Representing and Modifying Complex Surfaces

by

Henning Biermann

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Computer Science

New York University

November 2002

Denis Zorin

Preface

Henning Biermann died on July 1st, 2002, from leukemia. He fell ill in December of 2001, just a few months away from graduation. This thesis has been compiled from papers on which Henning was a primary author . All but one have appeared as published articles, which is a testament to the high quality of Henning's work.

Henning had all the qualities one can wish for in a colleague: he was highly intelligent, dedicated to his work, persistent and a pleasure to work with. During his years at NYU, he has co-authored ten research papers, many of which have appeared in the most respected publications in the field, a remarkable record for a graduate student. Henning was truly exceptional as a human being, always ready to help everyone around him, often putting the benefit of others above his own. He has made lots of friends at NYU and has left his trace in the lives of many people.

As his thesis advisor, I expected this thesis to be a beginning of a fruitful research career for Henning; his death is a great loss not only to his family, friends and colleagues but also to the field of computer science, where he has already left his mark.

Denis Zorin

New York, November, 2002

H. Biermann's Publications

[1] H. Biermann, I. Martin, F. Bernardini, and D. Zorin. Cut-and-paste editing of multiresolution surfaces. *ACM Transactions on Graphics, (Proceedings of ACM SIGGRAPH 2002)*, 21(3):312–321, July 2002.

[2] H. Biermann, D. Kristjansson, and D. Zorin. Approximate boolean operations on free-form solids. In *Proceedings of ACM SIGGRAPH 2001*, Computer Graphics Proceedings, Annual Conference Series, pages 185–194. ACM Press / ACM SIGGRAPH, August 2001.

[3] H. Biermann, I. M. Martin, D. Zorin, and F. Bernardini. Sharp features on multiresolution subdivision surfaces. In *9th Pacific Conference on Computer Graphics and Applications*, IEEE, October 2001.

[4] L. Velho, K. Perlin, L. Ying, H. Biermann. Procedural Shape Synthesis on Subdivision Surfaces. In *XIV Brazilian Symposium on Computer Graphics and Image Processing,* October 2001. Journal version to be published in *Computers and Graphics*, Elsevier Science, Vol. 26, No. 6.

[5] L. Ying, A. Hertzmann, H. Biermann, D. Zorin. Texture and Shape Synthesis on Surfaces In *Proceedings 12th Eurographics Workshop on Rendering*, June 2001.

[6] H. Biermann, A. Levin, and D. Zorin. Piecewise-smooth subdivision surfaces with normal control. In *Proceedings of ACM SIGGRAPH 2000*, Computer Graphics Proceedings, Annual Conference Series, pages 113–120. ACM Press / ACM SIGGRAPH, July 2000.

[7] , C. Bregler, A. Hertzmann, H. Biermann. Recovering Non-Rigid 3D Shape from Image Streams. In *Proceedings of IEEE CVPR 2000*, June 13-15, 2000, pages 690–696

[8] B. Moghaddam, H. Biermann, D. Margaritis. Image Retrieval with Local and Spatial Queries. *International Conference on Image Processing ICIP 2000*, September 2000.

[9] B. Moghaddam, H. Biermann, D. Margaritis. Regions-of-Interest and Spatial Layout in Content-based Image Retrieval. *European Workshop on Content-Based Multimedia Indexing*, CBMI'99, October 1999. Journal version published in *Multimedia Tools and Applications*,14(3), July 2001.

[10] B. Moghaddam, H. Biermann, D. Margaritis. Defining Image Content with Multiple Regions-of-Interest. *IEEE Workshop on Content-based Access of Image and Video Libraries,* CVPR'99, June 1999.

Acknowledgements

I would like to thank Henning's numerous collaborators: Daniel Kristjansson, Ioana Martin, Aaron Hertzmann, Adi Levin, Lexing Ying, Fausto Bernardini, Ken Perlin and Luiz Velho for their contributions to the papers they have co-authored with Henning. The help and support of the staff of the NYU Media Research Lab and the Center of Advanced Technology was indispensable.

My greatest gratitude and deepest condolences go to Henning's family: Hede and Christof Biermann, Christine Schindel, and his close friend Uta Hengst. I wish that he would not have worked so hard and would have spent more time with them during the past few years.

Denis Zorin

Abstract

The increasing demand for highly detailed geometric models poses new and important problems in computer graphics and geometric modeling. Applications for complex models range from geometric design and scientific simulations to feature movies and video games.

We focus on the fundamental problem of creating and manipulating complex surface models. We address the problem by designing an efficient and general surface representation, and develop algorithms for efficient modification of surfaces represented in this form. Our surface representation extends existing subdivision-based representations with explicit representation of sharp features and boundaries, which is crucial in many computer-aided design applications.

We consider two types of surface modifications: boolean operations on solids bounded by surfaces, and surface pasting. Our technique rapidly and robustly computes an *approximate* result rather than aiming for the precise solution. At the same time, our approach allows one to trade speed for accuracy, and, in most cases, compute the result with any desired accuracy. The second type of editing operations we consider address the problem of transferring geometric features between different objects. Our technique makes it easy to combine geometric data from various sources (e.g. 3D scanning, CAGD model) into a single model.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The increasing demand for highly detailed geometric models poses new and important problems in computer graphics and geometric modeling. Applications for complex models range from geometric-design[17] and scientific simulations[14] to feature movies[18] and video games (i.e. Id Soft's Quake).

The currently emerging field of *geometry processing* aims to process geometric data in a unified way. Key problems are the creation and representation of complex geometry[108], editing[42], transmission[35], compression[39], signal-processing[30], simulation, etc.

These problems are extremely challenging due to the enormous amount of data and the wide range of scale of geometric data. Moreover, many applications such as editing require processing at interactive rates. Many of these problems can only be approached by designing a geometry representation which supports certain processing operations very efficiently.

In our work, we focus on the fundamental problem of creating and manipulating a complex surface. We address the problem by designing an efficient representation for multiresolution surfaces and develop several algorithms for

the efficient modification of surfaces represented in this form.

Surface representation. Our surface representation, described in Chapters 2 and 3 extends the most commonly used techniques[60, 11], by representing sharp features explicitly. These features are of great practical relevance, because sharp creases and corners are common in geometric modeling applications, especially if boolean operations are used to create objects. The mathematical properties of our representation are considered in Chapter 4.

The goal of our work on surface modification is to develop algorithms for high-level operations which hide the specifics of the underlying surface representation from the user. At the same time, the processing operations exploit our computationally efficient representation.

Boolean operations. The first type of editing operations (Chapter 5) that we consider addresses the problem of creating complex models from scratch. In solid modeling, basic shapes are combined into more complex shapes with Boolean operations. However, the computation of intersections required for such operations is numerically unstable[34], and even state of the art methods are far from achieving interactive rates [46]. We propose the novel approach of rapidly computing an *approximate* result rather than aiming for the precise solution. Our approach allows one to trade speed for accuracy, and, in most cases, compute the result with any desired accuracy. While we cannot guarantee that we are able to produce an arbitrarily accurate result for all possible inputs, we do guarantee that the result is a valid closed surface, which is a property often missing in many existing systems.

Copy-and-paste surface editing. This type of editing operations considered in Chapter 6 addresses the problem of combining existing surfaces. This choice of the application is motivated by the typical work-flow in industrial design, where computer models are often based on conceptual designs created in clay. It is frequently necessary to rebuild the computer model, because changes in the clay design can not be incorporated due to the specific surface representation in the software.

We have developed a technique which allows geometric data from various sources (e.g. 3D scanning, CAGD modeling) to be combined into a single model. Our multi-resolution surface representation supports copy-and-paste operations: geometric features can be transferred between different objects and across different scales. The efficiency of the proposed pasting technique is based on using highly regular data structures and adapting image processing methods. Following the approach used for combining images in a typical image editing tool, sampling pattern of the target object is not changed whenever possible; instead geometric features transferred from the source object are resampled using the target object sampling pattern.

Chapter 2

Piecewise Smooth Subdivision Surfaces with Normal Control

This chapter has been published as a paper in the SIGGRAPH 2000 Conference Proceedings [6]. It introduces the subdivision schemes for representing piecewise-smooth surfaces with boundary, which are extensively used in subsequent chapters.

2.1 Introduction

Subdivision surfaces are rapidly gaining popularity in computer graphics. A number of commercial systems use subdivision as a surface representation: Alias-Wavefront's Maya, Pixar's Renderman, Nichimen's Mirai, and Micropace' Lightwave 3D, to name just a few. The greatest advantage of subdivision algorithms is that they efficiently generate smooth surfaces from arbitrary initial meshes. Subdivision algorithms are also attractive because they are conceptually simple and can be easily modified to create surface features without making

major changes to the algorithm.

At the same time, one of the drawbacks of subdivision is a lack of precise definition of the schemes with guaranteed behavior for a sufficiently general type of control meshes. Anyone who tries to implement a subdivision scheme can observe that more often than not it is unclear how rules should be specified in certain cases (most commonly on boundaries and creases). Ad hoc solutions have to be used, which often have unexpected undesirable behavior. The lack of precise and complete definition makes it more difficult to exchange data between applications, reuse control meshes, and design new algorithms based on subdivision.

The difficulty in defining a reasonably complete set of subdivision rules is related to the fact that subdivision algorithms allow a large variety of data as input: an arbitrary polygonal or triangular mesh, possibly with boundary, marked edges, and vertices. Subdivision rules for the interior of a control mesh are well understood, while the boundary rules have received less attention. Boundary rules are quite important for a variety of reasons. The boundary of the surface, together with the contour lines, forms the visual outline. Often, only an approximate definition is required for the interior of the surface, whereas the boundary conditions may be significantly more restrictive. For example, it is often necessary to join several surfaces along their boundaries. Boundary subdivision rules lead to rules for sharp creases [36] and soft creases [18]. In addition to specifying the boundary or crease curves, it is often desirable to be able to specify tangent planes on the boundary; existing subdivision schemes do not allow to control tangent plane behavior.

The goal of this paper is to present two complete sets of subdivision rules for generating piecewise-smooth, C^1-continuous, almost everywhere C^2 subdi-

vision surfaces, with tangent plane control. Our rules extend the well-known subdivision schemes of Catmull-Clark [11] and Loop [60]. The properties of our schemes were rigorously verified. We use a uniform approach to derive a set of rules, including new rules for concave corners, improved smooth boundary rules, new rules for tangent plane modification, and C^2 rules. While our approach is based on a number of known ideas, its advantage is that all desired features are handled in a unified framework.

Our approach to building a complete set of rules can be applied to any stationary subdivision scheme. In this paper, we focus on the Loop and Catmull-Clark subdivision schemes as schemes having the greatest practical importance. The code implementing our algorithms is available on the Web[1].

2.2 Previous Work

A number of subdivision schemes have been proposed since Catmull and Clark introduced subdivision surfaces in 1978 [11]. A detailed survey of subdivision can be found in [17].

Theoretical analysis of subdivision rules was performed in [86, 77, 32, 88, 106, 105]. Most of this work has focused on closed surfaces; while the general theory does not impose symmetry restrictions on the subdivision rules, almost all theoretical analysis of specific schemes relies on the rotational symmetry of the subdivision rules and applies only to the interior rules.

Subdivision rules for Doo-Sabin dual surfaces for the boundary were discussed by Doo [20] and Nasri [69, 70, 68], but only partial theoretical analysis was performed. Our work builds on the work of Hoppe et al. [36] and partially

[1] http://www.mrl.nyu.edu/biermann/sub

on the ideas of Nasri [71].

To the best of our knowledge, the boundary subdivision rules proposed in work [36] are the only ones that result in provably C^1-continuous surfaces (the analysis can be found in Schweitzer [88]). However, these rules suffer from two problems:

- The shape of the boundary of the generated surface depends on the control points in the interior;
- Only one rule for corners is defined, which works well for convex corners but does not work well for concave corners.

Standard Catmull-Clark rules, when applied to the boundary, suffer from the same problems.

Sederberg et al. [91] proposed a generalization of Catmull-Clark and Doo-Sabin subdivision rules that contains NURBS as a subset. For some applications it is important to include NURBS patches, however, the complexity of the algorithms is increased and the behavior of the surface near the extraordinary points becomes difficult to analyze and predict. The smooth crease effects that are obtained by manipulating NURBS weights for subdivision surfaces can be achieved using an elegant technique proposed by DeRose et al. [18]. Our approach to C^2 subdivision is similar to the approach of [83].

Levin recently introduced a combined subdivision scheme which interpolates a network of curves [53]. There are two main distinctions between the present work and [53]. First, we are solving a different problem: rather than assuming that we are given a network of smooth curves that has to be interpolated, we assume only a discrete mesh with tags, which controls the behavior of our surface, but no interpolation is required. Second, Levin's combined subdivision schemes are an interesting new research direction; not much is known and understood

about their behavior, especially on arbitrary meshes. In contrast, we focus on completing the subdivision toolbox with provably reliable tools.

Halstead et al. [33] describe a method of interpolating positions and normal direction on subdivision surfaces. However, this method involves the solution of a global system of equations, unlike our local subdivision rules.

2.3 Piecewise smooth surfaces

Piecewise smooth surfaces. Our goal is to design subdivision schemes for the class of *piecewise smooth surfaces*. This class includes common modeling primitives such as quadrilateral free-form patches with creases and corners. However, we exclude certain singularities (e.g., cone-like singularities and corners).

Here we give a somewhat informal description of piecewise-smooth surfaces. For simplicity, we consider only surfaces without self-intersection.

Recall that for a closed C^1-continuous surface in $\mathbf{R}^3$, each point has a neighborhood that can be smoothly deformed (that is, there is a C^1 map of maximal rank) into an open planar disk D. A surface with a *smooth boundary* can be described in a similar way, but neighborhoods of boundary points can be smoothly deformed into a half-disk H, with closed boundary (Figure 2.1). In order to allow *piecewise* smooth boundaries, we introduce two additional types of local charts: concave and convex corner charts, Q_3 and Q_1. We conclude that a C^1-continuous surface with piecewise smooth boundary looks locally like one of the domains D, H, Q_1, or Q_3. *Piecewise-smooth surfaces* are constructed out of surfaces with piecewise smooth boundaries joined together. If two surface patches have a common boundary, but different normal directions along the

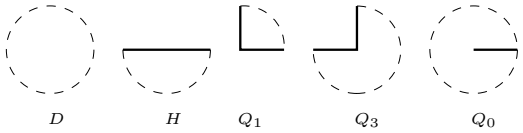


Figure 2.1: The charts for a surface with piecewise smooth boundary.

boundary, the resulting surface has a sharp crease.

