2008

PARAMETRIZATION AND SHAPE RECONSTRUCTION TECHNIQUES FOR DOO-SABIN SUBDIVISION SURFACES

Jiaxi Wang
University of Kentucky, jessie0927@hotmail.com

ABSTRACT OF THESIS

PARAMETRIZATION AND SHAPE RECONSTRUCTION TECHNIQUES FOR DOO-SABIN SUBDIVISION SURFACES

This thesis presents a new technique for the reconstruction of a smooth surface from a set of 3D data points. The reconstructed surface is represented by an everywhere C^1-continuous subdivision surface which interpolates all the given data points. And the topological structure of the reconstructed surface is exactly the same as that of the data points. The new technique consists of two major steps. First, use an efficient surface reconstruction method to produce a polyhedral approximation to the given data points. Second, construct a Doo-Sabin subdivision surface that smoothly passes through all the data points in the given data set. A new technique is presented for the second step in this thesis. The new technique iteratively modifies the vertices of the polyhedral approximation M until a new control mesh $\overline{M}$, whose Doo-Sabin subdivision surface interpolates M, is reached. It is proved that, for any mesh M with any size and any topology, the iterative process is always convergent with Doo-Sabin subdivision scheme. The new technique has the advantages of both a local method and a global method, and the surface reconstruction process can reproduce special features such as edges and corners faithfully.

KEYWORDS: Surface reconstruction, Doo-Sabin subdivision surfaces, interpolation, control mesh, polyhedral approximation

Jiaxi Wang
03/2008

PARAMETRIZATION AND SHAPE RECONSTRUCTION TECHNIQUES FOR DOO-SABIN SUBDIVISION SURFACES

By

Jiaxi Wang

Fuhua Cheng
Director of Thesis

Raphael A Finkel
Director of Graduate Studies

03/04/2008

Name	Date

THESIS

Jiaxi Wang

The Graduate School

University of Kentucky

2008

PARAMETRIZATION AND SHAPE RECONSTRUCTION TECHNIQUES FOR DOO-SABIN SUBDIVISION SURFACES

THESIS

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in the College of Engineering at the University of Kentucky

By

Jiaxi Wang

Lexington, Kentucky

Director: Dr. Fuhua Cheng, Professor of Computer Science

Lexington, Kentucky

2008

ACKNOWLEDGEMENTS

The following thesis, while an individual work, benefited from the insights and direction of several people. I would like to thank all of those people who helped make this thesis possible.

First, I especially would like to thank my advisor and thesis director, Dr. Fuhua Cheng, for all of his enlightening guidance, inspiring encouragement, and tremendous support, and for providing me with the unique opportunity to work in the research area of graphics. His knowledge of graphics and considerable experience often came to the rescue.

Next, I would like to thank my committee members, Dr. Jun Zhang and Dr. William (Brent) B Seales, for their acceptance of this task and for their helpful comments and suggestions. Each individual provided insights that guided and challenged my thinking, substantially improving the finished product.

In addition to the technical and instrumental assistance above, I received equally important assistance from my family and friends. Special thanks to my husband for providing on-going support throughout the thesis process. Special thanks to my parents for instilling in me, from an early age, the desire to obtain the Master's. Special thanks to my friends for their constant support.

Finally, I would like to thank God for giving me such a nice environment that made my project possible.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1 Introduction

The purpose of this thesis is to present an interpolation algorithm that allows a user to create a smooth reconstructed surface. This surface is defined and manipulated by a structured set of control points generated from the algorithm. The positions of the control points will determine the shape of the reconstructed surface. A user only needs to know the relationships among the original data points, control points and the reconstructed surface rather than the process and mathematics of the underlying implementation.

Many applications have been developed to represent surfaces. However, the only available information on a surface of most existing surface design schemes is a set of unorganized points sampled from that surface. This shall be discussed shortly. A set of unorganized sampled points can be quite restrictive from the point of view of design. This thesis will present a method for reconstructing a smooth surface that is not encumbered by this restriction, thereby giving a user more satisfiable and accurate shapes.

In most surface design schemes, computations on that surface require the construction of a piecewise linear approximation of the surface on a polynomial basis. Piecewise polynomial means that a curve or surface is represented by a collection of individual polynomial segments or patches. Before a computation can be performed on that surface, a representation of the surface has to be constructed from the sample points first, which will be used as control points in part of the implementation. This is the problem of surface reconstruction. Problems of this type occur in scientific and engineering applications such as CAD, medical imaging, visualization, computer graphics, computer vision, reverse engineering, and so on. Recent advances in modern laser technology have made it easier to generate a lot of sample points from the surface of an object, but this will result in a large amount of data and more storage space. A polyhedral approximation (mostly triangular) to the sampled surface is the goal to achieve in surface reconstruction. The reconstructed surface should be topologically equivalent to and geometrically close to the sampled surface. Reconstructing surfaces from unorganized sample points in a faithful way is still a difficult challenge, even though many fast and efficient algorithms proposed in the literature are able to achieve topologically correct surface reconstruction in most cases.

Traditional surface reconstruction methods always produce a set of triangles to approximate the surface shape. This usually is not precise enough when small details are needed. One can solve the precision problem by increasing the number of points sampled in the sampling process. This is possible because recent advances in laser technology have made it easier to generate a lot of sample points from the surface of an object. But there are occasions where a discrete representation is not good enough no matter how many points are used in the representation, such as 3D medical imaging where one needs to scale up an organ or a cross-section frequently. Smooth and precise surface representation for unorganized data is still needed. Especially in applications that require accurate representation. Take 3D medical imaging for example, people need to reconstruct surfaces as precisely as possible from range data, which usually are produced by laser range scanning systems or MRI. Construction of smooth representation of a surface from unorganized data has been studied for a while and some techniques have already been reported [1]. But the techniques do not guarantee interpolation of the sample points by the generated representation.

Control points schemes are generally either interpolating or approximating. The choice of which scheme to use depends upon the application. For all applications, the scheme should be coordinate free. This means that the relationship between the control points and the shape is independent of any coordinate system.

The shape defined by an approximating scheme will be constructed with polygons (most will be triangles). This type of scheme is well suited for local control. This means that modifying the shape by moving a control point will not affect the entire shape, i.e., if one of the control points is moved to another place, only the polygons owning it move with this point. Several such approximating schemes exist; the simplest and best known are B-splines (B for basic). The control points of a B-spline scheme are known as *de Boor points*. For B-spline curves, the de Boor points are an ordered set that forms the *de Boor polygon*. These points can be defined by a user. The resulting curve is a smooth approximation to the de Boor polygon. Figure 1.1 illustrates the relationship between the de Boor points and a B-spline curve. Doo-Sabin approximation method is used in this implementation.

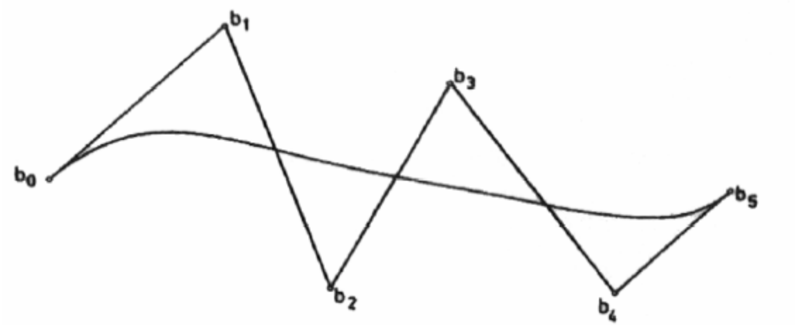


Figure 1.1 A B-Spline Curve

The shape defined by an interpolating scheme will pass through all of the control points. This type of scheme is well suited for representation, i.e., if the control points are known to belong to an existing shape. There are a number of interpolation methods. Linear interpolation (Figure 1.2) is the simplest method of getting values at positions in between the data points. Points are simply joined by straight line segments. Each segment (bounded by two data points) can be interpolated independently. The disadvantage of linear interpolation is the discontinuities at each point. Cubic interpolation (Figure 1.3) is the simplest method that offers true continuity between the segments. As such it requires more than just the two endpoints of the segment but also the two points on either side of them. Interpolation method used with Doo-Sabin scheme will be presented in this thesis.

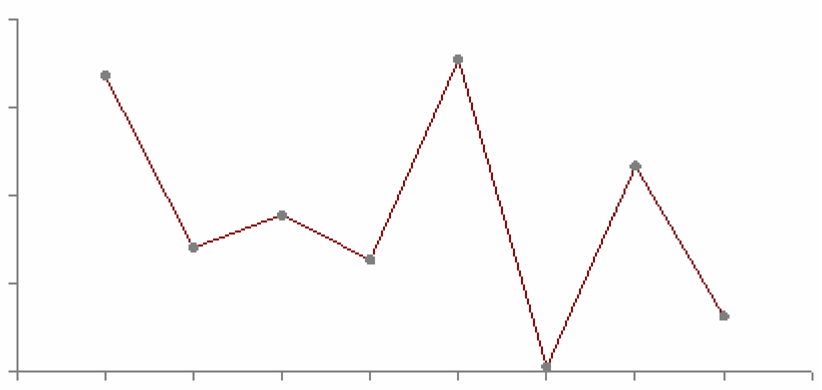


Figure 1.2 Linear Interpolation

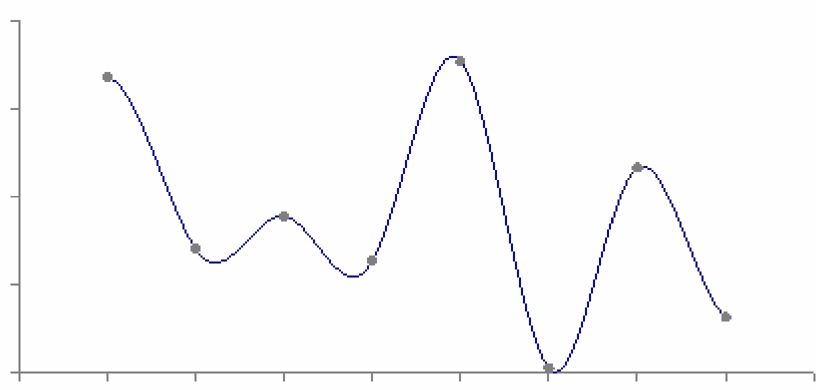


Figure 1.3 Cubic Interpolation

Approximation may not be the natural choice for representation while interpolation methods may not be the natural choice for local control. In this thesis a method, which is a combination of approximation and interpolation, is proposed to reconstruct a faithful surface from a set of data points, such that the reconstructed surface is not approximately represented by a polyhedron, but by an everywhere C^1-continuous subdivision surface. The subdivision surface interpolates all the given data points. Besides, the topological structure of the reconstructed surface is exactly the same as that of the data points. Therefore, the representation is guaranteed to be precise if the sampled points are taken directly from the sampled object. In addition, all the data points are guaranteed to lie precisely on the reconstructed surface. This is done in two steps:

- Use an efficient surface reconstruction method to produce a polyhedral approximation to the given sampled points. The polyhedral approximation obtained from the sampled points is the control mesh of a Doo-Sabin subdivision surface.
- Iteratively modify this control mesh to get a C^1-continuous subdivision surface to interpolate all the sampled points in the given data set.

While the first step is still a challenging step, it is the second step that is our focus here. Constructing a subdivision surface to interpolate an arbitrary mesh is not a well-solved problem when the number of vertices is large. So is the second step, especially when the number of sampled data points is huge. In this thesis a solution to this problem will be proposed, which will focus on how to construct an interpolating surface for the polyhedral approximation obtained from the first step.

