijk}(u, v, w)$ are Bernstein polynomial basis defined as

$$B^n_{ijk}(u, v, w) = \frac{n!}{i!j!k!}u^i v^j w^k, \quad i + j + k = n, \quad u, v, w \geq 0, \quad u + v + w = 1.$$

and

$$D_T = \{\,(u,v,w)|\; u+v+w=1, u,v,w \in [0,1]\}$$

is a triangle domain. A point in the domain is called a *parameter point*. A domain surface $\mathbf{P}(u,v,w)$ is a sub-surface of D_T.

$$\mathbf{P}(u,v,w) = \sum_{i+j+k=n} B^n_{ijk}(u,v,w)\mathbf{P}_{ijk}, \quad u,v,w \geq 0 \quad u+v+w=1. \qquad \text{(Eq.4.2)}$$

where $\mathbf{P}_{ijk}$ are parameter points, and the index (i,j,k) is called a *parameter index*. Several parameter indices are grouped together to form a *parameter index vector* $\mathbf{\Gamma}^m_n$ of degree m and size n:

$$\begin{aligned}
&\mathbf{\Gamma}^m_n = (\mathbf{I}^m_n, \mathbf{J}^m_n, \mathbf{K}^m_n)\,, \\
&\mathbf{I}^m_n = (I_1, \cdots, I_n), \mathbf{J}^m_n = (J_1, \cdots, J_n), \mathbf{K}^m_n = (K_1, \cdots, K_n), \\
&I_l + J_l + K_l = m, \qquad I_l, J_l, K_l \in \{0, \cdots, m\}.
\end{aligned} \qquad \text{(Eq.4.3)}$$

$\mathbf{\Gamma}^m_n$ contains n parameter indices. The norms for *parameter index vector* are

$$|\mathbf{I}^m_n| = \sum_{l=1}^{n} I_l, |\mathbf{J}^m_n| = \sum_{l=1}^{n} J_l, |\mathbf{K}^m_n| = \sum_{l=1}^{n} K_l, |\mathbf{\Gamma}^m_n| = |\mathbf{I}^m_n| + |\mathbf{J}^m_n| + |\mathbf{K}^m_n| = mn.$$

Based on Eq.4.3, a parameter point vector $\mathbf{P}^n_{\mathbf{\Gamma}^m_n} = \mathbf{P}^n_{\mathbf{I}^m_n\mathbf{J}^m_n\mathbf{K}^m_n}$ and its sub-vector $\mathbf{P}^s_{\mathbf{\Gamma}^m_n} = \mathbf{P}^s_{\mathbf{I}^m_n\mathbf{J}^m_n\mathbf{K}^m_n}$ can be obtained satisfying

$$\begin{aligned}
&\mathbf{P}^n_{\mathbf{\Gamma}^m_n} = \mathbf{P}^n_{\mathbf{I}^m_n\mathbf{J}^m_n\mathbf{K}^m_n} = [(u_{I_1J_1K_1}, v_{I_1J_1K_1}, w_{I_1J_1K_1}), \cdots, (u_{I_nJ_nK_n}, v_{I_nJ_nK_n}, w_{I_nJ_nK_n})] \\
&\mathbf{P}^s_{\mathbf{\Gamma}^m_n} = \mathbf{P}^s_{\mathbf{I}^m_n\mathbf{J}^m_n\mathbf{K}^m_n} = [(u_{I_1J_1K_1}, v_{I_1J_1K_1}, w_{I_1J_1K_1}),\ \cdots, (u_{I_sJ_sK_s}, v_{I_sJ_sK_s}, w_{I_sJ_sK_s})]\,, \\
&s = 0, 1, \cdots, n, \\
&(u_{I_iJ_iK_i}, v_{I_iJ_iK_i}, w_{I_iJ_iK_i}) \in D_T, \quad i = 0, 1, \cdots, n.
\end{aligned}$$

Hence, $\mathbf{P}^s_{\mathbf{\Gamma}^m_n}$ contains s parameter points. The *parameter index vector* can be used for the production of n Bernstein polynomials of degree m:

$$\prod_{l=1}^{n} B^m_{I_lJ_lK_l}(u,v,w) = C^{mn}_{\mathbf{I}^m_n\mathbf{J}^m_n\mathbf{K}^m_n} B^{mn}_{|\mathbf{I}^m_n|,|\mathbf{J}^m_n|,|\mathbf{K}^m_n|}(u,v,w). \qquad \text{(Eq.4.4)}$$

where

$$C^{mn}_{\mathbf{\Gamma}^m_n} = C^{mn}_{\mathbf{I}^m_n\mathbf{J}^m_n\mathbf{K}^m_n} = \left(\prod_{l=1}^{n}\frac{m!}{I_l!J_l!K_l!}\right)\Big/\left(\frac{(mn)!}{|\mathbf{I}^m_n|!\,|\mathbf{J}^m_n|!\,|\mathbf{K}^m_n|!}\right).$$

Lemma 4.8 Suppose $R+S+T=mn$, then

$$\sum_{|\mathbf{I}^m_n|=R,|\mathbf{J}^m_n|=S,|\mathbf{K}^m_n|=T} C^{mn}_{\mathbf{I}^m_n\mathbf{J}^m_n\mathbf{K}^m_n} = 1. \qquad \text{(Eq.4.5)}$$

Proof: Suppose we have mn different balls and put them into 3 different boxes B_1, B_2, B_3. There are $\frac{(mn)!}{R!S!T!}$ different cases for B_1 containing R balls, B_2 containing S balls, B_3 containing T balls with $R+S+T=mn$.

In another way, we can divide the mn balls into n groups $G_1, ..., G_n$ while each group G_l contains m balls. The box B_1 contains I_l balls from G_l, the box B_2 contains J_l balls from G_l, and the box B_3 contains K_l balls from G_l. Then the numbers of balls in B_1, B_2, B_3 are $|\mathbf{I}^m_n| = R, |\mathbf{J}^m_n| = S, |\mathbf{K}^m_n| = T$ where $\mathbf{I}^m_n = (I_1, \cdots, I_n), \mathbf{J}^m_n = (J_1, \cdots, J_n), \mathbf{K}^m_n = (K_1, \cdots, K_n), I_l + J_l + K_l = m$.

Distribute m balls in G_l into the three boxes, there are $\frac{m!}{I_l!J_l!K_l!}$ different cases. Therefore, for a given $(\mathbf{I}^m_n, \mathbf{J}^m_n, \mathbf{K}^m_n)$, there are $\prod_{l=1}^{n}\frac{m!}{I_l!J_l!K_l!}$ different cases to put mn different balls into B_1, B_2, B_3. Hence, we get

$$\sum_{|\mathbf{I}^m_n|=R,|\mathbf{J}^m_n|=S,|\mathbf{K}^m_n|=T}\left(\prod_{l=1}^{n}\frac{m!}{I_l!J_l!K_l!}\right) = \frac{(mn)!}{R!S!T!}.$$

This is equivalent to Eq.4.5.

Eq.4.6 is employed to product n objects

$$\left(\sum_{i+j+k=m} x_{ijk}\right)^n = \sum_{R+S+T=mn}\left(\sum_{|\mathbf{I}^m_n|=R,|\mathbf{J}^m_n|=S,|\mathbf{K}^m_n|=T}\prod_{l=1}^{n} x_{I_lJ_lK_l}\right) \qquad \text{(Eq.4.6)}$$

Set $\mathbf{T}_{ij} = \mathbf{T}_{i,j,n-i-j}$, then, Eq.4.1 equals

$$\mathbf{T}(u,v) = \sum_{i+j+k=0}^{n} T^n_{ijk}(u,v,1-u-v)\mathbf{T}_{ij}, \quad u,v \geq 0, \quad u+v \leq 1. \qquad \text{(Eq.4.7)}$$

The TB surface can be rewritten using shifting operators [108]

$$\mathbf{T}(u,v,w) = (uE_1 + vE_2 + wE_3)^n \mathbf{T}_{000} \tag{Eq.4.8}$$

where the shifting operators are

$$E_1\mathbf{T}_{ijk} = \mathbf{T}_{i+1,j,k}, E_2\mathbf{T}_{ijk} = \mathbf{T}_{i,j+1,k}, E_3\mathbf{T}_{ijk} = \mathbf{T}_{i,j,k+1}.$$

With $\mathbf{T}^0_{ijk} = \mathbf{T}_{ijk}$, given a parameter point vector $\mathbf{P}^n_{\mathbf{I}^m_n\mathbf{J}^m_n\mathbf{K}^m_n}$ for a TB surface, the de Casteljau algorithm [109] yields

$$\begin{aligned}
&\mathbf{T}^n_{ijk}(\mathbf{P}^n_{\mathbf{I}^m_n\mathbf{J}^m_n\mathbf{K}^m_n}) \\
&= u_{I_nJ_nK_n}\mathbf{T}^{n-1}_{i+1,j,k}(\mathbf{P}^{n-1}_{\mathbf{I}^m_n\mathbf{J}^m_n\mathbf{K}^m_n}) + v_{I_nJ_nK_n}\mathbf{T}^{n-1}_{i,j+1,k}(\mathbf{P}^{n-1}_{\mathbf{I}^m_n\mathbf{J}^m_n\mathbf{K}^m_n}) \\
&\quad + w_{I_nJ_nK_n}\mathbf{T}^{n-1}_{i,j,k+1}(\mathbf{P}^{n-1}_{\mathbf{I}^m_n\mathbf{J}^m_n\mathbf{K}^m_n}).
\end{aligned} \tag{Eq.4.9}$$

Alternatively, Eq.4.9 can be rewritten using shifting operators

$$\begin{aligned}
&\mathbf{T}^n_{ijk}(\mathbf{P}^n_{\mathbf{I}^m_n\mathbf{J}^m_n\mathbf{K}^m_n}) \\
&= \left(u_{I_nJ_nK_n}E_1 + v_{I_nJ_nK_n}E_2 + w_{I_nJ_nK_n}E_3\right)\mathbf{T}^{n-1}_{ijk}(\mathbf{P}^{n-1}_{\mathbf{I}^m_n\mathbf{J}^m_n\mathbf{K}^m_n}) \\
&= \prod_{l=1}^{n}(u_{I_lJ_lK_l}E_1 + v_{I_lJ_lK_l}E_2 + w_{I_lJ_lK_l}E_3)\mathrm{T}_{ijk}.
\end{aligned} \tag{Eq.4.10}$$

Finally, suppose $\mathbf{M}^s_d \subset Z^d$ is a power index set

$$\mathbf{M}^s_d = \left\{ \mathbf{M} | \, \mathbf{M} = (M_1, M_2, \cdots, M_d), \sum_{i=1}^{d} M_i = s, M_i = 0, 1, \cdots, s \right\}.$$

If $\mathbf{M} \in \mathbf{M}^s_d \subset Z^d$, $\mathbf{M}$ is a $1 \times d$ vector and the summation of its coordinates is s. From Lemma 4.9, there are $\begin{pmatrix} d-1+s \\ s \end{pmatrix}$ elements in $\mathbf{M}^s_d$.

Lemma 4.9 The number of different choices of the d variables $(N_1, ..., N_d)$ satisfying $\sum_{i=1}^{d} N_i = s$ is

$$\begin{pmatrix} s+d-1 \\ d-1 \end{pmatrix} = \begin{pmatrix} s+d-1 \\ s \end{pmatrix}.$$

Proof: This equals putting s balls into d boxes. List the s balls in a line and use $d-1$ bars to separate these balls. Firstly, there are $s+d-1$ positions for balls and bars. Select $d-1$ positions for the bars. Then put all balls to the left positions. Finally, the s balls are separated into d sets. Hence, we have $\begin{pmatrix} s+d-1 \\ d-1 \end{pmatrix}$ choices.

4.3 Triangle sub-patch from a triangle surface

In this section, an approach to extracting a triangle sub-patch from a triangle surface is presented. If a TB surface $\mathbf{T}(x,y,z)$ is defined over a triangular domain D_T, by assigning an area on the domain, a triangle sub-patch exists on the TB surface. The sub-patch is trimmed from the TB surface. We would like to represent the sub-patch as a new TB surface.

4.3.1 Domain surface

The domain surface is an area of the domain of the TB surface $\mathbf{T}(x,y,z)$. On the domain D_T, three boundary curves C_1, C_2, C_3 forms a closed sub-domain D_{C_1,C_2,C_3}. The surface D_{C_1,C_2,C_3} is a domain surface (Figure 4.1(a)).

Suppose the three boundary curves $C_1(u), C_2(u), C_3(u)$ are of degree n_1, n_2, n_3, respectively. Set $m = \max(n_1, n_2, n_3)$, we can elevate the degrees of $C_1(u), C_2(u), C_3(u)$ all to m. Suppose the control point of $C_i(u)\,(i=1,2,3)$ are $\mathbf{P}_{i,j}, j=0,...,m$ and the corresponding influence points are

$$\mathbf{Q}_{i,j} = C_i(\frac{j}{m}), \qquad \text{(Eq.4.11)}$$

Arrange the control point and the influence points as (Figure 4.1(b))

$$\mathbf{P}_{ijk}, \mathbf{Q}_{ijk} \in D_T, \min(i,j,k) = 0. \qquad \text{(Eq.4.12)}$$

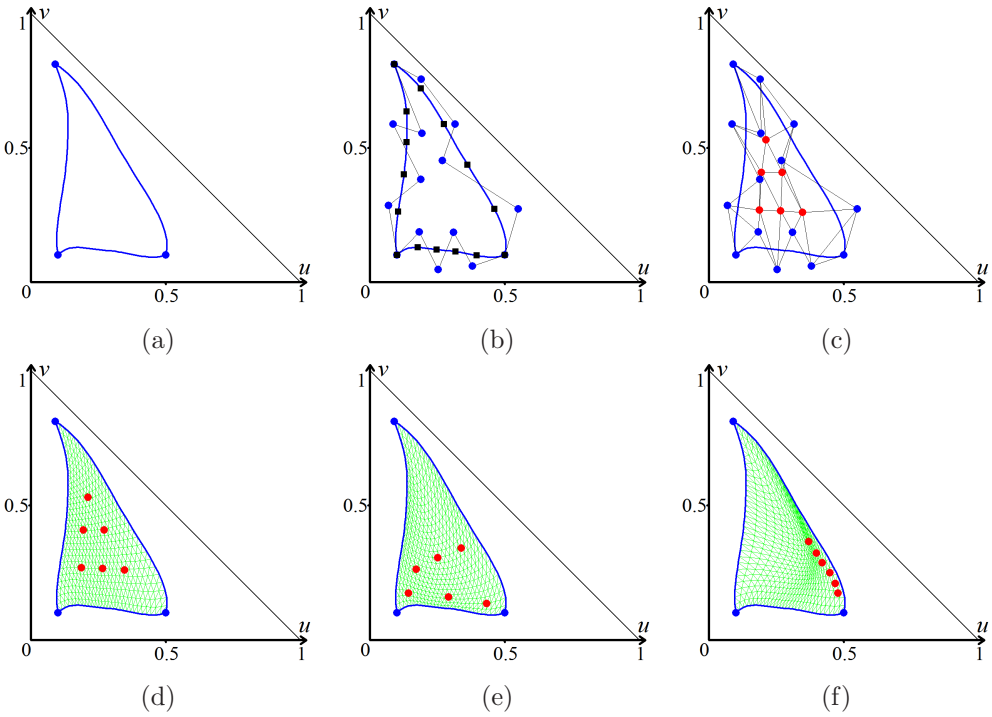


Figure 4.1: Domain surface.

(a) three boundary curves form a closed sub-domain; (b) the control point (blue dots) and the influence points (black squares); (c) the interior control point (red dots); and (d-f) different parameter curves by different interior control point.

Select the interior control point $\mathbf{P}_{ijk}, \min(i,j,k) \neq 0$ as (Figure 4.1(c))

$$\begin{aligned}\mathbf{P}_{ijk} = &\tfrac{1}{3}\left(\tfrac{k}{m-i}\mathbf{Q}_{i,0,m-i} + \tfrac{j}{m-i}\mathbf{Q}_{i,m-i,0}\right) + \tfrac{1}{3}\left(\tfrac{k}{m-j}\mathbf{Q}_{0,j,m-j} + \tfrac{i}{m-j}\mathbf{Q}_{m-j,j,0}\right) \\ &+\tfrac{1}{3}\left(\tfrac{j}{m-k}\mathbf{Q}_{0,m-k,k} + \tfrac{i}{m-k}\mathbf{Q}_{m-k,0,k}\right),\end{aligned} \quad \text{(Eq.4.13)}$$

Then the domain surface can be modeled as a TB surface (Figure 4.1(d))

$$\mathbf{P}(u,v,w) = \sum_{i+j+k=n} B^n_{ijk}(u,v,w)\mathbf{P}_{ijk}, \quad u,v,w \geq 0 \quad u+v+w = 1.$$

The interior control point are the linear combinations of the influence points. A combination involves the six points along the u, v, w directions (Figure 4.2). And, the number of interior control point is $(m-1)(m-2)/2$. For example, if $m = 3$, the only interior control point is

$$\mathbf{P}_{111} = (\mathbf{Q}_{102} + \mathbf{Q}_{120} + \mathbf{Q}_{012} + \mathbf{Q}_{210} + \mathbf{Q}_{201} + \mathbf{Q}_{021})/6.$$

But for $m = 1, 2$, no interior control point are involved.

The interior control point can be further modified if needed. Figure 4.1(e) and Figure 4.1(f) show two different choices of the interior control point. Different choice of the interior control point may lead to different parameter curves of the domain surface, hence affect the parameterization of the composite surface (To be illustrated later).

All the control point of the domain surface are derived from Eq.4.12 and Eq.4.13. They are named as parameter control point or parameter points.

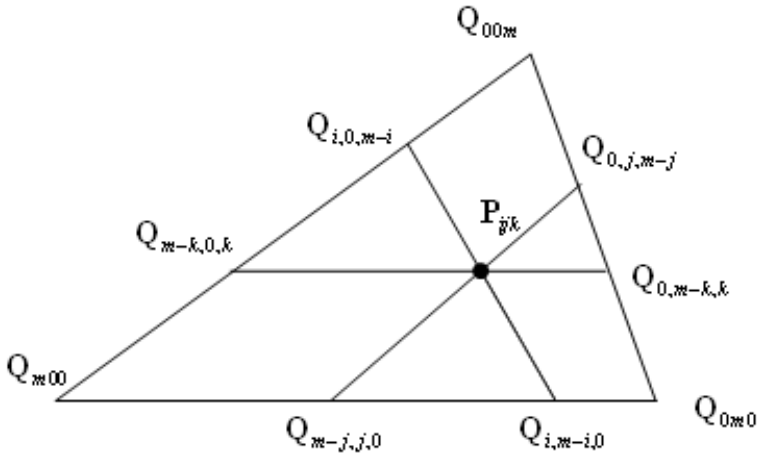


Figure 4.2: Construction of interior points.

4.3.1.1 Sub-patch via composition

The following theorem describes the composition of a TB surface $\mathbf{T}(x, y, z)$ and a domain surface. The surface $\mathbf{P}(u, v, w)$ is a sub-domain of the TB surface $\mathbf{T}(x, y, z)$. Therefore, the composition $\mathbf{S}(u, v, w) = \mathbf{T}(\mathbf{P}(u, v, w))$ is a sub-surface of the TB surface $\mathbf{T}(x, y, z)$ (Figure 4.3).

Theorem 4.1 (Composition of two TB surfaces). Suppose $\mathbf{T}(x, y, z)$ be a TB surface of degree n with control points $\mathbf{T}_{ijk} \in R^3, i + j + k = n$. And $\mathbf{P}(u, v, w)$ is a domain surface of degree m with parameter points

$$\mathbf{P}_{ijk} = (x_{ijk}, y_{ijk}, z_{ijk}), i + j + k = m, x_{ijk} + y_{ijk} + z_{ijk} = 1.$$

Then, the composition $\mathbf{S}(u, v, w) = \mathbf{T}(\mathbf{P}(u, v, w))$ is a TB surface of degree mn:

$$\mathbf{S}(u, v, w) = \sum_{R+S+T=mn} B^{mn}_{RST}(u, v, w)\mathbf{S}_{RST}, \qquad \text{(Eq.4.14)}$$

with control points

$$\mathbf{S}_{RST} = \sum_{|\mathbf{I}_n^m|=R,|\mathbf{J}_n^m|=S,|\mathbf{K}_n^m|=T} C^{mn}_{\mathbf{I}_n^m\mathbf{J}_n^m\mathbf{K}_n^m}\mathbf{S}^n_{\mathbf{I}_n^m\mathbf{J}_n^m\mathbf{K}_n^m}, \qquad \text{(Eq.4.15)}$$

where the construction point $\mathbf{S}^n_{\mathbf{I}_n^m\mathbf{J}_n^m\mathbf{K}_n^m} = \mathbf{T}^n_{000}(\mathbf{P}^n_{\mathbf{I}_n^m\mathbf{J}_n^m\mathbf{K}_n^m})$ is defined over the parameter vector

$$\mathbf{P}^n_{\mathbf{I}_n^m\mathbf{J}_n^m\mathbf{K}_n^m} = [\mathbf{P}_{I_lJ_lK_l} \mid \ l = 1, \cdots, n]\,. \qquad \text{(Eq.4.16)}$$

And $\mathbf{\Gamma}_n^m = (\mathbf{I}_n^m, \mathbf{J}_n^m, \mathbf{K}_n^m)$ is defined in Eq.4.3.

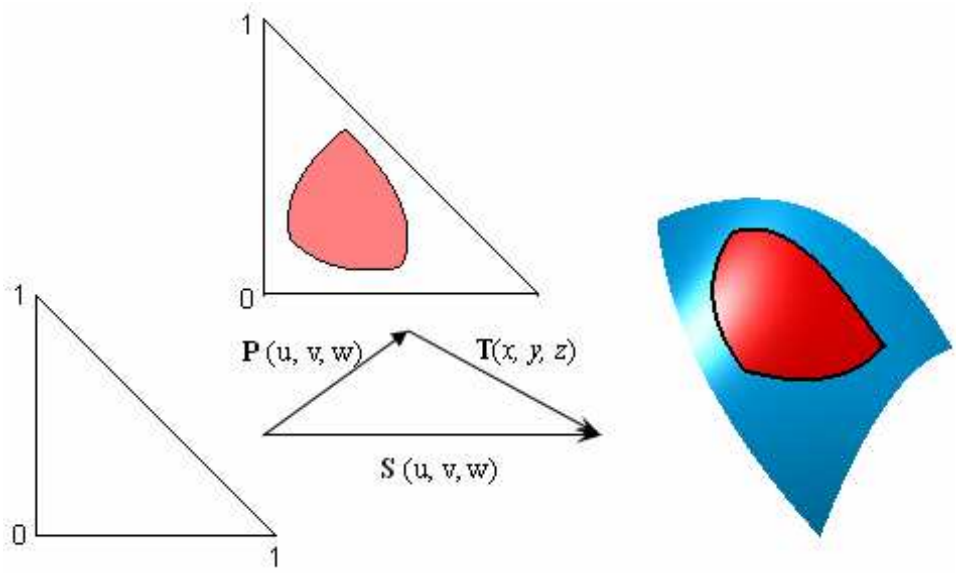


Figure 4.3: Composition of two TB surfaces.