We allow two adjacent smooth segments of a boundary to be joined, producing a crease ending in a *dart* (cf. [36]). For dart vertices an additional chart Q_0 is required; the surface near a dart can be deformed into this chart smoothly everywhere except at an open edge starting at the center of the disk.

It is important to observe that convex and concave corners, while being equivalent topologically, are not differentially equivalent. That is, there is no C^1 *nondegenerate* map from Q_1 to Q_3. Therefore, a single subdivision rule can not produce both types of corners [109]. In general, any complete set of subdivision rules should contain separate rules for all chart types. Most, if not all, known schemes miss some of the necessary rules.

2.4 Problems with common rules

In this section, we demonstrate some problems of existing subdivision rules. We will see that not all piecewise-smooth surfaces can be adequately represented in these schemes.

Concave corners. Concave corners often arise in modeling tasks (e.g., surfaces with holes). In an attempt to model such a corner with subdivision surfaces, one might arrange the control mesh in a concave configuration and expect

the surface to approximate the configuration. However, the corner rules of popular subdivision schemes (e.g., [36]) can only generate convex corners. If the control mesh is in a concave configuration, the rules force the surface to approach the corner from the outer, convex, side, causing the surface to develop a fold (Figure 2.2).

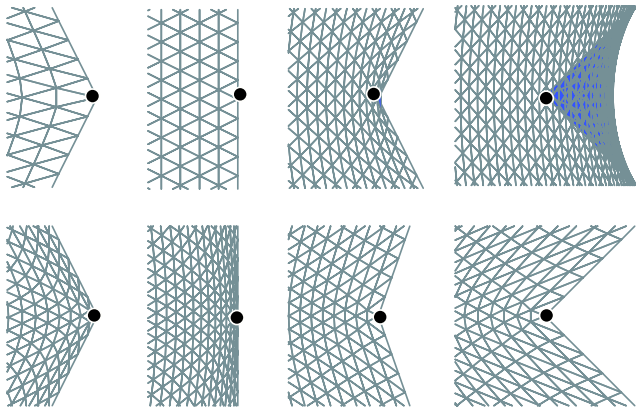


Figure 2.2: Upper row: behavior of a subdivision surface when rules of Hoppe et al. [36] are applied near a corner of the control mesh. As the corner of the control mesh is moved, the surface develops a fold. Lower row: our concave corner rules applied to the same mesh. The concave rules produce a small fold if applied to a convex control mesh configuration (not visible in the picture). For a concave configuration, our rule produces surfaces without folds.

Boundary rules. Hoppe et al. [36] observed that standard subdivision rules fail to produce smooth surfaces at extraordinary boundary vertices. They propose to change the subdivision scheme for the boundary curve in order to generate smooth surfaces. However, the boundary curve now depends on the interior of the control mesh. More specifically, the number of the interior vertices ad-

jacent to each boundary vertex. This side effect is undesirable if one wants to join surfaces along their boundary curves: Two separate meshes might initially have the same boundary, but after subdivision a gap between the meshes can appear (Figure 2.6).

Moreover, even though the rules of [36] are formally smooth, they might produce undesirable sharp normal transitions if the control mesh is twisted (Figure 2.7).

2.5 Subdivision and eigenanalysis

In this section, we briefly state several facts of the theory of subdivision [17], which are helpful to understand the problems described above and our solutions.

Subdivision algorithms recursively refine a control mesh, recomputing vertex positions and inserting new vertices on edges (and possibly faces).

Our method of constructing subdivision rules is based on manipulating the eigenstructure of *subdivision matrices* associated with most common subdivision rules. This idea can be traced back to [21]. Consider a vertex v, and let p be the vector of control points in a neighborhood of the vertex (Figure 2.3).

Let S be the matrix of subdivision coefficients relating the vector of control points p^m on subdivision level m to the vector of control points p^{m+1} on a similar neighborhood on the next subdivision level. Suppose the size of the matrix is N. Many properties of the subdivision scheme can be deduced from the eigenstructure of S. Let us decompose the vector of control points p with respect to the eigenbasis $\{x^i\}$, $i = 0..N-1$, of S, $p = \mathbf{a}_0 x^0 + \mathbf{a}_1 x^1 + \mathbf{a}_2 x^2 + \ldots$ (it exists in the cases of importance to us).

Note that we decompose a vector of 3D points: the coefficients $\mathbf{a}_i$ are 3D

vectors, which are componentwise multiplied with eigenvectors x^i.

We assume that the eigenvectors x^i are arranged in the order of non-increasing eigenvalues. For a convergent scheme, the first eigenvalue λ_0 is 1, and the eigenvector x_0 has all components equal to one; this is also required for invariance with respect to rigid and, more generally, arbitrary affine transformations.

Subdividing the surface m times means that the subdivision matrix is applied m times to the control point vector p.

$$S^m p = \lambda_0^m \mathbf{a}_0 x^0 + \lambda_1^m \mathbf{a}_1 x^1 + \lambda_2^m \mathbf{a}_2 x^2 + \cdots \quad \text{(Iterated Subdivision)}$$

If we further assume that λ_1 and λ_2 are real and equal, and $\lambda_1 = \lambda_2 = \lambda > |\lambda_3|$, we see from this formula that the vector of control points p^m can be approximated by $\mathbf{a}_0 x^0 + \lambda^m(\mathbf{a}_1 x^1 + \mathbf{a}_2 x^2)$; the rest of the terms decay to zero faster. If $\mathbf{a}_1 \times \mathbf{a}_2$ is not zero, then all of the control points p_i^m are close to the plane passing through $\mathbf{a}_0$ and spanned by vectors $\mathbf{a}_1$ and $\mathbf{a}_2$. As $m \to \infty$, the positions of all points converge to a_0.

This means that the limit position of the center vertex is $\mathbf{a}_0$; the tangent directions at this position are $\mathbf{a}_1$ and $\mathbf{a}_2$. We compute these values using the left eigenvectors of S (i.e., vectors l^i, satisfying $(l^i, x^i) = 1$ and $(l^i, x^j) = 0$ if $i \neq j$): $\mathbf{a}_i = (l^i, p)$.

These observations form the basis of our method: to ensure convergence to the tangent plane, we decrease the magnitudes of all eigenvalues except for those that correspond to the vectors $\mathbf{a}_1$, $\mathbf{a}_2$ spanning the desired tangent plane. We also modify the vectors $\mathbf{a}_1$ and $\mathbf{a}_2$ to change the direction of the normal. It should be noted that obtaining the correct spectrum of the subdivision matrix is not sufficient for smoothness analysis of subdivision; once our rules are formulated,

we still have to prove that the resulting surfaces are C^1, using the characteristic map analysis.

2.6 Algorithm

2.6.1 Tagged meshes

Before describing our set of subdivision rules, we start with the description of the tagged meshes which our algorithms accept as input. We use these meshes to represent piecewise-smooth surfaces: edges and vertices of the mesh can be tagged to generate the singularities described in Section 2.3.

The complete list of tags is as follows. Edges can be tagged as *crease edges*. A vertex with incident crease edges receives one of the following tags:

- *crease vertex:* joins exactly two incident crease edges smoothly.
- *corner vertex:* connects two or more creases in a corner (convex or concave).
- *dart vertex:* causes the crease to blend smoothly into the surface.

We require that all edges on the boundary of the mesh are tagged as crease edges. Boundary vertices are tagged as corner or crease vertices.

Crease edges divide the mesh into separate patches, several of which can meet in a corner vertex. At a corner vertex, the creases meeting at that vertex separate the ring of triangles around the vertex into sectors. We label each sector of the mesh as *convex sector* or *concave sector* indicating how the surface should approach the corner.

The only restriction that we place on sector tags is that we require concave sectors to consist of at least two faces. An example of a tagged mesh is given in Figure 2.4.

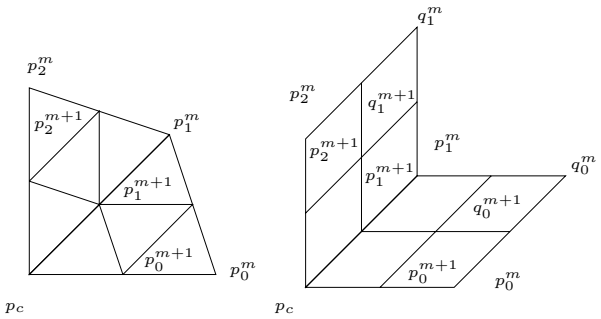


Figure 2.3: Neighborhoods of a vertex on different subdivision levels. The subdivision matrix relates the vector of control points p^m to the control points on the next level p^{m+1}. For a neighborhood of k triangles $p^m = \{p_c^m, p_0^m \dots p_{k-1}^m\}$, for k quadrilaterals $p^m = \{p_c^m, p_0^m \dots p_{k-1}^m, q_0^m \dots q_{k-1}^m\}$

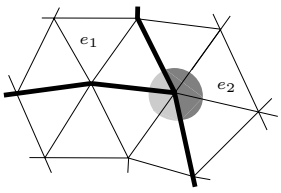


Figure 2.4: Crease edges meeting in a corner with two convex (light grey) and one concave (dark grey) sectors. Our subdivision scheme modifies the rules for edges incident to crease vertices (e.g., e_1) and corners (e.g. e_2).

In our implementation, the user applies the tags interactively, and the user interface prohibits an inconsistently tagged mesh (for example, there cannot be a corner vertex with some sector untagged). Also, the user can specify normal directions and *flatness parameters* for untagged vertices, crease vertices, and for each sector at a corner vertex. The flatness parameter determines how quickly the surface approaches the tangent plane in the neighborhood of a control point. This parameter is essential to our concave corner rules. Additionally, it improves the user control over the surface, for example, one can flatten a twist in the mesh (as shown in Figure 2.7). It is important to note however, that while manipulating these parameters is possible, it is not necessary: we provide default values reasonable for most situations (Section 2.6.2).

2.6.2 Subdivision rules

We describe our sets of rules for the triangular and quadrilateral schemes in parallel, as they are structurally very similar.

Our algorithm consists out of two stages, which, if desired, can be merged, but are conceptually easier to understand separately.

The first stage is a single iteration over the mesh during which we refine the position of existing vertices (vertex points) and insert new vertices on edges (edge points). For the quadrilateral scheme, we also need to insert vertices in the centers of faces (face points). The first stage is similar to one subdivision step of standard algorithms, but the weights that we use are somewhat different. In the following we refer to the rules of Loop and Catmull-Clark as standard rules.

Vertex points. We apply the standard vertex rules to reposition untagged vertices and dart vertices. The new control point at a vertex is the weighted average of the control points in its neighborhood.

If a vertex has k adjacent polygons, then its new position is a combination of the old position with weight $5/8$ and of the sum all surrounding control points with weight $3/8k$, for $k \neq 3$. In case $k = 3$ we use a special set of coefficients with the weight of the central vertex equal to $7/16$ [103]. For the quadrilateral scheme, the center vertex has weight $1 - \beta_1 - \beta_2$, while all adjacent vertices have weight β_1/k; the remaining vertices in the ring receive weight β_2/k with $\beta_1 = 3/(2k)$ and $\beta_2 = 1/(4k)$.

A crease vertex is refined as the average of its old position with weight $3/4$ and the two adjacent crease vertices with weight $1/8$ each. Corner vertices are interpolated.

Face points. For the quadrilateral scheme we insert a vertex at the centroid of each face; only one rule is necessary.

Edge points. This is the most complicated case. We choose the rule for an edge point depending on the tag of the edge and the tags of adjacent vertices and sectors. In the absence of tags, we apply the standard edge rules. The averaging masks are given in Figure 2.5.

We insert a new vertex on a crease edge as the average of the two adjacent vertices.

The remaining case of an untagged edge e adjacent to a tagged vertex v is illustrated in Figure 2.4. We modify the standard edge rule in the following way: we parameterize the rule by θ_k, which depends on the adjacent vertex tag

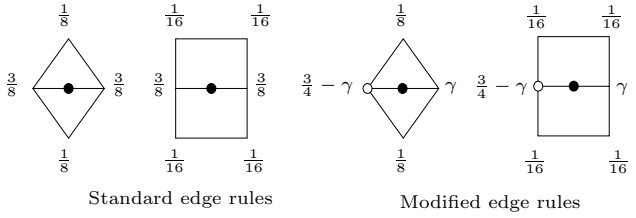


Figure 2.5: Edge rules for triangular and quadrilateral schemes. These rules apply to untagged edges. When both endpoints are untagged, we use standard rules. In case of a tagged endpoint we modify the rule such that the tagged endpoint (marked with a circle) receives coefficient $3/4 - \gamma$.

and sector tag. Let the vertices be labeled as in Figure 2.3, and let the position of the tagged endpoint be p_c^m, the other endpoint is p_i^m. We insert a vertex on the edge at position p_i^{m+1}. The edge rule for the triangular scheme is

$$p_i^{m+1} = (3/4 - \gamma)\, p_c^m + \gamma p_i^m + 1/8 \left(p_{i-1}^m + p_{i+1}^m\right).$$

We use a similar rule for the quadrilateral case:

$$p_i^{m+1} = (3/4 - \gamma)\, p_c^m + \gamma p_i^m + 1/16 \left(p_{i-1}^m + p_{i+1}^m + q_{i-1}^m + q_i^m\right).$$

The subdivision masks are illustrated in Figure 2.5. In each case γ is given in terms of parameter θ_k:

$$\begin{aligned} \gamma(\theta_k) &= 1/2 - 1/4 \cos\theta_k \quad \text{(triangular scheme)} \\ \gamma(\theta_k) &= 3/8 - 1/4 \cos\theta_k \quad \text{(quadrilateral scheme).} \end{aligned}$$

For a dart vertex v, we use $\theta_k = 2\pi/k$, where k is the total number of polygons adjacent to v. If v is a crease vertex, we use $\theta_k = \pi/k$, where k is the number of polygons adjacent to v in the sector of e.

At a corner vertex v we differentiate whether e is in a convex or concave sector. For a convex corner we use $\theta_k = \alpha/k$, where α is the angle between the two crease edges spanning the sector (k as above), for concave corners $\theta_k = (2\pi - \alpha)/k$.