Chapter 2 Conceptual Development

Background and Related Work

Surface Reconstruction from Unorganized Points

A number of applications ranging from CAD, computer graphics and mathematical modeling require the reconstruction of a smooth surface from a set of data points. The set of data points are the points on the surface which is being sampled. The data points could be densely sampled or sparsely taken from the surface such that they are the representative points of the surface. Up to now, many techniques have been proposed and developed to reconstruct an approximated surface from the set of 3D data points. Among them are greedy methods [2], implicit surfaces [3] and Delaunay triangulation, etc. However all of them only lead to a non-smooth polyhedral approximation to the given data points, or to a smooth surface that does not interpolate the input data point set [1]. Therefore without dense sampling of an object surface, none of the methods mentioned above can reconstruct the original surface precisely.

Subdivision Surfaces

Subdivision surfaces are popular now in Computer Animation, CAD and Geometric Modeling, etc. The ability to model arbitrary topology surfaces makes them more suitable than classical spline surfaces in some applications. The Catmull-Clark subdivision scheme [4] was proposed in 1978, which is the generalization of bi-cubic spline surface, while the Doo-Sabin subdivision method [5] is the generalization of quadratic spline surface. Later, the Loop subdivision scheme [6] was developed for triangular meshes which generalize the Box splines. All these three popular subdivision methods are approximating schemes. There are interpolating subdivision schemes that interpolate the given mesh. One of the most famous interpolating subdivision methods is the butterfly subdivision method [7] which was modified subsequently to generate smoother interpolation surfaces in [8]. An interpolating scheme for quadrilateral meshes was proposed in [9].

Surface Interpolation of Irregular Meshes

Interpolation is a popular technique used in surface design and shape modeling. There are plenty of publications dealing with the interpolation problem using various surface representations. As the appearance of recursive subdivision surfaces, interpolation methods based on subdivision surfaces have also been developed. One group of the methods is required to solve a global system of linear equations, like [10, 11]. To avoid the computational cost of solving a large system of linear equations, other methods have been developed. In [12], an always-working method solved the problem by using a two-phase subdivision method. The method proposed in [13] avoids solving a system of linear equations by using the concept of similarity. The approach presented in [14] avoids solving a system of linear equations by using quasi-interpolation.

In this thesis, based on the results obtained from traditional surface reconstruction methods which produce a polyhedral approximation to the given sample points, we present a new iterative interpolation method by using Doo-Sabin subdivision surface. Our iterative method is an extension of the *progressive iterative interpolation* method for B-splines [15, 16, 17]. The idea of our iterative interpolation method is to use the differences between the (original) mesh to be interpolated and the Doo-Sabin surface of current mesh to get a new mesh. This iterative process will converge to a Doo-Sabin surface interpolating the original mesh. The updating operation at each level of the iteration is done by a local operation for each vertex in current mesh. Therefore our method possesses the property of a local method. On the other hand, our method has the form of a global method due to its actual global linear effect. Therefore, our method has the advantages of both a local method and a global method. Experimental results demonstrate the efficiency and ability of our method in handling large meshes.

Definitions

In this subsection, several technical words related to the thesis will be explained .

Greedy algorithm

A *greedy algorithm* is any algorithm that follows the problem solving metaheuristic of making the locally optimum choice at each stage with the hope of finding the global optimum.

For example, applying the greedy strategy to the traveling salesman problem yields the following algorithm: "At each stage visit the unvisited city nearest to the current city".

In general, greedy algorithms have five pillars:

- A candidate set, from which a solution is created
- A selection function, which chooses the best candidate to be added to the solution
- A feasibility function, that is used to determine if a candidate can be used to contribute to a solution

- An objective function, which assigns a value to a solution, or a partial solution, and
- A solution function, which will indicate when we have discovered a complete solution

Greedy algorithms produce good solutions to some mathematical problems, but not to others.

Implicit Surface

In mathematical notation, a level set of a real-valued function f of n variables is a set of the form

$$\{(x_1,...,x_n) \mid f(x_1,...,x_n) = c\} \tag{2.1}$$

where c is a constant. That is, it is the set where the function takes on a given constant value. When the number of variables is two, this is a level curve (contour line), if it is three this is a level surface, and for higher values of n the level set is a level hypersurface. A level surface is sometimes called an *implicit surface* or an isosurface.

A set of the form

$$\{(x_1,...,x_n) \mid f(x_1,...,x_n) \le c\} \tag{2.2}$$

is called a sublevel set of f .

Delaunay triangulation

In mathematical notation and computational geometry, a Delaunay triangulation or Delone triangularization for a set P of points in the plane is a triangulation DT(P) such that no point in P is inside the circumcircle of any triangle in DT(P). Delaunay triangulations maximize the minimum angle of all the angles of the triangles in the triangulation; they tend to avoid "sliver" triangles. The triangulation was invented by Boris Delaunay in 1934.

Subdivision Surfaces

A subdivision surface, in the field of 3D computer graphics, is a method of representing a smooth surface via the specification of a coarser piecewise linear polygon mesh. The smooth surface can be calculated from the coarse mesh as the limit of an iterative process of subdividing each polygonal face into smaller faces that better approximate the smooth surface.

Catmull-Clark subdivision surface

The Catmull-Clark algorithm is used in subdivision surface modeling to create smooth surfaces. It was devised by Edwin Catmull (of Pixar) and Jim Clark, and won an Academy Award for Technical Achievement in 2006.

Doo-Sabin subdivision surface

A Doo-Sabin subdivision surface, in the field of computer graphics, is a type of subdivision surface based on a generalization of bi-quadratic uniform B-splines. It was developed in 1978 by Daniel Doo and Malcolm Sabin. There is a picture of a short subdivision process of this algorithm shown in picture 2.1.

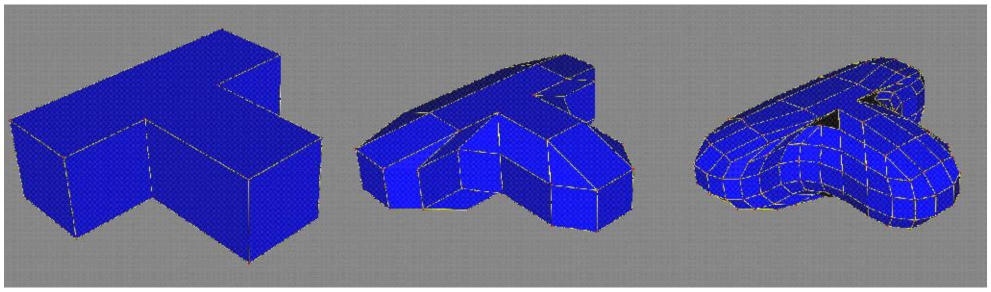


Figure 2.1 Doo-Sabin Subdivision Surface

Loop subdivision surface

A Loop subdivision surface is a subdivision scheme developed by Charles Loop in 1987 especially for triangular meshes.

Butterfly Subdivision

The Butterfly Subdivision method is a new interpolatory subdivision scheme for surface design. This scheme is designed for a general triangulation of control points and has a tension parameter that provides design flexibility. The resulting limit surface is C^1-continuous for a specified range of the tension parameter, with a few exceptions.

Parameterization

Parametrization (or parameterization) is the process of defining or deciding the 'parameters' —usually of some model— that are salient to the question being asked of that model.

Chapter 3 Methodology

Subdivision Methods

The Doo-Sabin algorithm

Doo and Sabin invented an algorithm for generating a smooth surface in 1978 [5] by generalizing the biquadratic B-spline subdivision rules to include arbitrary topologies. The corner clipping notion is used in this algorithm, which can generate smooth surfaces from an arbitrary set of control points or mesh. This algorithm can generate a biquadratic B-spline surface in the special case of a rectangle mesh.

The following picture is the subdivision masks of biquadratic B-Splines (Figure 3.1), which are used to derive the Doo-Sabin algorithm:

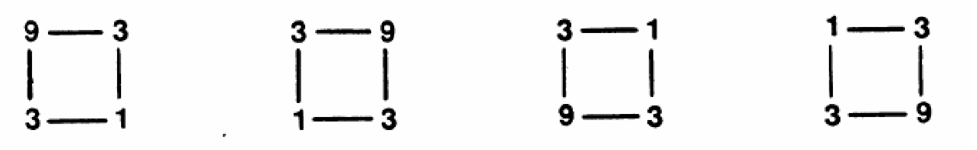


Figure 3.1 Subdivision masks for biquadratic B-splines

Four new de Boor points will be generated in each face of the original de Boor net when using the applications of these masks. One can get these new de boor points by taking a convex combination of the four vertices of each face of the de Boor net. When we observe the midpoint of the line segment connecting each vertex to the centroid of a face, we can see that the new de Boor points can also be located or replaced directly by the midpoint we have talked about above.

The situation described above is the special case for rectangle faces. This subdivision method can also be used to the generalized case of arbitrary topology control points or meshes. The faces may be n-sided, in a mesh of arbitrary topology, where $n \geq 3$. The centroid of the face can be easily found as the average of its vertices. A new control point may be found as the midpoint of the line segment connecting a face's centroid to each of its vertices, as in Figure 3.2. This is how the Doo-Sabin algorithm works.

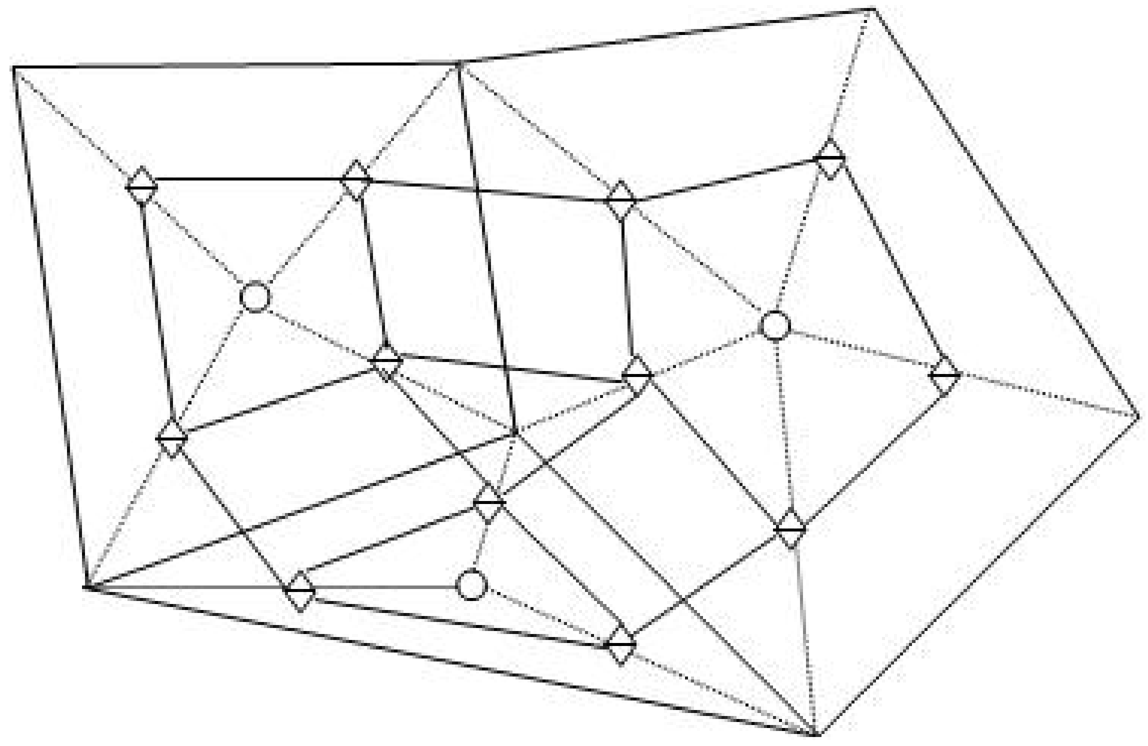


Figure 3.2 Construction of new control points using the Doo-Sabin algorithm

The above steps are executed repeatedly until the desired smoothness has been obtained by the result control points or mesh. Three iterations of the Doo-Sabin algorithm are illustrated in Figure 3.3. The faces, edges, and vertices of the old mesh are replaced or clipped to form the new mesh at every step. The renewed control points or mesh becomes locally rectangular everywhere as the subdivision process goes, except at a certain number of points. These points, also called *extraordinary points*, correspond to the vertices and faces of the original mesh. One can note that, which is very interesting, over every group of four rectangles with a valence 4 vertex in common, a biquadratic B-spline surface is local and exact representation. As the renewed mesh becomes increasingly rectangular or regular, more and more of the surface is exactly represented. However, the surface in the neighborhoods of the extraordinary points does not have this explicit representation. These regions are considered as holes in the paper of Doo and Sabin in an exact representation using biquadratic B-splines.