Proof: Set $\mathbf{P}(u,v,w) = (x(u,v,w), y(u,v,w), z(u,v,w))$. Then

$$\begin{aligned} x(u,v,w) &= \sum_{i+j+k=m} B^m_{ijk}(u,v,w)x_{ijk}, \\ y(u,v,w) &= \sum_{i+j+k=m} B^m_{ijk}(u,v,w)y_{ijk}, \\ z(u,v,w) &= \sum_{i+j+k=m} B^m_{ijk}(u,v,w)z_{ijk}. \end{aligned} \qquad \text{(Eq.4.17)}$$

Following Eq.4.8, the TB surface $\mathbf{T}(x,y,z)$ can be represented as

$$\mathbf{T}(x,y,z) = (xE_1 + yE_2 + zE_3)^n\,\mathbf{T}_{000}. \qquad \text{(Eq.4.18)}$$

Therefore, applying Eq.4.17 and Eq.4.18 yields

$$\begin{aligned}
\mathbf{S}(u,v,w) &= \mathbf{T}(\mathbf{P}(u,v,w)) \\
&= \left((x(u,v,w), y(u,v,w), z(u,v,w)) \begin{pmatrix} E_1 \\ E_2 \\ E_3 \end{pmatrix} \right)^n \mathbf{T}_{000} \\
&= \left(\sum_{i+j+k=m} (x_{ijk}, y_{ijk}, z_{ijk})\, B^m_{ijk}(u,v,w) \begin{pmatrix} E_1 \\ E_2 \\ E_3 \end{pmatrix} \right)^n \mathbf{T}_{000} \\
&= \left(\sum_{i+j+k=m} B^m_{ijk}(u,v,w)\, (x_{ijk}E_1 + y_{ijk}E_2 + z_{ijk}E_3) \right)^n \mathbf{T}_{000}.
\end{aligned}$$

Applying Eq.4.6 gives

$$\begin{aligned}
&\mathbf{S}(u,v,w) \\
&= \sum_{R+S+T=mn} \left(\sum_{|\mathbf{I}^m_n|=R, |\mathbf{J}^m_n|=S, |\mathbf{K}^m_n|=T} \prod_{l=1}^{n} \left(B^m_{I_l J_l K_l}(u,v,w)\, (x_{I_l J_l K_l} E_1 + y_{I_l J_l K_l} E_2 + z_{I_l J_l K_l} E_3) \right) \right) \mathbf{T}_{000}.
\end{aligned}$$

With Eq.4.4 and Eq.4.10, the composition is

$$\begin{aligned}
&\mathbf{S}(u,v,w) \\
&= \sum_{R+S+T=mn} \left(\sum_{|\mathbf{I}^m_n|=R, |\mathbf{J}^m_n|=S, |\mathbf{K}^m_n|=T} \left(C^{mn}_{\mathbf{I}^m_n \mathbf{J}^m_n \mathbf{K}^m_n} B^{mn}_{RST}(u,v,w) \mathbf{T}^n_{ijk}(\mathbf{P}^n_{\mathbf{I}^m_n \mathbf{J}^m_n \mathbf{K}^m_n}) \right) \right) \\
&= \sum_{R+S+T=mn} \left(\sum_{|\mathbf{I}^m_n|=R, |\mathbf{J}^m_n|=S, |\mathbf{K}^m_n|=T} \left(C^{mn}_{\mathbf{IJK}} \cdot \mathbf{T}^n_{ijk}(\mathbf{P}^n_{\mathbf{I}^m_n \mathbf{J}^m_n \mathbf{K}^m_n}) \right) \right) B^{mn}_{RST}(u,v,w) \\
&= \sum_{R+S+T=mn} \mathbf{S}_{RST} B^{mn}_{RST}(u,v,w).
\end{aligned}$$

This completes the proof.

As shown in Eq.4.9, the construction point $\mathbf{S}^n_{\mathbf{I}^m_n \mathbf{J}^m_n \mathbf{K}^m_n}$ is a linear combination of the control points $\mathbf{T}_{ijk}$. From Eq.4.5, the control points $\mathbf{S}_{RST}$ in Eq.4.15 are the linear combinations of $\mathbf{S}^n_{\mathbf{I}^m_n \mathbf{J}^m_n \mathbf{K}^m_n}$. Therefore, $\mathbf{S}_{RST}$ are the linear combinations of the original control points. As a result, the construction points are computed and which will be used to derive the control point $\mathbf{S}_{RST}$.

4.3.2 Number of different construction points

There are $(m+1)(m+2)/2$ parameter points from the domain surface. Hence, for every parameter point $\mathbf{P}_{I_lJ_lK_l}$, there are $(m+1)(m+2)/2$ different choices. Therefore, for every parameter point vector $\mathbf{P}^n_{\mathbf{I}^m_n\mathbf{J}^m_n\mathbf{K}^m_n}$ consisting of n parameter points, the total number of choices is $[(m+1)(m+2)/2]^n$ which implies $[(m+1)(m+2)/2]^n$ cases of construction points $\mathbf{S}^n_{\mathbf{I}^m_n\mathbf{J}^m_n\mathbf{K}^m_n} = \mathbf{T}^n_{000}(\mathbf{P}^n_{\mathbf{I}^m_n\mathbf{J}^m_n\mathbf{K}^m_n})$.

Define a *power index set* $\mathbf{B}^m_n = \mathbf{M}^n_{(m+1)(m+2)/2} \subset Z^{(m+1)(m+2)/2}$ as

$$\mathbf{B}^m_n = \left\{ (\beta_{0,0,m}, \beta_{0,1,m-1}, \beta_{1,0,m-1}, \cdots, \beta_{0,m,0}, \beta_{1,m-1,0}, \cdots, \beta_{m,0,0}) | \sum_{i+j+k=m} \beta_{ijk} = n, \beta_{ijk} > 0 \right\}.$$

Based on the $(m+1)(m+2)/2$ parameter points $\mathbf{P}_{ijk} = (x_{ijk}, y_{ijk}, z_{ijk})$ from the domain surface $\mathbf{P}(u,v,w)$, the *power point set* $\mathbf{Q}^m_n(\mathbf{P})$ is defined as

$$\mathbf{Q}^m_n(\mathbf{P}) = \left\{ \mathbf{Q}_{\mathbf{B}} \middle| \mathbf{Q}_{\mathbf{B}} = \prod_{i+j+k=m} (x_{ijk}E_1 + y_{ijk}E_2 + z_{ijk}E_3)^{\beta_{ijk}} \mathbf{T}_{000}, \ \mathbf{B} \in \mathbf{B}^m_n \right\}. \quad \text{(Eq.4.19)}$$

Each element in *power point set* $\mathbf{Q}^m_n(\mathbf{P})$ is a power point.

Lemma 4.10 A construction point is a power point: $\mathbf{S}^n_{\mathbf{I}^m_n\mathbf{J}^m_n\mathbf{K}^m_n} \in \mathbf{Q}^m_n(\mathbf{P})$.

Proof: From each construction point $\mathbf{S}^n_{\mathbf{I}^m_n\mathbf{J}^m_n\mathbf{K}^m_n} = \mathbf{T}^n_{000}(\mathbf{P}^n_{\mathbf{I}^m_n\mathbf{J}^m_n\mathbf{K}^m_n})$, suppose the parameter point $\mathbf{P}_{ijk} = (x_{ijk}, y_{ijk}, z_{ijk})$ repeats $\beta^{i,j,k}_{\mathbf{I}^m_n\mathbf{J}^m_n\mathbf{K}^m_n}$ times in the parameter point vector $\mathbf{P}^n_{\mathbf{I}^m_n\mathbf{J}^m_n\mathbf{K}^m_n}$, which means that the parameter index (i,j,k) repeats $\beta^{i,j,k}_{\mathbf{I}^m_n\mathbf{J}^m_n\mathbf{K}^m_n}$ times in the parameter index vector $\mathbf{I}^m_n\mathbf{J}^m_n\mathbf{K}^m_n$. Then, every construction point $\mathbf{S}^n_{\mathbf{I}^m_n\mathbf{J}^m_n\mathbf{K}^m_n}$ can be formulated as

$$\begin{aligned}
\mathbf{S}^n_{\mathbf{I}^m_n\mathbf{J}^m_n\mathbf{K}^m_n} &= \mathbf{T}^n_{000}(\mathbf{P}^n_{\mathbf{I}^m_n\mathbf{J}^m_n\mathbf{K}^m_n}) \\
&= \prod_{l=1}^{n} (x_{I_lJ_lK_l}E_1 + y_{I_lJ_lK_l}E_2 + z_{I_lJ_lK_l}E_3)\mathbf{T}_{000} \\
&= \prod_{i+j+k=m} (x_{ijk}E_1 + y_{ijk}E_2 + z_{ijk}E_3)^{\beta^{i,j,k}_{\mathbf{I}^m_n\mathbf{J}^m_n\mathbf{K}^m_n}} \mathbf{T}_{000} \\
&= \mathbf{Q}_{\mathbf{B}_{\mathbf{I}^m_n\mathbf{J}^m_n\mathbf{K}^m_n}},
\end{aligned} \quad \text{(Eq.4.20)}$$

where $\mathbf{B}_{\mathbf{I}^m_n\mathbf{J}^m_n\mathbf{K}^m_n} \in \mathbf{B}^m_n$ is a power index for $\mathbf{P}^n_{\mathbf{I}^m_n\mathbf{J}^m_n\mathbf{K}^m_n}$

$$\mathbf{B}_{\mathbf{I}^m_n\mathbf{J}^m_n\mathbf{K}^m_n} = \left(\beta^{0,0,m}_{\mathbf{I}^m_n\mathbf{J}^m_n\mathbf{K}^m_n}, \beta^{0,1,m-1}_{\mathbf{I}^m_n\mathbf{J}^m_n\mathbf{K}^m_n}, \beta^{1,0,m-1}_{\mathbf{I}^m_n\mathbf{J}^m_n\mathbf{K}^m_n}, \cdots, \beta^{0,m,0}_{\mathbf{I}^m_n\mathbf{J}^m_n\mathbf{K}^m_n}, \beta^{1,m-1,0}_{\mathbf{I}^m_n\mathbf{J}^m_n\mathbf{K}^m_n}, \cdots, \beta^{m,0,0}_{\mathbf{I}^m_n\mathbf{J}^m_n\mathbf{K}^m_n}\right).$$

This ends the proof.

From Lemma 4.10, the number of construction points is the number of points in $\mathbf{Q}_n^m(\mathbf{P})$, (i.e. the number of points in $\mathbf{B}_n^m$). Therefore the number of different construction points is

$$|\mathbf{Q}_n^m(\mathbf{P})| = |\mathbf{B}_n^m| = \left|\mathbf{M}^n_{(m+1)(m+2)/2}\right| = \begin{pmatrix} (m+1)(m+2)/2 - 1 + n \\ (m+1)(m+2)/2 - 1 \end{pmatrix}. \quad \text{(Eq.4.21)}$$

For example, if $m = 1$, the number of construction points is $(n+1)(n+2)/2$ which is the same as the number of control points $\mathbf{T}_{ijk}$.

4.3.3 Geometric algorithm for power points

A geometric algorithm is employed to derive the power point set $\mathbf{Q}_n^m(\mathbf{P})$. For a given $\mathbf{B} \in \mathbf{B}_n^m$, the power point $\mathbf{Q}_\mathbf{B} \in \mathbf{Q}_n^m(\mathbf{P})$ can be derived using the Pyramid algorithms [110]. Suppose

$$\mathbf{B} = (\beta_{0,0,m}, \beta_{0,1,m-1}, \beta_{1,0,m-1}, \cdots, \beta_{0,m,0}, \beta_{1,m-1,0}, \cdots, \beta_{m,0,0}). \quad \text{(Eq.4.22)}$$

The parameter index vector $\mathbf{I}_\mathbf{B}^n$ is defined by repeating (i,j,k) for β_{ijk} times

$$\mathbf{I}_\mathbf{B}^n = \left[\underbrace{(0,0,m),\cdots,(0,0,m)}_{\beta_{0,0,m}}, \underbrace{(0,1,m-1),\cdots,(0,1,m-1)}_{\beta_{0,1,m-1}}, \cdots, \underbrace{(0,m,0),\cdots,(m,0,0)}_{\beta_{m,0,0}}\right]. \quad \text{(Eq.4.23)}$$

The parameter point vector $\mathbf{P}_\mathbf{B}^n$ is defined by repeating $\mathbf{P}_{ijk}$ for β_{ijk} times

$$\begin{aligned} \mathbf{P}_\mathbf{B}^n &= \mathbf{P}_\mathbf{IJK}^n \\ &= \left[\underbrace{\mathbf{P}_{0,0,m},\cdots,\mathbf{P}_{0,0,m}}_{\beta_{0,0,m}}, \underbrace{\mathbf{P}_{0,1,m-1},\cdots,\mathbf{P}_{0,1,m-1}}_{\beta_{0,1,m-1}}, \cdots, \underbrace{\mathbf{P}_{0,m,0},\cdots,\mathbf{P}_{m,0,0}}_{\beta_{m,0,0}}\right]. \end{aligned} \quad \text{(Eq.4.24)}$$

Set the level n intermediate points as $\mathbf{R}_{ijk}^n = \mathbf{T}_{ijk}$. Suppose the α-th parameter point of $\mathbf{P}_\mathbf{B}^n$ is $\mathbf{P}_\alpha = (x_\alpha, y_\alpha, z_\alpha)$. Then there are $(\alpha+2)(\alpha+1)/2$ intermediate points $\mathbf{R}_{ijk}^\alpha$ at level α and $\mathbf{R}_{ijk}^\alpha$ are the linear combinations of $\mathbf{R}_{ijk}^{\alpha+1}$ as

$$\mathbf{R}_{ijk}^\alpha = x_{\alpha+1}\mathbf{R}_{i+1,j,k}^{\alpha+1} + y_{\alpha+1}\mathbf{R}_{i,j+1,k}^{\alpha+1} + z_{\alpha+1}\mathbf{R}_{i,j,k+1}^{\alpha+1}, i+j+k = \alpha, \alpha = 0, \cdots, n-1.$$

Then the 0 level intermediate point $\mathbf{R}^0_{000} = \mathbf{Q_B} \in \mathbf{Q}^m_n(\mathbf{P})$.

Figure 4.4 shows an example with $n = 3$. The parameter vector $\mathbf{P}^3_{\mathbf{B}}$ contains 3 parameter points $\mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3$. There are 10 intermediate points $\mathbf{R}^3_{ijk}$ in level 3, and 6 intermediate points $\mathbf{R}^2_{ijk}$ in level 2, and 3 intermediate points $\mathbf{R}^1_{ijk}$ in level 1, and the construction point $\mathbf{R}^0_{000}$ in level 0. So using Pyramid algorithms, all the power points $\mathbf{Q}^m_n(\mathbf{P})$ can be obtained easily.

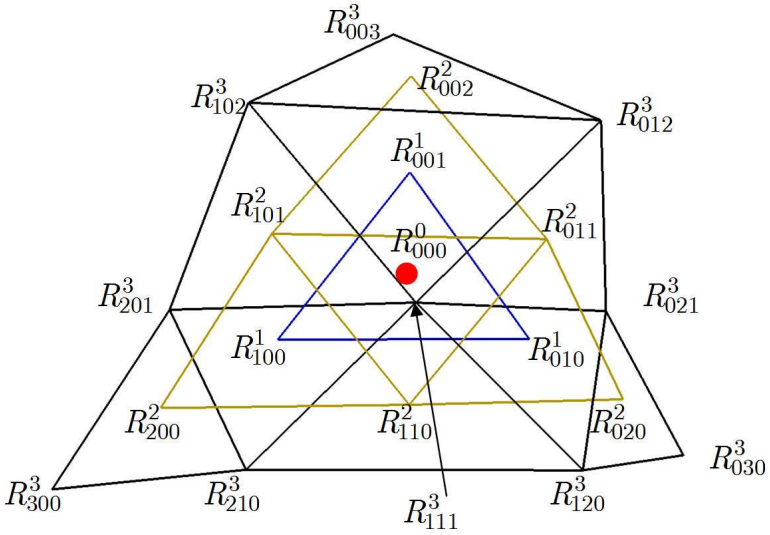


Figure 4.4: Pyramid algorithms for a construction point with $n = 3$.

4.3.4 Control points by power points

In Eq.4.15, different parameter point vector $\mathbf{P}^n_{\mathbf{I}^m_n\mathbf{J}^m_n\mathbf{K}^m_n} = [\mathbf{P}_{I_lJ_lK_l} \mid \ l = 1, \cdots, n]$ may lead to the same construction point $\mathbf{S}^n_{\mathbf{I}^m_n\mathbf{J}^m_n\mathbf{K}^m_n} = \mathbf{T}^n_{000}(\mathbf{P}^n_{\mathbf{I}^m_n\mathbf{J}^m_n\mathbf{K}^m_n})$. For example,

$$m = n = 2, R = \left|\mathbf{I}^2_2\right| = 0, S = \left|\mathbf{J}^2_2\right| = 1, T = \left|\mathbf{K}^2_2\right| = 3.$$

The construction points are

$$\bar{\mathbf{P}}^n_{\mathbf{I}^2_2\mathbf{J}^2_2\mathbf{K}^2_2} = [\mathbf{P}_{002}, \mathbf{P}_{011}], \tilde{\mathbf{P}}^n_{\mathbf{I}^2_2\mathbf{J}^2_2\mathbf{K}^2_2} = [\mathbf{P}_{011}, \mathbf{P}_{002}],$$

which are equal to the power point $\mathbf{P}^n_{\mathbf{B}} = \{\mathbf{P}_{002}, \mathbf{P}_{011}\}$ with $\mathbf{B} = (1, 1, 0, 0, 0, 0)$, hence $\mathbf{S}_{013}$ is defined by only one power point. Reformulating Eq.4.15 using power points

from $\mathbf{Q_B} \in \mathbf{Q}_n^m(\mathbf{P})$ in Eq.4.19 yields a control point $\mathbf{S}_{RST}$

$$\mathbf{S}_{RST} = \sum_{\mathbf{B}\in\mathbf{B}_n^m} F_{\mathbf{B}}^{RST} G_{\mathbf{B}}^{RST} Q_{\mathbf{B}}. \tag{Eq.4.25}$$

For a power vector $\mathbf{B}$ in Eq.4.22, $f_{\mathbf{B}}$ is defined

$$f_{\mathbf{B}} = \beta_{0,0,m}\begin{pmatrix}0\\0\\m\end{pmatrix} + \beta_{0,1,m-1}\begin{pmatrix}0\\1\\m-1\end{pmatrix} + \cdots + \beta_{0,m,0}\begin{pmatrix}0\\m\\0\end{pmatrix} + \cdots + \beta_{m,0,0}\begin{pmatrix}m\\0\\0\end{pmatrix}.$$

If a power point $\mathbf{Q_B}$, is used to construct $\mathbf{S}_{RST}$, then we have

$$f_{\mathbf{B}} - (R,S,T)^T = 0.$$

Hence, each power point is used for only one control point. By defining $F_{\mathbf{B}}^{RST}$ as

$$F_{\mathbf{B}}^{RST} = \begin{cases} 0, & f_{\mathbf{B}} - (R,S,T)^T \neq 0, \\ 1, & f_{\mathbf{B}} - (R,S,T)^T = 0. \end{cases} \tag{Eq.4.26}$$

The power points used for control points can be labeled. Given a power index $\mathbf{B}$, we get n parameter indices: the number of the index (i,j,k) is β_{ijk} with

$$i+j+k = m, \sum \beta_{ijk} = n.$$

From this power index, n parameter index (i,j,k) and $\frac{n!}{\prod(\beta_{ijk}!)}$ different parameter index vectors can be obtained completely. And from Eq.4.15 and Eq.4.23, for all these parameter index vectors $\mathbf{I}_n^m\mathbf{J}_n^m\mathbf{K}_n^m$, the coefficients of the construction point $\mathbf{S}^n_{\mathbf{I}_n^m\mathbf{J}_n^m\mathbf{K}_n^m}$ are $C^{mn}_{\mathbf{I}_n^m\mathbf{J}_n^m\mathbf{K}_n^m}$, which are equal to

$$C^{mn}_{\mathbf{I}_n^m\mathbf{J}_n^m\mathbf{K}_n^m} = \frac{\prod\limits_{i+j+k=m}\left(\prod\limits_{l=1}^{\beta_{ijk}}\left(\frac{m!}{i!j!k!}\right)\right)}{\frac{(mn)!}{R!S!T!}}.$$

As a result, for control point $\mathbf{S}_{RST}$, the coefficient of the power point $\mathbf{Q_B}$ is

$$G_{\mathbf{B}}^{RST} = \frac{\prod\limits_{i+j+k=m} \left(\prod\limits_{l=1}^{\beta_{ijk}} \left(\frac{m!}{i!j!k!} \right) \right)}{\frac{(mn)!}{R!S!T!}} \frac{n!}{\prod (\beta_{ijk}!)}. \qquad \text{(Eq.4.27)}$$

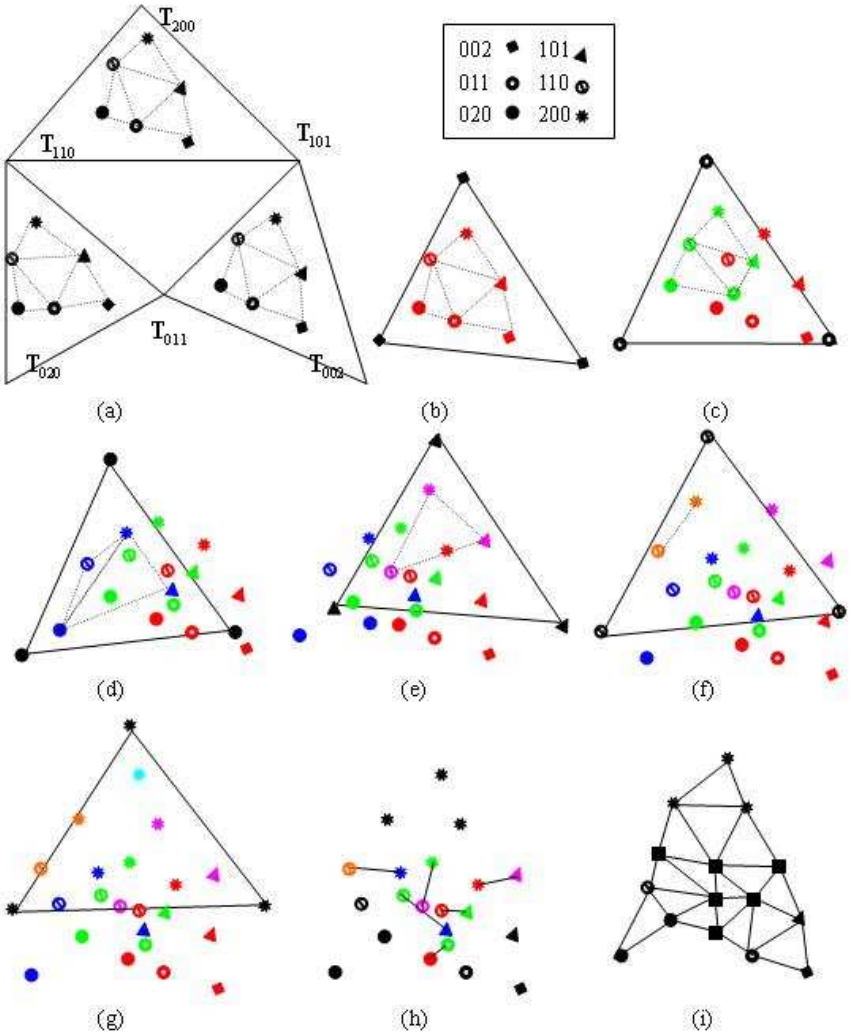


Figure 4.5: Geometric algorithm for control points with $m = n = 2$.
(a) the intermediate points at level 2 and 1; (b-g) construction points from triangles; (h) the control point from construction points; and (i) the control point.

Control points can be geometrically constructed from the power points. Consider $m = 2, n = 2$. Set $\mathbf{T}_{ijk}, i+j+k = 2$ be the control points for the TB surface $\mathbf{T}(x, y, z)$ and $\mathbf{P}_{ijk} = (x_{ijk}, y_{ijk}, z_{ijk}), i + j + k = 2$ be the control points for the domain surface

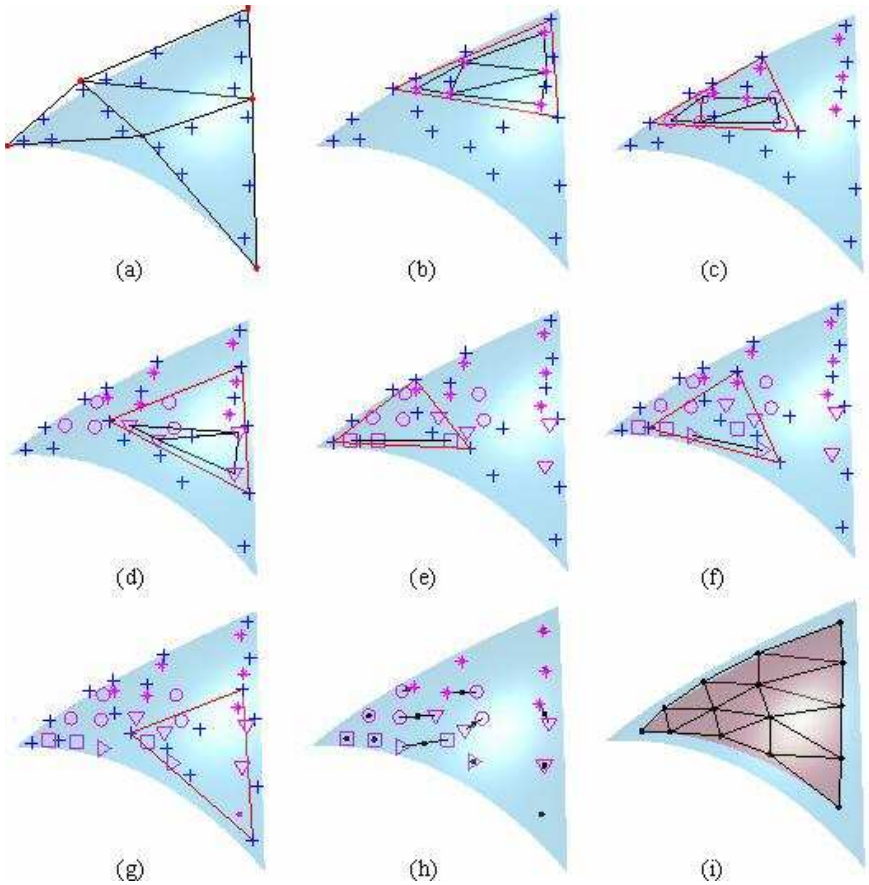


Figure 4.6: A 3D example of geometric algorithm with $m = n = 2$.
(a) the intermediate points at level 2 and 1; (b-g) construction points from triangles;(h) the control point from construction points; and (i) the control point.