2.6.3 Flatness and normal modification

The second stage of the algorithm is always applied at concave corner vertices and vertices with prescribed normals. It can be also applied at other boundary and interior vertices when it is desirable to increase flatness near a vertex or achieve C^2-continuity.

There are two slightly different types of position modifications performed at this stage: normal and flatness modification. Whenever we compute a vertex position in the neighborhood of a vertex subject to normal or flatness modification we compute the position using the rules above and modify it in a second step. The required eigenvectors for these modifications are listed in the appendix 3.6.

Flatness modification. We observe that we can control how quickly the control points in a neighborhood converge towards the tangent plane. The equation for iterated subdivision suggests to accelerate the convergence by reducing eigenvalues λ_i, $i = 3 \dots N-1$. We introduce a *flatness parameter* s and modify the subdivision rule to scale all eigenvalues except λ_0 and $\lambda = \lambda_1 = \lambda_2$ by factor $1 - s$. The vector of control points p after subdivision in a neighborhood of a point is modified as follows:

$$p^{\text{new}} = (1-s)\, p + s \left(\mathbf{a}_0 x^0 + \mathbf{a}_1 x^1 + \mathbf{a}_2 x^2 \right),$$

where $\mathbf{a}_i = (l^i, p)$, and $0 \leq s \leq 1$. Geometrically, the modified rule blends between control point positions before flatness modification and certain points in the tangent plane, which are typically close to the projection of the original control point. The limit position $\mathbf{a}_0$ of the center vertex remains unchanged.

The flatness modification is always applied at concave corner vertices; the default values for the parameter s is $s = 1 - 1/(2 + \cos\theta_k - \cos k\theta_k)$, which ensures that the surface is C^1 in this case. In other cases, s can be taken to be 0 by default.

C^2-modification. The flatness modification can be also used to make the subdivision scheme C^2, similar to the flat spot modifications [83]. It is known from the theory of subdivision that under certain conditions a scheme which is C^2 away from extraordinary vertices, generates surfaces which are C^2 at extraordinary vertices if all eigenvalues excluding 1 and λ are less than the squared subdominant eigenvalue. This can be easily achieved using flatness modification: s is taken to be less than $|\lambda|^2 / \max_{i>3} |\lambda_i|$. In general, values of s close to this quantity produce surfaces of better shape, but with greater curvature oscillations. It is worth noting that this approach has a fundamental problem: the resulting surface has zero curvature at the extraordinary vertex; the results of [82] indicate that for schemes with small support this is inevitable.

Normal modification. We introduce a similar modification, which allows one to interpolate given tangent and normal position at a vertex v. As above, we modify the control point positions in v's neighborhood after each subdivision step. In this case, the parameter t blends between the unmodified positions and positions in the prescribed tangent plane, while the limit position $\mathbf{a}_0$ of v remains

unchanged.

For a prescribed tangent vector pair $\mathbf{a}_1'$ and $\mathbf{a}_2'$, we modify

$$p^{\text{new}} = p + t\left((\mathbf{a}_1' - \mathbf{a}_1)\, x^1 + (\mathbf{a}_2' - \mathbf{a}_2)\, x^2\right);$$

where $\mathbf{a}_i = (l^i, p)$ and $0 \leq t \leq 1$. In case of a prescribed normal direction n we compute the tangent vectors as $\mathbf{a}_i' = \mathbf{a}_i - (\mathbf{a}_i, n)n$.

We observe that the subdivision rules are no longer applied to each coordinate of the control points separately; rather, the whole 3D vector is required. We can think of this as a generalized form of subdivision, where the coefficients are matrices rather than scalars. Thus, a control point position p_i^{m+1} in a neighborhood with prescribed normal n on level $m+1$ can be explicitly expressed as

$$p_i^{m+1} = \sum_j p_j^m \left(s_{ij}\mathbf{Id} - t\left(\sum_k x_i^1 l_k^1 s_{kj} + x_i^2 l_k^2 s_{kj}\right) n^T n \right)$$

where s_{ij} are entries of the original subdivision matrix S and $\mathbf{Id}$ the 3×3 identity matrix. It should be noted that our analysis applies only to the case $t = 1$, which we use as a default value; the analysis of the general case is still an open question.

2.7 Discussion

We have presented a number of simple extensions to the standard Catmull-Clark and Loop subdivision schemes that resolve some problems with existing rules.

Our rules are designed to coincide with cubic endpoint interpolating B-splines rules along a crease. As a consequence, the generated crease curves depend only on the crease control points. Therefore, it is possible to modify the interior of a surface patch without any effect on the bounding crease curves;

moreover, one can join piecewise-smooth surfaces without gaps and combine them with other surface representations supporting B-spline boundaries.

We can understand the behavior of the surface in a neighborhood of a corner or crease vertex from the eigenstructure of the corresponding subdivision matrix.

If we apply the standard rules in the neighborhood of a crease vertex, the eigenvalue $1/2$ corresponding to the tangent vector of the crease is not subdominant. As a result, the surface contracts at a different rate from the crease, leading to a degenerate configuration without tangent plane (Figure 2.2). The situation for corner vertices is similar as both tangent vectors are determined from crease curve segments with eigenvalue $1/2$.

Our subdivision rules ensure that $1/2$ is the subdominant eigenvalue in both cases. It it not difficult to see that $1/2$ is an eigenvalue: Consider a planar fan of k congruent polygons, where each polygon contributes an angle θ_k to the total angle $\theta = k\theta_k$. If we treat this configuration as a crease or corner neighborhood and apply our modified subdivision rules, then the center vertex does not change its position, and for each adjacent edge we insert a vertex at exactly the midpoint. Thus, the configuration is scaled down by a factor of $1/2$, i.e., $1/2$ is an eigenvalue.

It turns out that $\lambda = 1/2$ is indeed subdominant for crease vertices and convex corners. For concave corners we ensure subdominance by reducing all other eigenvalues (except $\lambda_0 = 1$) using the flatness modification with parameter s satisfying $(1 - s)(2 + \cos\theta_k - \cos k\theta_k) < 2$. Figure 2.11 demonstrates how the flatness modification pulls the neighborhood of a convex corner into its tangent plane.

Our implementation of the rules is available on the Web. We have also developed explicit evaluation rules for our schemes, extending [96].

2.8 Results and Conclusions

Surfaces with creases and corners of various types are illustrated in Figures 2.12 and 2.13(b). All the surfaces in Figure 2.12 are generated from the same control mesh by applying different tags. Note how convex and concave sectors meet along the crease of the torus.

Figures 2.8 and 2.10 demonstrate normal interpolation for boundary, corner and interior vertices; directions of normals are adjusted to obtain desired shapes without modifying the control mesh. Other applications are possible: we have applied normal modification to create certain surface characteristics: randomly perturbing the top-level normals produces a wavy doughnut from a torus-like control mesh; perturbing normals on the first subdivision levels creates a noisy doughnut (Figure 2.13(c) and (d)).

Conclusions and future work. We have presented a simple modification of the two most popular subdivision schemes that improves the behavior of the generated surfaces on boundary and creases and provides additional controls for surface modeling.

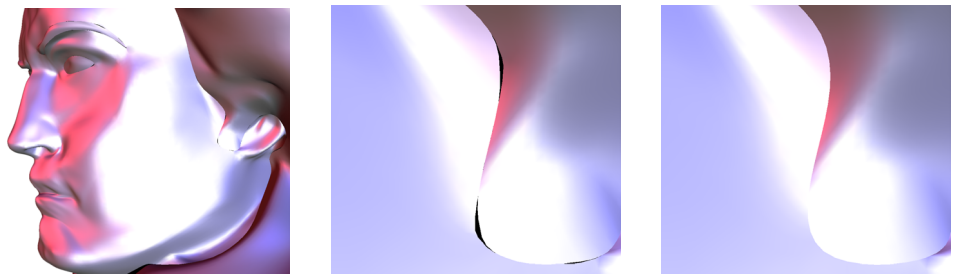
Even though the class of surfaces considered in this paper is quite general, we have excluded many types of surface singularities. Future work might explore which other singularities are useful for modeling purpose and how to construct subdivision rules to create such features.

2.9 Acknowledgments

We are greatly indebted to Peter Schröder for his support and suggestions, and for trying out the schemes and discovering numerous hard-to find typos in the

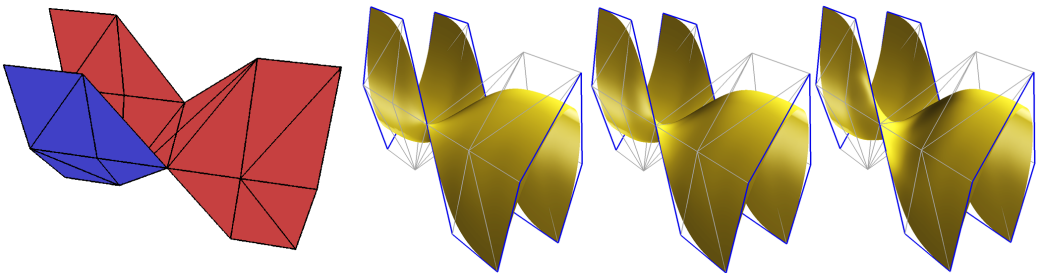
formulas. This work has its origin in discussions with Tom Duchamp. We would like to thank the anonymous reviewers for their comments.

A portion of this work was supported by NSF award ACI-9978147.



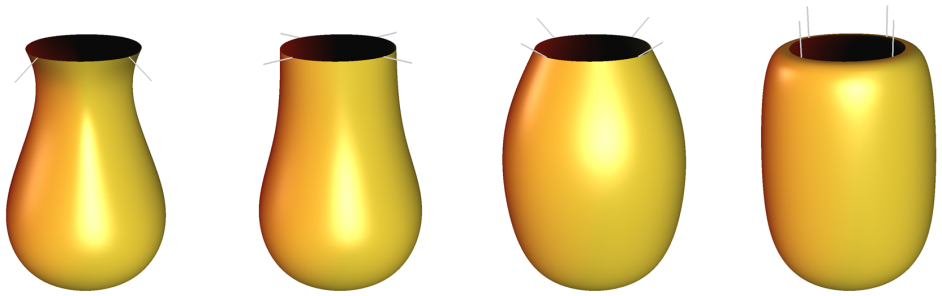
(a) (b) (c)

Figure 2.6: Subdivision on meshes with boundaries: Beethoven's face and hair are modeled as separate meshes with identical boundaries. (a) and (b): the rules of [36] result in a gap between the surfaces due to extraordinary vertices. (b) A close-up on the gaps at the ear. (c) With our rules no gap is created.



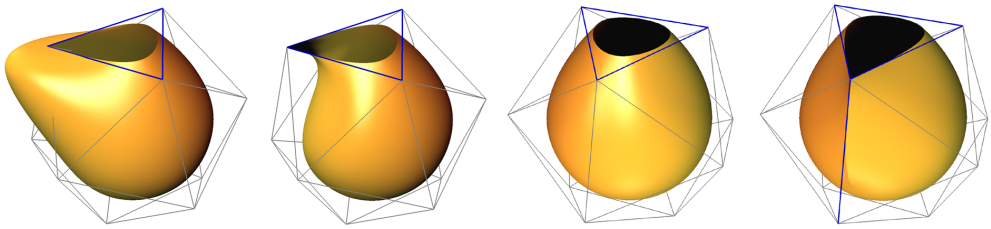
(a) (b) (c) (d)

Figure 2.7: (a) Control mesh with a twist on the boundary. (b) Normal varies rapidly near the point although the surface is formally smooth: there is a single bright spot on the front-facing boundary. (c), (d) Our algorithm reduces the variation: the highlights become larger.



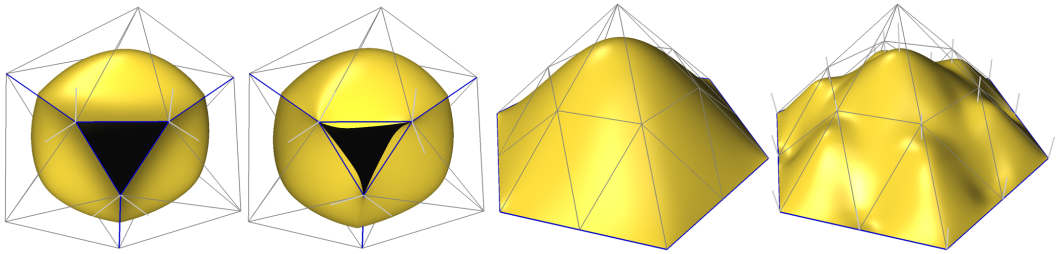
(a) (b) (c) (d)

Figure 2.8: Normal interpolation for quadrilateral subdivision. Prescribed directions: (a) tilted downwards, (b) horizontal, (c) no modification, (d) vertical.



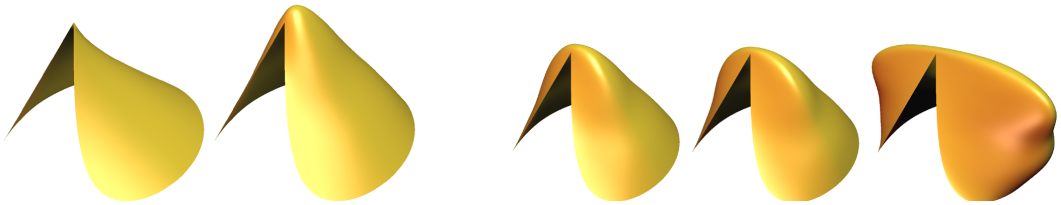
(a) (b) (c) (d)

Figure 2.9: Features: (a) concave corner, (b) convex corner, (c) smooth crease, (d) corner with two convex sectors.



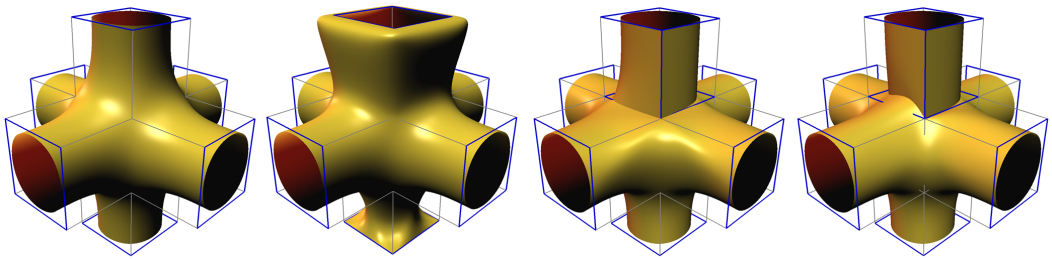
(a) (b) (c) (d)

Figure 2.10: Normal interpolation. (a) Surface with convex corners. (b) Prescribed directions: at each corner we tilt the normal for one surface sector slightly inwards. (c) Smooth surface. (d) Same control mesh but all normals vertical.