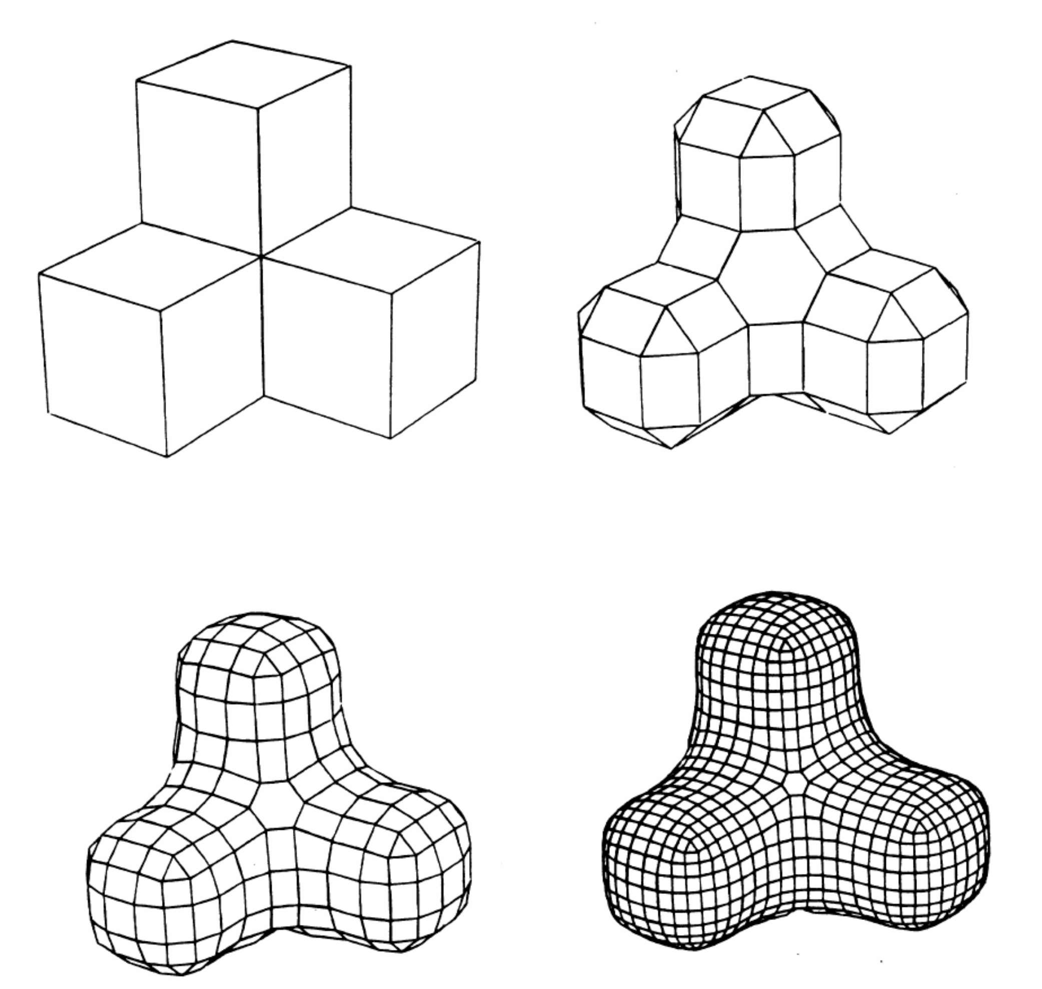


Figure 3.3 Three iterations of the Doo-Sabin subdivision surfaces

This representation maintains all the properties of piecewise biquadratic B-splines. Since biquadratic B-splines are C^1-continuous, the surfaces generated by the Doo-Sabin algorithm are considered locally C^1-continuous everywhere except at the extraordinary points. The refined mesh or the final subdivision surface is guaranteed to lie in the convex hull of the original control point mesh, since all new control points are found by taking convex combinations of the old control points. And each new control point is only dependent on a single face of the control point mesh. Thus, changing a control point will affect only a few faces, giving the Doo-Sabin surfaces a local control property. And this is also a property of all the approximation method.

The Catmull-Clark algorithm

In the same year of 1978, Catmull and Clark also presented an algorithm for generating a smooth surface from an arbitrary control point mesh [4]. Their approach is used to generalize bicubic B-spline subdivision surfaces instead of biquadratic B-spline subdivision surfaces. And since bicubic surfaces are higher order than biquadratic, the resulting algorithm is more complex.

The following picture is the subdivision masks of bicubic B-Splines (Figure 3.4), which is used to derive the Catmull-Clark algorithm:

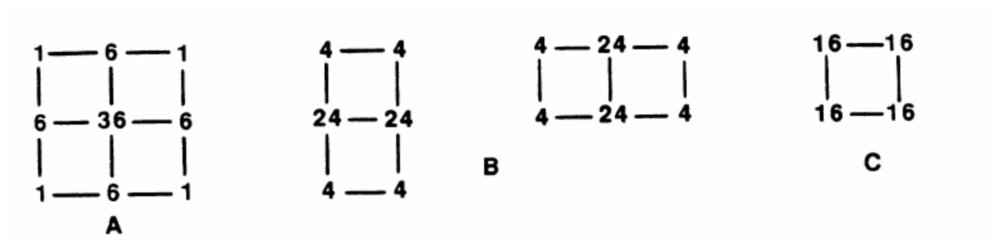


Figure 3.4 Subdivision Masks for bicubic B-splines

A new control point, which is also called vertex point, corresponding to each vertex of the old control point mesh will be generated when using the application of mask A. New points, which called edges points, corresponding to each edge, will be generated by the application of masks B, and mask C generates a new point, which is also called face point, corresponding to each face of the control point mesh. The geometric properties of the subdivision masks used to generate these points may be abstracted to control point meshes of arbitrary topology.

Considering mask C, the rule for computing new face points can be derived. We can see that this mask gives equal weight to all the vertices belonging to a face. And this can also be considered as computing the centroid of the face. It is obvious that we can create a new face point at the centroid of each face of the arbitrary mesh in general. To determine the rule for computing new edge points, considering masks of B. We can see that these masks generate new edge points as convex combinations of the vertices of the two faces adjacent at an edge. The point can also be found by taking the average of the centroids of the two adjacent faces along with the midpoint of the shared edge in the view of geometric topology. This idea is then easily applied to an arbitrary mesh. Generalizing the rules to generate a new vertex point is not that simple. Catmull and Clark determined that the vertex point $\overline{S}$ generated by mask A is found by taking a convex combination of three points. These points are: Q, the average of the new face points of all faces sharing an old vertex point; R, the average of the midpoints of all old edges incident on the old vertex point; S, the old vertex point. These three points may be similarly found for each vertex of an arbitrary mesh. The initial convex combination Catmull and Clark proposed was

$$\bar{S} = \frac{1}{4}Q + \frac{1}{2}R + \frac{1}{4}S. \tag{3.1}$$

Like the new face and edge point rules, the generalization of the new vertex point subdivision rule is equivalent to bicubic B-spline subdivision in the special case of a rectangular mesh. The rules described above were used by Catmull and Clark to generate smooth surfaces from arbitrary meshes initially. Like those of the Doo-Sabin algorithm, these surfaces are locally regular except at a constant number of extraordinary points. These points correspond to the faces and vertices of the original control point mesh. It was observed that in some arbitrary meshes, like the Doo-Sabin algorithm, continuity was not maintained at extraordinary points. This was remedied by modifying the generalization of the new vertex point rule to take the order of the vertex into account. The modified rule is :

$$\bar{S} = \frac{1}{N}Q + \frac{1}{N}R + \frac{N-3}{N}S. \tag{3.2}$$

Where N is the order of the vertex S. This rule generates surfaces that exhibit tangent plane continuity at all extraordinary points.

As the algorithm proceeds, we can note that the mesh becomes increasingly regular, like Doo-Sabin algorithm. The surface is exactly representable with bicubic B-splines over these regions. For this reason, Catmull-Clark subdivision surfaces inherit many of the important properties from bicubic B-splines. Catmull-Clark subdivision surfaces have the convex hull property, local control and are locally C^2-continuous everywhere except at the extraordinary points. A proof that Catmull-Clark surfaces have a continuous tangent plane at the extraordinary points was given by Doo and Sabin [5].

Both the Doo-Sabin and Catmull-Clark algorithms were derived from geometric properties of tensor product B-spline subdivision. Other non-tensor product B-spline surfaces have appeared since then. From properties of these surfaces, the Loop algorithm analogous to the Doo-Sabin and Catmull-Clark algorithm was derived.

The Loop Algorithm

This algorithm generalizes the subdivision of a regular triangle mesh [6]. A triangular mesh is a control point mesh whose faces are all triangles. Like the Doo-Sabin and Catmull-Clark algorithms, derivation of the generalized subdivision rules for Loop algorithm begins with an abstraction of the geometric properties of the subdivision masks. These masks are as follows (Figure 3.5):

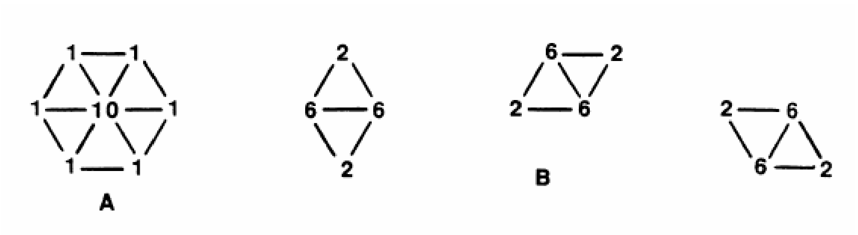


Figure 3.5 Subdivision Masks for triangle Splines

Mask A generates new control points for each vertex, and masks of B generate new control points for each edge of the original regular triangular mesh.

The masks of B compute the new edge points as convex combinations of the vertices of the two triangles that share the edge. In an arbitrary triangular mesh, each edge will be shared by two triangles. Therefore, an obvious generalization is to leave this subdivision rule intact. Like the Catmull-Clark algorithm, generalization of a vertex point is more difficult.

To derive the new vertex point rule, consider mask A. The new vertex point $\overline{V}$, can be computed as a convex combination of the old vertex, and all old vertices that share an edge with it. Alternatively, this same point may be found indirectly as a convex combination of two points. These points are: V, the old vertex point, and Q, the average of the old points that share an edge with V. The new vertex point is computed as

$$\overline{V} = \frac{5}{8}V + \frac{3}{8}Q \tag{3.3}$$

This can be applied to an arbitrary triangular mesh.

In the special case of a regular triangulation, the algorithm is equivalent to binary subdivision of a surface. Three iterations of the algorithm based on the rules just described are shown in Figure 3.6. As subdivision proceeds, the triangular control point mesh becomes locally regular, except at a fixed number of extraordinary points. For this new algorithm, only extraordinary points correspond to vertices of the original mesh rather than its faces. The surface of this algorithm is locally C^2-continuous everywhere, except at the extraordinary points.

Note that in Figure 3.6, tangent plane continuity is apparently lost at one of the extraordinary points. This situation is similar to the one encountered by Catmull and Clark in the initial formulation of their algorithm. This may be remedied by considering the order of the vertex when taking the convex combination of V and Q. This results in a new vertex point rule of the form:

$$\overline{V} = \alpha_N V + (1 - \alpha_N) Q \tag{3.4}$$

Where α_N is a function of the vertex order N. As long as $\alpha_6 = \frac{5}{8}$, the subdivision algorithm will be a superset of the subdivision algorithm. If $0 < \alpha_N < 1$, the resulting surface will lie in the convex hull of the control mesh.