$\mathbf{P}(u, v, w)$. Set

$$
\begin{aligned}
&\mathbf{S}^2_{\mathbf{IJK}} = \mathbf{T}^2_{000}(\mathbf{P}^2_{\mathbf{IJK}}) = \mathbf{T}^2_{000}(\mathbf{P}_{I_1J_1K_1}, \mathbf{P}_{I_2J_2K_2}) = \mathbf{F}_{I_1J_1K_1,I_2J_2K_2} \\
&= \left(x_{I_1J_1K_1}E_1 + y_{I_1J_1K_1}E_2 + z_{I_1J_1K_1}E_3\right)\left(x_{I_2J_2K_2}E_1 + y_{I_2J_2K_2}E_2 + z_{I_2J_2K_2}E_3\right)\mathbf{T}_{000} \\
&= x_{I_1J_1K_1}x_{I_2J_2K_2}\mathbf{T}_{200} + y_{I_1J_1K_1}y_{I_2J_2K_2}\mathbf{T}_{020} + z_{I_1J_1K_1}z_{I_2J_2K_2}\mathbf{T}_{002} \\
&\quad + (x_{I_1J_1K_1}y_{I_2J_2K_2} + y_{I_1J_1K_1}x_{I_2J_2K_2})\mathbf{T}_{110} + (x_{I_1J_1K_1}z_{I_2J_2K_2} + z_{I_1J_1K_1}x_{I_2J_2K_2})\mathbf{T}_{101} \\
&\quad + (z_{I_1J_1K_1}y_{I_2J_2K_2} + y_{I_1J_1K_1}z_{I_2J_2K_2})\mathbf{T}_{011} \\
&= (x_{I_1J_1K_1}, y_{I_1J_1K_1}, z_{I_1J_1K_1}) \begin{pmatrix} \mathbf{T}_{200} & \mathbf{T}_{110} & \mathbf{T}_{101} \\ \mathbf{T}_{110} & \mathbf{T}_{020} & \mathbf{T}_{011} \\ \mathbf{T}_{101} & \mathbf{T}_{011} & \mathbf{T}_{002} \end{pmatrix} \begin{pmatrix} x_{I_2J_2K_2} \\ y_{I_2J_2K_2} \\ z_{I_2J_2K_2} \end{pmatrix}.
\end{aligned}
$$

Hence there are 21 different power points. The 15 control points are

$$\begin{array}{lll}
\mathbf{S}_{004} = \mathbf{F}_{002,002} & \mathbf{S}_{022} = (\mathbf{F}_{002,020} + 2\mathbf{F}_{011,011})/3 & \mathbf{S}_{400} = \mathbf{F}_{200,200} \\
\mathbf{S}_{103} = \mathbf{F}_{002,101} & \mathbf{S}_{301} = \mathbf{F}_{101,200} & \mathbf{S}_{310} = \mathbf{F}_{110,200} \\
\mathbf{S}_{013} = \mathbf{F}_{002,011} & \mathbf{S}_{211} = (\mathbf{F}_{011,200} + 2\mathbf{F}_{101,110})/3 & \mathbf{S}_{220} = (\mathbf{F}_{020,200} + 2\mathbf{F}_{110,110})/3 \\
\mathbf{S}_{202} = (\mathbf{F}_{002,200} + 2\mathbf{F}_{101,101})/3 & \mathbf{S}_{121} = (\mathbf{F}_{020,101} + 2\mathbf{F}_{011,110})/3 & \mathbf{S}_{130} = \mathbf{F}_{020,110} \\
\mathbf{S}_{112} = (\mathbf{F}_{002,110} + 2\mathbf{F}_{011,101})/3 & \mathbf{S}_{031} = \mathbf{F}_{011,020} & \mathbf{S}_{040} = \mathbf{F}_{020,020}
\end{array}$$

There are 21 cases for $\mathbf{B} = (\beta_0, \beta_1, \beta_2, \beta_3, \beta_4, \beta_5) \in \mathbf{B}_2^2$.

$$\begin{array}{lll}
\mathbf{B}_1 = (0,0,0,0,0,2)\,, & \mathbf{B}_8 = (0,0,0,1,1,0)\,, & \mathbf{B}_{15} = (1,0,0,1,0,0)\,, \\
\mathbf{B}_2 = (0,0,0,0,1,1)\,, & \mathbf{B}_9 = (0,0,1,0,1,0)\,, & \mathbf{B}_{16} = (0,0,2,0,0,0)\,, \\
\mathbf{B}_3 = (0,0,0,1,0,1)\,, & \mathbf{B}_{10} = (0,1,0,0,1,0)\,, & \mathbf{B}_{17} = (0,1,1,0,0,0)\,, \\
\mathbf{B}_4 = (0,0,1,0,0,1)\,, & \mathbf{B}_{11} = (1,0,0,0,1,0)\,, & \mathbf{B}_{18} = (1,0,1,0,0,0)\,, \\
\mathbf{B}_5 = (0,1,0,0,0,1)\,, & \mathbf{B}_{12} = (0,0,0,2,0,0)\,, & \mathbf{B}_{19} = (0,2,0,0,0,0)\,, \\
\mathbf{B}_6 = (1,0,0,0,0,1)\,, & \mathbf{B}_{13} = (0,0,1,1,0,0)\,, & \mathbf{B}_{20} = (1,1,0,0,0,0)\,, \\
\mathbf{B}_7 = (0,0,0,0,2,0)\,, & \mathbf{B}_{14} = (0,1,0,1,0,0)\,, & \mathbf{B}_{21} = (2,0,0,0,0,0)\,.
\end{array}$$

Hence, we have

$$\begin{array}{llll}
\mathbf{Q}_{\mathbf{B}_1} = \mathbf{F}_{002,002}, & \mathbf{Q}_{\mathbf{B}_2} = \mathbf{F}_{002,110}, & \mathbf{Q}_{\mathbf{B}_3} = \mathbf{F}_{002,200}, & \mathbf{Q}_{\mathbf{B}_4} = \mathbf{F}_{002,011}, \\
\mathbf{Q}_{\mathbf{B}_5} = \mathbf{F}_{002,101}, & \mathbf{Q}_{\mathbf{B}_6} = \mathbf{F}_{002,002}, & \mathbf{Q}_{\mathbf{B}_7} = \mathbf{F}_{110,110}, & \mathbf{Q}_{\mathbf{B}_8} = \mathbf{F}_{110,200}, \\
\mathbf{Q}_{\mathbf{B}_9} = \mathbf{F}_{110,011}, & \mathbf{Q}_{\mathbf{B}_{10}} = \mathbf{F}_{110,101}, & \mathbf{Q}_{\mathbf{B}_{11}} = \mathbf{F}_{110,002}, & \mathbf{Q}_{\mathbf{B}_{12}} = \mathbf{F}_{200,200}, \\
\mathbf{Q}_{\mathbf{B}_{13}} = \mathbf{F}_{200,011}, & \mathbf{Q}_{\mathbf{B}_{14}} = \mathbf{F}_{200,101}, & \mathbf{Q}_{\mathbf{B}_{15}} = \mathbf{F}_{200,002}, & \mathbf{Q}_{\mathbf{B}_{16}} = \mathbf{F}_{011,011}, \\
\mathbf{Q}_{\mathbf{B}_{17}} = \mathbf{F}_{011,101}, & \mathbf{Q}_{\mathbf{B}_{18}} = \mathbf{F}_{011,002}, & \mathbf{Q}_{\mathbf{B}_{19}} = \mathbf{F}_{101,101}, & \mathbf{Q}_{\mathbf{B}_{20}} = \mathbf{F}_{101,002}, \\
\mathbf{Q}_{\mathbf{B}_{21}} = \mathbf{F}_{002,002}. & & &
\end{array}$$

Corresponding to the parameter set $\mathbf{P}^n_{\mathbf{B}_i}$, the power point $\mathbf{Q}_{\mathbf{B}_i}$ can be derive from the Pyramid algorithms. At level one, there are totally 18 intermediate points at level 1 as

$$\mathbf{G}^{ijk}_{rst} = x_{rst}\mathbf{T}_{i+1,j,k} + y_{rst}\mathbf{T}_{i,j+1,k} + z_{rst}\mathbf{T}_{i,j,k+1},\ i+j+k = 1, r+s+t = 2,$$

where $\mathbf{G}^{ijk}_{rst}$ means using the parameter point $\mathbf{P}_{rst}$ on the triangle $\Delta\mathbf{T}_{i+1,j,k}\mathbf{T}_{i,j+1,k}\mathbf{T}_{i,j,k+1}$.

Figure 4.5(a) shows the 18 intermediate points, we can derive 6 intermediate points at level 1 base on each of the three triangles

$$\Delta \mathbf{T}_{200}\mathbf{T}_{110}\mathbf{T}_{101}, \Delta \mathbf{T}_{110}\mathbf{T}_{020}\mathbf{T}_{011}, \Delta \mathbf{T}_{101}\mathbf{T}_{011}\mathbf{T}_{002}.$$

Finally, the 21 construction points at level 0 can be derived as

$$\mathbf{F}_{rst,ijk} = x_{ijk}\mathbf{G}^{100}_{rst} + y_{ijk}\mathbf{G}^{010}_{rst} + z_{ijk}\mathbf{G}^{001}_{rst}, i+j+k = 2.$$

The construction point $\mathbf{F}_{rst,ijk}$ means using the parameter point $\mathbf{P}_{ijk}$ on the triangle $\Delta \mathbf{G}^{100}_{rst}\mathbf{G}^{010}_{rst}\mathbf{G}^{001}_{rst}$. Figure 4.5(b) shows the construction points

$$\mathbf{F}_{002,002}, \mathbf{F}_{002,110}, \mathbf{F}_{002,200}, \mathbf{F}_{002,011}, \mathbf{F}_{002,101}, \mathbf{F}_{002,002}$$

from the triangle $\Delta \mathbf{G}^{100}_{002}\mathbf{G}^{010}_{002}\mathbf{G}^{001}_{002}$. Figure 4.5(c) shows the construction points

$$\mathbf{F}_{110,110}, \mathbf{F}_{110,200}, \mathbf{F}_{110,011}, \mathbf{F}_{110,101}, \mathbf{F}_{110,002}, \mathbf{F}_{200,200}$$

from the triangle $\Delta \mathbf{G}^{100}_{011}\mathbf{G}^{010}_{011}\mathbf{G}^{001}_{011}$. Figure 4.5(d) shows the construction points

$$\mathbf{F}_{200,200}, \mathbf{F}_{200,011}, \mathbf{F}_{200,101}, \mathbf{F}_{200,002}$$

from the triangle $\Delta \mathbf{G}^{100}_{200}\mathbf{G}^{010}_{200}\mathbf{G}^{001}_{200}$. Figure 4.5(e) shows the construction points

$$\mathbf{F}_{011,011}, \mathbf{F}_{011,101}, \mathbf{F}_{011,002}$$

from the triangle $\Delta \mathbf{G}^{100}_{011}\mathbf{G}^{010}_{011}\mathbf{G}^{001}_{011}$. Figure 4.5(f) shows the construction points

$$\mathbf{F}_{101,101}, \mathbf{F}_{101,002}$$

from the triangle $\Delta \mathbf{G}^{100}_{101}\mathbf{G}^{010}_{101}\mathbf{G}^{001}_{101}$. Figure 4.5(g) shows the construction point $\mathbf{F}_{002,002}$ which is from the triangle$\Delta \mathbf{G}^{100}_{002}\mathbf{G}^{010}_{002}\mathbf{G}^{001}_{002}$. Figure 4.5(h) shows that 9 construction points are control points and other control points are formed as the center point of

the line segments. Figure 4.5(i) shows all the 15 control points. Figure 4.6 is a 3D example corresponding to Figure 4.5.

4.4 Functions and algorithms

In the previous section, the formula for the control points of the composition (Eq.4.15) is derived. Since the control points are the linear combinations of the power points (Eq.4.25), a geometric algorithm for the power points and control points is presented. However, Eq.4.25 is complex, especially when m and n are big. Hence, computations of the control points with some useful functions are presented in this section.

4.4.1 Power index set

The power point set is based on the power index set. A function is provided to derive all the power indices for $\mathbf{M}_d^s$. The Algorithm 4.1 returns a matrix of size $\begin{pmatrix} d-1+s \\ s \end{pmatrix} \times d$. Each row is a vector of given dimension d. Each row is an element of $\mathbf{M}_d^s$.

If $d = 3, s = 2, c = 6$, Algorithm 4.1 provides

$$\left.\begin{array}{llllll}
i=3 & v=2 & l=2 & e=1 & \Rightarrow M_{1,3}=2 \\
 & & l=1 & e=2 & \Rightarrow M_{2,3}=M_{3,3}=1 \\
 & & l=0 & e=3 & \Rightarrow M_{4,3}=M_{5,3}=M_{6,3}=0 \\
i=2 & v=0 & l=0 & e=1 & \Rightarrow M_{1,2}=0 \\
 & v=1 & l=1 & e=1 & \Rightarrow M_{2,2}=1 \\
 & & l=0 & e=1 & \Rightarrow M_{3,2}=0 \\
 & v=2 & l=2 & e=1 & \Rightarrow M_{4,2}=2 \\
 & & l=1 & e=1 & \Rightarrow M_{5,2}=1 \\
 & & l=0 & e=1 & \Rightarrow M_{6,2}=0 \\
i=1 & v=0 & & & \Rightarrow M_{1,1}=0 \\
 & v=0 & & & \Rightarrow M_{2,1}=0 \\
 & v=1 & & & \Rightarrow M_{3,1}=1 \\
 & v=0 & & & \Rightarrow M_{4,1}=0 \\
 & v=1 & & & \Rightarrow M_{5,1}=1 \\
 & v=2 & & & \Rightarrow M_{6,1}=1
\end{array}\right\} \Rightarrow \mathbf{M}_3^2 = \begin{pmatrix} 0 & 0 & 2 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 2 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

FUNCTION: $\mathbf{M}$ = GetPowerMatrix(d, s)

$c = \begin{pmatrix} d-1+s \\ s \end{pmatrix}$;
FOR $i = d$ to 1 DO
  $r = 1$;
  WHILE $r < c$ DO
    $v = 0$;
    FOR $j = i+1$ to d DO
      $v+ = M(r, j)$;
    END
    $v = s - v$;
    IF $i == 1$ DO
      $\mathbf{M}(r, i) = v$;
      $r++$;
      CONTINUE;
    END
    FOR $l = v$ to 0 DO
      $e = \begin{pmatrix} i-2+v-l \\ v-l \end{pmatrix}$;
      FOR k = 1 to e DO
        $\mathbf{M}(r, i) = l$;
        $r++$;
      END
    END
  END
END

Algorithm 4.1: Algorithm for power matrix $\mathbf{M}_d^s$

4.4.2 Point index

The point index is computed based on the parameter point list $\mathbf{P}$, control point list $\mathbf{T}$ and the control point list $\mathbf{S}$ as below:

$$\begin{aligned}
\mathbf{P} &= \{\mathbf{P}_{00m}, \mathbf{P}_{0,1,m-1}, \mathbf{P}_{1,0,m-1}, \cdots, \mathbf{P}_{0,m,0}, \mathbf{P}_{1,m-1,0}, \cdots, \mathbf{P}_{m00}\}, \\
\mathbf{T} &= \{\mathbf{T}_{00n}, \mathbf{T}_{0,1,n-1}, \mathbf{T}_{1,0,n-1}, \cdots, \mathbf{T}_{0,n,0}, \mathbf{T}_{1,n-1,0}, \cdots, \mathbf{T}_{n00}\}, \\
\mathbf{S} &= \{\mathbf{S}_{0,0,mn}, \mathbf{S}_{0,1,mn-1}, \mathbf{S}_{1,0,mn-1}, \cdots, \mathbf{S}_{0,mn,0}, \mathbf{S}_{1,mn-1,0}, \cdots, \mathbf{S}_{mn,0,0}\}.
\end{aligned} \tag{Eq.4.28}$$

$\mathbf{P}_{ijk}$ is the I_{ijk}^m-th point in $\mathbf{P}$ (Algorithm 4.2),

$$I_{ijk}^m = 1 + i + \frac{1}{2}(m-k)(m-k+1) = 1 + i + \frac{1}{2}(i+j)(i+j+1) \tag{Eq.4.29}$$

FUNCTION: index = PointIndex(i, j, k)

index $= 1 + i + \frac{1}{2}(i+j)(i+j+1)$;

Algorithm 4.2: Algorithm for point index

4.4.3 Power points

Writing M, N and Q as

$$M = (m+1)(m+2)/2, N = \begin{pmatrix} M-1+n \\ n \end{pmatrix}, Q = (mn+1)(mn+2)/2. \quad \text{(Eq.4.30)}$$

Then, the power index set $\mathbf{B}$ of size $N \times M$ and all the parameter indices $\mathbf{I}$ of size $M \times 3$ are obtained

$$\begin{aligned} \mathbf{B} &= \text{GetPowerMatrix}\,(M,n)\,, \\ \mathbf{I} &= \text{GetPowerMatrix}\,(3,m)\,, \end{aligned} \quad \text{(Eq.4.31)}$$

A power point can be computed for each row of the geometric algorithm (Algorithm 4.3). The power point list $\mathbf{P_P}$ contains N points.

FUNCTION: $\mathbf{P_P}$ =GetAllPowerPoints($\mathbf{T}, n, \mathbf{P}, m, \mathbf{B}, \mathbf{I}, M, N$)

FOR $i = 1$ to N DO
  $L = n$;
  $\mathbf{I_P} = \mathbf{T}$;
  FOR $j = 1$ to M DO
    $r = \mathbf{B}(i,j)$;
    (u_0, v_0, w_0) =$\mathbf{P}$(PointIndex($\mathbf{I}(j,1), \mathbf{I}(j,2), \mathbf{I}(j,3)$))
    FOR $k = 1$ to r DO
      $L = L - 1$;
      $\mathbf{N_L}$ = GetPowerMatrix(3,L);
      FOR each row (i_0, j_0, k_0) in $\mathbf{N_L}$ DO
        I_0 = PointIndex(i_0, j_0, k_0);
        I_1 = PointIndex$(i_0 + 1, j0, k_0)$;
        I_2 = PointIndex$(i_0, j_0 + 1, k_0)$;
        I_3 = PointIndex$(i_0, j_0, k_0 + 1)$;
        $\mathbf{T_P}(I_0) = u_0 * \mathbf{I_P}(I_1) + v_0 * \mathbf{IP}(I_2) + w_0 * \mathbf{I_P}(I_3)$;
      END
      $\mathbf{I_P} = \mathbf{T_P}$;
    END
  END
  $\mathbf{P_P}$ (i)= $\mathbf{I_P}$;
END

Algorithm 4.3: Algorithm for power points

4.4.4 Power points to control points

Each control point $\mathbf{S}_{RST}$ is a linear combination of the power points. For each point of $\mathbf{S}$ in Eq.4.28, A set of index is assigned to indicate which power vectors are contributing to this point. Setting $\mathbf{J} = \mathbf{B} \cdot \mathbf{I}$, the i-th row of $\mathbf{B}$ corresponding to the i-th row of $\mathbf{J}$ which is corresponding to an index for $\mathbf{S}$, say (R, S, T). Hence, i-th row of $\mathbf{B}$ contributes to the only one control point (R, S, T). Since $\mathbf{S}$ contains $\mathbf{Q}$, a matrix C_M of with MN rows can be established (Algorithm 4.4). Each row contains all the i-th row of $\mathbf{J}$. For example, if $m = n = 2$, we have

$$C_M = \begin{pmatrix} 21 & 20 & 18 & 15 & 11 & 6 & 14 & 10 & 5 & 4 & 12 & 8 & 3 & 2 & 1 \\ 0 & 0 & 0 & 19 & 17 & 16 & 0 & 13 & 9 & 0 & 0 & 0 & 7 & 0 & 0 \end{pmatrix}^{\top}.$$

The fifth row of C_M has values 11 and 17 which implies that the fifth control point $\mathbf{S}_{112}$ is constructed by the 11th and 17th power points. And $\mathbf{S}_{013}$ is the second point which is constructed by the 20-th power point.

FUNCTION: $\mathbf{C_M}$ =GetCorrespondenceMatrix($\mathbf{B}$, $\mathbf{I}$, m, n, M, N)

$\mathbf{J} = \mathbf{B} \cdot \mathbf{I}$;
$X[1 : Q] = 0$;
FOR $i = 1$ to N DO
    $(i0, j0, k0) = \mathbf{J}(i)$;
    j = PointIndex$(i0, j0, k0)$;
    $X(j) + +$;
    $\mathbf{C_M}(j, X(j)) = i$;
END

Algorithm 4.4: Algorithm for correspondence

4.4.5 Coefficients of the power points

For each power point, a coefficient following Eq.4.15 is calculated in Algorithm 4.5.

4.4.6 Complete control points

With the previous 5 functions, Algorithm 4.6 provides the construction points $\mathbf{S}$.

FUNCTION: $\mathbf{C_V}$ =GetCoefficientVector($\mathbf{B}$, $\mathbf{I}$, m, n, M, N)

FOR $s = 1$ to N DO
  $B = \mathbf{B}(s) = (B_1, \cdots, B_M)$;
  $(R, S, T) = B \cdot \mathbf{I}$;
  $a = 1$;
  $b = 1$;
  FOR $i = 0$ to m DO
    FOR $j = 0$ to $m - i$ DO
      $k = m - i - j$;
      l= PointIndex(i, j, k);
      $b = b \cdot (B_l!)$;
      For $t = 1$ to B_l DO
        $a = a \cdot \frac{m!}{i!j!k!}$;
      END
    END
  END
  $\mathbf{C_V}(s) = \frac{a \cdot n! \cdot R! \cdot S! \cdot T!}{b \cdot (mn)!}$;
END

Algorithm 4.5: Algorithm for coefficient

FUNCTION: $\mathbf{S}$ =GetAllControlPoints($\mathbf{T}, n, \mathbf{P}, m$)

Get M, N, Q in Eq.4.30;
$\mathbf{B}, \mathbf{I}$ in Eq.4.31;
$\mathbf{A}$ = GetAllPowerPoints($\mathbf{T}, n, \mathbf{P}, m, \mathbf{B}, \mathbf{I}, M, N$);
$\mathbf{C_M}$ = GetCorrespondenceMatrix($\mathbf{B}, \mathbf{I}, m, n, M, N$);
$\mathbf{C_V}$ = GetCoefficientVector($\mathbf{B}, \mathbf{I}, m, n, M, N$);
FOR $i = 1$ to Q DO
  $\mathbf{S}(i) = 0$;
  $X = \mathbf{C_M}(i)$;
  $j = 0$;
  WHILE $X(j) > 0$ DO
    $B = \mathbf{A}(X(j))$;
    $b = \mathbf{C_V}(X(j))$;
    $\mathbf{S}(i) = \mathbf{S}(i) + B * b$;
  END
END

Algorithm 4.6: Algorithm for complete control points

4.5 Examples and discussion

In this part, examples from reference 16 are cited and the proposed algorithm is applied to show the geometric properties of the control points.