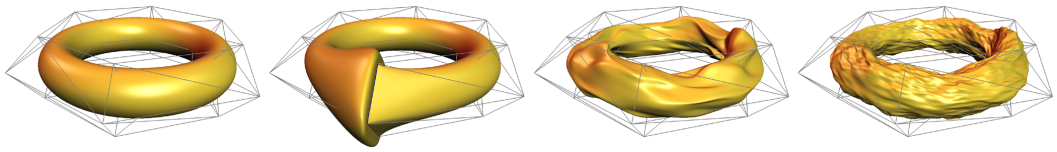
(a) (b) (c) (d) (e)

Figure 2.11: Concave corner rules. (a) A corner without flatness modification. (b) Flatness modification lifts the surface into its tangent plane. (c-e) The corner shape for different values of θ_k.



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

Figure 2.12: Surface manipulation with corners. (a) Smooth boundary curves. (b) Concave corners on top, convex corners on bottom. (c) Corners with convex and concave sectors. (d) Creases and corners as for (c) but with prescribed normal direction on concave sectors.



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

Figure 2.13: Manipulating a torus. (a) The original surface. (b) A surface with creases and convex/concave corners. (c) Wavy torus: we deform the torus by randomly perturbing normals of the control mesh. (d) Noisy torus: we perturb the normals on the first four subdivision levels.

Chapter 3

Sharp Features on Multiresolution Subdivision Surfaces

The contents of this chapter is going to appear in an article published in the journal *Graphical Models*. An earlier version of the article was published in the Proceedings of Pacific Graphics 2001 [8]. It describes an approach to introducing sharp features into multiresolution surfaces on different resolution levels. This approach extends some of the techniques described in the previous chapter on multiresolution setting.

3.1 Introduction

Interactive editing is of great importance for creating geometric models for a variety of applications, ranging from mechanical design to movie character creation. Often times, modeling begins with an existing object on which the user

performs a sequence of editing operations that lead to the desired shape. Of particular interest are small-scale features such as engravings and embossed details that are encountered on many real-life objects.

Traditionally, geometric modeling has relied on non-uniform rational B-splines (NURBS) for surface design. However, NURBS have well-known limitations such as the inability to address arbitrary topology, tedious cross-boundary continuity management, and difficulty representing different resolution levels. In addition, editing operations such as those considered in this paper typically require features to be aligned with iso-parameter lines or patch boundaries, or other complex manipulations in parameter space.

While techniques such as free-form deformations [90], wires [95], and procedural modeling [76] offer alternative ways to edit three-dimensional objects, typically they do not present a unified representation that includes both the original surface and the edits. Thus, in many cases, the resulting representation is not the same as the original, but an extension of it. The main drawback of this approach is that algorithms that have been developed for the original representation are not directly applicable to the result and special cases may have to be considered.

The past few years have seen considerable advances in subdivision theory and many common NURBS operations have been translated into the subdivision setting. Subdivision theory [109], parametric evaluation [96], and applications such as interactive editing [108, 42], trimming [59], boolean operations [5], and geometry compression [39] have contributed to the increasing popularity of multiresolution subdivision surfaces. To date, they have been used in commercial modelers (e.g., Alias/Wavefront's Maya, Pixar's Renderman, Nichimen's Mirai, and Micropace's Lightwave 3D) and are currently making their way through in

game engines and hardware implementations. Subdivision algorithms are attractive because of their conceptual simplicity and efficiency with which they can generate smooth surfaces starting from arbitrary meshes. Multiresolution subdivision surfaces offer additional flexibility by allowing modeling of details at different resolution levels and ensuring that fine-scale edits blend naturally with coarse shape deformations.

In this paper, we address the problems of feature placement and feature creation by providing a set of tools that allow fine-scale editing and trimming operations to be applied anywhere on a surface. We use multiresolution subdivision surfaces as our representation and we ensure that this representation is preserved after editing. This gives us the flexibility to integrate our technique with other algorithms developed for multiresolution subdivision surfaces. In our implementation, we use a subdivision scheme for quadrilateral meshes, however a similar algorithm can be developed for triangular meshes. Our contributions include:

1. An algorithm to produce sharp features at arbitrary locations on a multiresolution surface without remeshing the control mesh. The sharp features are created interactively, along curves drawn by the user on the target surface.

2. An extended set of rules for the Catmull-Clark subdivision scheme that allow the creation of creases and boundaries along diagonals of quadrilateral mesh faces.

3. A unified solution to offsetting and trimming operations. Using our technique, a sharp crease having a user-defined profile may be applied along

a given curve. Alternatively, the portion of the surface delimited by the curve can be trimmed off, creating a hole in the surface.

The remaining sections of the paper are organized as follows: in section 3.2 we overview surface modeling methods and we emphasize the main differences between our approach and existing techniques. The core of our method is presented in section 3.3. In section 3.4 we present our results and we illustrate applications. Finally, in section 3.5 we summarize our work and we point out open issues.

3.2 Background and Related Work

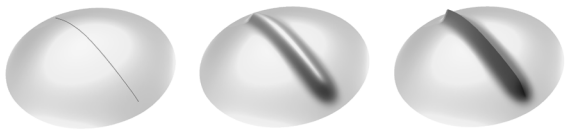


Figure 3.1: Surface editing. Features are added to an initial surface (left). Smooth and sharp features are fundamentally different: smooth features (center) add bumps to the surface, while sharp features create tangent plane discontinuities (right).

Subdivision surfaces [17] efficiently represent free-form surfaces of arbitrary topology. A subdivision surface is defined over an initial control mesh and a subdivision scheme is used to recursively refine the control mesh by recomputing vertex positions and inserting new vertices according to certain rules (masks). Recursive subdivision produces a hierarchy of meshes converging to a smooth limit surface. Most objects of interests to geometric design, however, are only piecewise smooth and exhibit sharp creases and corners. To model them us-

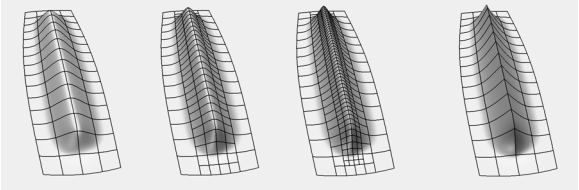


Figure 3.2: Smooth surface representations do not capture sharp features. Multiresolution Catmull-Clark surfaces can approximate sharp features by adding detail coefficients at finer levels (left three pictures). Instead, we use piecewise-smooth multiresolution surfaces to exactly represent sharp features without any detail coefficients (right).

ing subdivision, special rules are needed to avoid smoothing of sharp details (Figure 3.1 and 3.2). Previous work in this area has focused on defining such special rules. Hoppe et al. introduce rules to create sharp features on subdivision surfaces in [36]. The work of DeRose et al. [18] extends Hoppe's approach to achieve creases of controllable sharpness by using subdivision rules parameterized by a sharpness factor. Our work builds upon subdivision schemes for piecewise-smooth surfaces [6] where control mesh vertices and edges are tagged in order to generate singularities, such as creases, darts, and corners. We also draw upon the curve interpolation work of Nasri [72]. In all of these techniques, there is the common requirement that features need to be aligned with the edges of the underlying control mesh. Therefore, the control mesh has to be designed with a particular feature in mind. However, a designer might want to first model an initial shape and apply small scale features in later stages of the design. It is our goal to support this kind of a modeling approach and to allow features to be placed at arbitrary locations on the surface.

Multiresolution subdivision surfaces are a natural extension of subdivision

surfaces that accommodates editing of details at different scales, allowing general shape deformations as well as the creation of minute features. Multiresolution, however, does not solve the problem of sharp features as they can only be placed along edges at discrete locations in the mesh hierarchy.

Our technique removes this constraint by allowing sharp features to be created and edited along any user-defined set of curves on the mesh. The main idea is to view subdivision surfaces as parametric surfaces defined over the coarsest-level mesh similar to MAPS [49]. Similar to the approach in [5] we compute the image of a given curve in the parametric domain and we reparameterize the surface to align the parameterization with the curve on some level of the multiresolution hierarchy. Subsequently, we apply special rules to generate non-smooth features.

The closest work to the technique presented in this paper is that of Khodakovsky and Schröder. In [38] they describe a method for interactive creation of feature curves at arbitrary locations on a surface. To create a feature along a curve, a perturbation according to a given profile is applied in the neighborhood of the curve, while maintaining smooth boundary conditions. There are no restrictions on the position of the curve with respect to the underlying surface, however, the representation used is no longer a pure multiresolution surface. In order to create a sharp feature with this technique, it is necessary to enforce the feature profile at each level of the multiresolution hierarchy. Both the surface and the feature curve are needed to represent the resulting surface. Thus, one cannot directly use techniques developed for subdivision surfaces (i.e., evaluation). In our method, we address the issue of arbitrarily placed sharp features within the multiresolution subdivision setting, thus allowing for greater flexibility in combining feature editing with other existing subdivision tools.

A by-product of our method is the ability to perform trimming of surfaces by simply discarding the portion of surface inside a given curve. Trimming is an important design operation that has been traditionally difficult to perform on parametric surfaces. Our work is complementary to the trimming approach of Litke et al. [59] where quasi-interpolation is used to approximate a trimmed surface with a combined subdivision surface [54]. Similarly, quasi-interpolation may be combined with our approach to obtain trimmed multiresolution surfaces within a specified tolerance.

3.3 Feature Editing Algorithm

The input to our feature editing algorithm consists of a Catmull-Clark [11, 21] multiresolution subdivision surface, a set of feature curves on the surface, and a user-selected profile to be applied along these curves. The result is a multiresolution subdivision surface with offset or trim features along the given curves.

The basic idea is to model sharp features with piecewise-smooth surfaces. We view feature curves as boundaries between smooth surface patches. Sharp features occur along patch boundaries where patches with distinct tangent planes are joined.

Our multiresolution surface representation enables us to represent sharp features as boundaries or creases by tagging the control mesh edges. In general, the creases generated in this fashion do not coincide with the user specified feature curves. Moreover, it may not be possible to create topologically equivalent curves due to the control mesh topology. Therefore, before we can represent a feature, we need to change the surface parameterization.

Our algorithm proceeds in two steps: *Reparameterization* and *Feature Creation*. We first reparameterize the surface by sliding the control mesh along the surface in order to sample the feature curve with vertices of the mesh. Hence, we are able to approximate the feature curve by edges or face diagonals of the control mesh. In the following subdivision step, we treat these edges and diagonals as creases in the control mesh, and apply piecewise-smooth subdivision rules to obtain a surface with a sharp feature. The surface patches on each side of the feature may be controlled separately. This allows to shape the feature according to a specified profile. Moreover, for trimming we can discard the surface on one side of the feature without changing the surface on the other side.

3.3.1 Reparameterization

The goal of this step is to align the parameterization of the given surface with a given feature curve. Recall that a multiresolution subdivision surface can be naturally parameterized over the coarsest level control mesh (Figure 3.3).

Some notation is necessary to describe the reparameterization. Let c denote an input curve defined on the parameter domain X of the surface, $c : [0, 1] \to X$. In general, c traverses the domain X at arbitrary positions. We want to reparameterize the domain X such that c passes through the vertices of X. Therefore, we compute a one-to-one mapping $\Pi : X \to X$ which maps vertices of X to curve points: $\Pi(v_i) = c(t_i)$, for some vertices $\{v_0, v_1, \dots\}$ and curve parameters $\{t_0, t_1, \dots\}$. The mapping Π is built to satisfy the following *approximation property* (AP):

(AP): *the piecewise linear curve* $[v_0, v_1, \dots]$ *has the same topology as* c *and either follows along mesh edges or crosses mesh faces diagonally.*

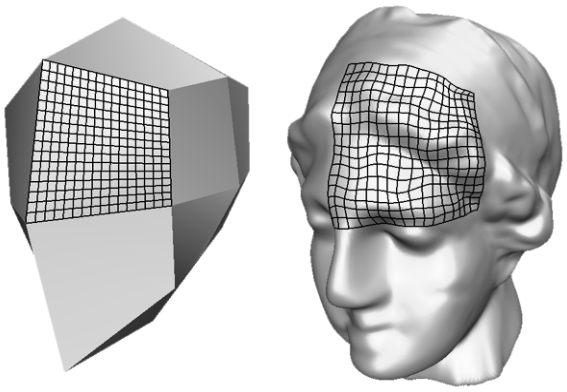


Figure 3.3: Parametric domain and surface. Multiresolution subdivision surfaces can be parameterized over the coarsest level control mesh (left). The subdivision operator maps vertices of the parametric domain to their image on the surface (right).

The reparameterization algorithm proceeds iteratively, alternating *Snapping* and *Refinement* steps. The snapping step moves mesh vertices onto the curve if they are sufficiently close. In the refinement step we simply subdivide the parameterization linearly. The algorithm terminates if the sequence of vertices $\{v_0, v_1, \cdots\}$ along c satisfies the approximation property (AP). Property (AP) is guaranteed to be satisfied after a finite number of steps for piecewise-linear curves c. After the reparameterization, the surface is *Resampled* to reflect the new parameterization.

Snapping. This step moves vertices onto the curve c if they are sufficiently close to it. First, the algorithm traverses the mesh along the curve on a given subdivision level. For all vertices of the traversed faces we compute the closest points on the curve. Distances are measured by parameterizing each quad intersected by c over the unit square and by computing distances to the curve in this

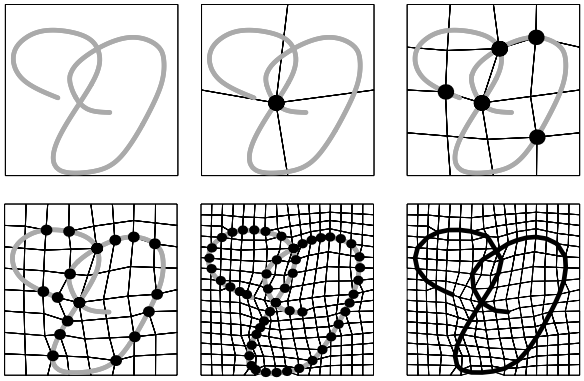


Figure 3.4: Reparameterization matching a feature curve. The quad is recursively split and vertices are snapped to the curve. After four subdivision steps, the curve is approximated by a sequence of mesh vertices: The approximated curve follows edges or passes through quads diagonally.

parameter space. This approach presents an advantage over computing geometric distances in that it does not undersample small quads. Vertices v within a certain distance ε to the curve are snapped to the corresponding closest points $c(t)$ on the curve (see Figures 3.5 and 3.4). We assign $\Pi(v) := c(t)$.