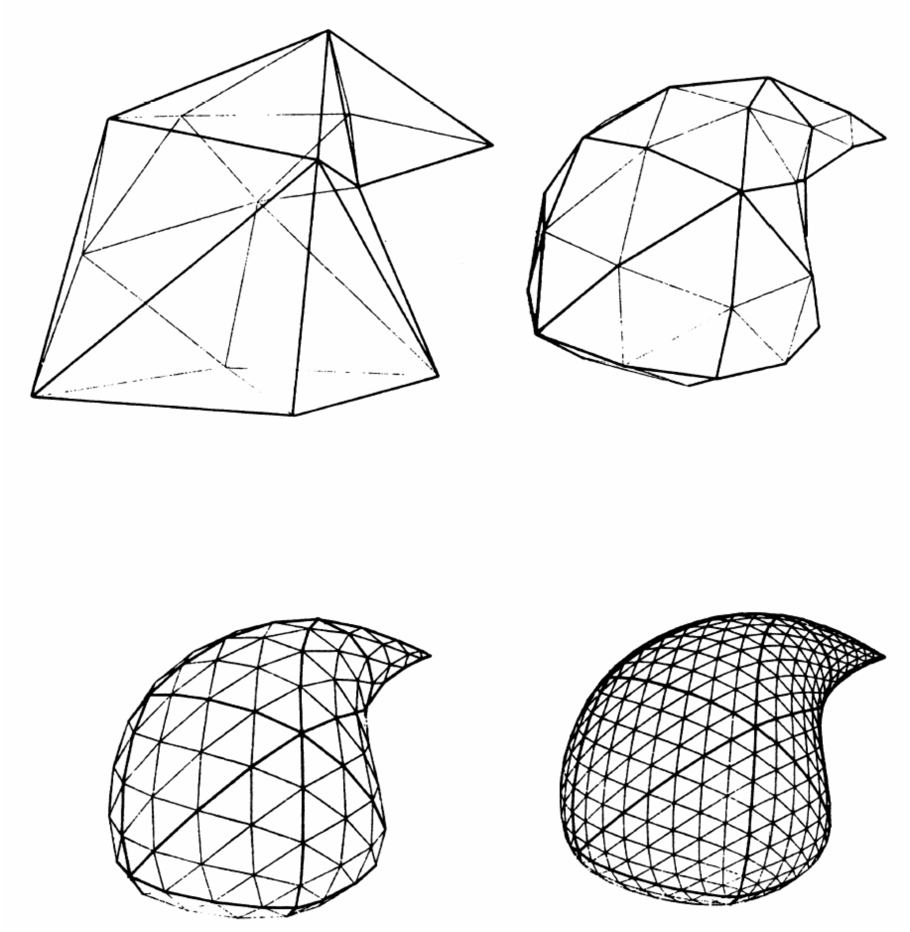


Figure 3.6 Three iterations of triangular subdivision algorithm

Parametrization Methods

Parametrization of Catmull-Clark subdivision Surfaces

In 1998, Jos Stam presented a parametrization method for Catmull-Clark subdivision surfaces at arbitrary parameter values [18]. The Catmull-Clark subdivision surface and all of its derivatives can be evaluated in terms of a set of eigenbasis functions which only depend on the subdivision scheme, and Jos Stam derived analytical expressions for these basis functions. This method allows many algorithms developed for parametric surfaces to be applied to Catmull-Clark subdivision surfaces, which makes subdivision surfaces an even more attractive tool for free-form surface modeling.

The initial mesh used in this method is assumed to be subdivided at least twice, in order to isolate the extraordinary vertices so that each face is a quadrilateral and contains at most one extraordinary vertex. There is a sample picture of a subdivision patch (see Figure 3.7). The valence of the extraordinary vertex is denoted by N. The task is then to find a surface patch $s(u,v)$ defined over the unit square $\Omega = [0,1] \times [0,1]$ that can be evaluated directly in terms of the $K = 2N + 8$ vertices that influence the shape of the patch corresponding to the face. Jos Stam assumes that the surface point corresponding to the extraordinary vertex is $s(0,0)$ and that the orientation of Ω is chosen such that $s_u \times s_v$ points outside of the surface.

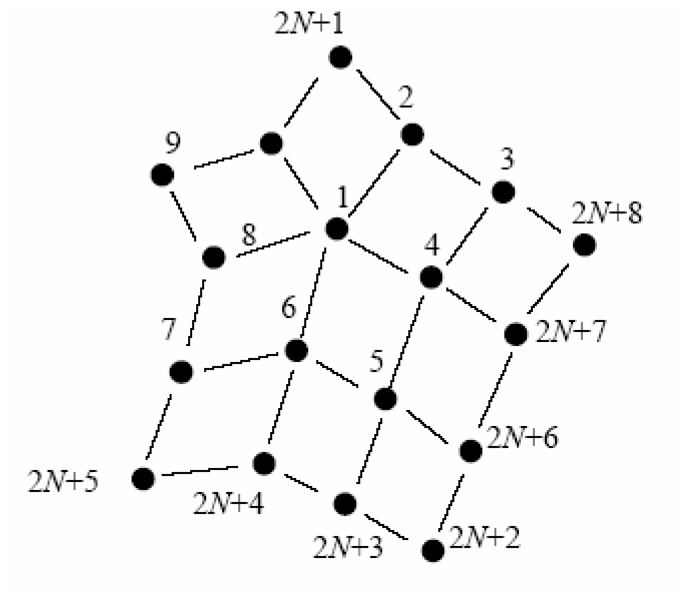


Figure 3.7 Surface patch near an extraordinary vertex with its control vertices

The initial control points set is denoted by matrix $C_0^T = (c_{0,1}, \cdots, c_{0,K})$,where $K = 2N + 8$. After one subdivision, a new set of $M = K + 9$ control points (see Figure 3.8) will be generated. Subsets of these new vertices are the control vertices of three uniform B-spline patches. Therefore, three-quarters of the surface patch is parametrized, and could be evaluated as simple bicubic B-splines (see top left of Figure 3.9). This new set of vertices is denoted by $C_1^T = (c_{1,1}, \cdots, c_{1,K})$ and $\overline{C}_1^T = (C_1^T, c_{1,K+1}, \cdots, c_{1,M})$. Then, the subdivision step is simplified to a multiplication process:

$$C_1 = AC_0 \tag{3.5}$$

where matrix A is a $K \times K$ *subdivision matrix*. And it has the following block structure:

$$A = \begin{pmatrix} S & 0 \\ S_{11} & S_{12} \end{pmatrix} \tag{3.6}$$

The additional points needed to evaluate the three B-spline patches are defined using a bigger matrix $\overline{A}$ of size $M \times K$:

$$\overline{C}_1 = \overline{A} C_0 \tag{3.7}$$

where

$$\overline{A} = \begin{pmatrix} S & 0 \\ S_{11} & S_{12} \\ S_{21} & S_{22} \end{pmatrix} \tag{3.8}$$

The subdivision step of Equation 3.5 can be repeated to create an infinite sequence of control vertices: $C_n = AC_{n-1} = A^n C_0$ and $\overline{C}_n = \overline{A} C_{n-1} = \overline{A} A^{n-1} C_0$, $n \geq 1$.

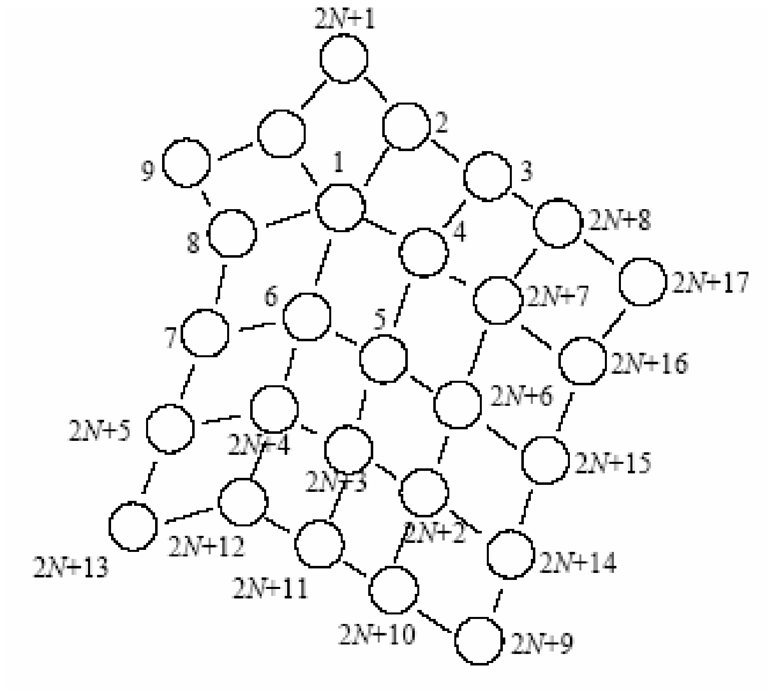


Figure 3.8 Addition of new vertices by applying the Catmull-Clark subdivision rule to the vertices in Figure 3.7

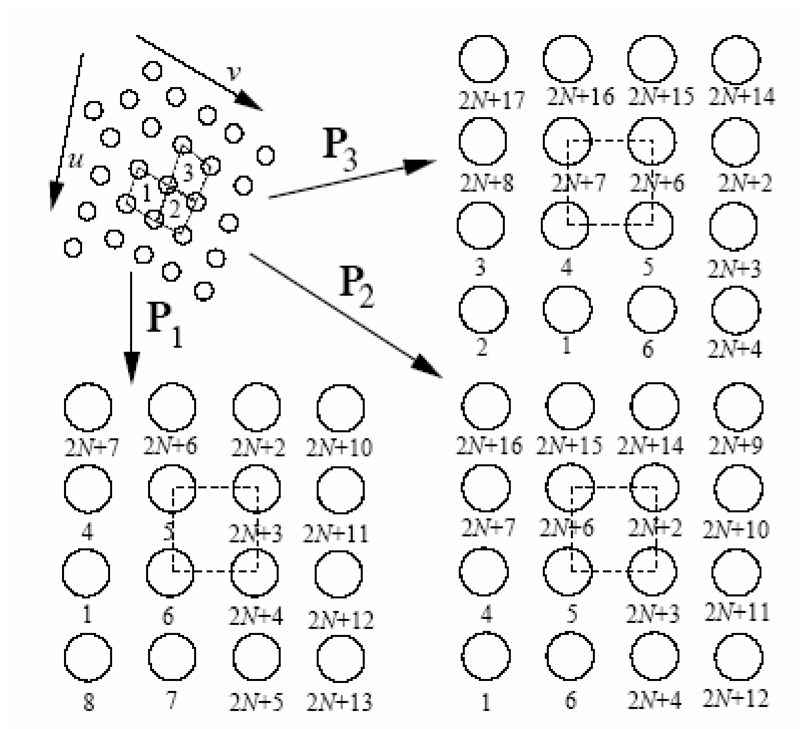


Figure 3.9 Indices of the control vertices of the three bi-cubic B-spline patches obtained from $\overline{C}_n$

As noted above, for each level $n \geq 1$, a subset of the vertices of $\overline{C}_n$ becomes the control vertices of three B-Spline patches. These control vertices can be defined by selecting 16 control vertices from $\overline{C}_n$ and storing them in 16×3 matrices: $B_{k,n} = P_k \overline{C}_n$, where P_k is a $16 \times M$ "picking" matrix and $k = 1,2,3$. Then, the surface patch corresponding to each matrix of control vertices is defined as:

$$s_{k,n}(u,v) = B_{k,n}^T b(u,v) = \vec{C}_n^T P_k^T b(u,v) \tag{3.9}$$

Where $b(u,v)$ is the vector containing the 16 cubic B-spline basis functions (see Appendix A), $(u,v) \in \Omega, n \geq 1$ and $k = 1,2,3$. We can partition the unit square Ω into an infinite set of tiles $\{\Omega_k^n\}, n \geq 1, k = 1,2,3$, as shown in Figure 3.10. A parametrization for $s(u,v)$ is constructed by defining its restriction to each tile Ω_k^n to be equal to the B-spline patch defined by the control vertices $B_{k,n}$:

$$s(u,v)\Big|_{\Omega_k^n} = s_{k,n}(t_{k,n}(u,v)) \tag{3.10}$$

The transformation $t_{k,n}$ maps the tile Ω_k^n onto the unit square Ω_k^n.