Example 1: Set $m = 1$, the domain surface $\mathbf{P}(u, v, w)$ is defined by three parameter control point

$$\mathbf{P}_{100} = (x_{100}, y_{100}, z_{100}),$$
$$\mathbf{P}_{010} = (x_{010}, y_{010}, z_{010}),$$
$$\mathbf{P}_{001} = (x_{001}, y_{001}, z_{001}).$$

Then, the control point for the composition will be

$$\mathbf{S}_{RST} = \sum_{|\mathbf{I}_n^m|=R,|\mathbf{J}_n^m|=S,|\mathbf{K}_n^m|=T} C^{1n}_{\mathbf{I}_n^m\mathbf{J}_n^m\mathbf{K}_n^m}\mathbf{T}^n_{000}(\mathbf{P}^n_{\mathbf{I}_n^m\mathbf{J}_n^m\mathbf{K}_n^m}).$$

When $|\mathbf{I}_n^m| = R, |\mathbf{J}_n^m| = S, |\mathbf{K}_n^m| = T \ \forall I_l, J_l, K_l \in \{0,1\}\ I_l, J_l, K_l \in \{0,1\}$, the power vector for the parameter vector $\mathbf{P}^n_{\mathbf{I}_n^m\mathbf{J}_n^m\mathbf{K}_n^m}$ is $\mathbf{B}_{\mathbf{I}_n^m\mathbf{J}_n^m\mathbf{K}_n^m} = (R, S, T)$. Hence

$$\mathbf{P}_{RST} = \mathbf{T}^n_{000}(\mathbf{P}^n_{\mathbf{I}_n^m\mathbf{J}_n^m\mathbf{K}_n^m}) = \prod_{l=1}^{n}(x_{I_lJ_lK_l}E_1 + y_{I_lJ_lK_l}E_2 + z_{I_lJ_lK_l}E_3)\mathbf{T}_{000}$$

$$= (x_{100}E_1 + y_{100}E_2 + z_{100}E_3)^R (x_{010}E_1 + y_{010}E_2 + z_{010}E_3)^S (x_{001}E_1 + y_{001}E_2 + z_{001}E_3)^T \mathbf{T}_{000}$$

Therefore,

$$\begin{aligned}\mathbf{S}_{RST} &= \sum_{|\mathbf{I}_n^m|=R,|\mathbf{J}_n^m|=S,|\mathbf{K}_n^m|=T} C^{1,n}_{\mathbf{I}_n^m\mathbf{J}_n^m\mathbf{K}_n^m}\mathbf{T}^n_{000}(\mathbf{P}^n_{\mathbf{I}_n^m\mathbf{J}_n^m\mathbf{K}_n^m}) \\ &= \mathbf{P}_{RST}\sum_{|\mathbf{I}_n^m|=R,|\mathbf{J}_n^m|=S,|\mathbf{K}_n^m|=T} C^{1n}_{\mathbf{I}_n^m\mathbf{J}_n^m\mathbf{K}_n^m} \\ &= \mathbf{P}_{RST}.\end{aligned}$$

This result is the same as that of Chang and Davis [111]. In this case, the number of different construction points is $(n+2)(n+1)/2$, and they are just the control point for the composition (Figure 4.7). While $m = 1, n = 2$, the algorithms yields

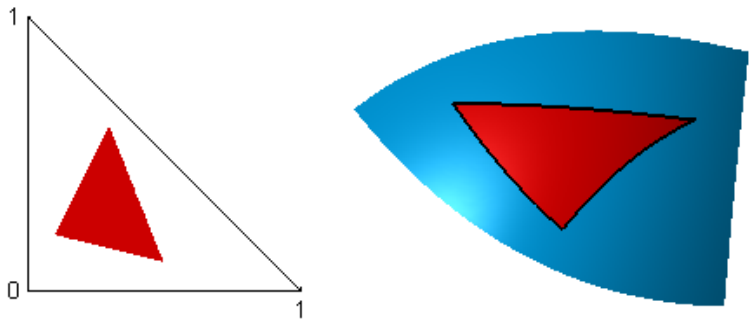
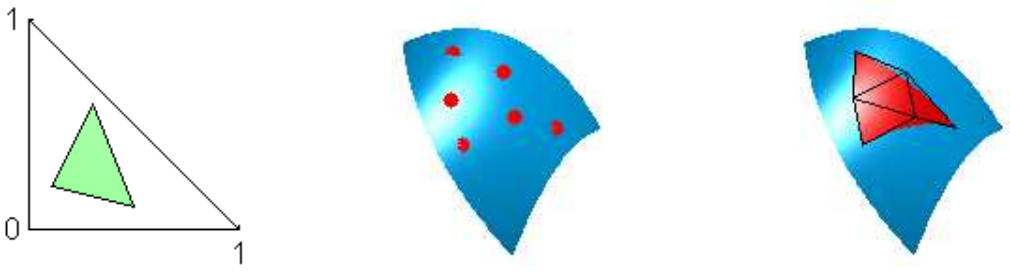


Figure 4.7: A triangular sub-patch from a TB surface.



(a) (b) (c)

Figure 4.8: A composite surface with $m = 1, n = 2$.
(a) the domain surfaces; (b) the power points; and (c) the composition surface with the control mesh.

$M = 3, N = Q = 6$ and

$$\mathbf{J} = \mathbf{B} \cdot \mathbf{I} = \begin{pmatrix} 0 & 0 & 2 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 2 & 0 \\ 1 & 1 & 0 \\ 2 & 0 & 0 \end{pmatrix} \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 2 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 2 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 2 \end{pmatrix}, \mathbf{C_M} = \begin{pmatrix} 6 \\ 5 \\ 3 \\ 4 \\ 2 \\ 1 \end{pmatrix}, \mathbf{C_V} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}.$$

The value from $\mathbf{C_M}, \mathbf{C_V}$ indicate that each control point equals one power point. Figure 4.8 shows the example.

Example 2: This example shows the surface subdivision.

Set $m = 1$, the TB surface $\mathbf{P}_1(u, v, w)$ is defined by three parameter points

$$(0, 0, 1), (0, 1, 0), (0.5, 0.5, 0).$$

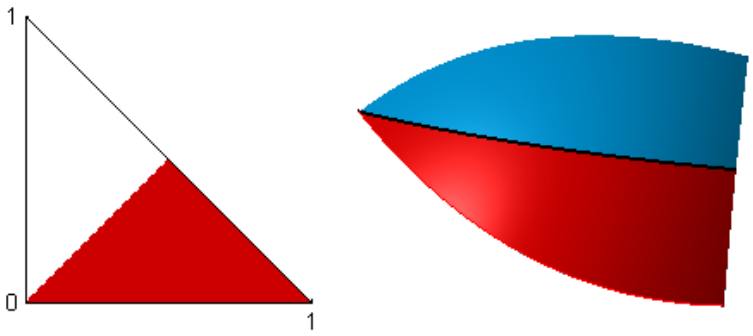


Figure 4.9: Subdivision of a TB surface.

And another TB surface $\mathbf{P}_2(u, v, w)$ is defined by three parameter points

$$(0, 0, 1), (1, 0, 0), (0.5, 0.5, 0).$$

From **Example 1**,

$$\mathbf{D}^1_{RST} = (E_3)^R (E_2)^S (0.5E_1 + 0.5E_2)^T \mathbf{T}_{000} = \sum_{i=0}^{T} \begin{pmatrix} T \\ i \end{pmatrix} \mathbf{T}_{i,S+(T-i),R} \Big/ 2^T,$$

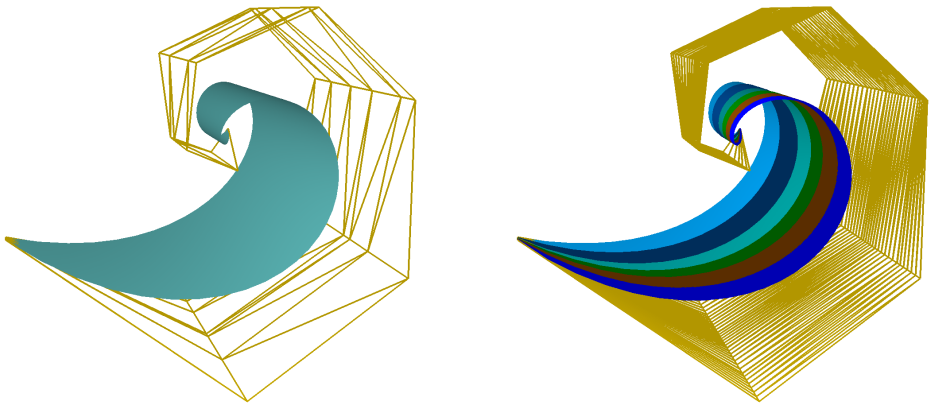
And

$$\mathbf{D}^2_{RST} = (E_3)^R (E_1)^S (0.5E_1 + 0.5E_2)^T \mathbf{T}_{000} = \sum_{i=0}^{T} \begin{pmatrix} T \\ i \end{pmatrix} \mathbf{T}_{S+i,T-i,R} \Big/ 2^T,$$

Then these two composition surfaces

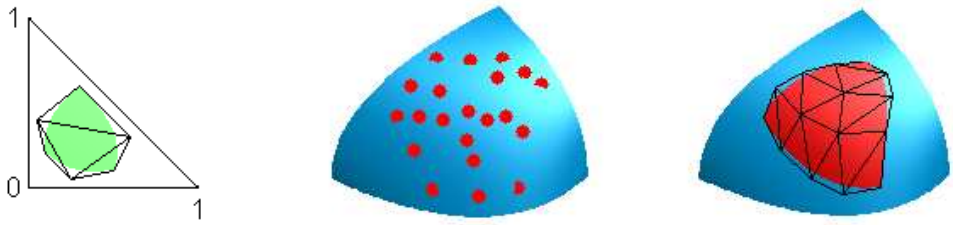
$$\mathbf{S}^i(u, v, w) = \sum_{R+S+T=mn} B^{mn}_{RST}(u, v, w)\mathbf{D}^i_{RST}, i = 1, 2$$

form a subdivision of the original surface (Figure 4.9). Figure 4.10 shows the subdivision of a leaf surface into 6 sub-patches.



(a) (b)

Figure 4.10: Subdivision of a leaf surface.
(a) the leaf surface with its control nets; and (b) the 6 sub-patches with their control nets.



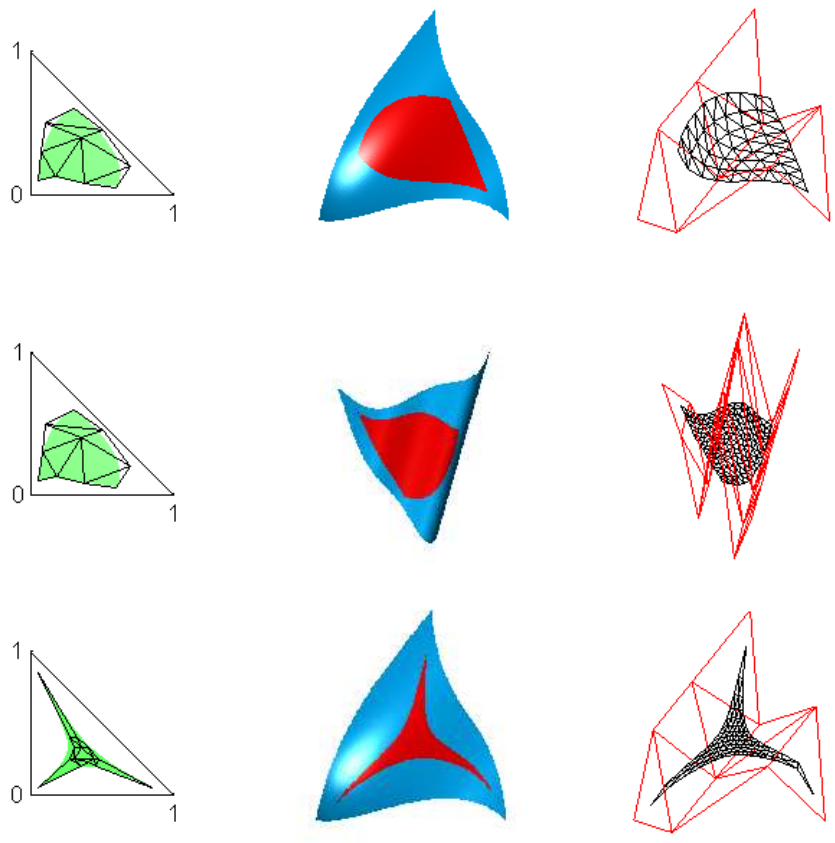
(a) (b) (c)

Figure 4.11: A composite surface with $m = 2, n = 2$.
(a) the domain surface; (b) the power points; and (c) the composition surface and the control mesh.

Example 3: Consider $m = 2, n = 2$. Then $M = 6, N = 21, Q = 15$, And

$$\mathbf{J}=\mathbf{B}\cdot\mathbf{I}=\begin{pmatrix}
0&0&0&0&0&2\\0&0&0&0&1&1\\0&0&0&1&0&1\\0&0&1&0&0&1\\0&1&0&0&0&1\\1&0&0&0&0&1\\
0&0&0&0&2&0\\0&0&0&1&1&0\\0&0&1&0&1&0\\0&1&0&0&1&0\\1&0&0&0&1&0\\
0&0&0&2&0&0\\0&0&1&1&0&0\\0&1&0&1&0&0\\1&0&0&1&0&0\\
0&0&2&0&0&0\\0&1&1&0&0&0\\1&0&1&0&0&0\\0&2&0&0&0&0\\1&1&0&0&0&0\\2&0&0&0&0&0
\end{pmatrix}\begin{pmatrix}0&0&2\\0&1&1\\1&0&1\\0&2&0\\1&1&0\\2&0&0\end{pmatrix}=\begin{pmatrix}
4&0&0\\3&1&0\\2&2&0\\3&0&1\\2&1&1\\2&0&2\\2&2&0\\1&3&0\\2&1&1\\1&2&1\\1&1&2\\
0&4&0\\1&2&1\\0&3&1\\0&2&2\\2&0&2\\1&1&2\\1&0&3\\0&2&2\\0&1&3\\0&0&4
\end{pmatrix},\mathbf{C_M}=\begin{pmatrix}21&0\\20&0\\18&0\\15&19\\11&17\\6&16\\14&0\\10&13\\5&9\\4&0\\12&0\\8&0\\3&7\\2&0\\1&0\end{pmatrix},\mathbf{C_V}=\begin{pmatrix}
1\\1\\1/3\\1\\1/3\\1/3\\2/3\\1\\2/3\\2/3\\1/3\\1\\1/3\\1\\1/3\\2/3\\2/3\\1\\2/3\\1\\1
\end{pmatrix}.$$

Hence, 21 different power points and 15 control points are generated. Figure 4.11 shows the surface.



(a) (b) (c)

Figure 4.12: Composite surfaces.
first row: $m = 3, n = 3$. second row: $m = 3, n = 5$. third row $m = 4, n = 3$. (a) the domain surface; (b) the power points; and (c) the composition surfaces and the control meshes.

In fact, the proposed algorithm can handle any couple of m, n. Figure 4.12 shows some other examples with $m = 3, n = 3$, $m = 3, n = 5$ and $m = 4, n = 3$.

Example 4: Different parameterization of the composite surface.

Different choices of the interior control point for the domain surfaces may lead to different parameterizations of the composite surfaces. The interior control point can be used as parameters to make adjustment of the composite surface. Figure 4.13 shows an example of a composition with $m = n = 5$. Figure 4.13(b-f) are the different choices of interior control point. In each case, the domain surface is uniform sample on the surface (the green curves are parameter curves). Uniform parameter curves (Figure 4.13(b)) in domain surface leads to uniform parameter curves (Figure 4.13(h)) in the composite surface. Moving the interior control point causes the change in the density of parameter curve for both the domain surface (Figure 4.13(c)

and Figure 4.13(d)) and the composite surface (Figure 4.13(i) and Figure 4.13(j)). It can also cause the parameter curves intersecting with each other (Figure 4.13(e), Figure 4.13(f), Figure 4.13(k) and Figure 4.13(l)).

Example 5: Surface extensions

We can also allow the domain surface to extend outside D_T. In this case, the composite surface will not be a sub-patch of the TB surface. Figure 4.14(a) shows a TB surface with three boundary curves in red, blue and yellow. In Figure 4.14(b), a domain surface in black is defined as an extension of the domain D_T. The composite surface (Figure 4.14(e)) becomes a nature extension of the original TB surface (Figure 4.14(d)) at the yellow boundary. Figure 4.14(c) and Figure 4.14(f) show the extension of the blue boundary.

4.6 Summary

In this chapter, an approach to generate a TB sub-patch from a TB surface is presented. Firstly, the TB sub-patch can be formed by composition of the TB surface and the domain surface. Secondly, an explicit formula for computing the control points of the composition is derived. These new control points are the linear combinations of the control points of the TB surface while the coefficients are given by the parameter points of the domain surface. Thirdly, the geometric algorithm for the power points is analyzed. The power points can be calculated using the Pyramid algorithms. Then the control points of the TB sub-patch are the linear combinations of these power points. Several examples are examined.

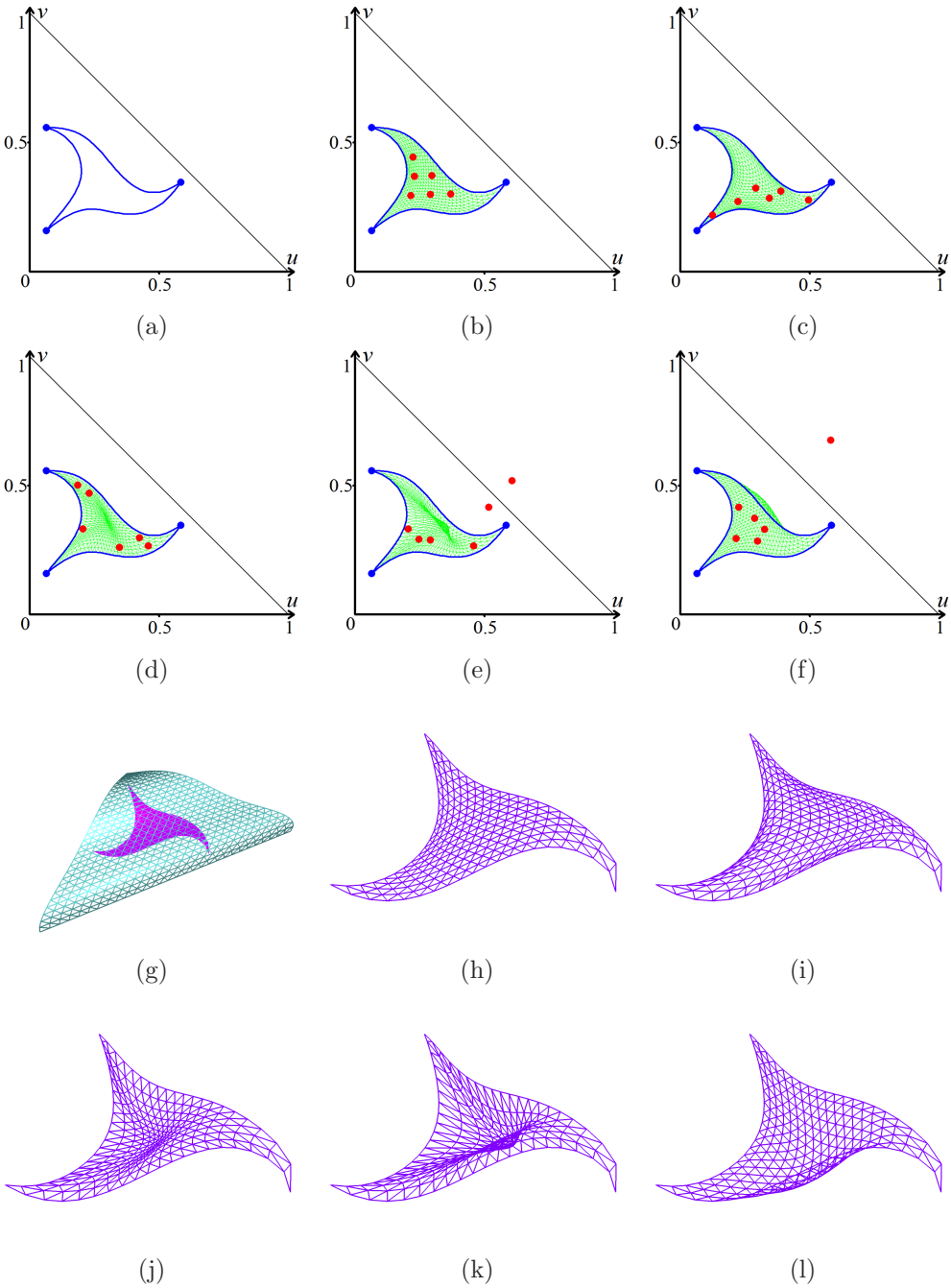
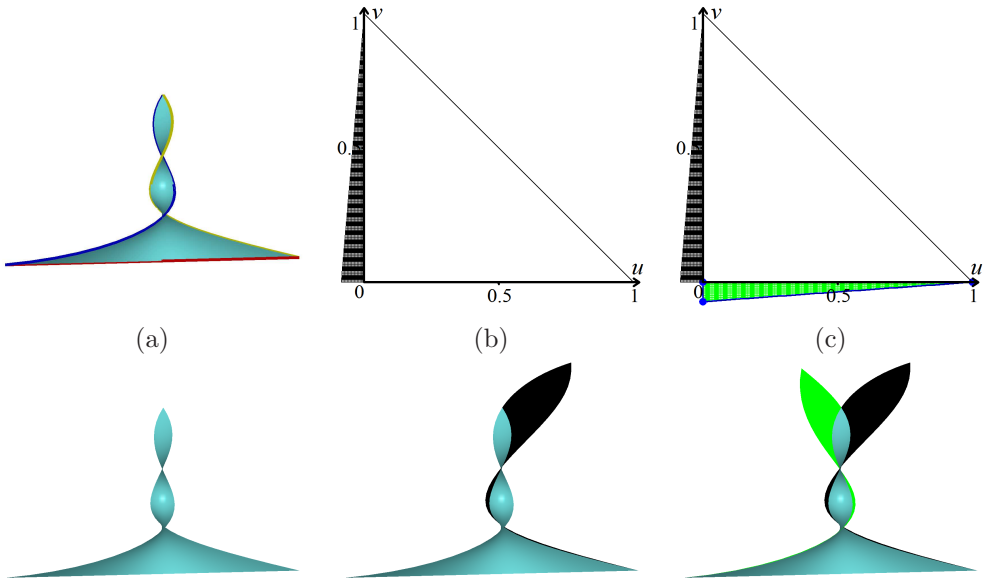


Figure 4.13: Different parameterization of the composite surface.
(a) an area bounded by three curves; (b-f) different choices of interior control point; (g) the sub-patch defined by the three curves in (a); and (h-l) different parameterization of the sub-patch.



(a) (b) (c)

(d) (e) (f)

Figure 4.14: Surface extensions.

(a) a surface with three boundary curves in red, blue and yellow; (b) one domain surface in black; (c) another domain surface in green; (d) the surface; (e) the composite surface from (b); and (f) the composite surface from (c).

Chapter 5

H-NURBS Surfaces[4]

Texture mapping is an efficient and effective tool in computer graphics and animation. While computationally very cost-effective, texture mapping may produce non-realistic appearances of shapes in a 3D environment, especially when viewing closely. To improve the realism of 3D modeling, bump mapping technique is developed to add details with the 3D models on top of texture mapping. Bump mapping, however, offers only simple and visual enhancement. Displacement mapping technique can further improve the localized detail of geometry. In this chapter, Monge mapping technique is developed for detail and local shape modifications of NURBS represented geometry in a 3D environment. Based on multiresolution and refinement schemes, hierarchical NURBS (H-NURBS) is first investigated to design a mechanism for the purpose of carrying localized geometric information. Monge mapping on H-NURBS patch can be easily performed via simple cut-&-paste operations. Parametric control of the local shapes is developed to facilitate easier and better 3D local modeling.