The parameter ε controls the distortion of the reparameterization: small values keep vertices from moving too far, but require more snapping steps. In

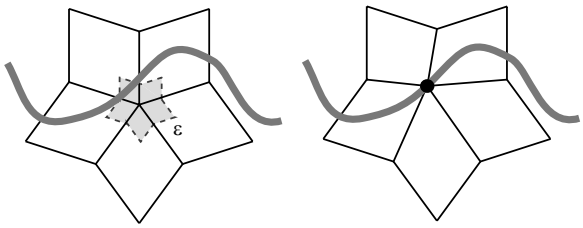


Figure 3.5: Snapping step. Vertices are snapped to closest curve vertex if the distance is less than a certain ε. Note that distances are measured in parameter space, where each face corresponds the unit square. A single snapping step reparameterizes the neighborhood of the snapped vertex.

all cases, ε is less than 0.5 to ensure that the snapping region around each vertex is disjoint from the snapping regions of its neighbors. In our examples, we obtained good results with $\varepsilon = 0.3$.

In some cases, it may be necessary to align specific points in the parameter domain with mesh vertices. This is the case, for instance, where several curves intersect in a corner. Due to the chosen surface representation, we need a mesh vertex at the corner that connects the separate curve branches. Such constraints are enforced during snapping: constrained curve points have higher priority than unconstrained points. For a given vertex, the algorithm first tries to snap to the unmatched constrained vertices. If no snapping is possible, unconstrained points are considered as snap targets.

Refinement. The parameterization Π is piecewise linearly subdivided to increase the resolution and allow future snapping. The subdivision is done similarly to [49]. As in [49], local charts are used to refine the parameterization Π where neighborhoods are mapped to different faces of the coarsest level control mesh.

Resampling. After reparameterization, we resample the surface at the new parameter positions. Intuitively, this moves the control mesh on the surface and places mesh vertices on the feature curve. With the notation from above, we iterate over the vertices of the finest level control mesh. For every vertex v of X, we assign a new position by evaluating the input surface at parameter position $X(v)$. Finally, we apply multiresolution analysis to obtain a multiresolution representation with detail coefficients.

3.3.2 Feature Creation

The reparameterization step approximates the feature curves as chains of mesh edges or diagonals and mesh diagonals. Our idea is to view these curves as crease or boundary curves in the control mesh. In order to create surfaces with sharp features, we apply crease subdivision rules along the feature curves. The shape of the feature can be controlled by offsetting mesh vertices in the feature neighborhood according to a user given profile.

The profiles shown in Figures 3.14 have been created by changing the positions of mesh vertices that are close to a feature curve. Vertices are displaced normal to the surface and the displacement is determined by their distance to the curve. Note that for closed curves different shape profiles may be used for displacing points in the interior and points on the outside.

Trimming can also be performed by simply interpreting a feature curve as a boundary of the mesh. The feature curve cuts the control mesh into separate pieces. Each piece can be subdivided separately and processed further. The resulting surfaces are independent from each other, moreover, their boundary curves are cubic B-splines.

3.3.3 Subdivision Rules for Sharp Features

In this section, we explain how to subdivide control meshes with feature curves (Figure 3.6). As in the standard Catmull-Clark subdivision, we iterate over the mesh on a given level and we compute positions of the vertices on the next level by refining the positions of existing vertices (vertex points) and by inserting new vertices on edges (edge points) and in the faces centers (face points). We apply special rules in the vicinity of the curves and standard rules everywhere

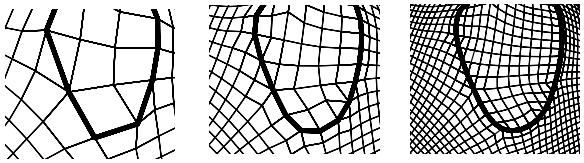


Figure 3.6: Subdivision of a control mesh with a feature curve. The feature curve is a sequence of mesh edges or mesh diagonals acting as a boundary in the mesh. The resulting surface consists of two smooth patches joined along a cubic B-spline boundary. Our subdivision scheme extends piecewise-smooth subdivision with rules for diagonally split faces.

else. Our rules extend the piecewise-smooth subdivision of [6]. The idea is to treat the feature curve as a crease and to refine the mesh on each side of the curve independently to create a tangent-plane discontinuity. As feature curves may pass through quads diagonally, we introduce new rules, that account for such situations.

We use vertex tags to identify the subdivision rules to be applied when traversing the mesh. Initially, we tag all vertices traversed by the feature curves as crease vertices. Additionally, we mark the faces that are cut diagonally by the curve.

Vertex points. A refined control point position corresponding to an untagged vertex is computed as a weighted average of control points in its neighborhood. For a vertex c with valence k (i.e., with k adjacent polygons), its new position c^{i+1} is a linear combination of its old position weighted by $(1 - \beta - \gamma)$ and the positions of the vertices in its 1-ring each weighted by β/k if situated on an edge incident to c or by γ/k otherwise. We use coefficients $\beta = 3/(2k)$ and $\gamma = 1/(4k)$.

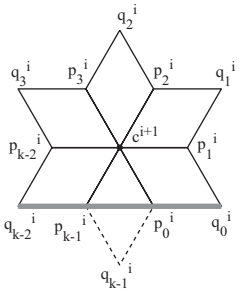


Figure 3.7: Special vertex rule in the neighborhood of a curve that passes through some of the quads diagonally (gray line). Point q^i_{k-1} is on the opposite side of the curve from c^i and it is not used in the computation of the refined position c^{i+1}. The reflection of c^i across the diagonal (p^i_{k-1}, p^i_0) is used instead.

A special situation occurs when some of the quads in the 1-ring of c are split by a curve (see Figure 3.7). The previous rule is modified to ignore vertices that are not on the same side. We use $q' := p^i_0 + p^i_{k-1} - c$.

$$c^{i+1} = (1 - \beta - \gamma)c^i + \frac{1}{k}\left(\beta p^i_{k-1} + \gamma q' + \sum_{j=0}^{k-2} \beta p^i_j + \gamma q^i_j\right)$$

A crease vertex is refined as the average of its old position with weight 3/4 and the two adjacent crease vertices with weights equal to 1/8.

Corner vertices (i.e., where two or more creases meet) require additional rules depending on the neighboring topology, similar to our discussion of creases. For the sake of brevity, we do not include them in this paper. Darts (i.e., smooth transitions of a crease into a surface) require no special consideration, as the standard rules are directly applicable at such points.

Face points. For faces that are not split diagonally by crease curves, we insert a point in the centroid of each face. For faces with a diagonal on the curve, the

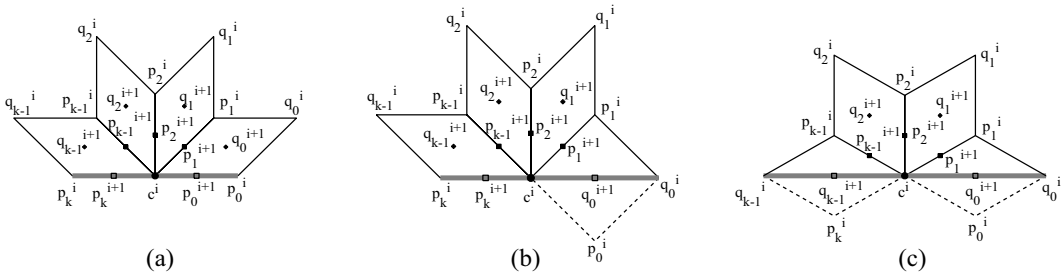


Figure 3.8: Special rules for edge points on edges with one endpoint on a curve (gray line). (a) Sector delimited by the curve consists of quads only: in this case the rules from [6]. (b) Sector begins with a split quad and ends with a full quad. In this case only p_0^i is on the opposite side of the curve and the only rule that needs to be modified is that for p_1^{i+1}. The reflection of c^i across the edge (p_1^i, q_0^i) is used. (c) Sector begins and ends with quads that are split by the curve. Points p_k^i and p_0^i are on the opposite side of the curve. The rules that would normally take into account these points to compute p_1^{i+1} and p_{k-1}^{i+1} are modified to use the reflections of the points p_1^i and p_k^i across the curve instead.

center point is computed as the average of the two diagonal endpoints on the crease.

Edge points. On an edge with both endpoints tagged, we insert a new vertex as the average of the endpoints. When both endpoints are untagged, the standard edge mask applies. The remaining case is that of an edge with one vertex tagged and the other untagged. A curve passing through a mesh vertex partitions the mesh in the neighborhood of that vertex into two *sectors*. We select the rule to be applied to an edge point adjacent to a tagged vertex c depending on the topology of the sectors around the tagged vertex. We distinguish three types of sectors:

1. *Consisting of quads only:* in this case the curve follows two edges incident to c and we can apply the crease rules described in [6]. This case is illustrated in Figure 3.8(a). We choose $\theta_k = \pi/k$, where k is the number of polygons adjacent to c in the sector considered, and apply an edge rule which is parameterized by $\gamma = 3/8 - 1/4\cos\theta_k$. The new edgepoints p_j^{i+1} $(j = 1, \cdots, k-1)$ are computed as

$$p_j^{i+1} = (\frac{3}{4} - \gamma)c^i + \gamma p_j^i + \frac{1}{16}(p_{j-1}^i + p_{j+1}^i + q_{j-1}^i + q_j^i).$$

2. *Beginning with a triangle (i.e., a split quad) and ending with a quad or vice versa:* in this case the curve passes through the vertex following a mesh edge and a mesh diagonal. This case is illustrated in Figure 3.8(b). We use θ_k as above and apply the same edge rule to compute $p_2^{i+1} \cdots, p_{k-1}^{i+1}$. The edge point p_1^{i+1} between the triangle and the first quad is obtained as

$${p_1}^{i+1} = (\frac{11}{16} - \gamma)c^i + (\gamma + \frac{1}{16})p_1^i + \frac{1}{16}(p_2^i + 2q_0^i + q_1^i).$$

3. *Beginning and ending with triangles:* in this case the curve follows two diagonals incident to c (Figure 3.8(c)). We choose $\theta_k = k - 1$ and apply the edgerule of the first case to find $p_2^{i+1}, \ldots p_{k-2}^{i+1}$. The edge points p_1^{i+1} between first triangles and quads is computed as

$${p_1}^{i+1} = (\frac{13}{16} - \gamma)c^i + (\gamma - \frac{1}{16})p_1^i + \frac{1}{16}(p_2^i + 2q_0^i + q_1^i).$$

   The rule for p_{k-1}^{i+1} is symmetric to this. In the special case of $k = 2$, we use $p_1^{i+1} = 1/4c^i + 1/2p_1^i + 1/8(q_0^i + q_1^i)$.

Tangents and normals. Our subdivision scheme has well-defined limit and tangent properties. We can efficiently evaluate limit positions of vertices and

tangent directions by applying specific masks [33]. These masks correspond to the left eigenvectors of the subdivision matrix used at a given vertex. The masks are listed in appendix 3.6.

3.3.4 Discussion of the Subdivision Rules

In this section, we briefly discuss some properties of the previous subdivision rules and we motivate our choice of the special rules for crease neighborhoods.

As we use cubic B-Spline subdivision along the features, the resulting curves are B-Splines defined only in terms of control points along the curves. Moreover, the surfaces on either side of the feature do not depend on each other. This is a consequence of our rules, as no stencil uses vertices from the opposite side of a feature.

Our rules have an easy geometric interpretation (Figure 3.8), but some subdivision theory is needed to understand the rules in more detail. We follow the usual eigen-analysis approach [21] to understand the asymptotic behavior of the subdivision operation. Consider a neighborhood of a crease vertex c as shown in Figure 3.8(a). Iterated subdivision contracts the neighborhoods to a single point. We want to ensure a well-shaped limit configuration and design rules which preserve a specifically chosen configuration. Technically speaking, we design rules that have certain desired subdominant eigenvectors (see Appendix 3.6). We use geometric reasoning to reduce the cases of neighborhoods with triangles to the case of neighborhoods without. The reflections previously mentioned are chosen to map the desired eigenvectors to the eigenvectors of the no-triangle case (Figure 3.8(a)). Thus, we can design subdivision rules as follows: (i) apply reflection to complete triangles, (ii) subdivide using the

usual no-triangle rules and (iii) discard the unnecessary points. The subdivision scheme defined in this way has the desired eigenvectors.

For a complete analysis a larger neighborhood and the corresponding subdivision matrix need to be analyzed. This is beyond the scope of this paper, and instead, we visualize here only the asymptotic behavior of the subdivision rules with illustrations of the characteristic maps (Figure 3.9).

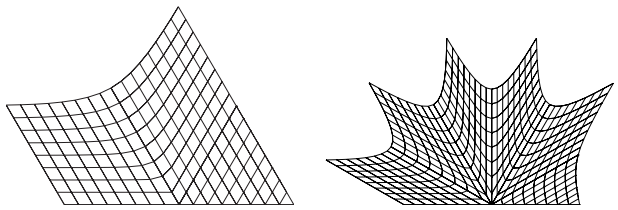


Figure 3.9: Characteristic map for crease vertex neighborhoods with a single triangle. We show maps for valence 3 (left) and valence 6 (right). The maps show the behavior near the curve vertex. Note that the maps are smooth and one-to-one.

3.4 Applications and Results

Figure 3.13 illustrates the steps of the algorithm for a trimming sequence. The creation of sharp features and trim regions is illustrated in figures Figures 3.10 and 3.11. Note the arbitrary position of the curves with respect to the underlying meshes. The models were created interactively on a Pentium III workstation.

Figure 3.14 illustrates the creation of offset features along a curve according to various user-specified profiles. The profiles are based on distance to the curve. In general, computing distances on surfaces is a difficult problem [65]. In our examples (i.e., Figure 3.14), we measure the distance in space, which is

a reasonable approximation for small distances on surfaces with low curvature. Alternatively, one could use geodesic distances on the surface [38].