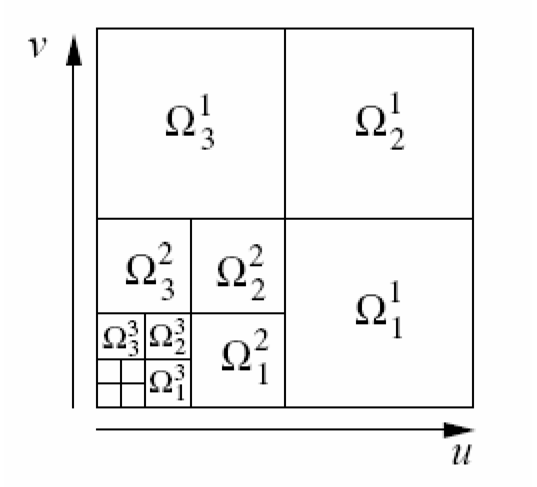


Figure 3.10 Partition of the unit square into an infinite family of tiles

The eigenstructure of matrix A can be written as $A = V\Lambda V^{-1}$. Then the subdivided control vertices at level n are now equal to $\overline{C}_n = \overline{A}A^{n-1}C_0 = \overline{A}V\Lambda^{n-1}V^{-1}C_0 = \overline{A}V\Lambda^{n-1}\hat{C}_0$, where $\hat{C}_0 = V^{-1}C_0$. Then equation 3.9 can be rewritten in the following form:

$$s_{k,n}(u,v) = \hat{C}_0^T \Lambda^{n-1}(P_k \overline{A}V)^T b(u,v) \tag{3.11}$$

By observing that the right most terms in this equation are independent of the control vertices and the power n, we can pre-compute this expression and define the following three vectors:

$$x(u,v,k) = (P_k \overline{A}V)^T b(u,v) \quad k = 1,2,3 \tag{3.12}$$

The components of these three vectors correspond to a set of K bi-cubic splines. In Appendix A we will show how to compute these splines. Now, the equation for each patch can be rewritten more compactly as:

$$s_{k,n}(u,v) = \hat{C}_0^T \Lambda^{n-1} x(u,v,k) \quad k = 1,2,3 \tag{3.13}$$

To make the expression for the evaluation of the surface patch more concrete, let p_i^T denote the rows of $\hat{C}_0$. Then the surface patch can be evaluated as:

$$s(u,v)\Big|_{\Omega_k^n} = \sum_{i=1}^{K} (\lambda_i)^{n-1} x_i(t_{k,n}(u,v),k)\, p_i \tag{3.14}$$

Alternatively, the bi-cubic spline functions $x(u,v,k)$ can be used to define a set of *eigenbasis functions* for the subdivision. For a given eigenvalue λ_i we define the function φ_i by its restrictions on the domains Ω_k^n as follows:

$$\varphi_i(u,v)\Big|_{\Omega_k^n} = (\lambda_i)^{n-1} x_i(t_{k,n}(u,v),k) \tag{3.15}$$

Now, the evaluation of the surface patch given by equation 3.14 can be rewritten exactly as:

$$s(u,v)\Big|_{\Omega_k^n} = \sum_{i=1}^{K} \varphi_i(u,v)\, p_i \qquad (3.16)$$

This is the key result of this method.

Parametrization of Doo-Sabin subdivision Surfaces

Parametrization of Doo-Sabin subdivision Surfaces is similar to the Jos Stam's method on Catmull-Clark subdivision surfaces. However, we use bi-quadratic B-splines in our method instead of bi-cubic B-splines. And the control vertices needed for bi-quadratic B-splines is 9 instead of 16.

The initial mesh used in this method is assumed to be subdivided at least once, in order to get rid of extraordinary vertices and isolate the extraordinary faces so that each patch, which will be described later, contains at most one extraordinary face. There is a sample picture of a subdivision patch (see Figure 3.11). The vertex number of the extraordinary face is denoted by N . The task is then to find a surface patch $s(u,v)$ defined over the unit square $\Omega = [0,1] \times [0,1]$ that can be evaluated directly in terms of the $K = N + 5$ vertices that influence the shape of the patch corresponding to the face. We assume that the surface point corresponding to the original extraordinary vertex is $s(0,0)$ and that the orientation of Ω is chosen such that $s_u \times s_v$ points outside of the surface.

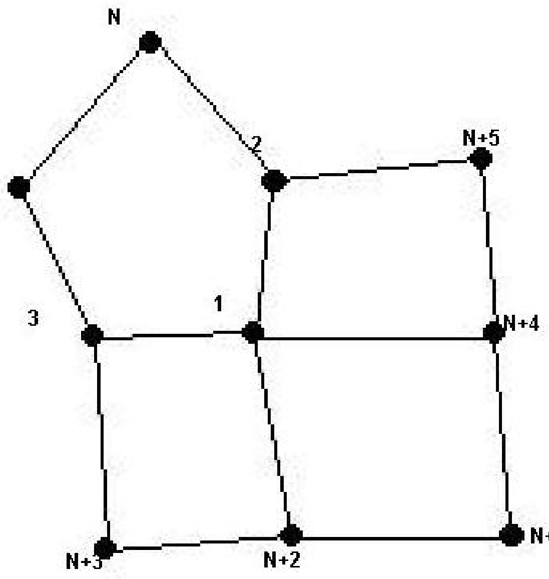


Figure 3.11 Surface patch near an extraordinary face with its control vertices

The initial control point set is denoted by matrix $C_0^T = (c_{0,1}, \cdots, c_{0,K})$,where $K = N + 5$. After one subdivision, a new set of $M = K + 7$ control points (see Figure 3.12) will be generated. Subsets of these new vertices are the control vertices of three uniform B-spline patches. Therefore, three-quarters of the surface patch is parametrized, and could be evaluate as simple biquadratic B-splines. This new set of vertices is denoted by $C_1^T = (c_{1,1}, \cdots, c_{1,K})$ and $\overline{C}_1^T = (C_1^T, c_{1,K+1}, \cdots, c_{1,M})$. Then, the subdivision step is simplified to a multiplication process:

$$C_1 = AC_0 \tag{3.17}$$

where matrix A is a $K \times K$ *subdivision matrix*. And it has the following block structure:

$$A = \begin{pmatrix} S & 0 \\ S_{11} & S_{12} \end{pmatrix} \tag{3.18}$$

The additional points needed to evaluate the three B-spline patches are defined using a bigger matrix $\overline{A}$ of size $M \times K$:

$$\bar{C}_1 = \bar{A}C_0 \tag{3.19}$$

where

$$\bar{A} = \begin{pmatrix} S & 0 \\ S_{11} & S_{12} \\ S_{21} & S_{22} \end{pmatrix} \tag{3.20}$$

The subdivision step of Equation 3.17 can be repeated to create an infinite sequence of control vertices: $C_n = AC_{n-1} = A^n C_0$ and $\bar{C}_n = \bar{A}C_{n-1} = \bar{A}A^{n-1}C_0$, $n \geq 1$.

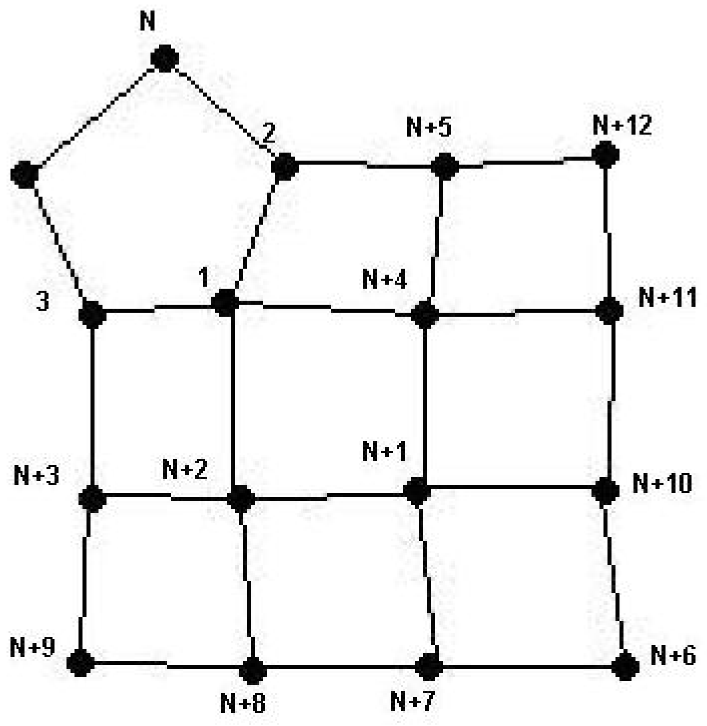


Figure 3.12 Addition of new vertices by applying the Doo-Sabin subdivision rule to the vertices in Figure 3.11

As noted above, for each level $n \geq 1$, a subset of the vertices of $\bar{C}_n$ becomes the control vertices of three B-Spline patches. These control vertices can be defined by selecting 9 control vertices from $\bar{C}_n$ and storing them in 9×3 matrices: $B_{k,n} = P_k\bar{C}_n$, where P_k is a $9 \times M$ "picking" matrix and $k = 1,2,3$. Then, the surface patch corresponding to each matrix of control vertices is defined as:

$$s_{k,n}(u,v) = B_{k,n}^T b(u,v) = \vec{C}_n^T P_k^T b(u,v) \tag{3.21}$$

Where $b(u,v)$ is the vector containing the 9 quadratic B-spline basis functions (see Appendix B), $(u,v) \in \Omega, n \geq 1$ and $k = 1,2,3$. We can partition the unit square Ω into an infinite set of tiles $\{\Omega_k^n\}, n \geq 1, k = 1,2,3$, as shown in Figure 3.10. A parametrization for $s(u,v)$ is constructed by defining its restriction to each tile Ω_k^n to be equal to the B-spline patch defined by the control vertices $B_{k,n}$:

$$s(u,v)\Big|_{\Omega_k^n} = s_{k,n}(t_{k,n}(u,v)) \tag{3.22}$$

The transformation $t_{k,n}$ maps the tile Ω_k^n onto the unit square Ω_k^n.

The eigenstructure of matrix A can be written as $A = V \Lambda V^{-1}$. Then the subdivided control vertices at level n are now equal to $\overline{C}_n = \overline{A} A^{n-1} C_0 = \overline{A} V \Lambda^{n-1} V^{-1} C_0 = \overline{A} V \Lambda^{n-1} \hat{C}_0$, where $\hat{C}_0 = V^{-1} C_0$. Then equation 3.21 can be rewritten in the following form:

$$s_{k,n}(u,v) = \hat{C}_0^T \Lambda^{n-1} (P_k \overline{A} V)^T b(u,v) \tag{3.23}$$

By observing that the most right-side terms in this equation are independent of the control vertices and the power n, we can pre-compute this expression and define the following three vectors:

$$x(u,v,k) = (P_k \overline{A} V)^T b(u,v) \quad k = 1,2,3 \tag{3.24}$$

The components of these three vectors correspond to a set of K bi-quadratic splines. In Appendix B we will show how to compute these splines. Now, the equation for each patch can be rewritten more compactly as:

$$s_{k,n}(u,v) = \hat{C}_0^T \Lambda^{n-1} x(u,v,k) \quad k = 1,2,3 \tag{3.25}$$

To make the expression for the evaluation of the surface patch more concrete, let p_i^T denote the rows of $\hat{C}_0$. Then the surface patch can be evaluated as:

$$s(u,v)\Big|_{\Omega_k^n} = \sum_{i=1}^{K} (\lambda_i)^{n-1} x_i(t_{k,n}(u,v),k)\, p_i \tag{3.26}$$

Alternatively, the bi-quadratic spline functions $x(u,v,k)$ can be used to define a set of *eigenbasis functions* for the subdivision. For a given eigenvalue λ_i we define the function φ_i by its restrictions on the domains Ω_k^n as follows:

$$\varphi_i(u,v)\Big|_{\Omega_k^n} = (\lambda_i)^{n-1} x_i(t_{k,n}(u,v),k) \tag{3.27}$$

Now, the evaluation of the surface patch given by equation 3.26 can be rewritten exactly as:

$$s(u,v)\Big|_{\Omega_k^n} = \sum_{i=1}^{K} \varphi_i(u,v)\, p_i \tag{3.28}$$

This is the final result of our method.