[4]The following publications are based on the results of this chapter:
Chen,W.Y., Zheng,J.M., and Cai,Y.Y., (2008). **Generalized hierarchical NURBS for interactive shape modification**,*Proceedings of The 7th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and Its Applications in Industry*, ACM.
Chen,W.Y., Zheng,J.M., and Cai, Y.Y., (2010). **Monge Mapping Using Hierarchical NURBS**,*The Visual Computer*, 26(6-8), 779-789.
Chen,W.Y., Cai,Y.Y., and Zheng,J.M., (2010). **Freeform-based form feature modeling using a hierarchical & multi-resolution NURBS method**,*Proceedings of The 9th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and Its Applications in Industry*, ACM.

5.1 Prior art

Stereographic visualization is quickly entering homes and offices with 3D TVs and displays commercially available. In a high definition (HD) environment, detail of 3D geometry becomes necessary in modeling and visualization. In computer graphics, techniques, such as texture mapping [112, 113], bump mapping [114], and displacement mapping [115, 116], have been developed to improve the realism of the 3D scene. However, these methods focus on efficient rendering.

In 3D modeling, complex objects can be represented using Non-uniform Rational B-spline (NURBS). NURBS has been a de facto standard in CAD and CG industries [117]. Most of the current commercial modeling and animation systems take NURBS as a fundamental representation for free-form shapes. NURBS can easily achieve any Level of Detail (LOD). The modification of a complex NURBS shape is, however, not that straightforward. Control point (or equivalent) modification is often used in an interactive fashion. Knots insertion technique is applied in order to limit the shape change in a controlled region which might not be intuitive to most of the end users. Typically, designers have to do trial and error before getting a satisfied change.

5.1.1 Texture mapping

Shape features of an object usually can be described as surfaces. The modeling and rendering of these surfaces, especially complex ones, are crucial to improve the realism of the scene especially in 3D. Texture mapping [112, 113] is very efficient to create the appearance of complexity without modeling and rendering of the complex surfaces. Basically, texture mapping adds 2D detail using images on top of surfaces. In general forms of texture mapping, attributes such as specular reflection [118], normal [114], transparency [119], and diffuse reflection [120] can be added to surfaces.

5.1.2 Bump mapping

Surface normal plays a vital role in graphics rendering. It is an intuitive way to change the lighting effect. Bump mapping [114] adds perturbations to the surface normal, making a smooth surface appear bumpy. A traditional bump map uses an image to

define the bumps: the brighter the color, the higher the bump. Normal mapping [121] uses the actual normal instead of just a perturbation of the normal to make a smooth surface appear bumpy.

5.1.3 Displacement mapping

A height map is a grayscale texture where the brightness of each pixel represents its height. Displacement mapping is the first technique using a height map to represent a fine scale surface. It is initially introduced by Cook in the context of Shade Trees [115] and later in REYES architecture [116]. Different from bump mapping and texture mapping, displacement mapping uses a height map to change the geometric position of the surface points. A common displacement mapping subdivides a surface into a large number of mesh points and displaces each surface point in its normal direction according to the value of the height map. This can produce a great sense of depth and detail. However, the rendering is either space-inefficient or computationally expensive.

The height map is also used in other techniques. The View-dependent Displacement Mapping (VDM) method [122] preprocesses a height map to get a set of VDM images based on different view directions and different curvatures. Although VDM method provides good quality, it requires more space to store the preprocessed VDM data. Relief Texture Mapping (RTM) [123] uses multiple height maps to do the rendering.

5.1.4 Level of detail

LOD is applied to display geometry detail at different levels. Generally, texture mapping, bump mapping, and displacement mapping are independent of LOD. With the help of Mipmapping [124], these techniques can have some kind of LOD. However, these techniques have a focus on rendering rather than modeling and cannot achieve local modifications of a model. Texture mapping, bump mapping, and displacement mapping are not designed for local depth geometry modeling.

5.2 Research aims

This chapter focuses on the local shape modeling based on hierarchical NURBS (H-NURBS) representation. Monge mapping technique will be developed for the purpose of detail and local shape modeling with NURBS represented geometry in a 3D environment. More specifically, multiresolution and refinement schemes will be investigated to design an H-NURBS based mechanism to carry localized geometric information. Cut-&-paste operations and parametric control with Monge mapping will be developed on H-NURBS to facilitate easier and better 3D local modeling.

5.3 Hierarchical NURBS

A NURBS surface of order $m \times n$ is defined as

$$\mathbf{R}(u,v) = \frac{\mathbf{P}(u,v)}{W(u,v)} = \frac{\sum_i \sum_j P_{ij} w_{ij} N_{i,m}(u) N_{j,n}(v)}{\sum_i \sum_j w_{ij} N_{i,m}(u) N_{j,n}(v)}, \qquad \text{(Eq.5.1)}$$

where P_{ij} are the control points, w_{ij} are the weights, $N_{i,m}(u)$ are the normalized B-spline basis functions of order m over knot vector $\mathbf{u} = \{\cdots, u_1, u_2, \cdots, u_k, \cdots\}$, and $N_{i,n}(v)$ are the normalized B-spline basis functions of order n over knot vector $\mathbf{v} = \{\cdots, v_1, v_2, \cdots, v_l, \cdots\}$.

5.3.1 Refinement of a NURBS surface

B-spline basis function can be refined. If we insert a set of new knots into the original knot vector $\mathbf{u}$, yielding a new refined knot vector $\bar{\mathbf{u}}$, which is a superset of $\mathbf{u}$, the original B-spline basis functions $N_{i,m}(u)$ defined over $\mathbf{u}$ can be formulated as a linear combination of the refined basis functions $\bar{N}_{r,m}(u)$ defined over the new knot vector $\bar{\mathbf{u}}$,

$$N_{i,m}(u) = \sum_r \alpha_i(r) \bar{N}_{r,m}(u). \qquad \text{(Eq.5.2)}$$

Similarly, if we refine $\mathbf{v}$ to yield $\bar{\mathbf{v}}$, we have

$$N_{j,n}(v) = \sum_{s} \alpha_j(s) \bar{N}_{s,n}(v). \tag{Eq.5.3}$$

The above two coefficients $\alpha_i(r)$ and $\alpha_j(s)$ are known as the discrete B-spline and they can be computed by Oslo algorithm [125, 126]. NURBS refinement is derived from B-spline basis functions. If we substitute Eq.5.2 and Eq.5.3 into Eq.5.1, we can reexpress the NURBS surface as

$$\mathbf{R}(u,v) = \frac{\sum_{r}\sum_{s} \bar{P}_{rs}\bar{w}_{rs}\bar{N}_{r,m}(u)\bar{N}_{s,n}(v)}{\sum_{r}\sum_{s} \bar{w}_{rs}\bar{N}_{r,m}(u)\bar{N}_{s,n}(v)}, \tag{Eq.5.4}$$

where the new control points $\bar{P}_{rs}$ and weights $\bar{w}_{rs}$ satisfy

$$(\bar{P}_{rs}\bar{w}_{rs}, \bar{w}_{rs}) = \sum_{i}\sum_{j} \alpha_i(r)\alpha_j(s)(P_{ij}w_{ij}, w_{ij}). \tag{Eq.5.5}$$

In the rest of the chapter, we take $N_{i,m}^{j,n} = N_{i,m}(u)N_{j,n}(v)$.

5.3.2 From H-spline to H-NURBS

For NURBS curves and surfaces, study shows that refinement [109, 117, 127] is an effective and interactive approach for local editing and manipulating. In this process, the influence of each control point is restricted to a surface area. For NURBS curves, refinement is obviously a local operation. As shown above, for tensor-product surfaces, however, refinement is not local. Inserting one knot causes the creation of new control points along a whole row or a whole column. To localize the effect of refinement, Forsey and Bartels [128] introduce a hierarchical B-spline surface concept for refinements through the use of overlays. The overlay keeps the original surface as the basic description of the surface and it is used as the replacement of local portions. This could restrict the influence of refinement to the locality at which the overlay is defined. In other words, the overlay will superimpose only the local area in the original surface. The approach can be repeated on the overlay which is, in turn, regarded as a surface subjected to refinement for the creation of further overlays. The result of this is that

the surface will contain a collection of overlays at different levels of refinement.

Hierarchy and local refinement have been widely studied. Local B-spline multiresolution [16, 129, 130] is a solution to provide a tool for decomposing data into a hierarchy. To achieve a better control with multiresolution models, reverse subdivision method is used to construct local multiresolution filters for B-spline [16]. Local refinement is extended to surfaces of arbitrary topology by Gonzalez-Ochoa and Peters [131]. The hierarchical triangular spline [132] is introduced to enable LOD construction and surface editing by interpolating the points of a hierarchy of locally refined meshes. From an arbitrary triangular mesh, Yvart *et al.* [133] propose a method to fit a refinable triangular spline surface of arbitrary topology. However, all these methods cannot be directly used for NURBS.

Three essential features of H-spline are (1) only the modified portions of a hierarchical surface are needed in a data structure; (2) each level of refinement is represented as an offset from reference position derived from a level of lower refinement; and (3) editing can be done by operating on points selected directly from the composite surface itself, rather than through control points. The three features lead H-spline to a compact and efficient means for fitting surfaces to data [134]. However, H-spline is not general enough as the name itself indicates that surface representation is limited to B-spline only. In addition, the H-spline implementation is also limited to point-based superimposition. This chapter addresses the generalization: 1) from H-spline to H-NURBS; 2) from point-based to parameter-based shape modifications; and 3) from superimposition to Monge mapping.

5.3.3 From B-patch to C-patch

With H-NURBS, we are able to modify surfaces on a patch-by-patch basis. Typically defined on a rectangular parametric domain, a base patch (B-patch) is a local area (NURBS represented) to be modified. An attaching patch (A-patch) defined by Monge patch is used to be integrated into the new design at the B-patch. The integration provides the combined patch (C-patch). The process of integration is called Monge mapping, which can be used to add detail or reduce the detail by adjusting three pa-

rameters. In [135], we have proposed two mechanisms to do local shape modifications for H-NURBS: the base plus detail mechanism and the replacement mechanism. The limitation with the approach is that not only the boundary conditions for A-patch are restrictive but also the adjustments of C-patch are not very intuitive. In Monge mapping, the A-patch is much easy to obtain and three parameters can be adopted to adjust the shape of the C-patch.

5.4 Monge patch in NURBS Form

Named after Gaspard Monge (1746-1818), a Monge patch $(u, v, f(u, v))$ is defined from a function f,

$$f : U \subset R^2 \rightarrow R, \quad (u, v) \in U.$$

A Monge patch can be described as the summary of one reference and one offset. Each surface point can be viewed as the offset from the point $(u, v, 0)$ along the z direction for the length $f(u, v)$. We can generalize the Monge patch for NURBS surfaces by providing the offset information for each control points.

5.4.1 Local coordinates of the control points

The control point P_{ij} in Eq.5.1 corresponds to a Greville point $\mathbf{\Omega}_{ij}$ [109] on the surface $\mathbf{R}(u, v)$ as

$$\mathbf{\Omega}_{ij} = \mathbf{R}(\phi_i, \varphi_j), \qquad \text{(Eq.5.6)}$$

where (ϕ_i, φ_j) is the Greville abscissae given by

$$\phi_i = \frac{1}{m} \sum_{k=i}^{i+m-1} u_k, \quad \varphi_j = \frac{1}{n} \sum_{k=j}^{j+n-1} v_k.$$

A local coordinate system and the s-curvature for each control point is necessary for Monge mapping.

Definition 5.1 The local coordinate system $(\mathbf{X}_{ij}, \mathbf{Y}_{ij}, \mathbf{Z}_{ij})$ at the control points P_{ij}

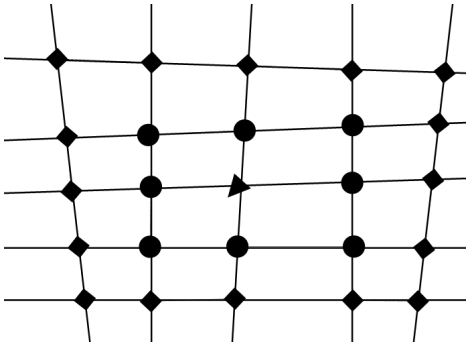


Figure 5.1: s-neighbors of a Greville point.
a Greville point $\mathbf{\Omega}_{ij}$(the triangle), 1-neighbor (dots) and 2-neighbor (squares).

is

$$\begin{cases} \mathbf{X}_{ij} = \mathbf{R}_u(\phi_i, \varphi_j), \\ \mathbf{Y}_{ij} = \mathbf{R}_v(\phi_i, \varphi_j), \\ \mathbf{Z}_{ij} = \mathbf{N}_{ij}, \end{cases} \tag{Eq.5.7}$$

where $\mathbf{N}_{ij}$ is the normal direction at $\mathbf{\Omega}_{ij}$.

Definition 5.2 The s-neighbor $\mathbf{\Omega}^s_{ij}$ is

$$\begin{aligned} \mathbf{\Omega}^s_{ij} = \{\mathbf{\Omega}_{i_0 j_0} \,| & i_0 = i \pm s \;\text{ or }\; j_0 = j \pm s, \\ & 0 \le i_0 \le k,\; 0 \le j_0 \le l\}\,, \end{aligned} \tag{Eq.5.8}$$

Figure 5.1 shows the 1-neighbor $\mathbf{\Omega}^1_{ij}$ and the 2-neighbor $\mathbf{\Omega}^2_{ij}$.

Definition 5.3 The s-curvature h^s_{ij} at P_{ij} is the average of the mean curvatures at all points in $\mathbf{\Omega}^s_{ij}$. The curvature descriptor θ_{ijs} is the average of the mean curvatures with all s-neighbor ($t \le s$) as

$$\theta_{ijs} = \frac{1}{s+1}\left(h_{ij} + \sum_{t=1}^{s} h^t_{ij}\right). \tag{Eq.5.9}$$

Suppose that P($\mathbf{\Omega}_{ij}$) is the portion of the surface around the Greville point $\mathbf{\Omega}_{ij}$. In general, if θ_{ijs} is near to zero, P($\mathbf{\Omega}_{ij}$) is flat, otherwise, P($\mathbf{\Omega}_{ij}$) is curved.

5.4.2 Monge patch and Monge mapping

Adding a NURBS surface $\mathbf{R}_1(u, v)$ onto another NURBS surface $\mathbf{R}_2(u, v)$ is an intuitive way to modify the local detail of $\mathbf{R}_2(u, v)$. The sum of two NURBS surfaces

produces a new surface,

$$\mathbf{R}_3(u,v) = \mathbf{R}_1(u,v) + \mathbf{R}_2(u,v) = \frac{\mathbf{P}_1(u,v)}{W_1(u,v)} + \frac{\mathbf{P}_2(u,v)}{W_2(u,v)}. \qquad \text{(Eq.5.10)}$$

However, if the two NURBS surfaces are of different weights, in order to get the new NURBS, we have to calculate the production of two B-spline $W_2(u,v)\cdot W_1(u,v)$, which is very complex [136, 137]. Moreover, it is difficult to adjust the shape of $\mathbf{R}_3(u,v)$ by directly summing the two surfaces.

Monge mapping uses a Monge patch to modify a NURBS surface while avoiding the product operation and provides parameters to adjust the final shape. A Monge patch contains two types of information: at which directions the control points move and how far they move.

Definition 5.4 A Monge patch $\mathbf{M}(u,v)$ is a NURBS surface that can be formulated as the sum of a planar reference patch $\mathbf{S}(u,v)$ and an A-patch $\mathbf{A}(u,v)$ as

$$\mathbf{M}(u,v) = \mathbf{S}(u,v) + \mathbf{A}(u,v) = \frac{\sum_{i=1}^{k}\sum_{j=1}^{l} S_{ij}\bar{w}_{ij}N_{i,m}^{j,n}}{\sum_{i=1}^{k}\sum_{j=1}^{l} \bar{w}_{ij}N_{i,m}^{j,n}} + \frac{\sum_{i=1}^{k}\sum_{j=1}^{l} A_{ij}\bar{w}_{ij}N_{i,m}^{j,n}}{\sum_{i=1}^{k}\sum_{j=1}^{l} \bar{w}_{ij}N_{i,m}^{j,n}}.$$

If the B-patch $\mathbf{B}(u,v)$ is

$$\mathbf{B}(u,v) = \frac{\sum_{i=1}^{k}\sum_{j=1}^{l} B_{ij}w_{ij}N_{i,m}^{j,n}}{\sum_{i=1}^{k}\sum_{j=1}^{l} w_{ij}N_{i,m}^{j,n}}, \qquad \text{(Eq.5.11)}$$

Monge mapping derives the C-patch $\mathbf{C}(u,v)$ as

$$\mathbf{C}(u,v) = \mathbf{B}(u,v) \oplus \mathbf{A}(u,v) = \frac{\sum_{i}\sum_{j} C_{ij}\hat{w}_{ij}N_{i,m}^{j,n}}{\sum_{i}\sum_{j} \hat{w}_{ij}N_{i,m}^{j,n}},$$

where

$$C_{ij} = B_{ij} + \alpha_{ij}\left|A_{ij}\right| d_{ij},$$

with

$$\begin{aligned} d_{ij} &= \frac{A^0_{ij}\mathbf{X}_{ij}+A^1_{ij}\mathbf{Y}_{ij}+A^2_{ij}\mathbf{Z}_{ij}}{\left|A^0_{ij}\mathbf{X}_{ij}+A^1_{ij}\mathbf{Y}_{ij}+A^2_{ij}\mathbf{Z}_{ij}\right|}, \\ \alpha_{ij} &= \eta e^{-\lambda\theta^2_{ijs}}, \quad \eta, \lambda \in R, \quad \lambda \geq 0, \quad s = 0, 1, 2, 3, ..., \end{aligned} \tag{Eq.5.12}$$

and $A_{ij} = (A^0_{ij}, A^1_{ij}, A^2_{ij})$, $(\mathbf{X}_{ij}, \mathbf{Y}_{ij}, \mathbf{Z}_{ij})$ is the local coordinate system at B_{ij}, θ_{ijs} is defined in Eq.5.9, and η, λ, s are shape parameters.

5.4.3 Boundary conditions

The index (i, j) is a boundary index if

$$\begin{aligned} &i = 1, \cdots, m-1, k-m+2, \cdots, k, \\ &or\ j = 1, \cdots, n-1, l-n+2, \cdots, l. \end{aligned} \tag{Eq.5.13}$$

The set Ψ consists of all boundary indices. A control point (weight) with a boundary index is a boundary control point (weight), otherwise, it is an inner control point (weight).

Definition 5.5 Two NURBS surfaces are matched if and only if they share the same u-order, the same v-order, the same u-knots and the same v-knots.

During a local modification, Monge mapping replaces the B-patch using the C-patch. To keep the continuity between the C-patch and its neighboring patches, the C-patch and the B-patch should be matched while sharing the same boundary points (weights) as follows:

$$d_{ij} = 0, \quad \hat{w}_{ij} = w_{ij}, \quad (i, j) \in \Psi. \tag{Eq.5.14}$$

5.4.4 Monge patch from an image

Using a gray image as height field, NURBS surface fitting techniques can provide a Monge patch with control point $M_{ij} = (x_{ij}, y_{ij}, z_{ij})$. The Monge patch is formulated in the form of $S_{ij} = (x_{ij}, y_{ij}, 0)$ and $A_{ij} = (0, 0, z_{ij})$, from Eq.5.12, Monge mapping will only move a control point B_{ij} along its local $\mathbf{Z}_{ij}$ coordinate. Figure 5.2 gives some Monge patches as bi-cubic NURBS surfaces.

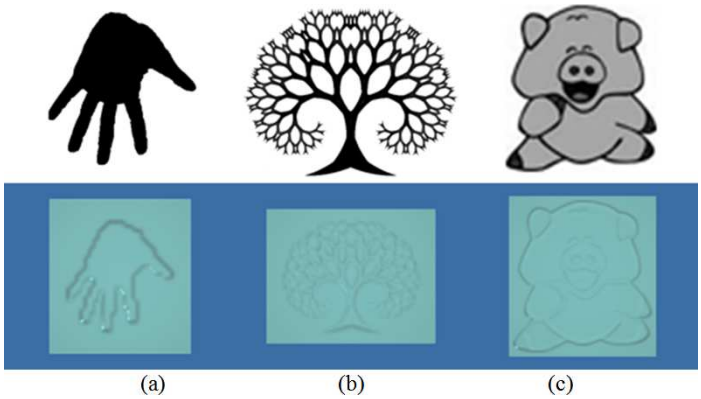


Figure 5.2: Monge patches from images.
the first row shows the gray images and the second row presents the NURBS surface. (a) a hand print; (b) a tree; and (c) a pig.

5.4.5 Monge patch from a NURBS surface

We need to modify the selected NURBS patch $\mathbf{R}(u, v)$ in order to obtain a Monge patch $\mathbf{S}(u, v)$. The key issue is how to derive the reference patch. A plane Γ can be derived by fitting its boundary points. Suppose $\mathbf{P}_\Gamma(p)$ is the projection of the point p on the plane and the domain of $\mathbf{R}(u, v)$ is $[u_{\min}, u_{\max}] \times [v_{\min}, v_{\max}]$. A planar surface can be obtained as

$$\mathbf{Q}(u, v) = \sum_{i=0}^{1} \sum_{j=0}^{1} Q_{ij} (1-u)^{1-i} u^i (1-v)^{1-j} v^j,$$

where

$$Q_{00} = \mathbf{P}_\Gamma \left(\mathbf{R}(u_{\min}, v_{\min}) \right), \qquad Q_{01} = \mathbf{P}_\Gamma \left(\mathbf{R}(u_{\min}, v_{\max}) \right),$$
$$Q_{10} = \mathbf{P}_\Gamma \left(\mathbf{R}(u_{\max}, v_{\min}) \right), \qquad Q_{11} = \mathbf{P}_\Gamma \left(\mathbf{R}(u_{\max}, v_{\max}) \right).$$

We can apply a transformation on the coordinate system such that the planar surface is the XY-plane, then derive the control points for the reference patch as

$$S_{ij} = \mathbf{Q} \left(\frac{\phi_i - u_{\min}}{u_{\max} - u_{\min}}, \frac{\varphi_j - v_{\min}}{v_{\max} - v_{\min}} \right).$$

Thus, the Monge patch is defined by taking

$$M_{ij} = \begin{cases} S_{ij}, & (i, j) \in \Psi, \\ P_{ij}, & (i, j) \notin \Psi. \end{cases}$$

5.4.6 Postprocessing of an A-patch

In above two sections, the A-patch is derived as an offset from the XY-plane. Algorithm 5.1 reformulates the A-patch using the idea of local coordinate system. After this, each control point of an A-patch is an offset in a local coordinate system. Later when applying Monge mapping, we can directly use the local coordinate system.

Step 1: Calculate the Greville abscissae (ϕ_i, φ_j) for each control point A_{ij}.

Step 2: Derive a corresponding point on the planar surface $\mathbf{Q}(\phi_i, \varphi_j)$.

Step 3: Obtain the local coordinate system at $\mathbf{Q}(\phi_i, \varphi_j)$ as $(\mathbf{X}_{ij}, \mathbf{Y}_{ij}, \mathbf{Z}_{ij})$.

Step 4: Replace A_{ij} with its new coordinates under $(\mathbf{X}_{ij}, \mathbf{Y}_{ij}, \mathbf{Z}_{ij})$.

Algorithm 5.1: Postprocessing of an A-patch.

5.5 H-NURBS based patch formation

5.5.1 B-patch Formation

A B-patch $\mathbf{B}(u, v)$, defined in a rectangular domain, is a selected NURBS patch to be modified. NURBS refinement allows forming much compact B-patches for local shape modifications. A B-patch can be formed in two styles: patch-based formation and point-based formation. Figure 5.3 shows the patch-based formation of a B-patch for a nose shape. The red lines are iso-curves and the green patches are selected patches. Before refinement, by selecting two corner patches (Figure 5.3(a)), we can define a local area containing the nose shape (Figure 5.3(b)). On the selected patch, a local refinement is performed (Figure 5.3(c)). Following the same way, a new local area can be selected (Figure 5.3(d)). Figure 5.3(e) shows the final B-patch. Figure 5.4 shows the point-based formation of a B-patch for the nose shape. A B-patch can be formed by selecting two corner points (Figure 5.4(b)). In this procedure, the local refinement is automatically performed (Figure 5.4(c)).