Also, we use our algorithm to create features on multiresolution surfaces (Figure 3.12(a)). Feature curves with intersections are illustrated in Figures 3.13 and 3.12(b).

Finally, we demonstrate how our trimming approach approximates the result of a precise trimming operation (Figure 3.15). It is the nature of our technique that the resulting surface is different from the input surface (even for a flat feature profile). The differences are due to resampling and the use of piecewise-smooth base functions in the feature neighborhood. Also, the specified feature curves are resampled only at vertices of the control mesh. However, we can control the approximation by resampling surface and feature curves at different levels in the hierarchy. In general, resampling on a finer level reduces the error, but is computationally more expensive. In our implementation the user controls the level on which the resampling takes place.

3.5 Conclusions and Future Work

In this paper we present an efficient method for creating sharp features along an arbitrarily positioned set of curves on a Catmull-Clark multiresolution subdivision surface. We view our surface as a parametric surface defined over the initial control mesh and we change the parameterization to align it with the pre-image of the feature curves in the parameter domain. The result is a surface represented in the same way as the input surface with the curves passing through mesh edges or face diagonals. This property allows us to apply special subdivision rules along the curve to create sharp profiles. Another application

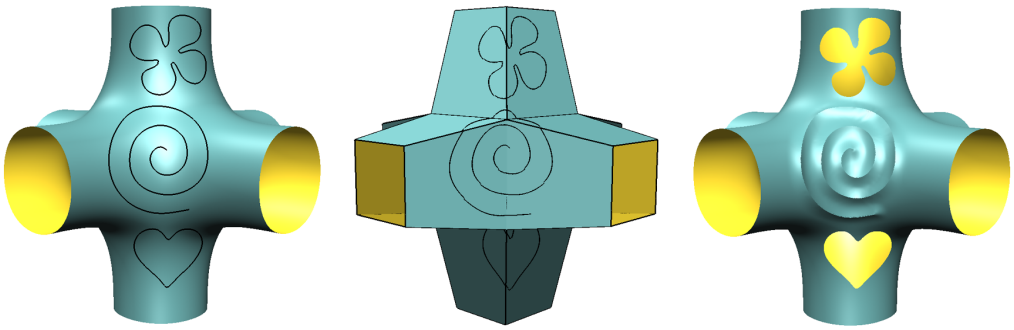


Figure 3.10: Surface editing. Left: surface with user specified feature curves. Closed curves are trim curves, the open curve indicates an offset feature. Center: image of the feature curves in the parametric domain. Right: resulting surface with sharp features and trimmed regions. The displacement of the points along the offset curve is a quadratic function of the distance to the curve.

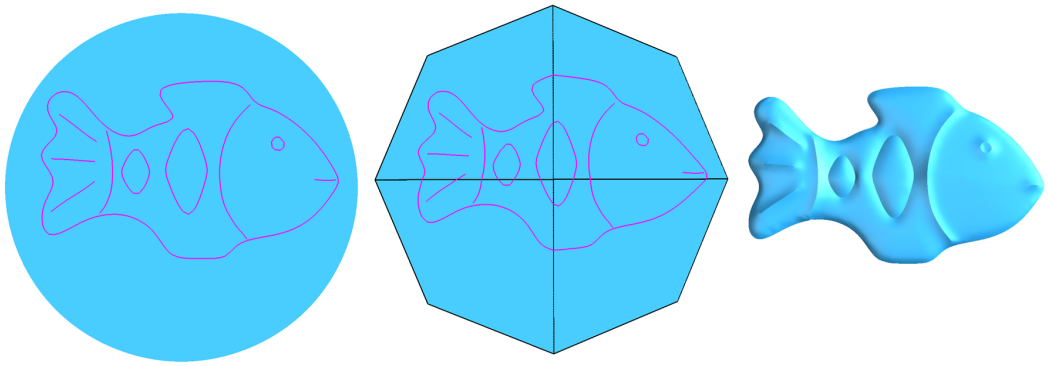
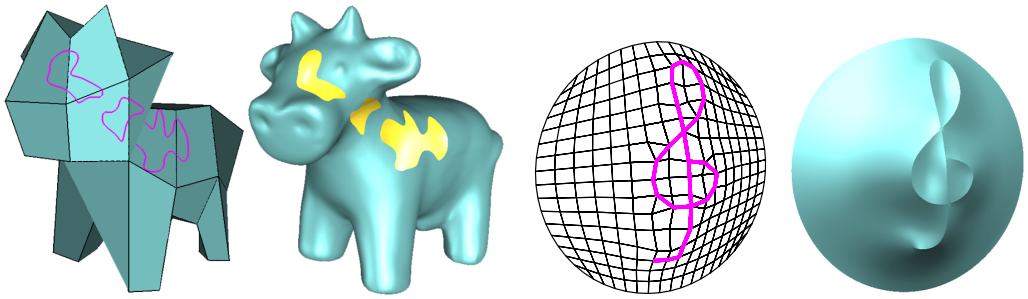
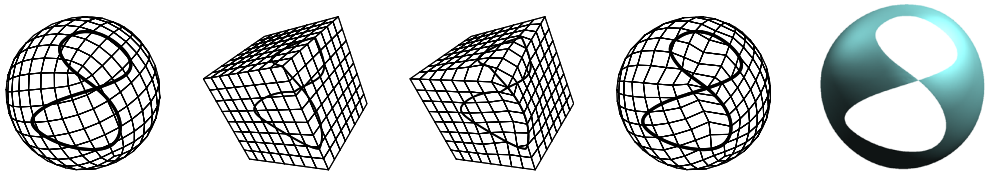


Figure 3.11: Surface shaping. A fish pin is cut from a disk using a trim curve along its outer contour and shaped with offset curves in the interior.



(a) (b)

Figure 3.12: Left image pair: trim features on multiresolution surfaces. Right image pair: offset features with intersections. The control mesh on the right shows how reparameterization aligns the feature with edges and diagonals. Mesh vertices are placed at curve intersections.



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

Figure 3.13: Steps of the algorithm. (a) Wireframe rendering of a surface with feature curve. (b) The surface is parameterized over a cube. (c) Reparameterization aligns mesh edges with the feature. (d) Resampling with respect to new parameterization. (e) Trimming by discarding a piece of the control mesh and subsequent subdivision.

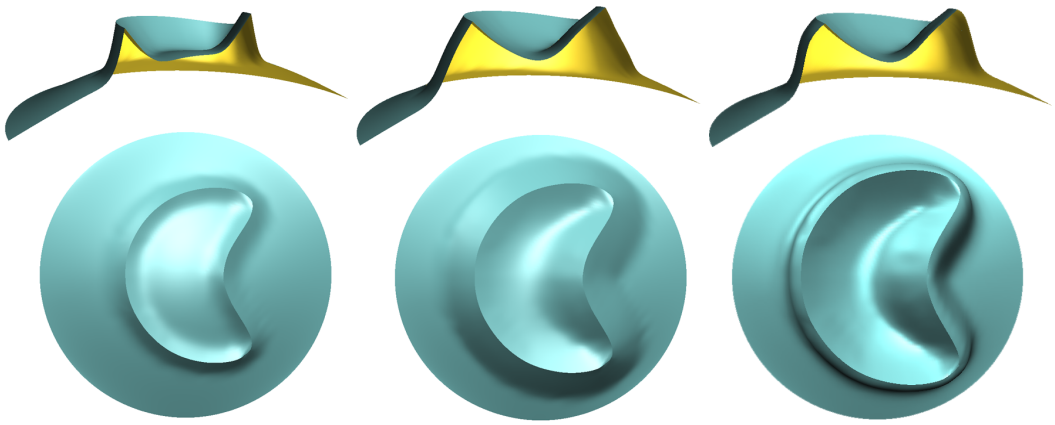


Figure 3.14: Different offset profiles for a feature curve. In all three cases, the interior profile is a quadratic function of the distance to the curve. Left: linear exterior profile. Middle: linear exterior profile; the size of the neighborhood altered is doubled with respect to the previous image. Right: Gaussian exterior profile.

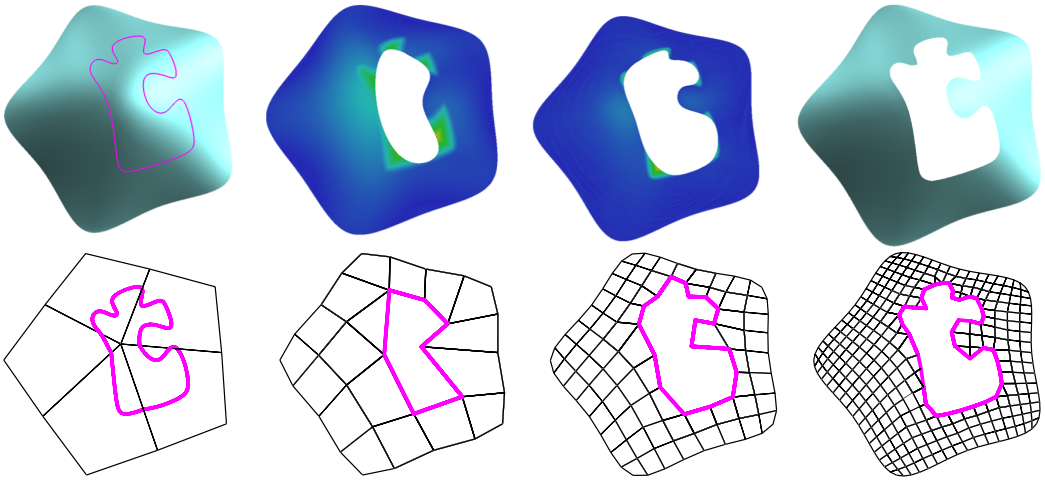


Figure 3.15: Trim operations are applied on different levels of the hierarchy. Top row: distance between the resulting surface and an analytically trimmed surface. Bottom row: corresponding control meshes. The largest error occurs where the surface boundary does not capture the intended trim curve. The error of surface obtained by trimming on level 4 is less than 1% (of the length of a base mesh edge).

of this algorithm is trimming, which can be achieved simply by discarding the portion of the mesh situated inside the trim curve.

Our algorithm takes as input arbitrarily shaped curves, with or without self-intersections, as well as multiple intersecting curves. The number of curves intersecting at a point, however, is limited by the number of connections available between a vertex and its neighbors along edges and diagonals (eight in the regular case).

Multiresolution surfaces that allow for topology changes within the hierarchy [29] could be used to resolve this restriction in future research. Other research might combine quasi-interpolation or surface fitting with our trimming approach and study the approximation along the lines of [59]. Also, we are interested to see whether the special diagonal subdivision rules are useful in a general geometric modeling context. For practical applications, it might be useful to work out expressions for direct evaluation.

3.6 Eigenvectors

We list the right and left eigenvectors corresponding to the special subdivision rules previously described. We denote the dominant left eigenvector by l^0 and the left subdominant eigenvectors by l^1 and l^2, respectively. We denote the dominant right eigenvector by x^0 and we use x^1 and x^2 for the right subdominant eigenvectors. Recall that the eigenvector coefficients are applied to the control points of a polygon ring/fan. The eigenvector l^0 is used to compute the limit position of a point, whereas l^1 and l^2 are necessary for computing the tangents. The *crease degree* is the number of polygons adjacent to a crease or corner vertex with respect to a specific sector. We use the following notation: k denotes

the crease degree of a crease vertex, the subscript c denotes the coefficient corresponding to the center vertex, we mark edgepoint coefficients with the subscript p, and facepoint coefficients with q. The dominant right eigenvector x^0 is the vector consisting of ones. Unlisted coefficient values are null.

- Sector consisting of quads only. Let $\theta_k = \pi/k$.

$$l^0_c = 2/3,\ l^0_{p0} = l^0_{pk} = 1/6$$

For $k = 1$,

$$\begin{array}{llll} x^1_c = 1/18, & x^1_{p0} = -2/18, & x^1_{p1} = -2/18, & x^1_{q0} = -5/18 \\ x^2_c = 0, & x^2_{p0} = -1/2, & x^2_{p1} = 1/2, & x^2_{q0} = 0 \\ l^1_c = 6, & l^1_{p0} = -3, & l^1_{p1} = -3, & l^1_{q0} = 0 \\ l^2_c = 0, & l^2_{p0} = -1, & l^2_{p1} = 1, & l^2_{q0} = 0 \end{array}$$

otherwise $l^2_c = x^1_c = x^2_c = 0$ and

$$x^1_{pi} = \cos i\theta_k, \quad x^2_{pi} = \sin i\theta_k, \quad i = 0, \cdots, k$$

$$x^1_{qi} = \cos i\theta_k + \cos(i+1)\theta_k, \quad i = 0, \cdots, k-1$$

$$x^2_{qi} = \sin i\theta_k + \sin(i+1)\theta_k, \quad i = 0, \cdots, k-1$$

$$l^2_{p0} = 1/2,\ l^2_{pk} = -1/2$$

$$\alpha = \frac{\cos\theta_k + 1}{k \sin\theta_k (3 + \cos\theta_k)}$$

$$l^1_c = 4\alpha(\cos\theta_k - 1), \quad l^1_{p0} = l^1_{pk} = -\alpha(1 + 2\cos\theta_k)$$

$$l^1_{pi} = \frac{4\sin i\theta_k}{(3 + \cos\theta_k)k}, i = 0, \cdots, k-1$$

$$l^1_{qi} = \frac{(\sin i\theta_k + \sin(i+1)\theta_k)}{(3 + \cos\theta_k)k}, \quad i = 0, \cdots, k-1$$

• Sector beginning and ending with triangles. Let $\theta_k = \pi/k$.

$$l_c^0 = 2/3,\ l_{pk}^0 = l_{q0}^0 = 1/6,\ l_{pk}^1 = 1/2,\ l_{q0}^1 = -1/2$$

$$x_c^1 = x_c^2 = 0$$

$$x_{pi}^1 = \cos i\theta_k - \theta_k/2, x_{pi}^2 = \sin i\theta_k - \theta_k/2, i = 1, \cdots, k-1$$

$$x_{qi}^1 = \cos i\theta_k - \theta_k/2 + \cos(i+1)\theta_k - \theta_k/2, i = 0, \cdots, k-1$$

$$x_{qi}^2 = \sin i\theta_k - \theta_k/2 + \sin(i+1)\theta_k - \theta_k/2, i = 0, \cdots, k-1$$

$$l_c^2 = -4\sin(\theta_k/2)$$

$$l_{p_k}^2 = l_{q_0}^2 = \frac{3(\sin^2(\theta_k/2) - 1)}{\sin(\theta_k/2)}$$

$$l_{p_i}^2 = 4\sin(i\theta_k - \theta_k/2), i = 1, \cdots, k-1$$

$$l_{q_i}^2 = \sin(i\theta_k - \theta_k/2) + \sin((i+1)\theta_k - \theta_k/2), i = 1, \cdots, k-1$$

• Sector beginning with triangle, ending with quad.