Chapter 4 Surface Reconstruction using Doo-Sabin Subdivision Surfaces

As mentioned in Chapter 1, there are two major steps in the new surface reconstruction process. First we apply an efficient surface reconstruction method to producing a polyhedral approximation to the given data points, then we find an interpolatory surface for the obtained polyhedral approximation in the first step. There are many efficient approaches that we can use for the first step [1, 2, 3]. In this thesis the polyhedral approximation obtained from the first step is regarded as the control mesh of a Doo-Sabin subdivision surface and this chapter focuses on how to construct an interpolating surface for the control mesh.

Polyhedral Approximation

Doo-Sabin subdivision scheme is used for the first step. In this subdivision scheme, we use the Catmull quadratic method. New polygons are built from the old mesh in the following way. An edge point is formed from the midpoint of each edge. A face point is formed as the centroid of each polygon of the mesh. Finally, each vertex in the new mesh is formed as the average of a vertex in the old mesh, a face point for a polygon that is incident to that old vertex, and the edge points for the two edges that belong to that polygon and are adjacent to that old vertex.

The new vertices then are connected. There will be two vertices along each side of each edge in the old mesh, by construction. These pairs are connected, forming quadrilaterals across the old edges. Within each old polygon, there will be as many new vertices as there were vertices in the polygon. These are connected to form a new, smaller, inset polygon. And finally, around each old vertex there is a new vertex in the adjoining corner of each old polygon. These are connected to form a new polygon with as many edges as there were polygons around the old vertex. The new mesh, therefore, will create quadrilaterals for each edge in the old mesh, will create a smaller m-sided polygon for each m-sided polygon in the old mesh, and will create an n-sided polygon for each n-valence vertex. After one application of the scheme all vertices have a valence of four. So, subsequent applications will create quadrilaterals for the vertices only. All n-sided polygons are retained in the subdivision process, and shrink to extraordinary points as the subdivision scheme is repeatedly applied.

For a vertex V of valance n (see Figure 4.1), if its adjacent edge points are E_i, $1 \le i \le n$ and its adjacent face points are F_j^i, $1 \le i \le n, 1 \le j \le m_i - 3$, where m_i is the number of edges in the ith adjacent face, then after one subdivision we have

$$V_i^{'} = (\frac{1}{2} + \frac{1}{4m_i})V + (\frac{1}{8} + \frac{1}{4m_i})E_i + (\frac{1}{8} + \frac{1}{4m_i})E_{i+1} + \frac{1}{4m_i}\sum_{j=1}^{m_i-3} F_j^i \tag{4.1}$$

where $V_i^{'}$, $1 \le i \le n$ is one of the newly generated vertex points around vertex V after one subdivision (see Figure 4.1).

After each subdivision we have an n-sided polygon around vertex V, which will remain to be n-sided in the subdivision process, and shrink to a limit point as the scheme is repeatedly applied. The limit point corresponding to V on the limit surface can be calculated as follows:

$$V_\infty = \frac{1}{n}\sum_{i=1}^{n} V_i^{'} \tag{4.2}$$

The above formula can be expanded and hence V_∞ can be more precisely rewritten as follows:

$$\begin{aligned} V_\infty &= \frac{1}{n}\left(\sum_{i=1}^{n}\frac{4m_i+2}{8m_i}V + \sum_{i=1}^{n}\left(\frac{m_i+2}{8m_i}+\frac{m_{i-1}+2}{8m_{i-1}}\right)E_i + \sum_{i=1}^{n}\left(\sum_{j=1}^{m_i-3}\frac{2}{8m_i}F_j^i\right)\right) \\ &= \frac{1}{n}\sum_{i=1}^{n}\left(\frac{4m_i+2}{8m_i}V + \left(\frac{m_i+2}{8m_i}+\frac{m_{i-1}+2}{8m_{i-1}}\right)E_i + \sum_{j=1}^{m_i-3}\frac{2}{8m_i}F_j^i\right) \end{aligned} \tag{4.3}$$

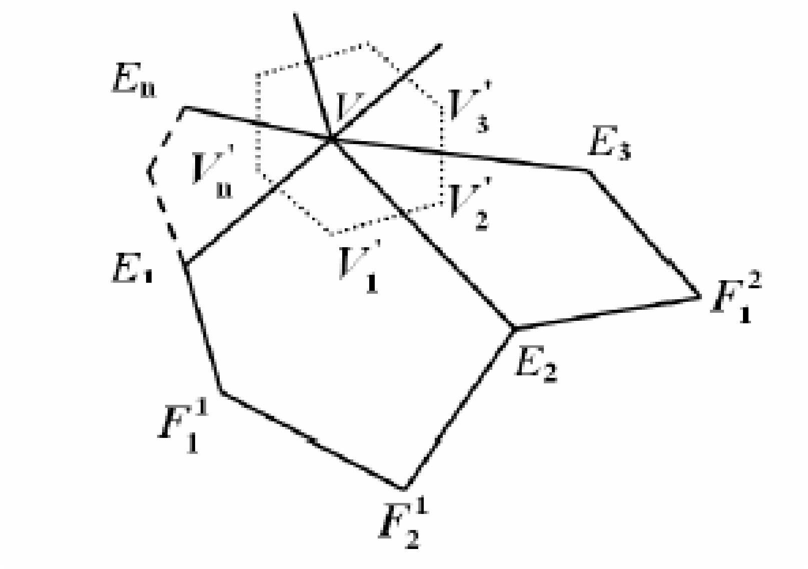


Figure 4.1 A vertex V of valence n and the new, adjacent vertex points generated after one Doo-Sabin subdivision.

Progressive Interpolation

For a given mesh M^0, we will find a new mesh M whose Doo-Sabin limit surface interpolates all vertices of M^0. Instead of solving a global system of linear equations, we develop a progressive iterative method which only locally manipulates vertices of the control mesh by an affine operation at each level of iteration. The iteration process is described as follows.

Initially, for each vertex V^0 of M^0, we compute the difference vector between this vertex and its limit point on the Doo-Sabin surface S^0 calculated from the equation (4.3),

$$D^0 = V^0 - V_\infty^0 \tag{4.4}$$

and add the differences D^0 to the vertex V^0.

$$V^1 = V^0 + D^0 \tag{4.5}$$

Therefore, we get a new control mesh M^1 whose vertices are computed as V^1. By iteratively repeating this process, we get a sequence of control meshes $M^0, M^1, M^2 \cdots$.

In general, if V^k,($0 \le k < \infty$), is the new location of vertex V after k iterations of the above process and M^k is the control mesh consists of all new V^ks, then we denote the Doo-Sabin limit surface of M^k or S^k. We first compute the distance between V^0 and the limit point V_∞^k of V^k on S^k:

$$D^k = V^0 - V_\infty^k \tag{4.6}$$

We then add this distance to V^k to get V^{k+1} as follows:

$$V^{k+1} = V^k + D^k \tag{4.7}$$

The set of new vertices is called M^{k+1}.

This process generates a sequence of control meshes M^k and a sequence of corresponding Doo-Sabin surfaces S^k. S^k converges to an interpolating surface of M^0 if the distance between S^k and M^0 converges to zero (i.e., $D^k \to 0$). Therefore the key task here is to prove that D^k converges to zero when k tends to infinity.

Chapter 5 Proof of Convergence

To prove the convergence of the above iterative process, we need a lemma about the eigenvalues of the product of positive definite matrices.

Lemma 1 Eigenvalues of the product of positive definite matrices are positive.

The proof of *Lemma 1* follows immediately from the fact that if P and Q are square matrices of the same dimension, then PQ and QP have the same eigenvalues (see, e.g., [18], p.14).

As mentioned above, to prove that the iterative interpolation process converges, we must prove that the difference D^k approaches zero when k tends to infinity.

Note that D^k can be expanded as follows:

$$
\begin{aligned}
D^k &= V^0 - V_\infty^k \\
&= V^0 - \frac{1}{n}\Big(\sum_{i=1}^{n}\frac{4m_i+2}{8m_i}V^k + \sum_{i=1}^{n}\Big(\frac{m_i+2}{8m_i}+\frac{m_{i-1}+2}{8m_{i-1}}\Big)E_i^k + \sum_{i=1}^{n}\Big(\sum_{j=1}^{m_i-3}\frac{2}{8m_i}(F_j^i)^k\Big)\Big) \\
&= V^0 - \frac{1}{n}\Big(\sum_{i=1}^{n}\frac{4m_i+2}{8m_i}V^{k-1} + \sum_{i=1}^{n}\Big(\frac{m_i+2}{8m_i}+\frac{m_{i-1}+2}{8m_{i-1}}\Big)E_i^{k-1} + \sum_{i=1}^{n}\Big(\sum_{j=1}^{m_i-3}\frac{2}{8m_i}(F_j^i)^{k-1}\Big)\Big) \qquad (5.1) \\
&\quad - \frac{1}{n}\Big(\sum_{i=1}^{n}\frac{4m_i+2}{8m_i}D^{k-1} + \sum_{i=1}^{n}\Big(\frac{m_i+2}{8m_i}+\frac{m_{i-1}+2}{8m_{i-1}}\Big)D_{E_i}^{k-1} + \sum_{i=1}^{n}\Big(\sum_{j=1}^{m_i-3}\frac{2}{8m_i}D_{F_j^i}^{k-1}\Big)\Big) \\
&= D^{k-1} - \frac{1}{n}\Big(\sum_{i=1}^{n}\frac{4m_i+2}{8m_i}D^{k-1} + \sum_{i=1}^{n}\Big(\frac{m_i+2}{8m_i}+\frac{m_{i-1}+2}{8m_{i-1}}\Big)D_{E_i}^{k-1} + \sum_{i=1}^{n}\Big(\sum_{j=1}^{m_i-3}\frac{2}{8m_i}D_{F_j^i}^{k-1}\Big)\Big)
\end{aligned}
$$

Equation (5.1) can be represented in a compact matrix form as follows:

$$[D_1^k, D_2^k, \ldots, D_m^k]^T = (I-B)\begin{bmatrix} D_1^{k-1} \\ D_2^{k-1} \\ \vdots \\ D_m^{k-1} \end{bmatrix} = (I-B)^k \begin{bmatrix} D_1^0 \\ D_2^0 \\ \vdots \\ D_m^0 \end{bmatrix} \tag{5.2}$$

where m is the number of vertices in the given mesh, I is an identity matrix of size $m \times m$, and B is a matrix of the following form:

$$B = \begin{pmatrix} \frac{1}{n_1}(\sum_{i=1}^{n_1} \frac{4m_i+2}{8m_i}) & \cdots & \frac{1}{n_1}(\frac{m_i+2}{8m_i} + \frac{m_{i-1}+2}{8m_{i-1}}) & \cdots & \frac{1}{n_1}(\frac{2}{8m_i}) & \cdots \\ \vdots & \ddots & & & & \\ \frac{1}{n_i}(\frac{m_i+2}{8m_i} + \frac{m_{i-1}+2}{8m_{i-1}}) & & \frac{1}{n_i}(\sum_{i=1}^{n_1} \frac{4m_i+2}{8m_i}) & & & \\ \vdots & & & \ddots & & \\ \frac{1}{n_j}(\frac{2}{8m_i}) & & & & \frac{1}{n_j}(\sum_{i=1}^{n_j} \frac{4m_i+2}{8m_i}) & \\ \vdots & & & & & \ddots \end{pmatrix} \tag{5.3}$$

Each entry of matrix B can be directly derived from Equation (4.3). Now, to prove D^k approaches zero when k tends to infinity, we just need to show that $(I-B)^k$ approaches zero when k tends to infinity.