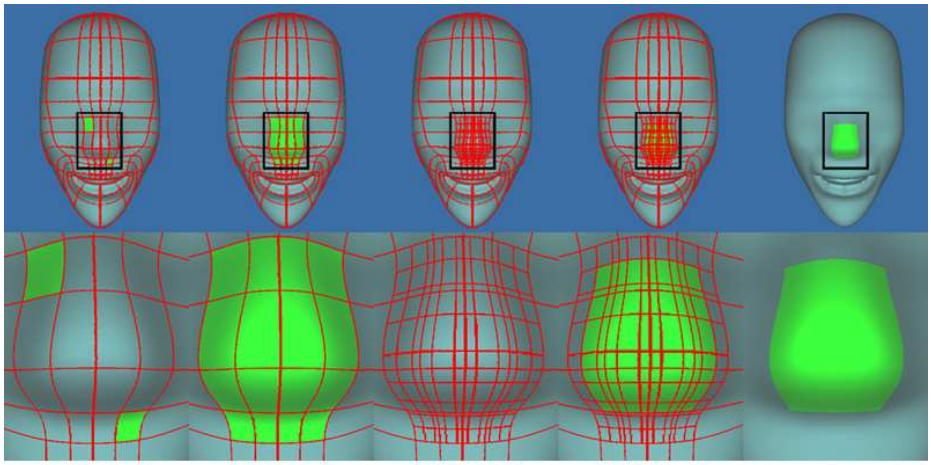


Figure 5.3: Patch-based formation of the B-patch.

the second row is the zoomed view of the rectangular portion in the first row. (a) two patches at level 0 are selected as the corner patches; (b) several patches are merged to one patch; (c) the selected patch is refined; (d) a new patch is formed at level 1; and (e) the B-patch.

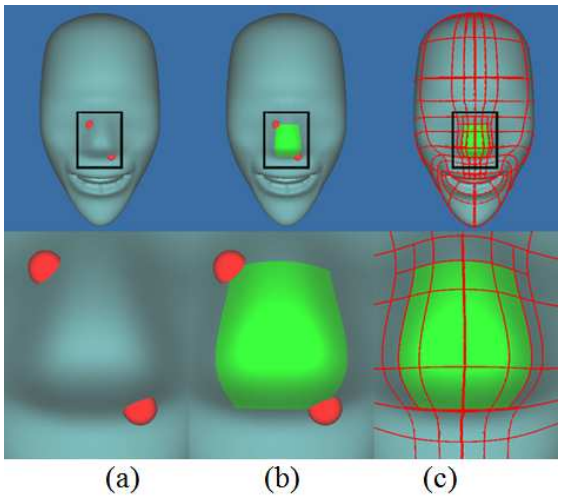


Figure 5.4: Point-based formation of the B-patch.

the second row is the zoomed view of the rectangular portion in the first row. (a) two points are selected as corner points; (b) a B-patch is selected using the two corner points; and (c) the B-patch is formed by local refinement.

5.5.2 A-patch formation

An A-patch $\mathbf{P}(u, v)$, which is derived from a Monge patch, is to be added onto the B-patch. We can create the A-patch either from any existing NURBS surface (Section 5.4.5) or from an image (Section 5.4.4). With the former method, we can easily achieve cut-&-paste operations. However, the A-patch and the B-patch may neither be matched nor share the same boundary weights. Surface approximation (Section 5.6) will be applied to produce a new A-patch before applying Monge mapping. With the latter method, during the surface fitting, we can adopt the orders, knots and boundary weights from a selected B-patch, therefore it requires no surface approximation.

Generally, before applying Monge mapping, we need to do the following three steps

Step 1: Degree elevation is used on the A-patch to match the orders.

Step 2: Translating, scaling and knots inserting are applied to match the u-knots and v-knots of both A-patch and B-patch.

Step 3: Surface approximation is employed on the A-patch to match the weights when necessary.

The second step requires a local refinement on both A-patch and B-patch. After the second step, the number of the control points of A-patch is the same as the number of the control points of B-patch. If the B-patch is provided by Eq.5.11, the A-patch is formulated as

$$\mathbf{P}(u, v) = \frac{\sum_{i=1}^{k} \sum_{j=1}^{l} P_{ij} \tilde{w}_{ij} N_{i,m}^{j,n}}{\sum_{i=1}^{k} \sum_{j=1}^{l} \tilde{w}_{ij} N_{i,m}^{j,n}}. \qquad \text{(Eq.5.15)}$$

A new A-patch $\mathbf{A}(u, v)$, which meets the boundary conditions, will be constructed (Section 5.6).

$$\mathbf{A}(u, v) = \frac{\sum_{i=1}^{k} \sum_{j=1}^{l} A_{ij} \bar{w}_{ij} N_{i,m}^{j,n}}{\sum_{i=1}^{k} \sum_{j=1}^{l} \bar{w}_{ij} N_{i,m}^{j,n}}. \qquad \text{(Eq.5.16)}$$

5.5.3 C-patch formation

The B-patch $\mathbf{B}(u,v)$ and the A-patch $\mathbf{A}(u,v)$ share the same number $(k \times l)$ of control points. Monge mapping calculates the C-patch $\mathbf{C}(u,v)$ using Eq.5.12. Different choices of the weights $\hat{w}_{ij}$ produce different shapes. Since we want to keep the continuity of the C-patch with nearby patches of the B-patch, we adopt the weights from the B-patch, $\hat{w}_{ij} = w_{ij}$.

5.6 Approximation for Monge mapping

From the B-patch $\mathbf{B}(u,v)$ and the A-patch $\mathbf{P}(u,v)$, surface approximation aims to derive a new A-patch $\mathbf{A}(u,v)$, such that the A-patch satisfies the boundary conditions. $\mathbf{P}(u,v)$ will be approximated by $\tilde{\mathbf{A}}(u,v)$ which is further approximated by $\mathbf{A}(u,v)$.

5.6.1 Approximations by boundary control points

$\tilde{\mathbf{A}}(u,v)$ is an approximation to $\mathbf{P}(u,v)$ satisfying $\tilde{A}_{ij} = 0$ on the boundary,

$$\tilde{\mathbf{A}}(u,v) = \frac{\sum_{i=1}^{k}\sum_{j=1}^{l} \tilde{A}_{ij}\tilde{w}_{ij}N_{i,m}^{j,n}}{\sum_{i=1}^{k}\sum_{j=1}^{l} \tilde{w}_{ij}N_{i,m}^{j,n}}, \qquad \text{(Eq.5.17)}$$

$\tilde{\mathbf{A}}(u,v)$ can be derived by minimizing the following function

$$\min \iint_{R^2} \left|\tilde{\mathbf{A}}(u,v) - \mathbf{P}(u,v)\right|^2 dudv. \qquad \text{(Eq.5.18)}$$

5.6.2 Approximations by weights

$\tilde{w}_{ij}$, the weights of $\tilde{\mathbf{A}}(u,v)$, may be different from w_{ij}, the weights of $\mathbf{B}(u,v)$, both on the boundary points and the inner points. We will approximate $\tilde{\mathbf{A}}(u,v)$ to using a new A-patch $\mathbf{A}(u,v)$ with weights $\bar{w}_{ij}$. There are three options for $\bar{w}_{ij}$:

i) No change with the weights of $\tilde{\mathbf{A}}(u,v)$ $(\bar{w}_{ij} = \tilde{w}_{ij})$.

ii) The boundary weights changed $(\bar{w}_{ij} = w_{ij})$ and the inner weights unchanged $(\bar{w}_{ij} = \tilde{w}_{ij})$.

iii) All the weights changed ($\bar{w}_{ij} = w_{ij}$).

For each option, we can solve the following problem to get $\mathbf{A}(u, v)$

$$\min \iint_{R^2} \left| \tilde{\mathbf{A}}(u, v) - \mathbf{A}(u, v) \right|^2 dudv. \tag{Eq.5.19}$$

This minimizes the distance between $\tilde{\mathbf{A}}(u, v)$ and $\mathbf{A}(u, v)$. We adopt option 3 in this work.

5.6.3 Surface approximations

We need to solve two approximations (Eq.5.18 and Eq.5.19) to get the desired A-patch. Both approximations only change N number of inner control points($N = (k - 2m + 2)(l - 2n + 2)$). We will show how to derive unknown inner control points from Eq.5.19, and the other one is similar. We have

$$\min \iint_{R^2} \left| \frac{\sum_{i=1}^{k} \sum_{j=1}^{l} \tilde{A}_{ij} \tilde{w}_{ij} N_{i,m}^{j,n}}{\sum_{i=1}^{k} \sum_{j=1}^{l} \tilde{w}_{ij} N_{i,m}^{j,n}} - \frac{\sum_{i=1}^{k} \sum_{j=1}^{l} A_{ij} \bar{w}_{ij} N_{i,m}^{j,n}}{\sum_{i=1}^{k} \sum_{j=1}^{l} \bar{w}_{ij} N_{i,m}^{j,n}} \right|^2 dudv, \tag{Eq.5.20}$$

where $\tilde{A}_{ij}, \tilde{w}_{ij}, \bar{w}_{ij}$ are provided. Suppose

$$A_{ij} = (D_{ij}^0, D_{ij}^1, D_{ij}^2), \tilde{A}_{ij} = (\tilde{D}_{ij}^0, \tilde{D}_{ij}^1, \tilde{D}_{ij}^2).$$

We generalize Eq.5.20 to

$$\begin{aligned} \min L = \min \iint_{R^2} \Bigg| & \sum_{i=1}^{k} \sum_{j=1}^{l} \tilde{A}_{ij} \tilde{w}_{ij} N_{i,m}^{j,n} \sum_{i=1}^{k} \sum_{j=1}^{l} \bar{w}_{ij} N_{i,m}^{j,n} \\ & - \sum_{i=1}^{k} \sum_{j=1}^{l} A_{ij} \bar{w}_{ij} N_{i,m}^{j,n} \sum_{i=1}^{k} \sum_{j=1}^{l} \tilde{w}_{ij} N_{i,m}^{j,n} \Bigg|^2 dudv. \end{aligned} \tag{Eq.5.21}$$

Thus, we have $3N$ different equations

$$\frac{\partial L}{\partial D_{i_0 j_0}^{k_0}} = 0, \quad k_0 = 0, 1, 2, \quad (i_0, j_0) \notin \Psi. \tag{Eq.5.22}$$

From each (k_0, i_0, j_0), we get

$$\begin{aligned}\frac{\partial L}{\partial D^{k_0}_{i_0 j_0}} = 2 \iint_{R^2} \bar{w}_{i_0 j_0} N^{j_0,n}_{i_0,m} \sum_{i=1}^{k} \sum_{j=1}^{l} \tilde{w}_{ij} N^{j,n}_{i,m} \cdot \\ \left(\sum_{i=1}^{k} \sum_{j=1}^{l} \tilde{D}^{k_0}_{ij} \tilde{w}_{ij} N^{j,n}_{i,m} \sum_{i=1}^{k} \sum_{j=1}^{l} \bar{w}_{ij} N^{j,n}_{i,m} \right. \\ \left. - \sum_{i=1}^{k} \sum_{j=1}^{l} D^{k_0}_{ij} \bar{w}_{ij} N^{j,n}_{i,m} \sum_{i=1}^{k} \sum_{j=1}^{l} \tilde{w}_{ij} N^{j,n}_{i,m} \right) dudv = 0.\end{aligned} \tag{Eq.5.23}$$

Since $N_{i_0,m}(u)$ is zero outside $[u_{i_0}, u_{i_0+m}]$ and $N_{j_0,n}(v)$ is zero outside $[v_{j_0}, v_{j_0+n}]$, reformulate the above equations into

$$\begin{aligned}&\int_{u_{i_0}}^{u_{i_0+m}} \int_{v_{j_0}}^{v_{j_0+n}} \bar{w}_{i_0 j_0} N^{j_0,n}_{i_0,m} \sum_{i=i_0-m+1}^{i_0+m-1} \sum_{j=j_0-n+1}^{j_0+n-1} \tilde{w}_{ij} N^{j,n}_{i,m} \\ &\sum_{i=i_0-m+1}^{i_0+m-1} \sum_{j=j_0-n+1}^{j_0+n-1} \tilde{D}^{k_0}_{ij} \tilde{w}_{ij} N^{j,n}_{i,m} \sum_{i=i_0-m+1}^{i_0+m-1} \sum_{j=j_0-n+1}^{j_0+n-1} \bar{w}_{ij} N^{j,n}_{i,m} dudv \\ =& \int_{u_{i_0}}^{u_{i_0+m}} \int_{v_{j_0}}^{v_{j_0+n}} \bar{w}_{i_0 j_0} N^{j_0,n}_{i_0,m} \sum_{i=i_0-m+1}^{i_0+m-1} \sum_{j=j_0-n+1}^{j_0+n-1} \tilde{w}_{ij} N^{j,n}_{i,m} \\ &\sum_{i=i_0-m+1}^{i_0+m-1} \sum_{j=j_0-n+1}^{j_0+n-1} D^{k_0}_{ij} \bar{w}_{ij} N^{j,n}_{i,m} \sum_{i=i_0-m+1}^{i_0+m-1} \sum_{j=j_0-n+1}^{j_0+n-1} \tilde{w}_{ij} N^{j,n}_{i,m} dudv.\end{aligned} \tag{Eq.5.24}$$

Denote the left hand side of Eq.5.24 as $f^{k_0}_{i_0,j_0}$ and rewrite the right hand side as

$$\begin{aligned}\sum_{i_1=i_0-m+1}^{i_0+m-1} \sum_{j_1=j_0-n+1}^{j_0+n-1} D^{k_0}_{i_1 j_1} \left(\int_{u_{i_0}}^{u_{i_0+m}} \int_{v_{j_0}}^{v_{j_0+n}} \bar{w}_{i_0 j_0} N^{j_0,n}_{i_0,m} \bar{w}_{i_1 j_1} N^{j_1,n}_{i_1,m} \right. \\ \left. \left(\sum_{i=i_0-m+1}^{i_0+m-1} \sum_{j=j_0-n+1}^{j_0+n-1} \tilde{w}_{ij} N^{j,n}_{i,m} \right)^2 dudv \right).\end{aligned} \tag{Eq.5.25}$$

Set the coefficients of $D^{k_0}_{i_1 j_1}$ be $h^{i_1,j_1}_{i_0,j_0}$, Eq.5.24 becomes

$$\sum_{i_1=\max(i_0-m+1,m)}^{\min(i_0+m-1,k-m+1)} \sum_{j_1=\max(j_0-n+1,n)}^{\min(j_0+n-1,l-n+1)} D^{k_0}_{i_1 j_1} \cdot h^{i_1,j_1}_{i_0,j_0} = f^{k_0}_{i_0,j_0}. \tag{Eq.5.26}$$

Suppose

$$g_{i_0,j_0}^{i_1,j_1} = \begin{cases} h_{i_0,j_0}^{i_1,j_1}, & |i_1 - i_0| \le m-1, |j_1 - j_0| \le n-1, \\ 0, & otherwise, \end{cases}$$

$$\mathbf{D}_{j_1}^{k_0} = \left(D_{m,j_1}^{k_0}, \cdots, D_{k-m+1,j_1}^{k_0}\right),$$

$$\mathbf{F}_{j_1}^{k_0} = \left(f_{m,j_1}^{k_0}, \cdots, f_{k-m+1,j_1}^{k_0}\right),$$

$$\mathbf{G}_{i_0 j_0}^{j_1} = \left(g_{i_0,j_0}^{m,j_1}, \cdots, g_{i_0,j_0}^{k-m+1,j_1}\right),$$

$$\mathbf{D}^{k_0} = \left(\mathbf{D}_{n}^{k_0}, \mathbf{D}_{n+1}^{k_0}, \cdots, \mathbf{D}_{k-n+1}^{k_0}\right),$$

$$\mathbf{F}^{k_0} = \left(\mathbf{F}_{n}^{k_0}, \mathbf{F}_{n+1}^{k_0}, \cdots, \mathbf{F}_{k-n+1}^{k_0}\right),$$

$$\mathbf{G}_{i_0 j_0} = \left(\mathbf{G}_{i_0 j_0}^{n}, \mathbf{G}_{i_0 j_0}^{n+1}, \cdots, \mathbf{G}_{i_0 j_0}^{k-n+1}\right).$$

Eq.5.26 becomes

$$\begin{aligned} f_{i_0,j_0}^{k_0} &= \sum_{j_1=n}^{l-n+1} \sum_{i_1=m}^{k-m+1} D_{i_1 j_1}^{k_0} \cdot g_{i_0,j_0}^{i_1,j_1} \\ &= \sum_{j_1=n}^{l-n+1} \mathbf{D}_{j_1}^{k_0} \cdot \mathbf{G}_{i_0 j_0}^{j_1} = \mathbf{D}^{k_0} \cdot \mathbf{G}^{i_0 j_0}. \end{aligned} \qquad \text{(Eq.5.27)}$$

Finally, Eq.5.22 can be rewritten in a matrix form

$$\mathbf{G} \cdot \mathbf{D}^{k_0} = \mathbf{F}^{k_0}, \quad k_0 = 0, 1, 2. \qquad \text{(Eq.5.28)}$$

where the matrix $\mathbf{G}$ is of size $N \times N$.

$\mathbf{G}_{i_0 j_0}$ is the K-th row of $\mathbf{G}$ with $K = i_0 - m + (j_0 - n)(k - 2m + 2)$. Since $\mathbf{G}_{i_0 j_0}$ contains only $(2m-1)(2n-1)$ nonzero value at most, $\mathbf{G}$ is a sparse matrix. By solving Eq.5.28, we can get A_{ij}.

5.7 Patch-based Monge mapping

As shown in Eq.5.12, the control point C_{ij} is derived by moving the control point B_{ij} along the direction d_{ij} and the coefficient α_{ij} decides the moving distance. For complete control points, instead of changing the value α_{ij} for each control point B_{ij}, we adjust only three parameters to obtain different α_{ij} for different B_{ij}.

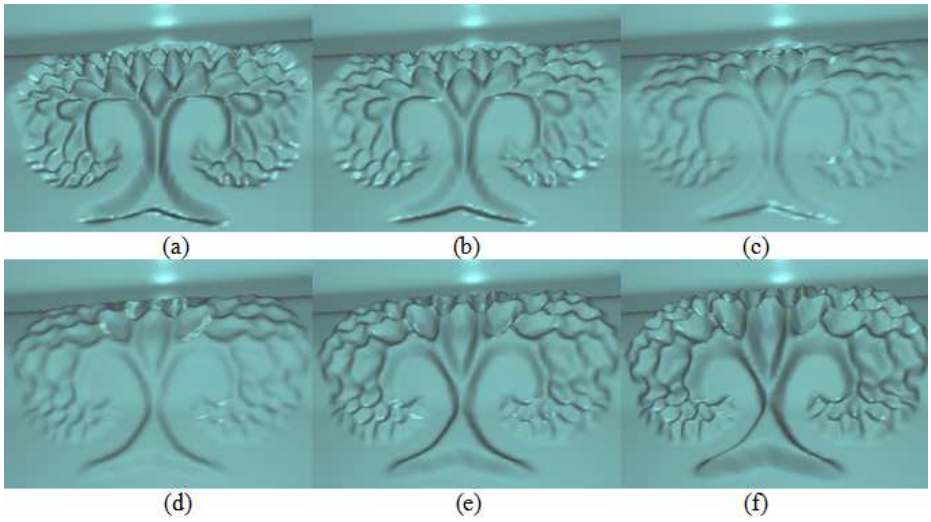


Figure 5.5: The rilievo and intaglio effects.
(a) $\eta = -1.5$; (b) $\eta = -1$; (c) $\eta = -0.5$; (d) $\eta = 0.5$; (e) $\eta = 1$; and (f) $\eta = 1.5$.

5.7.1 The coefficients

The control point C_{ij} is derived from the control point B_{ij}, which corresponds to the point $\mathbf{\Omega}_{ij}$. The curvature descriptor θ_{ijs} reflects the curvature of the area around $\mathbf{\Omega}_{ij}$. If θ_{ijs} is relatively high, a small change of the control point B_{ij} may change the surface a lot, hence α_{ij} should be smaller. Therefore we choose the normal distribution

$$\alpha_{ij} = \eta e^{-\lambda\theta_{ijs}^2}, \quad \eta, \lambda \in R, \quad \lambda \geq 0, \quad s = 0, 1, 2, 3, ... \qquad \text{(Eq.5.29)}$$

The three parameters η, λ and s decide the final shape of the C-patch.

5.7.2 The influence of parameter η

The parameter η affects the whole C-patch. Figure 5.5 shows an example of applying the A-patch shown in Figure 5.2(b). Different η values lead to different C-patches. If η is negative, the C-patch provides the intaglio effects (Figure 5.5(a-c)). Otherwise, if η is positive, rilievo effects can be obtained (Figure 5.5(d-f)).

5.7.3 The influence of parameter λ

The parameter λ affects the portion of the B-patch where the mean curvature is nonzero. Figure 5.6 shows an example of applying the A-patch shown in Figure 5.2(c). Figure 5.6(a) is the B-patch whose mean curvature is nonzero only at the central part.

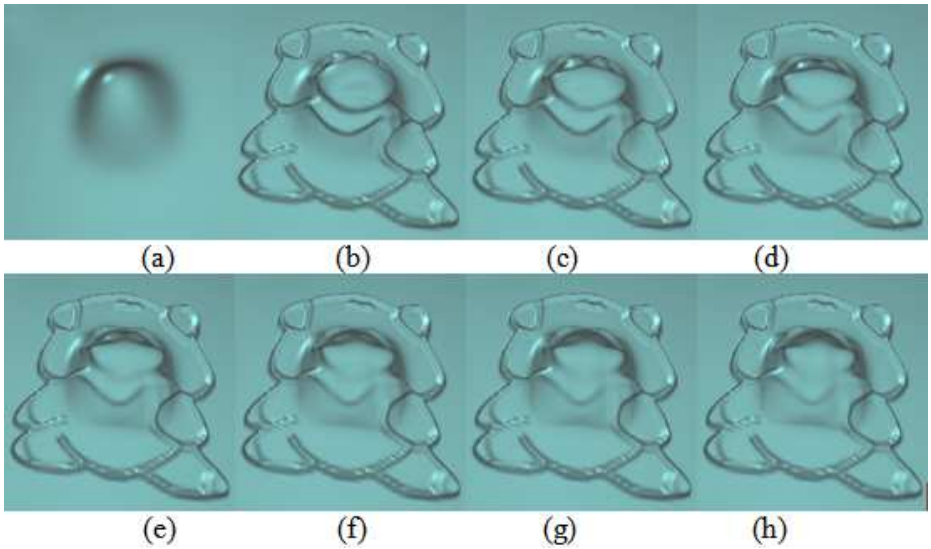


Figure 5.6: C-patch under different values of λ.
(a) the B-patch; (b) $\lambda = 0$; (c) $\lambda = 0.1$; (d) $\lambda = 0.2$; (e) $\lambda = 0.3$; (f) $\lambda = 0.4$; (g) $\lambda = 0.5$; and (h) $\lambda = 0.6$.

As the increase of λ, only the central part is changed. Compared with the features in the central part of Figure 5.6(b), the central part of Figure 5.6(h) is smoother.

5.7.4 The influence of parameter s

The parameter s controls the neighborhood area of each point. It determines the influence of the mean curvature, due to the fact that the mean curvature of each point affects not only the corresponding point but also the nearby points. Figure 5.7 shows an example of applying the A-patch shown in Figure 5.2(a). Figure 5.7(a,b) are two sides of the B-patch. The C-patch is derived with $\eta = 1, \lambda = 0, s = 0$ (Figure 5.7(c,d)). If we increase the η value from 1 to 3, the C-patch is self-intersecting (Figure 5.7(e,f)) where the original curvature is high. By increasing the λ value to 0.1, we can eliminate the self-intersection but we lost the details of the high curvature portion (Figure 5.7(g,h)). To change the shape in the high curvature portion, we can increase the s value to 5 (Figure 5.7(i,j)). Therefore, by selecting a combination of η, λ and s, we can eliminate the self-intersection and achieve desire shapes (Figure 5.7(k,l)).