Let $\theta_k = \pi/(k-1)$.

$$l_c^0 = 2/3,\ l_{pk}^0 = l_{q0}^0 = 1/6,\ l_{pk}^1 = -1/2,\ l_{q0}^1 = 1/2$$

$$x_c^1 = x_c^2 = 0,\ x_{q0}^1 = 1,\ x_{q0}^2 = 0$$

$$x_{pi}^1 = \cos i\theta_k, \quad x_{pi}^2 = \sin i\theta_k, \quad i = 1, \cdots, k$$

$$x_{qi}^1 = \cos i\theta_k + \cos(i+1)\theta_k, \quad i = 1, \cdots, k-1$$

$$x_{qi}^2 = \sin i\theta_k + \sin(i+1)\theta_k, \quad i = 1, \cdots, k-1$$

$$\alpha = \frac{1}{\cos^2\theta_k + k\cos\theta_k + 3k - 1}$$

$$l_c^2 = -4\sin\theta_k$$

$$l_{p_k}^2 = \alpha\frac{\sin\theta_k(2 + 5\cos\theta_k - \cos^2\theta_k)}{2(\cos\theta_k - 1)}$$

$$l_{q_0}^2 = \alpha\frac{\sin\theta_k\cos\theta_k(5 + \cos\theta_k)}{2(\cos\theta_k - 1)}$$

$$l_{p_i}^2 = 4\alpha\sin i\theta_k,\ l_{q_i}^2 = \alpha\sin i\theta_k + \sin(i+1)\theta_k, i = 1, \cdots, k-1$$

Chapter 4

Analysis of Subdivision Schemes For Surfaces with Boundaries

This chapter presents an analysis of the smoothness properties of the subdivision rules introduced in Chapter 4. This is a preliminary version of a paper which could not be completed due to the author's illness.

4.1 Introduction

Subdivision is a way to construct smooth surfaces out of polygonal meshes used in a variety of computer graphics and geometric modeling applications. Two features of subdivision algorithms are particularly important for applications. The first is the ability to handle a large variety of input meshes, including meshes with boundary. The second is the ease of modification of subdivision rules, which makes it possible to generate different surfaces (e.g. surfaces with sharp or soft creases) from the same input mesh.

The importance of special boundary and crease rules has been recognized

for some time [69, 70, 36, 88]. However, most of the theoretical analysis of subdivision [86, 77, 106, 105] focused on the case of surfaces without boundaries and schemes invariant with respect to rotations. The goal of this paper is to develop the necessary theoretical foundations for analysis of subdivision rules for meshes with boundary, and present analysis for rules for the boundary rules extending several well-known subdivision schemes described in [6].

The starting point for our theory is a precise description of the class of surfaces that we would like to be able to model using subdivision. We introduce the definition of surfaces with piecewise-smooth boundary. This class readily extends to a broader class of piecewise-smooth surfaces, which is sufficiently broad for many practical applications. We demonstrate how the standard constructions of subdivision theory (subdivision matrices, characteristic maps etc.) generalize to the case of surfaces with piecewise-smooth boundary. Remarkably, even at this abstract stage we make a simple, yet important observation, with substantial practical implications: convex and concave corner singularities of the boundary require separate subdivision rules.

We proceed to extend the techniques for the analysis of C^1-continuity developed in [105] to the case of piecewise-smooth surfaces with boundary. While we briefly consider C^k-continuity, we focus on C^1-continuity conditions.

The result allowing one to analyze C^1-continuity of most subdivision schemes for surfaces without boundaries is the sufficient condition of Reif [86]. This condition reduces the analysis of stationary subdivision to the analysis of a single map, called the *characteristic map*, uniquely defined for each valence of vertices in the mesh. The analysis of C^1-continuity is performed in three steps for each valence:

1. compute the control net of the characteristic map;

2. prove that the characteristic map is regular;

3. prove that the characteristic map is injective.

We show that similar conditions hold for surfaces with boundary, and under commonly satisfied assumptions injectivity of the characteristic map for schemes for surfaces with boundary can be inferred from regularity. To avoid the need to evaluate the characteristic map in closed form, we obtain convergence estimates for subdivision schemes acting on regular grids with boundary. These estimates allow us to use sufficiently close linear approximations to draw conclusions about the regularity of the characteristic map. We describe the elements of the theory of schemes acting on regular grids with boundary which we need to perform C^1-continuity analysis.

Subdivision schemes acting on grids with boundary were introduced in [109] where they were referred to as *crease subdivision schemes*. A generalization of this class of schemes was studied in [51] where they are referred to as *quasi-uniform* subdivision schemes.

Finally, we use the theory that we have developed to derive and analyze several specific boundary subdivision rules, initially proposed in [6].

Previous work The theory presented in this paper is based on the theory developed for closed surfaces in [86, 106, 105]. As far as we know, analysis of C^1-continuity of subdivision rules for surfaces with boundary was performed only in [88], where a particular choice of rules extending Loop subdivision was analyzed.

At the same time, a substantial number of papers proposed various boundary rules, starting with the first papers on subdivision by Doo and Sabin, and Catmull and Clark [11, 20, 69, 71, 36]. Most recently, a method for generating soft creases was proposed in [18].

In our C^1-continuity verification method we use estimates of the convergence rate of quasi-uniform subdivision schemes, considered briefly in [109] and in greater generality and detail in [51].

Our estimates of the errors of linear approximations rely on the work of Cavaretta, Dahmen and Micchelli [12], and on the work of Cohen, Dyn and Levin [15] on matrix subdivision.

Finally, we extensively use interval arithmetics (see, for example, [66]).

4.2 Surfaces with Piecewise-smooth Boundary

4.2.1 Definitions

In this section we define surfaces with piecewise-smooth boundaries. Unlike the case of open surfaces, there is no single commonly accepted definition that would be suitable for our purposes. We consider several definitions of surfaces with boundaries and motivate the choice that we make (Definition 4).

The least restrictive definition of a closed surface with boundary is a closed part of an open surface. This definition is too general for our purposes but provides a useful starting point. More formally, we define a closed surface with boundary as follows. Recall that an open C^k-continuous surface in $\mathbf{R}^p$ can be defined as a topological space M with a map $f : M \to \mathbf{R}^p$ such that for any point $x \in M$ there is a neighborhood U_x such that $f(U_x)$ can be reparameterized

over the open unit disk D using a C^k nondegenerate map $g: D \to f(U_x)$.

Definition 1. *Let M be a closed topological space with boundary, and f a map from M to $\mathbf{R}^p$. We say that (M, f) is a closed C^k surface with boundary, if there is an open C^k-continuous surface (M', f') and an injective inclusion map $\iota: M \to M'$, such that $f' \circ \iota = f$.*

Note that this definition places very few restrictions on the boundary: for example, any subset of the plane from this point of view is a C^k-continuous surface for any k. Typically, additional restrictions are added. Most commonly the boundary is required to be a union of nonintersecting C^k-continuous curves (see [67],[22]). Assuming that the domains of these curves are separated in M', this type of surfaces can be defined using two local charts, the open unit disk D and the half-disk $Q_2 = H \cap D$, where H is the closed halfplane defined by $H = \{(x, y) | y \geq 0\}$.

Definition 2. *Consider a surface (M, f) where M is a topological space, and f is a map $f: M \to \mathbf{R}^p$. The surface (M, f) is called a* **closed C^k-continuous surface with C^k boundary** *if for any $x \in M$ there a neighborhood U_x and a regular C^k-continuous parameterization h of $f(U_x)$ over an open disk D (internal point) or a half-disk Q_2 (boundary point).*

This definition is too narrow for geometric modeling, as surfaces with corners (e.g. surfaces obtained by smooth deformations of a rectangle) are quite common. To include corners, we have to allow isolated singularities for the boundary curves. We consider a broader class of surfaces, which we call C^k-continuous surfaces with piecewise C^k-continous boundary.

Definition 3. *let (M, f), $f : M \to \mathbf{R}^p$ be a closed C^k-continuous surface with boundary as defined above, Let $\gamma_i : [0,1] \to M$, $i \in I$, where I is finite, be a set of curve segments, such that each endpoint is shared by exactly 2 segments, and the curve segments intersect only at endpoints. Suppose the boundary of M coincides with $\cap_i \mathrm{Im}\gamma_i$, the curves $f \circ \gamma_i$ are C^k-continuous. Then we call (M, f) a* **C^k-continuous surface with piecewise C^k-continous boundary.**

The definition implies exisitince of the tangents to the boundary curves at the endpoints. However, these tangents may coincide for two adjacent curves, and result in either a *cusp* of degree m or a C^m-continuous joint for $m \leq k$. In either case, k different charts are required to parameterize the surface, as two curves with a contact point of order m are clearly not C^k-diffeomorphic to two curves with a contact point of order $n \neq m$, for $n, m \leq k$. Moreover, the boundary of the surface is not Lipschitz if it contains cusps, which means that surfaces of this type require special treatment when we consider functions defined on such surfaces (cf. [97]).

Transversality assumption. We assume that the adjacent boundary curve segments intersect transversally, that is, their tangents are different at the shared endpoint. We call such endpoints of boundary curve segments **nondegenerate corners.** Thus, the surfaces that we consider do not contain cusps of C^m-continuous joints for $0 < m < k$, We leave analysis of surfaces with higher degree contact points as future work. There are two reasons for this.

First, mathematical description of such surfaces is more complex and best done separately, once the framework for surfaces with boundaries with non-degenerate corners is established.

Second, it appears that this type of features in most cases is best modeled

using degenerate configurations of control points rather than special subdivision rules. In the paper we consider only the behavior of surfaces for generic configuarations of control points.

It is clear, however, that higher-order boundary singularities are useful in applications, a simple example being a surface filling a gap between a cylinder and a tangent plane.

Once we exclude the higher-order contact cases, we can use a more constructive equivalent definition of surfaces with piecewise C^k-continuous boundary with nondegenerate corners. We use four charts for all possible types of points of the surface. In addition to the disk D and the halfdisk Q_2, we use a quarter of the disk Q_1 and three quarters of the disk Q_3. The domains Q_i $i = 1, 3$ are defined as follows: $Q_1 = \{(x, y)|y \geq 0 \text{ and } x \geq 0\} \cap D$, $Q_3 = \{(x, y)|y \geq 0 \text{ or } x \geq 0\} \cap D$.

Now we can give an alternative definition of a C^k-continuous surface with piecewise smooth boundary with nondegenerate corners:

Definition 4. *Consider a surface (M, f) where M is a topological space, and f is a map $f : M \to \mathbf{R}^p$. The surface (M, f) is called* **C^k-continuous with piecewise C^k-continuous boundary with non-degenerate corners** *if for any x there is a neighborhood U_x and a regular C^k-continuous parameterization of $f(U_x)$ over one of the domains Q_i, $i = 1, \ldots 3$, or over the disk D. In the first case, we call the point x a boundary point, in the second case an interior point. We distinguish two main types of boundary points: if U_x is diffeomorphic to Q_2, the boundary point is called smooth; othwise it is called a corner. There are 2 types of corners:*

- *convex corners (U_x is diffeomorphic to Q_1);*
- *concave corners (U_x is diffeomorphic to Q_3);*

For brevity, we will also use the term C^k-continuous surface with piecewise-smooth boundary.

The equivalence of Definitions 4 and 3 with degenerate corners excluded is straightforward to show using the well known facts about existence of the extensions of functions defined on Lipschitz domains to the plane.

Surfaces satisfying Definition 4 can be used to model a large variety of features; for example, by joining the surfaces along boundary lines, we can obtain surfaces with creases. However, in addition to boundary cusps, a number of useful features such as cones cannot be modeled, unless degenerate configurations of control points are used.

4.2.2 Tangent Plane Continuity and C^1-continuity

As we will see in Section 4.3, analysis of subdivision focuses on the behavior of surfaces which are known to be at least C^1-continuous in a neighborhood of a point, but nothing is known about the behavior at the point. In this case, it is convenient to split the analysis into several steps, the first being **tangent plane continuity**. In the definition below, we use $\wedge$ to denote the exterior product (vector product for $p = 3$) and $[\cdot]_+$ to denote normalization of a vector.

Definition 5. *Let D be the unit disk in the plane. Suppose a surface (M, f) in a neighborhood of a point $x \in M$ is parameterized by $h : U \to \mathbf{R}^p$, where U is a subset of the unit disk D containing 0, which is regular everywhere except 0, and $h(0) = f(x)$. Let $\pi(y) = [\partial_1 h \wedge \partial_2 h]_+$. where $\partial_1 h$ and $\partial_2 h$ are derivatives with respect to the coordinates in the plane of the disk D. The surface is* **tangent plane continuous** *at x if the limit $\lim_{y \to 0} \pi(y)$ exists.*

For an interior point x for which the surface is known to be C^1-continuous

in a neighborhood of the point x excluding x, the surface is C^1-continuous at x if and only if it is tangent plane continuous and the projection of the surface into the tangent plane is injective ([106], Proposition 1.2). The proof of this proposition does not assume that the surface is defined on an open neighborhood of x. C^1 continuity for an interior point x is inferred from existence and C^1 continuity of two indepenent derivatives of reparameterization of the surface over the tangent plane. This fact alone is not sufficient to guarantee that the surface has piecewise continuous bounary with nondegenerate corners: we need to impose an additional condition on the boundary curve. We will say that the boundary of (M, f) has a nondegenerate corner at x if there is a neighborhood U_x such that $f(U_x) \cap f(\partial M)$ admits a parameterization by two C^1-continuous curves $\gamma_i : (0, 1] \to \mathbf{R}^p$, $i = 1, 2$, such that $\gamma_1(1) = \gamma_2(1) = f(x)$, and the tangents to the curves are different at the common endpoint x.