Obviously, V^{i+1}, limit points of the mesh control points V^i, lying on the Doo-Sabin subdivision surface S^i, now can be represented in matrix form as $V^{i+1} = BV^i$. Note that B can be decomposed into the product of a diagonal matrix Λ and a symmetric matrix T as follows:

$$B = \Lambda T \tag{5.4}$$

where Λ is of the following form:

$$\Lambda = \begin{pmatrix} \frac{1}{n_1} & 0 & \cdots & 0 \\ 0 & \frac{1}{n_2} & \cdots & 0 \\ \vdots & & \ddots & \\ 0 & & & \frac{1}{n_m} \end{pmatrix} \tag{5.5}$$

and T is of the following form:

$$T = \begin{pmatrix} \sum_{i=1}^{n_1} \frac{4m_i+2}{8m_i} & \cdots & \frac{m_i+2}{8m_i} + \frac{m_{i-1}+2}{8m_{i-1}} & \cdots & \frac{2}{8m_i} & \cdots \\ \vdots & \ddots & & & & \\ \frac{m_i+2}{8m_i} + \frac{m_{i-1}+2}{8m_{i-1}} & & \sum_{i=1}^{n_1} \frac{4m_i+2}{8m_i} & & & \\ \vdots & & & \ddots & & \\ \frac{2}{8m_i} & & & & \sum_{i=1}^{n_j} \frac{4m_i+2}{8m_i} & \\ \vdots & & & & & \ddots \end{pmatrix} \tag{5.6}$$

Note that if (V_i, V_j) is an edge of a mesh, then (V_j, V_i) is an edge of this mesh as well; if (V_i, V_j) is an edge of a face, then so is (V_j, V_i). In other words, the relationship between two edge vertices or two face vertices is symmetric. It is then easy to see that T is symmetric. Furthermore, it can be proved that matrix T is positive definite.

Proposition 1 The matrix T is positive definite.

Proof: It is well-known that a symmetric and strictly diagonally dominant matrix with positive diagonal entries is a positive definite matrix. Because all the coefficients in the Doo-Sabin subdivision process are non-negative, it is easy to check that the diagonal entries of T are positive numbers. Therefore we just need to show that T is a strictly diagonally dominant matrix. According to equation (4.3), each row of matrix T satisfies:

$$\begin{aligned} T_{kk} - \sum_{l=1,l\neq k}^{n_k} T_{lk} &= \sum_{i=1}^{n_k} \frac{4m_i+2}{8m_i} - 2\sum_{i=1}^{n_k} \frac{m_i+2}{8m_i} - \sum_{i=1}^{n_k} (\sum_{j=1}^{m_i-3} \frac{2}{8m_i}) \\ &= \sum_{i=1}^{n_k} \frac{4}{8m_i} > 0 \end{aligned} \tag{5.7}$$

Hence, T is strictly diagonally dominant and, consequently, T is positive definite.

With the above results, we are ready to prove the convergence of the iterative interpolation process.

Proposition 2 The iterative interpolation process for Doo-Sabin subdivision surface is convergent.

Proof: As mentioned above, we just need to prove that $(I-B)^k$ approaches zero when k tends to infinity, where B is defined above and I is an identity matrix. Recall that matrix T is a symmetric positive definite matrix, and so is the diagonal matrix Λ . According to *Lemma 1*, $B = \Lambda T$, we can conclude that B only has positive eigenvalues. Since Doo-Sabin subdivision scheme satisfies the convex hull property, we have $\|B\|_\infty = 1$, which implies all eigenvalue λ_i of B satisfy $|\lambda_i| \le 1$. Therefore, all eigenvalues of B satisfy $0 < \lambda_i \le 1$. Based on this result, it is easy to see that the eigenvalues of matrix $(I - B)$ satisfy $0 \le 1 - \lambda_i < 1$. Consequently, $(I - B)^k$ approaches zero when k tends to infinity. The convergence of the iterative interpolation process for Doo-Sabin subdivision surfaces then is a direct consequence.

Chapter 6 Implementation Results

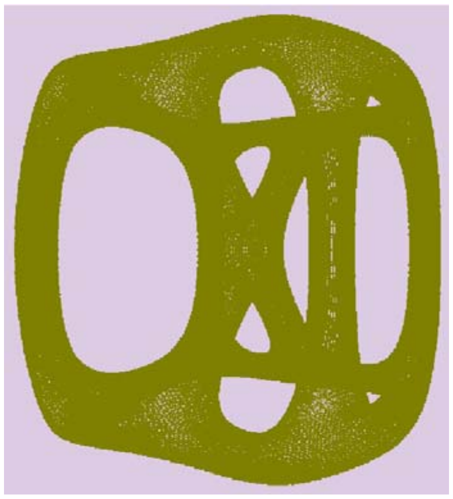
Implementation of the surface reconstruction technique using Doo-Sabin subdivision surfaces is done on a Windows platform using *OpenGL* as the supporting graphics system. Due to the combination of local and global advantages, the iterative interpolation method is very efficient and can handle very large data sets easily. Besides, our experiment results show that our approach can generate visually pleasing surfaces though there is no fairness parameter in the interpolation scheme.

Model	# of data points	#of vertices in poly. Approx.	# of iterations	Error
CubeHC	81920	7666	9	10^{-6}
Goblet	129280	8082	7	10^{-6}
Rockarm	203904	13984	5	10^{-6}
Beethoven	262016	16378	5	10^{-6}

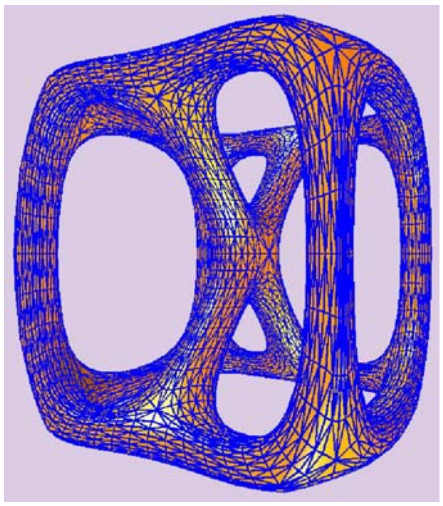
Table 6.1 Doo-Sabin surface based progressive interpolation: test results

Many examples have been tested and some examples are presented in Figure 6.1, 6.2, 6.3, 6.4. In these figures, the input 3D data points for these examples are listed in the first row, the corresponding polyhedral approximations, obtained after applying the surface reconstruction method [2], are listed in the second row, and the reconstructed C^1-continuous Doo-Sabin subdivision surfaces which interpolate the corresponding polyhedral approximations are shown in the third row. We also tabulate some of the testing parameters (see Table 6.1), such as the number of data points in the input model, the number of vertices in the polyhedral approximation obtained from applying a traditional surface reconstruction method [2, 3], the number of iterations used in the iterative interpolation process to get the interpolating surface and error tolerance used to stop the iteration.

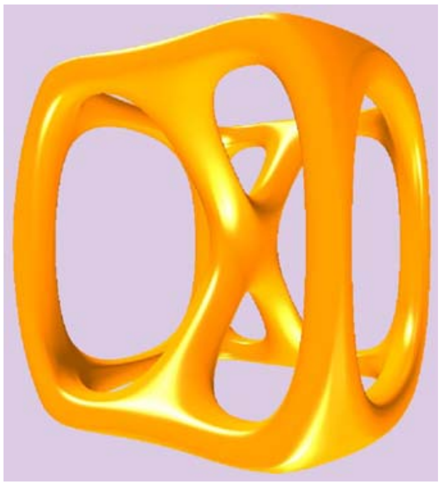
Note that the number of data points in the input 3D model is not the same as the number of vertices in the obtained polyhedral approximation. This is because we made some simplification such that the obtained polyhedral approximations are not as dense as the input data set and meanwhile, without losing much precision (by tolerating a small given error, say 10^{-6}).