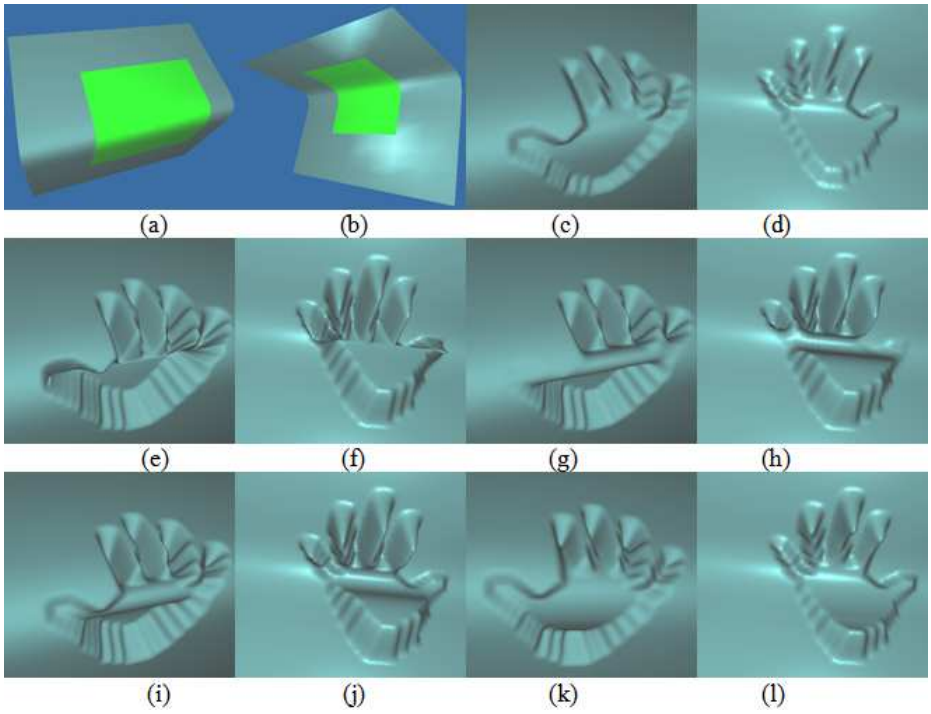


Figure 5.7: C-patch under different values of s.
(a,b) different views of the B-patch; (c,d) $\eta = 1, \lambda = 0, s = 0$; (e,f) $\eta = 3, \lambda = 0, s = 0$; (g,h) $\eta = 3, \lambda = 0.1, s = 0$; and (i,j) $\eta = 3, \lambda = 0, s = 5$; (k,l) $\eta = 4, \lambda = 0.1, s = 11$.

5.7.5 Cut-&-paste operations

In displacement mapping, displacement points are limited to move along its normal direction. Our Monge mapping allows the control points to move along any predefined directions. This is useful in cut-&-paste operations.

With the patch formation technique, we can easily identify a region or a feature (Figure 5.8(a)) as an A-patch from an existing model. The identification can be either done by selecting the region with two corner patches or with two corner points (Section 5.5). This A-patch will be represented by a Monge patch (Figure 5.8(b)). A B-patch (Figure 5.8(d)) can be similarly selected either using two corner patches or two corner points. However, direct cut of the A-patch and paste it to the B-patch may produce undesired results. For example, the selected A-patch can be much bigger than the B-patch in terms of scale. This can be seen from the reference patch (Figure 5.8(c)) of the A-patch and the green area with the B-patch (Figure 5.8(d)). The parameter control technique can be applied here to adjust the shape. H-NURBS mechanism enables cut-&-paste operations in an efficient and effective fashion.

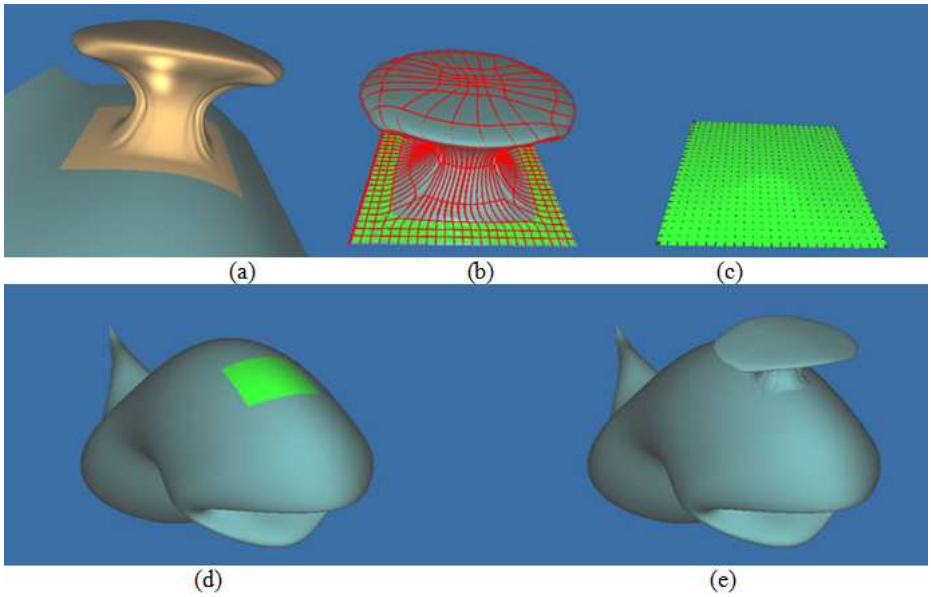


Figure 5.8: Cut-&-paste operations.

(a) a feature selected from a NURBS model; (b) the corresponding Monge patch with the reference patch in green and iso-curves in red; (c) the reference patch and its control points; (d) selected B-patch; and (e) the result of cut-&-paste operations.

5.7.6 Examples

We show some examples by using H-NURBS to achieve complex shape modeling. We can maintain the hierarchy and describe the whole surface using H-NURBS. H-NURBS based Monge mapping can be used to create special effects on existing models. The Monge mapping becomes easy if the selected B-patch is planar. When the B-patch is curved, we need to adjust three parameters in Eq.5.29. Different combinations of η, λ and s may lead to different shapes. A comparison of Monge mapping and texture mapping is listed in Figure 5.9.

Monge mapping can be used to either add detail to a surface or model an object. Figure 5.10 shows how we use Monge mapping to create a dolphin model. The H-NURBS created reaches level 2.

Figure 5.11 shows the Monge patches reconstructed from the images of the dinosaur fossils. These patches are mapped on to different parts of the torus (Figure 5.12). As a final example, Figure 5.13 shows how the Monge mapping is applied on a curved surface. Since the curvature of the B-patch varies a lot, we need to select a proper λ to avoid the self-intersections. The second row of Figure 5.13 shows the head portion of the dinosaur fossil. The mouth portion is divided into two parts to

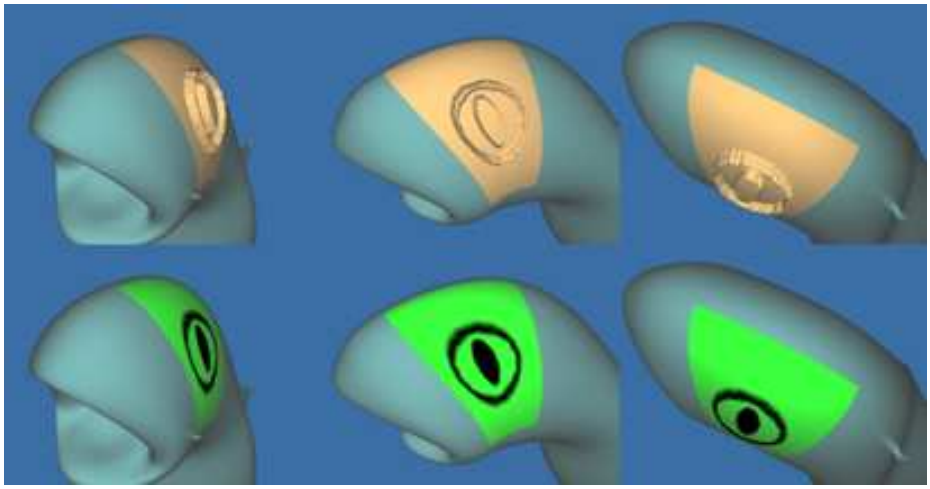


Figure 5.9: Monge mapping vs. texture mapping.
the first row is Monge mapping and the second row is texture mapping.

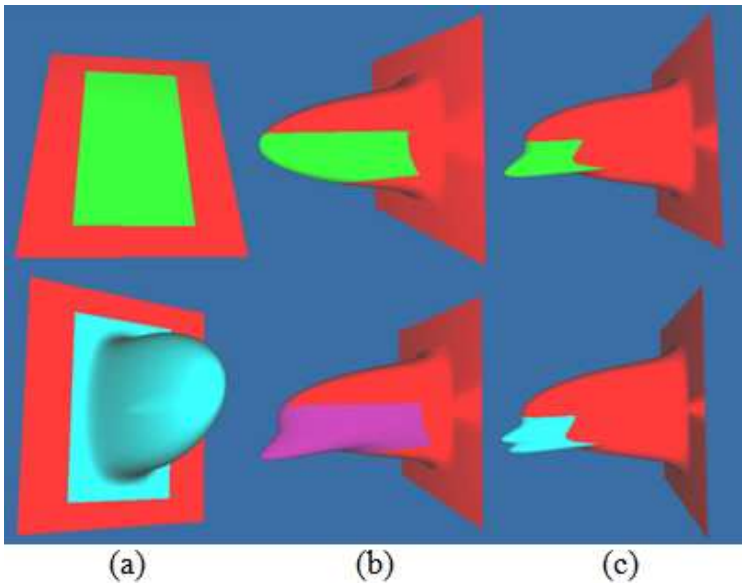


Figure 5.10: Creating an H-NURBS model for a dolphin.
the first row labels all the B-patches and the second row labels all the C-patches. (a) select a B-patch at level 0 and create the body; (b) select a B-patch at level 1 and modify the head part; and (c) select a B-patch at level 2 and modify the mouth part.

avoid self-intersection, which is similar to the example in Figure 5.7, where the hand print is divided into two.

5.8 Summary

In a 3D environment, NURBS is a proven tool for powerful shape modeling. NURBS represented shapes, however, can be difficult to end users (CAD or media designers) for local shape controls and local modifications to obtain a desired result due

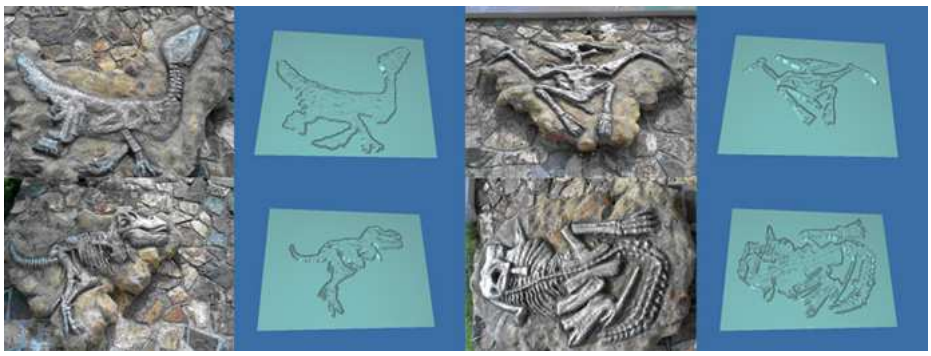


Figure 5.11: Monge patches for dinosaur fossils.

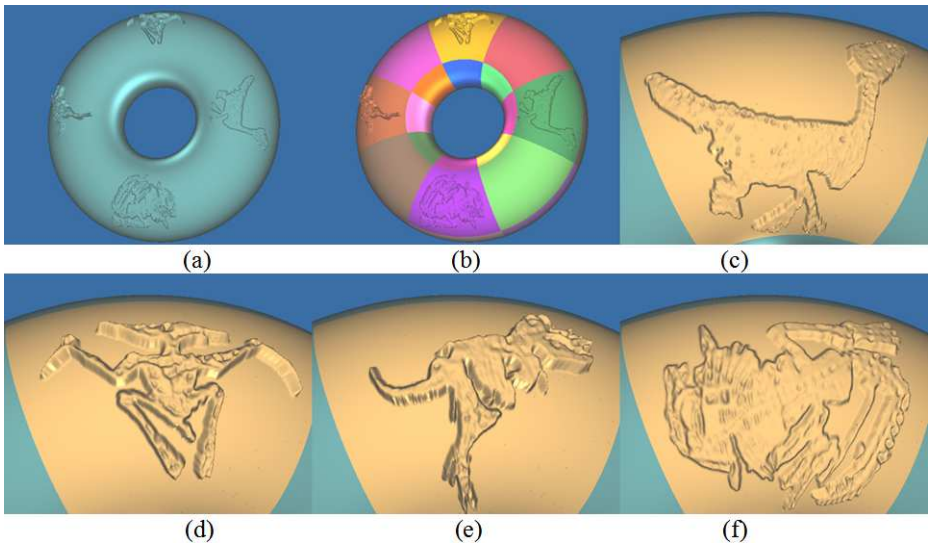


Figure 5.12: Torus of dinosaur fossils.

(a) Torus of dinosaur fossils; (b) H-NURBS patches; (c) the first fossil; (d) the second fossil; (e) the third fossil; and (f) the fourth fossil.

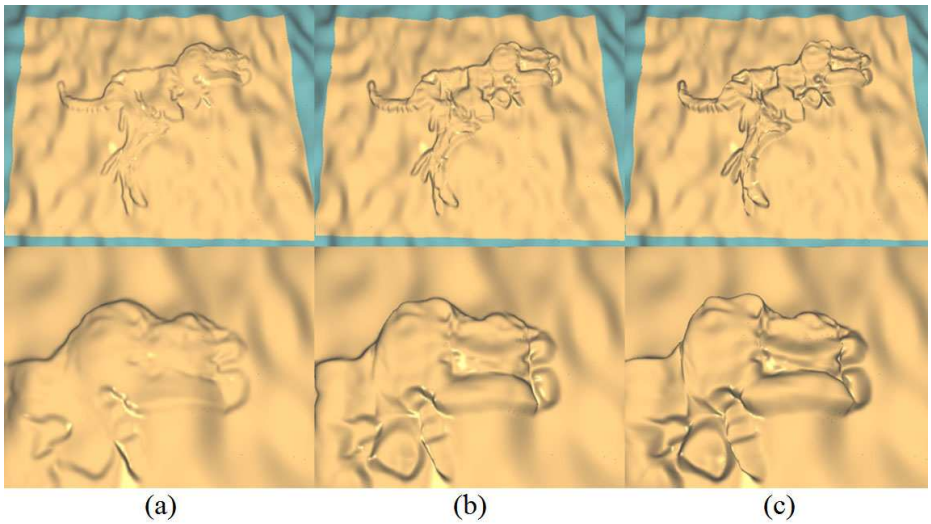


Figure 5.13: Monge mapping on a curved surface.

the second row is the head part of the first row. (a) $\eta = 0.2, \lambda = 0, s = 0$; (b) $\eta = 0.4, \lambda = 0.4, s = 1$; and (c) $\eta = 0.6, \lambda = 0.4, s = 2$.

to its complexity. This chapter proposes Monge mapping using parameters to do shape modifications basing on H-NURBS. Instead of direct modifying control points or feature-based design (e.g., corner rounding, extrusion, etc.), Monge mapping allows local shape construction by adjusting three parameters to achieve various effects. The local shape construction is achieved by adding height field information from an attaching patch (A-patch) onto a base patch (B-patch). The B-patch is the local shape to modify and the A-patch can be derived from any 2D image or from existing NURBS model. With such mechanism, cut-&-paste operations become easy.

Chapter 6

Conclusions and Future Work

This thesis discusses the modeling of four types of complex surfaces: molecular surfaces, minimal surfaces, composite surfaces, and H-NURBS surfaces. Figure 6.1 shows the screen snap shots of four programs developed for the modeling of the four types of complex surfaces. Each program is developed with:

- Implementation platform: Normal PC.
- Coding platform: Microsoft Visual C++ 2005.
- Operating system: Windows XP.
- CPU: Intel(R) Core(TM)2 Duo CPU T8100.
- RAM: 2GB.

6.1 Contributions

We develop a set of algorithms to explore the techniques for complex surface modeling. Four types of complex surfaces are covered in this thesis. The followings are our contributions:

6.1.1 Molecular surfaces

We have proposed a uniform approach to modeling three types of molecular surfaces. Each molecular surface (vdWS, SES and SAS) can be created through topology mod-

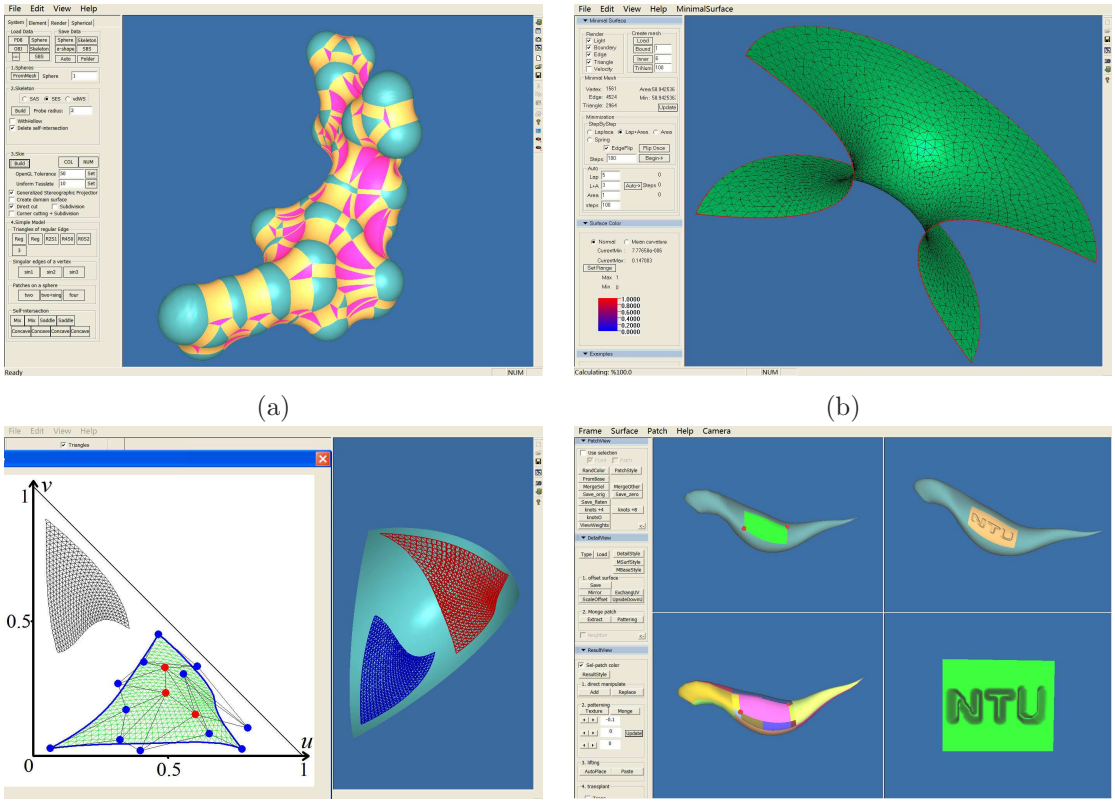


Figure 6.1: Screen snap shots of the four complex surfaces
(a) molecular surface; (b) minimal surface; (c) composite surface; and (d) H-NURBS.

eling, boundary modeling and surface modeling. In the first step, topology modeling creates two topological networks using a weighted α-shape. The networks are used to maintain the neighboring relationship of the atoms. The vdWS network is used for the vdWS. The solvent network is used for the SES and the SAS. For each network, there are boundary vertices, edges and triangles. In the second step, three types of boundary arcs are created: SES arcs, SAS arcs and vdWS arcs. SES arcs and SAS arcs are created from the solvent network. From the vdWS network, vdWS arcs are created. Singularity processing is used to further modify the SES arcs by removing the self-intersections. In the third step, all the arc-bounded patches are modeled in rational Bézier form. Each molecular surface is modeled as a collection of rational Bézier surfaces.

6.1.2 Minimal surfaces

We have proposed an algorithm to construct triangular meshes of minimal area from all possible triangular meshes with the prescribed boundary and number of triangles. The core techniques of the algorithm are three processes: area minimizing, Laplacian fairing, and edge swapping. They are combined to provide an automatic approach. The algorithm has been shown by the examples to be reliable and effective. On the other hand, since these three processes can be done very fast, they can be used in interactive environments. Especially in digital geometry modeling, after users sketch the boundary and specify the level of detail, the three processes can then be interactively performed in various orders to achieve users' specific requirements.

6.1.3 Composite surfaces

An approach to generate a TB sub-patch from a TB surface is presented. Firstly, the TB sub-patch can be formed by composition of the TB surface and the domain surface. Secondly, an explicit formula for computing the control points of the composition is derived. These new control points are the linear combinations of the control points of the TB surface while the coefficients are given by the parameter points of the domain surface. Thirdly, the geometric algorithm for the power points is analyzed. The power points can be calculated using the Pyramid algorithms. Then the control points of the TB sub-patch are the linear combinations of these power points. Several examples are examined.

6.1.4 H-NURBS surfaces

In a 3D environment, NURBS is a proven tool for powerful shape modeling. NURBS represented shapes, however, can be difficult to end users (CAD or media designers) for local shape controls and local modifications to obtain a desired result due to its complexity. This thesis proposes Monge mapping using parameters to do shape modifications basing on H-NURBS. Instead of direct modifying control points or feature-based design (e.g., corner rounding, extrusion, etc.), Monge mapping allows local shape construction by adjusting three parameters to achieve various effects. The

local shape construction is achieved by adding height field information from an attaching patch (A-patch) onto a base patch (B-patch). The B-patch is the local shape to modify and the A-patch can be derived from any 2D image or from existing NURBS model. With such mechanism, the cut-&-paste operations become easy.

6.2 Future Work

6.2.1 Future work for molecular surfaces

In Section 2.8, a rectangular spherical patch, bounded by four arcs, is modeled as a rational Bézier surface $\mathbf{P}(u, v)$ of degree $(2, 4)$ following the idea of generalized stereographic projection (GSP) [12, 13].

$$\mathbf{P}(u,v) = \frac{\sum_{i=0}^{2}\sum_{j=0}^{4} w_{ij}P_{ij}N_i(u)N_j(v)}{\sum_{i=0}^{2}\sum_{j=0}^{4} w_{ij}N_i(u)N_j(v)},$$

where P_{ij} are control points and w_{ij} are weights. For each patch, there are 15 control points and 15 weights. GSP method first describes the unit sphere as a rational surface $\mathbf{S}$ of degree $(2, 2)$ and derives a surface $\mathbf{Q}$ of degree $(1, 2)$ from the four boundary arcs. Finally, the spherical patch is obtained as a composition of $\mathbf{S}$ and $\mathbf{Q}$. Such a composition can not guarantee that all 15 weights are positive. It is useful to find out one method to provide all positive weights.

The uniform model builds a network to derive a molecular surface. Since the atoms always keep moving, the network of the molecule will dynamically change. Consequently, the molecular surface will also dynamically change. With the current method, every change of the molecule leads to the re-calculation of the network and molecular surface. Therefore, it is necessary to find our a dynamic algorithm to simplify the calculation.

The uniform model can be used in ligand-protein docking. We need to develop a new docking algorithm. The algorithm may include some aspects: identify pockets, locate the binding sites, define a new scoring function and take biochemical properties

(hydrophobic/hydrophilic, electrostatic, etc.) of the molecule into account. Then, we also have to compare our algorithm with the existing algorithms.

6.2.2 Future work for minimal surfaces

The algorithm for minimal surfaces has some theoretical issues we are yet to address. One of them is the convergence analysis of the algorithm. Though the algorithm succeeds for all our testing examples, we do not know exactly what conditions guarantee the convergence. Meanwhile, the existence and uniqueness properties of our minimal surface is another issue. The existence of triangular meshes of minimal area can be proved. In fact, the area functional (Eq.3.3) can be considered as a continuous real function defined on $R^{3(N-n)}$ since we have $N-n$ free vertices $P_{n+1}, \cdots, P_N$ and each vertex has three coordinates. The area cannot be negative and thus it is bounded from below. Furthermore, when we look for a minimum, we can restrict the function to a suitable compact subset such that if a vertex goes outside of the compact subset, then the area functional becomes greater than some bound. Thus calculus affirms that a minimum exists and it can be attained.

Applications of the algorithm to other fields are also possible and we are currently studying the application to mesh fairing and modeling of human vessel network. In our current algorithm, area minimization is equivalent to minimize the mean curvature of each point to zero. More general, a constant mean curvature (CMC) problem is to provide a surface with mean curvature at each point equal to a constant value. This problem is equivalent to minimize the area functional while preserving volume. Further development of our algorithm will be able to model CMC surface.

6.2.3 Future work for composite surfaces

Three boundary curves connected end-to-end on the domain defined a trimmed surface from a triangular Bernstein-Bézier surface. The trimmed surface is a sub-patch of the TB surface, and it can be obtained via composition. However, the control points of the sub-patch can not be uniquely decided by the trimmed curves. Therefore, there are lots of Bézier representations for the sub-patch. From the application point of

view, by assigning three trimmed curves, we should give a solution. Coupling with minimal energy conditions might be a future research direction to get an optimal solution to trim a sub-patch from the triangular Bézier surface.

6.2.4 Future work for H-NURBS surfaces

The Monge mapping provides easy cut-&-paste operations. It can cut a shape from one existing model (A-patch) and paste the shape onto a local area of another model (B-patch). The result (C-patch) is dynamically control by three parameters. It will not be easy to make A-patch and C-patch exactly the same during cut-&-paste operations. Hence, a new mechanism for transplanting purpose can be a new research area. Coupling with some techniques such as the surface blending and the Poisson operator, it is possible to make the C-patch exactly the same as an A-patch if exact shapes are pasted after the transplantation.

References

[1] Biederman, I. (1987). Recognition-by-Components: A Theory of Human Image Understanding. *Psychological review*, 94(2), 115–147.

[2] Homan, D. & Richards, W. (1985). Parts of Recognition. *Cognition*, 18, 65–96.

[3] Palmer, S. (1977). Hierarchical Structure in Perceptual Representation. *Cognitive Psychology*, 9(4), 441–474.

[4] Marr, D. & Nishihara, H. (1978). Representation and Recognition of the Spatial Organization of Three-Dimensional Shapes. In *Proceedings of the Royal Society of London. Series B. Biological sciences*, (269–294).

[5] Rom, H. & Medioni, G. (1993). Hierarchical Decomposition and Axial Shape Description. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(10), 973–981.

[6] Pasupathy, A. & Connor, C. (2001). Shape Representation in Area V4: Position-Specific Tuning for Boundary Conformation. *Journal of Neurophysiology*, 86(5), 2505.

[7] Pasupathy, A. & Connor, C. (1999). Responses to Contour Features in Macaque Area V4. *Journal of Neurophysiology*, 82(5), 2490–2502.

[8] Pasupathy, A. & Connor, C. (2002). Population Coding of Shape in Area V4. *Nature Neuroscience*, 5(12), 1332–1338.

[9] Wikipedia. *Visual cortex*. `http://en.wikipedia.org/wiki/Visual_cortex` (accessed on August 1, 2010).

[10] Putnam, L. & Subrahmanyam, P. (1986). Boolean Operations on n-Dimensional Objects. *IEEE Computer Graphics and Applications*, 6(6), 43–51.

[11] Mortenson, M. (2006). *Geometric Modeling*. Industrial Press.

[12] Dietz, R., Hoschek, J., & Jüttler, B. (1993). An Algebraic Approach to Curves and Surfaces on the Sphere and on Other Quadrics. *Computer Aided Geometric Design*, 10(3-4), 229.

[13] Dietz, H. & Jüttler, B. (1995). Rational Patches on Quadric Surfaces. *Computer-Aided design*, 27(1), 27–40.

[14] Connolly, M. (1983). Analytical Molecular Surface Calculation. *Journal of Applied Crystallography*, 16(5), 548–558.

[15] Hua, W., Bao, H., & Peng, Q. (2002). Hierarchical Surface Fragments. *Progress in Natural Science*, 12(9), 701–709.

[16] Samavati, F., Bartels, R., & Olsen, L. (2007). Local B-spline Multiresolution with Example in Iris Synthesis and Volumetric Rendering. *Series in Machine Perception and Artificial Intelligence*, 67, 65.

[17] Bash, P., Pattabiraman, N., Huang, C., Ferrin, T., & Langridge, R. (1983). Van der Waals Surfaces in Molecular Modeling: Implementation with Real-Time Computer Graphics. *Science*, 222, 1325–1327.

[18] Lee, B. & Richards, F. (1971). The Interpretation of Protein Structures: Estimation of Static Accessibility. *Journal of Molecular Biology*, 55(3), 379–380.

[19] Whitley, D. (1998). Van der Waals Surface Graphs and Molecular Shape. *Journal of Mathematical Chemistry*, 23(3), 377–397.

[20] Whitley, D. (1998). Van der Waals Surface Graphs and the Shape of Small Rings. *Journal of Chemical Information and Computer Sciences*, 38(5), 906–914.

[21] Ugliengo, P., Viterbo, D., & Chiari, G. (1993). MOLDRAW: Molecular Graphics on a Personal Computer. *Zeitschrift für Kristallographie*, 207(1), 9–23.

[22] Adams, G., O'Keeffe, M., & Ruoff, R. (1994). Van der Waals Surface Areas and Volumes of Fullerenes. *Journal of Physical Chemistry*, 98(38), 9465–9469.

[23] Hermann, R. (1972). Theory of Hydrophobic Bonding. II. Correlation of Hydrocarbon Solubility in Water with Solvent Cavity Surface Area. *The Journal of Physical Chemistry*, 76(19), 2754–2759.

[24] Connolly, M. (1983). Solvent-accessible Surfaces of Proteins and Nucleic Acids. *Science*, 221(4612), 709–713.

[25] Zauhar, R. (1995). SMART: a Solvent-Accessible Triangulated Surface Generator for Molecular Graphics and Boundary Element Applications. *Journal of Computer-Aided Molecular Design*, 9(2), 149–159.

[26] Juffer, A. & Vogel, H. (1998). A Flexible Triangulation Method to Describe the Solvent-Accessible Surface of Biopolymers. *Journal of Computer-Aided Molecular Design*, 12(3), 289–299.

[27] Deanda, F. & Pearlman, R. (2002). A Novel Approach for Identifying the Surface Atoms of Macromolecules. *Journal of Molecular Graphics and Modelling*, 20(5), 415–425.

[28] Alden, C. & Kim, S. (1979). Solvent-Accessible Surfaces of Nucleic Acids. *Journal of molecular biology*, 132(3), 411.

[29] Sridharan, S., Nicholls, A., & Sharp, K. (1995). A Rapid Method for Calculating Derivatives of Solvent Accessible Surface Areas of Molecules. *Journal of Computational Chemistry*, 16(8), 1038–1044.

[30] Gogonea, V. & Osawa, E. (1994). Implementation of Solvent Effect in Molecular Mechanics. III: The First- and Second-Order Analytical Derivatives of Excluded Volume. *Journal of molecular structure - Theochem*, 117, 305–324.

[31] Weiser, J., Shenkin, P., & Still, W. (1999). Approximate Solvent - Accessible Surface Areas from Tetrahedrally Directed Neighbor Densities. *Peptide Science*, 50(4), 373–380.

[32] Guvench, O. & Brooks, C. (2004). Efficient Approximate All - Atom Solvent Accessible Surface Area Method Parameterized for Folded and Denatured Protein Conformations. *Journal of computational chemistry*, 25(8), 1005–1014.

[33] Hou, T., Qiao, X., Zhang, W., & Xu, X. (2002). Empirical Aqueous Solvation Models Based on Accessible Surface Areas with Implicit Electrostatics. *The Journal of Physical Chemistry B*, 106(43), 11295–11304.

[34] Richards, F. (1977). Areas, Volumes, Packing, and Protein Structure. *Annual Review of Biophysics and Bioengineering*, 6(1), 151–176.

[35] Sanner, M., Olson, A., & Spehner, J. (1996). Reduced Surface: an Efficient Way to Compute Molecular Surfaces. *Peptide Science*, 38(3), 305–320.

[36] Sanner, M., Olson, A., & Spehner, J. (1995). Fast and Robust Computation of Molecular Surfaces. In *Proceedings of the Eleventh Annual Symposium on Computational Geometry*, (406–407).

[37] Ryu, J., Cho, Y., Park, R., Seo, J., & Kim, D. (2007). Beta-Shape Based Computation of Blending Surfaces on a Molecule. In *ISVD 2007: The 4th International Symposium on Voronoi Diagrams in Science and Engineering 2007*, (189–198).

[38] Ryu, J., Park, R., & Kim, D. (2007). Molecular Surfaces on Proteins via Beta Shapes. *Computer-Aided Design*, 39(12), 1042–1057.

[39] Blinn, J. (1982). A generalization of algebraic surface drawing. *ACM Transactions on Graphics (TOG)*, 1(3), 235–256.

[40] Wyvill, G., McPheeters, C., & Wyvill, B. (1986). Soft Objects. In *Proceedings of Computer Graphics Tokyo on Advanced Computer Graphics*, (113–128).

[41] Wyvill, G., McPheeters, C., & Wyvill, B. (1986). Data structure for soft objects. *The visual computer*, 2(4), 227–234.

[42] Zhang, Y., Xu, G., & Bajaj, C. (2006). Quality meshing of implicit solvation models of biomolecular structures. *Computer aided geometric design*, 23(6), 510–530.

[43] Lorensen, W. & Cline, H. (1987). Marching cubes: A high resolution 3D surface construction algorithm. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, (163–169).

[44] Edelsbrunner, H. (1999). Deformable smooth surface design. *Discrete and Computational Geometry*, 21(1), 87–115.

[45] Edelsbrunner, H. & Üngör, A. (2003). Relaxed scheduling in dynamic skin triangulation. *Discrete and Computational Geometry*, (135–151).

[46] Cheng, H., Dey, T., Edelsbrunner, H., & Sullivan, J. (2001). Dynamic skin triangulation. *Discrete and Computational Geometry*, 25(4), 525–568.

[47] Cheng, H. & Shi, X. (2005). Guaranteed quality triangulation of molecular skin surfaces. In *Visualization, 2004. IEEE*, (481–488). IEEE.

[48] Cheng, H. & Shi, X. (2009). Quality mesh generation for molecular skin surfaces using restricted union of balls. *Computational Geometry*, 42(3), 196–206.

[49] Boissonnat, J. & Oudot, S. (2003). Provably good surface sampling and approximation. In *Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, (9–18). Eurographics Association.

[50] Kruithof, N. & Vegter, G. (2007). Meshing skin surfaces with certified topology. *Computational Geometry*, 36(3), 166–182.

[51] Max, N. (1982). Computer Representation of Molecular Surfaces. *Journal of Medical Systems*, 6(5), 485–499.

[52] Moon, J. & Howe, W. (1989). A Fast Algorithm for Generating Smooth Molecular Dot Surface Representations. *Journal of molecular graphics*, 7(2), 109.

[53] Weber, J., Morgantini, P., Fluekiger, P., & Roch, M. (1989). Molecular Graphics Modeling of Organometallic Reactivity. *Computers & Graphics*, 13(2), 229–235.

[54] Zauhar, R. & Morgan, R. (1990). Computing the Electric Potential of Biomolecules: Application of a New Method of Molecular Surface Triangulation. *Journal of Computational Chemistry*, 11(5), 603–622.

[55] Bajaj, C., Lee, H., Merkert, R., & Pascucci, V. (1997). NURBS Based B-rep Models for Macromolecules and their Properties. In *Proceedings of the Fourth ACM Symposium on Solid Modeling and Applications*, (217–228).

[56] Bajaj, C., Pascucci, V., Shamir, A., Holt, R., & Netravali, A. (2003). Dynamic Maintenance and Visualization of Molecular Surfaces. *Discrete Applied Mathematics*, 127(1), 23–51.

[57] Akenine-Moller, T., Moller, T., & Haines, E. (2002). *Real-Time Rendering*. A.K. Peters, Ltd.

[58] Zhao, W., Xu, G., & Bajaj, C. (2007). An Algebraic Spline Model of Molecular Surfaces. In *Proceedings of the 2007 ACM Symposium on Solid and Physical Modeling*, (302).

[59] Aurenhammer, F. (1987). Power Diagrams - Properties, Algorithms, and Applications. *SIAM Journal on Computing*, 16(1), 78–96.

[60] Kim, D., Seo, J., Kim, D., Ryu, J., & Cho, C. (2006). Three-Dimensional Beta Shapes. *Computer-Aided Design*, 38(11), 1179–1191.

[61] Edelsbrunner, H. (1992). Weighted Alpha Shapes. Technical report, Technical Report UIUCDCS.

[62] Edelsbrunner, H. (1995). The Union of Balls and its Dual Shape. *Discrete and Computational Geometry*, 13(1), 415–440.

[63] Laug, P. & Borouchaki, H. (2002). Molecular Surface Modeling and Meshing. *Engineering with Computers*, 18(3), 199–210.

[64] Fabri, A. & Pion, S. (2009). CGAL: the Computational Geometry Algorithms Library. In *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, (538–539).

[65] Wang, G. (1991). Rational Cubic Circular Arcs and their Application in CAD. *Computers in Industry*, 16(3), 288.

[66] Eyal, E. & Halperin, D. (2005). Dynamic Maintenance of Molecular Surfaces under Conformational Changes. In *Proceedings of the Twenty-first Annual Symposium on Computational Geometry*, (54).

[67] Halperin, D. & Shelton, C. (1998). A Perturbation Scheme for Spherical Arrangements with Application to Molecular Modeling. *Computational Geometry: Theory and Applications*, 10(4), 273–287.

[68] Sussman, J., Lin, D., Jiang, J., Manning, N., Prilusky, J., Ritter, O., & Abola, E. (1998). Protein Data Bank (PDB): Database of Three-Dimensional Structural Information of Biological Macromolecules. *Acta Crystallographica. Section D: Biological Crystallography*, 54(6), 1078–1084.

[69] Rost, R. (2006). *OpenGL® Shading Language*. Addison Wesley.

[70] Moreton, H. & Sequin, C. (1992). Functional Minimization for Fair Surface Design. In *Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques*, (167–176).

[71] O'Rourke, J. (1981). Polyhedra of Minimal Area as 3D Object Models. In *Proceedings of the International Joint Conference on Artificial Intelligence*, (664–666).

[72] Fuchs, H., Kedem, Z., & Uselton, S. (1977). Optimal Surface Reconstruction from Planar Contours. *Communications of the ACM*, 20(10), 693–702.

[73] Lawson, H. (1980). *Lectures on Minimal Submanifolds*. Publish or Perish in Berkeley.

[74] Rado, T. (1930). On Plateau's Problem. *Annals of Mathematics*, 31(3), 457–469.

[75] Douglas, J. (1931). Solution of the Problem of Plateau. *Transactions of the American Mathematical Society*, 33(1), 263–321.

[76] Douglas, J. (1928). A Method of Numerical Solution of the Plateau Problem. *Annals of Mathematics*, 29(2), 180–188.

[77] Wilson, W. (1961). On Discrete Dirichlet and Plateau Problems. *Numerische Mathematik*, 3(1), 359–373.

[78] Wang, H., Qin, Q. H., & Arounsavat, D. (2007). Application of Hybrid Trefftz Finite Element Method to Non-Linear Problems of Minimal Surface. *International Journal for Numerical Methods in Engineering*, 69(6), 1262–1277.

[79] Tsuchiya, T. (1986). On Two Methods for Approximating Minimal Surfaces in Parametric Form. *Mathematics of Computation*, 46(174), 517–529.

[80] Tsuchiya, T. (1987). Discrete Solution of the Plateau Problem and its Convergence. *Mathematics of Computation*, 49(179), 157–165.

[81] Tsuchiya, T. (1990). A Note on Discrete Solutions of the Plateau-Problem. *Mathematics of Computation*, 54(189), 131–138.

[82] Dziuk, G. (1991). An Algorithm for Evolutionary Surfaces. *Numerische Mathematik*, 58(6), 603–611.

[83] Hinze, M. (1997). On the Numerical Approximation and Computation of Minimal Surface Continua Bounded by One-Parameter Families of Polygonal Contours. *Applied Numerical Mathematics*, 25(1), 89–116.

[84] Arnal, A., Lluch, A., & Monterde, J. (2003). Triangular Bézier Surfaces of Minimal Area. *Computational Science and its Applications - ICCSA 2003*, (986–986).

[85] Monterde, J. (2004). Bézier Surfaces of Minimal area: the Dirichlet Approach. *Computer Aided Geometric Design*, 21(2), 117–136.

[86] Cosín, C. & Monterde, J. (2002). Bézier Surfaces of Minimal Area. *Computational Science–ICCS 2002*, (72–81).

[87] Taubin, G. (1995). A Signal Processing Approach to Fair Surface Design. In *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*, (351–358).

[88] Desbrun, M., Meyer, M., Schröder, P., & Barr, A. (1999). Implicit Fairing of Irregular Meshes Using Diffusion and Curvature Flow. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, (324).

[89] Schroder, P. & Sweldens, W. (2001). Digital Geometry Processing. In *Sixth Annual Symposium on Frontiers of Engineering*, (41–44).

[90] Pinkall, U. & Polthier, K. (1993). Computing Discrete Minimal Surfaces and their Conjugates. *Experimental Mathematics*, 2(1), 15–36.

[91] Callahan, M., Hoffman, D., & Hoffman, J. (1988). Computer Graphics Tools for the Study of Minimal Surfaces. *Communications of the ACM*, 31(6), 661.

[92] Nitsche, J. (1989). *Lectures on Minimal Surfaces*. Cambridge University Press New York.

[93] Dyn, N., Levin, D., & Rippa, S. (1990). Data Dependent Triangulations for Piecewise Linear Interpolation. *IMA Journal of Numerical Analysis*, 10(1), 137.

[94] Baszenski, G. & Schumaker, L. (1991). Use of Aimulated Annealing to Construct Triangular Facet Surfaces. In *Curves and Surfaces*, (27–32). Boston: Academic Press.

[95] Meyer, M., Desbrun, M., Schröder, P., & Barr, A. (2002). Discrete Differential-Geometry Operators for Triangulated 2-Manifolds. *Visualization and Mathematics*, 3, 35–57.

[96] DeRose, T. (1988). Composing Bézier Simplexes. *ACM Transactions on Graphics*, 7(3), 221.

[97] Jüttler, B. & Wang, W. (2003). The shape of spherical quartics. *Computer Aided Geometric Design*, 20(8-9), 621–636.

[98] Hu, S., Wang, G., & Jin, T. (1996). Generalized Subdivision of Bézier Surfaces. *Graphical Models and Image Processing*, 58(3), 218–222.

[99] Brueckner, I. (1980). Construction of Bézier Points of Quadrilaterals from those of Triangles. *Computer-Aided Design*, 12(1), 21–24.

[100] Waggenspack Jr, W. & Anderson, D. (1986). Converting Standard Bivariate Polynomials to Bernstein Form over Arbitrary Triangular Regions. *Computer-Aided Design*, 18(10), 529–532.

[101] Jie, T. (1990). A Geometric Condition for Smoothness between Adjacent Rational Bézier Surfaces. *Computers in Industry*, 13(4), 355–360.

[102] Goldman, R. & Filip, D. (1987). Conversion from Bézier Rectangles to Bézier Triangles. *Computer-Aided Design*, 19(1), 25–27.

[103] Hu, S. (1996). Conversion of a Triangular Bézier Patch into Three Rectangular Bézier Patches. *Computer Aided Geometric Design*, 13(3), 219–226.

[104] Sheng, X. & Hirsch, B. (1992). Triangulation of Trimmed Surfaces in Parametric Space. *Computer-Aided Design*, 24(8), 437–444.

[105] Lasser, D. (2002). Tensor Product Bézier surfaces on Triangle Bézier Surfaces. *Computer Aided Geometric Design*, 19(8), 625 – 643.

[106] Feng, J. & Peng, Q. (1999). Functional Compositions via Shifting Operators for Bézier Patches and Their Applications. *Chinese Journal of Software*, 10(12), 1316–1322.

[107] Farm, G. (1986). Triangular Bernstein-Bézier Patches. *Computer Aided Geometric Design*, 3, 83–127.

[108] Chang, G. (1984). Bernstein Polynomials via the Shifting Operator. *American Mathematical Monthly*, 91(10), 634–638.

[109] Farin, G. & Farin, G. (2002). *Curves and Surfaces for CAGD: a Practical Guide*. Morgan Kaufmann Pub.

[110] Goldman, R. (2003). *Pyramid Algorithms: A Dynamic Programming Approach to Curves and Surfaces for Geometric Modeling*. Morgan Kaufmann Pub.

[111] Chang, G. & Davis, P. (1984). The Convexity of Bernstein Polynomials over Triangles. *Journal of Approximation Theory*, 40(1), 11–28.

[112] Catmull, E. (1974). *A Subdivision Algorithm for Computer Display of Curved Surfaces*. Ph.D. thesis, The University of Utah.

[113] Heckbert, P. (1989). *Fundamentals of Texture Mapping and Image Warping*. Master's thesis, University of California at Berkeley.

[114] Blinn, J. (1978). Simulation of Wrinkled Surfaces. In *Proceedings of the 5th Annual Conference on Computer Graphics and Interactive Techniques*, (286–292).

[115] Cook, R. (1984). Shade Trees. In *Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques*, (223–231).

[116] Cook, R., Carpenter, L., & Catmull, E. (1987). The Reyes Image Rendering Architecture. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, (102).

[117] Piegl, L. & Tiller, W. (1997). *The NURBS Book*. Springer Verlag.

[118] Blinn, J. & Newell, M. (1976). Texture and Reflection in Computer Generated Images. *Communications of the ACM*, 19(10), 542–547.

[119] Gardner, G. (1985). Visual Simulation of Clouds. *ACM SIGGRAPH Computer Graphics*, 19(3), 297–304.

[120] Miller, G. & Hoffman, C. (1984). Illumination and Reflection Maps: Simulated Objects in Simulated and Real Environments. In *SIGGRAPH 84 Advanced Computer Graphics Animation seminar notes*, volume 190.

[121] Fournier, A. (1992). Normal Distribution Functions and Multiple Surfaces. In *Graphics Interface'92 Workshop on Local Illumination*, (45–52).

[122] Tong, X., Wang, L., & Wang, X. (2003). View-Dependent Displacement Mapping. *ACM Transactions on Graphics*, 22(3), 334.

[123] Oliveira, M., Bishop, G., & McAllister, D. (2000). Relief Texture Mapping. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, (359–368).

[124] Williams, L. (1983). Pyramidal Parametrics. In *Proceedings of the 10th Annual Conference on Computer Graphics and Interactive Techniques*, (1–11).

[125] Bartels, R., Beatty, J., & Barsky, B. (1995). *An Introduction to Splines for Use in Computer Graphics and Geometric Modeling*. Morgan Kaufmann Pub.

[126] Cohen, E., Lyche, T., & Riesenfeld, R. (1980). Discrete B-splines and Subdivision Techniques in Computer-Aided Geometric Design and Computer Graphics. *Computer Graphics and Image Processing*, 14(2), 87–111.

[127] Lane, J. & Riesenfeld, R. (1980). A Theoretical Development for the Computer Generation and Display of Piecewise Polynomial Surfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2(1), 35–46.

[128] Forsey, D. & Bartels, R. (1988). Hierarchical B-spline Refinement. *ACM SIGGRAPH Computer Graphics*, 22(4), 205–212.

[129] Olsen, L., Samavati, F., & Bartels, R. (2005). Multiresolution B-splines Based On Wavelet Constraints. In *poster presentation at the Third Eurographics Symposium on Geometry Processing*.

[130] Olsen, L., Samavati, F., & Bartels, R. (2007). Multiresolution for Curves and Surfaces Based on Constraining Wavelets. *Computers & Graphics*, 31(3), 449–462.

[131] Gonzalez-Ochoa, C. & Peters, J. (1999). Localized-Hierarchy Surface Splines (LeSS). In *Proceedings of the 1999 Symposium on Interactive 3D Graphics*, (7–15).

[132] Yvart, A., Hahmann, S., & Bonneau, G. (2005). Hierarchical Triangular Splines. *ACM Transactions on Graphics*, 24(4), 1374–1391.

[133] Yvart, A., Hahmann, S., & Bonneau, G. (2005). Smooth Adaptive Fitting of 3D Models Using Hierarchical Triangular Splines. In *International Conference on Shape Modeling and Applications*, (13–22).

[134] Forsey, D. & Bartels, R. (1995). Surface Fitting with Hierarchical Splines. *ACM Transactions on Graphics*, 14(2), 134–161.

[135] Chen, W., Cai, Y., & Zheng, J. (2008). Generalized Hierarchical NURBS for Interactive Shape Modification. In *Proceedings of The 7th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and Its Applications in Industry*, (24).

[136] Mørken, K. (1991). Some Identities for Products and Degree Raising of Splines. *Constructive Approximation*, 7(1), 195–208.

[137] Strøm, K. (1993). Products of B-patches. *Numerical Algorithms*, 4(3), 323–337.