(a) Data Points

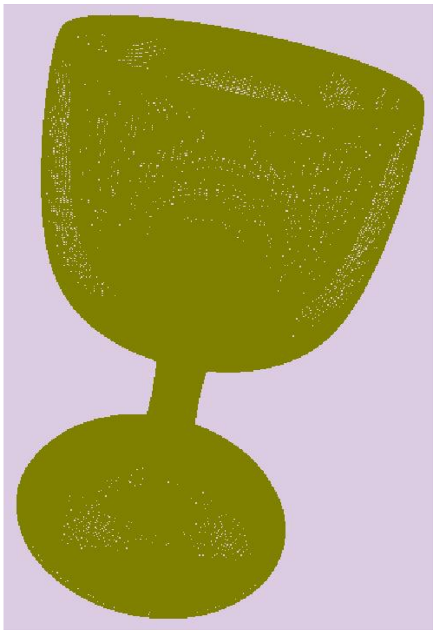


(b) Polyhedral Approximation

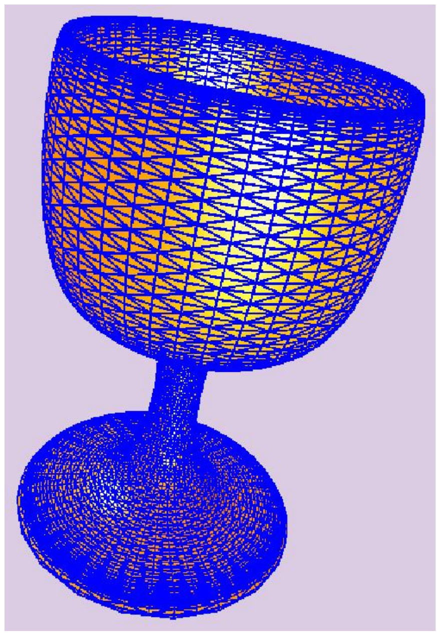


(c) Reconstruction by Interpolation

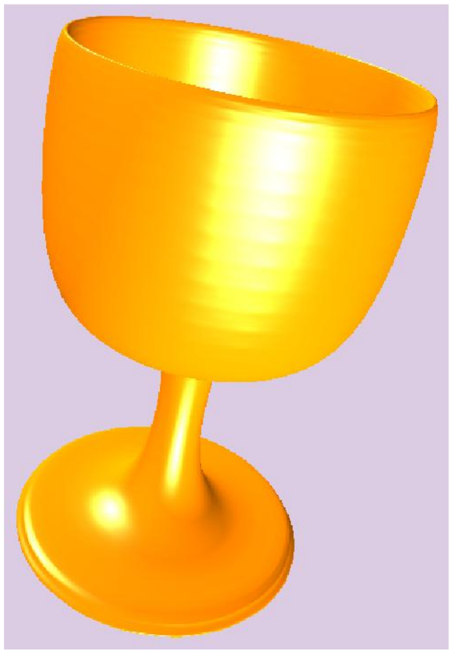
Figure 6.1 CubeHC



(a) Data Points

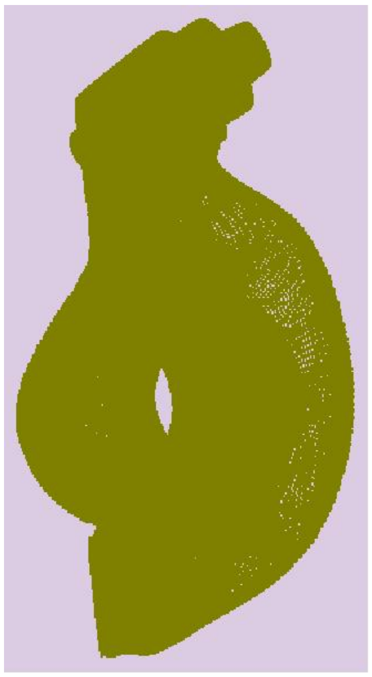


(b) Polyhedral Approximation

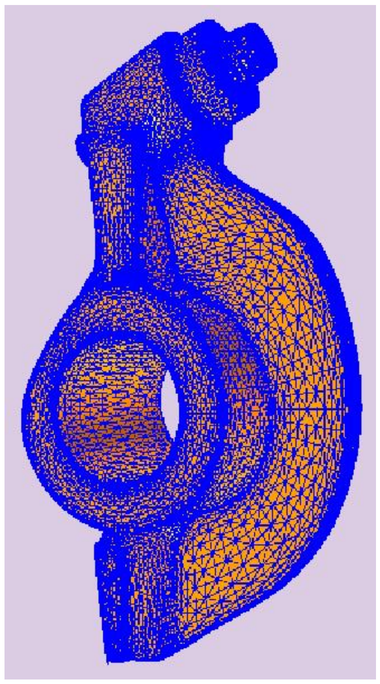


(c) Reconstruction by Interpolation

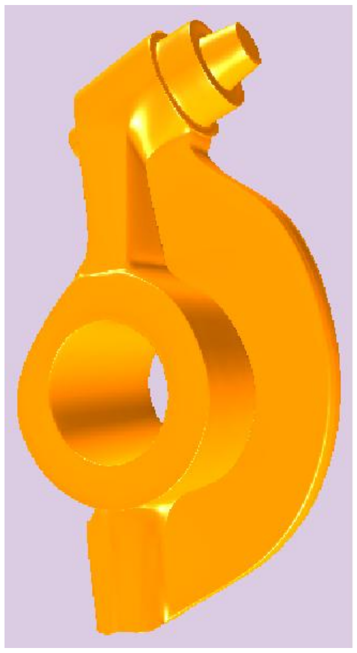
Figure 6.2 Goblet



(a) Data Points



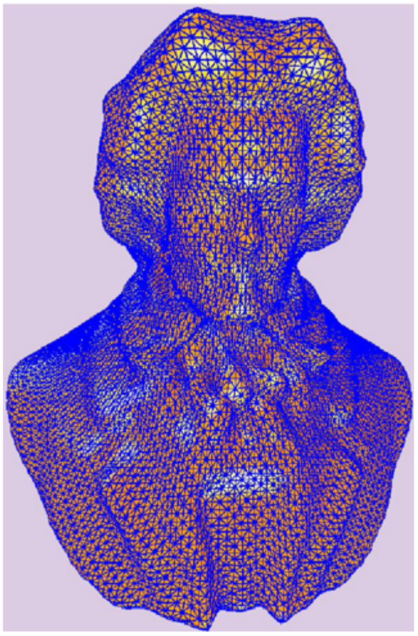
(b) Polyhedral Approximation



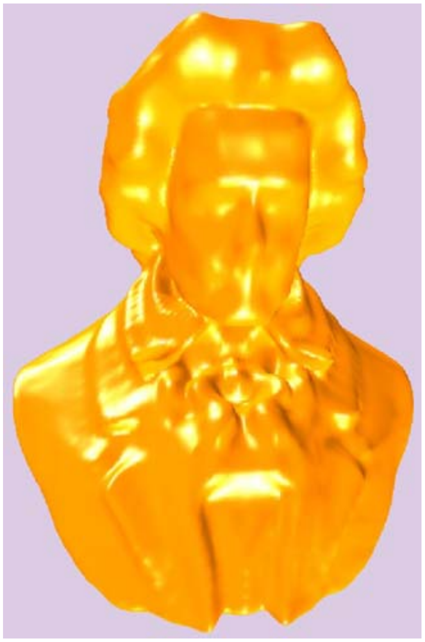
(c) Reconstruction by Interpolation

Figure 6.3 Rockarm

(a) Data Points



(b) Polyhedral Approximation



(c) Reconstruction by Interpolation

Figure 6.4 Beethoven

Chapter 7 Concluding Remarks

A new technique for the reconstruction of a smooth surface from a set of 3D sample points is presented. The reconstructed surface is not represented by a polyhedral approximation, but an everywhere C^1-continuous subdivision surface which interpolates all the sample points. Meanwhile all the data points are guaranteed to precisely lie on the reconstructed surface. The reconstruction process employs a two-step approach: a surface reconstruction step and a surface interpolation step. The first step produces a polyhedral approximation to the sampled surface from the sample points. The second step produces a Doo-Sabin subdivision surface that interpolates all the sample points. The second step is the focus of this thesis. The interpolating surface is generated by iteratively modifying the vertices of the polyhedral approximation M until a control mesh $\overline{M}$, whose Doo-Sabin subdivision surface interpolates M, is reached. It is proved that, for any mesh M with any size and any topology, the iterative process is convergent with Doo-Sabin subdivision surfaces. Therefore the surface reconstruction process is well-defined. The new technique has the advantages of both a local method and a global method. Therefore it can handle data set of any size while capable of generating a faithful approximation of the sampled surface no matter how complicated the shape and topology of the surface. The surface reconstruction process can also reproduce special features such as edges and corners faithfully.

Appendices

A. Bi-cubic splines

In this appendix [18] we compute the bi-cubic spline pieces $x(u,v,k)$ of the eigen basis defined in Equation 3.12. The vector $b(u,v)$ contains the 16 tensor B-spline basis functions ($i=1,\cdots,16$): $b(u,v)=N_{(i-1)\%4}(u)N_{(i-1)/4}(v)$, where "%" and "/" stand for the remainder and the division respectively. The functions $N_i(t)$ are the uniform B-spline basis functions:

$$\begin{aligned}
6N_0(t) &= 1-3t+3t^2-t^3,\\
6N_1(t) &= 4-6t^2+3t^3,\\
6N_2(t) &= 1+3t+3t^2-3t^3,\\
6N_3(t) &= t^3.
\end{aligned}$$

The projection matrices P_1, P_2 and P_3 are defined by introducing the following three permutation vectors (see Figure 3.10):

$$\begin{aligned}
q^1 &= (8,7,2N+5,2N+13,1,6,2N+4,2N+12,\\
&\quad 4,5,2N+3,2N+11,2N+7,2N+6,2N+2,2N+10),\\
q^2 &= (1,6,2N+4,2N+12,4,5,2N+3,2N+11,\\
&\quad 2N+7,2N+6,2N+2,2N+10,2N+16,2N+15,2N+14,2N+9),\\
q^3 &= (2,1,6,2N+4,3,4,5,2N+3,\\
&\quad 2N+8,2N+7,2N+6,2N+2,2N+17,2N+16,2N+15,2N+14).
\end{aligned}$$

Since for the case $N=3$ the vertices c_2 and c_8 are the same vertex, $q_1^1=2$ instead of 8 for $N=3$. Using these permutation vectors we can compute each bi-cubic spline as follows:

$$x_i(u,v,k)=\sum_{j=1}^{16}\Big(\sum_{l=1}^{K}\overline{A}_{q_j^k,l}V_{l,i}\Big)b_j(u,v)$$

where $i=1,\cdots,K$ and V are the eigenvectors of the subdivision matrix.

B. Bi-quadratic splines

In this appendix we compute the bi-quadratic spline pieces $x(u,v,k)$ of the eigen basis defined in Equation 3.24. The vector $b(u,v)$ contains the 9 tensor B-spline basis functions ($i=1,\cdots,9$): $b(u,v)=N_{(i-1)\%3}(u)N_{(i-1)/3}(v)$, where “%” and “/” stand for the remainder and the division respectively. The functions $N_i(t)$ are the uniform B-spline basis functions:

$$\begin{aligned}
2N_0(t) &= t^2-2t+1,\\
2N_1(t) &= -2t^2+2t-1,\\
2N_2(t) &= t^2.
\end{aligned}$$

The projection matrices P_1 , P_2 and P_3 are defined by introducing the following three permutation vectors (see Figure 3.10):

$$\begin{aligned}
q^1 &= (3, N+3, N+9, 1, N+2, N+8, N+4, N+1, N+7),\\
q^2 &= (1, N+2, N+8, N+4, N+1, N+7, N+11, N+10, N+6),\\
q^3 &= (2, 1, N+2, N+5, N+4, N+1, N+12, N+11, N+10)
\end{aligned}$$

Using these permutation vectors we can compute each bi-quadratic spline as follows:

$$x_i(u,v,k)=\sum_{j=1}^{9}\left(\sum_{l=1}^{K}\overline{A}_{q_j^k,l}V_{l,i}\right)b_j(u,v)$$

where $i=1,\cdots,K$ and V are the eigenvectors of the subdivision matrix.

References

[1] T. K. Dey and J. Sun, *An Adaptive MLS Surface for Reconstruction with Guarantees*, Symposium on Geometry Processing 2005, 43--52.
[2] David Cohen-Steiner, Frank Da, *Greedy Delaunay Based Surface Reconstruction Algorithm*, technical report, ECG-TR-124202-01, http://cgal.inria.fr/Reconstruction.
[3] H. Hoppe, T. DeRose, T. Duchamp, etc., *Surface reconstruction from unorganized points*, SIGGRAPH 1992, 71-78.
[4] Catmull E, Clark J, Recursively generated B-spline surfaces on arbitrary topological meshes, *Computer-Aided Design* 1978, 10(6):350-355.
[5] Doo D, Sabin M, Behavior of recursive division surfaces near extraordinary points, *Computer-Aided Design* 1978, 10(6):356-360.
[6] Loop C, Smooth Subdivision Surfaces Based on Triangles, *Master'thesis*, Dept. of Math., Univ. of Utah, 1987.
[7] Dyn N, Levin D, Gregory JA, A Butterfly Subdivision Scheme for Surface Interpolation with Tension Control, *ACM Trans. Graphics* 1990, 9(2):160-169.
[8] Zorin D, Schroder P, Sweldens W, Interpolating Subdivision for Meshes with Arbitrary Topology, *Computer Graphics, Ann. Conf. Series*, 1996, 30:189-192.
[9] Kobbelt L, Interpolatory Subdivision on Open Quadrilateral Nets with Arbitrary Topology, *Comput. Graph. Forum* 1996, 5(3):409-420.
[10] Halstead M, Kass M, DeRose T, Efficient, fair interpolation using Catmull-Clark surfaces, *Proc.SIGGRAPH 1993*, 47-61.
[11] Nasri AH, Surface interpolation on irregular networks with normal conditions, *Computer Aided Geometric Design* 1991, 8:89-96.
[12] Zheng J, Cai YY, Interpolation over arbitrary topology meshes using a two-phase subdivision scheme, *IEEE Trans. Visualization and Computer Graphics* 2006, 12(3):301-310.
[13] Lai S, Cheng F, Similarity based Interpolation using Catmull-Clark Subdivision Surfaces, *The Visual Computer* 2006, 22(9):865-873.
[14] Litke N, Levin A, Schröder P, Fitting Subdivision Surfaces, *Proc. Visualization* 2001, 319-324.
[15] Qi D, Tian,Z, Zhang Y, Zheng JB, The method of numeric polish in curve fitting, *Acta Mathematica Sinica* 1975, 18:173-184 (in Chinese).
[16] de Boor C, How does Agee's method work? *Proc. 1979 Army Numerical Analysis and Computers Conference*, ARO Report 79-3, Army Research Office, 299-302.
[17] Lin H, Wang, G, Dong C, Constructing iterative non-uniform B-spline curve and surface to fit data points, Science in China, Series F, 2004, 47(3):315-331.
[18] Stam J, Exact Evaluation of Catmull-Clark Subdivision Surfaces at Arbitrary Parameter Values, *Proc. SIGGRAPH* 1998.

Vita

- Birthday: 09/27/1983
- Education: BE, Computer Science, North China Electric Power University
- Name: Jiaxi Wang