Proposition 1. *Suppose a surface (M, f) is C^1-continuous with C^1-continuous boundary in a neighborhood U_x of a boundary point x excluding x. The surface is C^1-continuous at x with piecewise C^1-continuous boundary if and only if it is*

1. *tangent plane continuous,*
2. *the projection of the surface into the tangent plane is injective,*
3. *the boundary either has a nondegenerate corner at x or is C^1-continuous at x.*

Proof. Necessity of these conditions is straightforward. Most of the proof of sufficiency coincides with the proof of Proposition 1.2 from [106]: if we assume only that the surface is tangent plane continuous and the projection into the tangent plane is injective, we can show that the derivatives in two independent

directions of π, the inverse of a projection of the surface into the tangent plane, exist and are continuous at point x.

It remains to be shown that the surface is C^1-diffeomorphic to one of the domains Q_i, $i = 1, 2, 3$.

As the boundary curves γ_i are C^1-continuous, and their tangents are in the tangent plane to the surface at all points, their projections $P\gamma_i$ into the tangent plane at x are also C^1-continuous. At the point x the tangents to the curves are in the tangent plane at x, and coincide with the tangents to the projections. By construction, the domain of π, the image of the projection of the surface into the tangent plane, is homeomorphic to a halfdisk. We have shown that the image of the boundary diameter of the halfdisk is a C^1-continuous or C^1-continuous with a nondegenerate corner at x. The neighborhood U_x can be chosen in such a way that $Pf(\partial U_x \setminus \partial M)$, the image of the part of the boundary of U_x which is not the boundary of M, is a semicircle centered at x and intersects the curves $P\gamma_i$ only at a single point each. (We omit the somewhat tedious but straightforward proof of this fact.)

Thus, our surface can be parameterized in a neighborhood of x over a planar domain $Pf(U_x)$ which is a subset of an open disk $D_{Pf(x)}$ bounded by two C^1 curve segments connecting the center $Pf(x)$ to to the boundary. Let l_1 and l_2 be the rays along tangent directions to γ_1 and γ_2 at x (possibly collinear). Then for sufficiently small radius of the neighborhood, we can assume that orthogonal projections of γ_i to l_i is injective. Note that $l_1 \cap l_2$ split the disk $D_{Pf(x)}$ into two parts D_1 and D_2; either both parts are half-disks, or one part is convex and the other concave. Now we can directly construct a C^1-diffeomorphism of the domain $Pf(U_x)$ to one of the domains D_1 and D_2. For example, in the simplest case of l_1 and l_2 being collinear, we can use a coordinate system (s, t) in which l_1

and l_2 form the s axis, and γ_1 and γ_2 form a graph of a function $\gamma(s)$. Assuming that the disk $D_{Pf(x)}$ has radius 1 the formula

$$(s,t) \to \left(s, \sqrt{1-s^2}\frac{t-\gamma(s)}{\sqrt{(1-s^2)-\gamma(s)}} \right)$$

defines the desired diffeomorphism.

We have shown that the surface has a parameterization g over one of the domains Q_i $i = 1, 2, 3$ in the neighborhood of x, which has C^1-continuous derivatives everywhere on Q_i with nowhere degenerate Jacobian. □

This proposition provides a general strategy for establishing C^1-continuity of surfaces, which is particularly convenient for subdivision surfaces. Moreover, as we shall see, in most cases of practical importance we can infer the injectivity of the projection from the other two conditions, so only local tests need to be performed.

4.3 Subdivision Schemes on Complexes with Boundary

In this section we summarize the main definitions and facts about subdivision on complexes that we use; more details for the case of surfaces without boundaries can be found in [106, 104]. The changes that have to be made to make the constructions work for the boundary case are relatively small. We restrict the presentation to the case of schemes for triangle meshes to avoid making the notation excessively complex. However, the results equally apply to quadrilateral schemes; only minor changes in notation are necessary.

Simplicial complexes. Subdivision surfaces are naturally defined as functions on two-dimensional polygonal complexes. A simplicial complex K is a set of vertices, edges and planar simple polygons (faces) in $\mathbf{R}^N$, such that for any face its edges are in K, and for any edge, its vertices are in K. We assume that there are no isolated vertices or edges. $|K|$ denotes the union of faces of the complex regarded as a subset of $\mathbf{R}^N$ with induced metric. We say that two complexes K_1 and K_2 are *isomorphic* if there is a homeomorphism between $|K_1|$ and $|K_2|$ that maps vertices to vertices, edges to edges and faces to faces.

A *subcomplex* of a complex K is a subset of K that is a complex. A 1-neighborhood $N_1(v, K)$ of a vertex v in a complex K is the subcomplex formed by all faces that have v as a vertex. The m-neighborhood of a vertex v is defined recursively as a union of all 1-neighborhoods of vertices in the $(m-1)$-neighborhood of v. We omit K in the notation for neighborhoods when it is clear what complex we refer to.

Recall that a *link* of a vertex is the set of edges of $N_1(v, K)$ that do not contain v. We consider only complexes with all vertices having links that are connected simple polygonal lines, open or closed. If the link of a vertex is an open polygonal line, this vertex is a boundary vertex, otherwise it is an internal vertex.

In the analysis of schemes for surfaces without boundary the regular complex $\mathcal{R}$ and k-regular complexes $\mathcal{R}_k$ are are commonly used [106]. We are primarily interested in schemes that work on quadrilateral and triangle meshes, and we consider k-regular complexes with all faces being identical triangles or quads; however, similar complexes can be defined for the remaining regular tiling, with all faces being hexagons, and more generally for any Laves tiling. For schemes acting on meshes with boundary we use regular and k-regular complexes with

boundary. A regular complex with boundary is isomorphic to a regular tiling of the upper half-plane.A k-regular complex $\mathcal{R}_k^\alpha$ with apex angle α is isomorphic to the regular tiling of a sector with apex angle α, consisting of identical polygons, with all internal vertices of equal valence and all vertices on the boundary of equal valence, excluding the vertex C at the apex which has valence $k+1$. For triangle meshes, this valence of regular interior vertices is six, and for boundary vertices it is three.

Note that the complex is called k-regular, when the number of *faces* sharing the vertex C is k. In the case of complexes without boundary these numbers are equal, but for complexes with boundary the number of edges sharing C is $k+1$.

Tagged complexes. The vertices, edges or faces of a complex can be assigned tags or, more formally, a map can be defined from the sets of vertices, edges or faces to a finite set of tags. These tags can be used to choose a type of subdivision rules applied at a vertex. In this paper, we use tags in a very limited way: specifically, a boundary vertex can be tagged as a *convex* or *concave* corner, or a smooth boundary vertex. However, as is discussed below, the tags can be used to create creases in the interior of meshes and for other purposes. Subdivision on tagged complexes merits a separate detailed consideration in a future paper.

Isomorphisms of tagged complexes with identical tag sets can be defined as isomorphisms of complexes which preserve tags, i.e. if a vertex has a tag τ its image also has a tag τ.

Subdivision of simplicial complexes. We can construct a new complex $D(K)$ from a complex K by subdivision. For a triangle scheme, $D(K)$ is constructed by adding a new vertex for each edge of the complex and replacing each old triangular face with four new triangles. If some faces of the initial complex are not triangular, they have to be split into triangles first. For a quadrilateral scheme, $D(K)$ is constructed by adding a vertex for each edge and face, and replacing each n-gonal face with n quadrilateral faces. Note that k-regular complexes and k-regular complexes with boundary are self-similar, that is, $D(\mathcal{R}_k)$ and $\mathcal{R}_k$, as well as $D(\mathcal{R}_k^\alpha)$ and $\mathcal{R}_k^\alpha$, are isomorphic.

We use notation K^j for j times subdivided complex $D^j(K)$ and V^j for the set of vertices of K^j. Note that the sets of vertices are nested: $V^0 \subset V^1 \subset \ldots$.

If a complex is tagged, it is also necessary to define rules for assigning tags to the new edges, vertices and faces. For our vertex tags, we use a trivial rule: all newly inserted boundary vertices are tagged as smooth boundary.

Subdivision schemes. Next, we attach values to the vertices of the complex; in other words, we consider the space of functions $V \to B$, where B is a vector space over $\mathbf{R}$. The range B is typically $\mathbf{R}^l$ or $\mathbf{C}^l$ for some l. We denote this space $\mathcal{P}(V, B)$, or $\mathcal{P}(V)$, if the choice of B is not important.

A *subdivision scheme* for any function $p^j(v)$ on vertices V^j of the complex K^j computes a function $p^{j+1}(v)$ on the vertices of the subdivided complex $D(K) = K^1$. More formally, a subdivision scheme is a collection of operators $S[K]$ defined for every complex K, mapping $\mathcal{P}(K)$ to $\mathcal{P}(K^1)$. We consider only subdivision schemes that are linear; that is, the operators $S[K]$ are linear functions on $\mathcal{P}(K)$. In this case the subdivision operators are defined by equations

$$p^1(v) = \sum_{w \in V} a_{vw} p^0(w)$$

for all $v \in V^1$. The coefficients a_{vw} may depend on K.

We restrict our attention to subdivision schemes which are finitely supported, locally invariant with respect to a set of isomorphisms of tagged complexes and affinely invariant.

A subdivision scheme is *finitely supported* if there is an integer M such that $a_{vw} \neq 0$ only if $w \in N_M(v, K)$ for any complex K (note that the neighborhood is taken in the complex K^{j+1}). We call the minimal possible M the *support size* of the scheme.

We assume our schemes to be *locally defined* and *invariant with respect to isomorphisms of tagged complexes*.

Together these two requirements can be defined as follows: there is a constant L such that if for two complexes K_1 and K_2 and two vertices $v_1 \in V_1$ and $v_2 \in V_2$ there is a tag-preserving isomorphism $\rho : N_L(v_1, K_1) \to N_L(v_2, K_2)$, such that $\rho(v_1) = v_2$, then $a_{v_1 w} = a_{v_2 \rho(w)}$. In most cases, the *localization size* $L = M$.

The final requirement that we impose on subdivision schemes is *affine invariance*: if T is a linear transformation $B \to B$, then for any $v \quad Tp^{j+1}(v) = \sum a_{vw} T p^j(v)$. This is equivalent to requiring that all coefficients a_{vw} for a fixed v sum up to 1.

For each vertex $v \in \cup_{j=0}^{\infty} V^j$ there is a sequence of values $p^i(v)$,... where i is the minimal number such that V^i contains v.

Definition 6. *A subdivision scheme is called convergent on a complex K, if for any function $p \in \mathcal{P}(K, B)$ there is a continuous function f defined on $|K|$ with*

values in B, such that

$$\lim_{j\to\infty} \sup_{v\in V^j} \left\| p^j(v) - f(v) \right\|_2 \to 0$$

The function f is called the limit function of subdivision.

Notation: $f[p]$ is the limit function generated by subdivision from the initial values $p \in \mathcal{P}(K)$.

It is easy to show that if a limit function exists, it is unique. A *subdivision surface* is the limit function of subdivision on a complex K with values in $\mathbf{R}^3$. In this case we call the initial values $p^0(v)$ the *control points* of the surface.

Assuming the trivial rule for assigning tags to the newly inserted boundary vertices, we observe that locally any surface generated by a subdivision scheme on an arbitrary complex can be thought of as a part of a subdivision surface defined on a k-regular complex or a k-regular complex with boundary.

Note that this fact alone does not guarantee that it is sufficient to study subdivision schemes only on k-regular complexes and k-regular complexes with boundary [106]. If the number of control points of the initial complex for a k-gonal patch is less than the number of control points of the central k-gonal patch in the k-regular complex, then only a proper subspace of all possible configurations of control points on the subdivided complexes can be realized. Although it is unlikely, it is possible that for such complexes almost all configurations of control points will lead to non-smooth surfaces, while the scheme is smooth on the k-regular complexes.

Subdivision matrices. Consider the part of a subdivision surface $f[y]$ with $y \in U_1^j = |N_1(0, \mathcal{R}_k^j)|$, defined on the domain formed by faces of the subdivided complex $\mathcal{R}_k^j$ adjacent to the central vertex. It is straightforward to show that

the values at all dyadic points in $|N_1(0, \mathcal{R}_k^j)|$ can be computed given the initial values $p^j(v)$ for $v \in N_L(0, \mathcal{R}_k^j)$. In particular, the control points $p^{j+1}(v)$ for $v \in N_L(0, R_k^{j+1})$ can be computed using only control points $p^j(w)$ for $w \in N_L(0, \mathcal{R}_k^j)$. Let $\bar{p}^j$ be the vector of control points $p^j(v)$ for $v \in N_L(0, \mathcal{R}_k^j)$. Let $p+1$ be the number of vertices in $N_L(0, \mathcal{R}_k)$. As the subdivision operators are linear, $\bar{p}^{j+1}$ can be computed from $\bar{p}^j$ using a $(p+1) \times (p+1)$ matrix S^j: $\bar{p}^{j+1} = S^j \bar{p}^j$

If for some m and for all $j > m$, $S^j = S^m = S$, we say that the subdivision scheme is *stationary on the k-regular complex*, or simply stationary, and call S the *subdivision matrix* of the scheme.

Eigenbasis functions. let $\lambda_0 = 1, \lambda_i, \ldots \lambda_J$ be different eigenvalues of the subdivision matrix in nonincreasing order, the condition $\lambda_0 > \lambda_1$ is necessary for convergence.

For any λ_i let J_j^i, $j = 1 \ldots$ be the complex cyclic subspaces corresponding to this eigenvalue.

Let n_j^i be the *orders* of these cyclic subspaces; the order of a cyclic subspace is equal to its dimension minus one.

Let b_{jr}^i, $r = 0 \ldots n_j^i$ be the complex generalized eigenvectors corresponding to the cyclic subspace J_j^i. The vectors b_{jr}^i satisfy

$$Sb_{jr}^i = \lambda_i b_{jr}^i + b_{j\,r-1}^i \quad \text{if } r > 0, \quad Sb_{j0}^i = \lambda_i b_{j0}^i \tag{4.1}$$

The complex *eigenbasis functions* are the limit functions defined by $f_{jr}^i = f[b_{jr}^i] : U_1 \to \mathbf{C}$

Any subdivision surface $f[p] : U_1 \to \mathbf{R}^3$ can be represented as

$$f[p](y) = \sum_{i,j,r} \beta^i_{jr} f^i_{jr}(y) \tag{4.2}$$

where $\beta^i_{jr} \in \mathbf{C}^3$, and if $b^i_{jr} = \overline{b^k_{lt}}$, $\beta^i_{jr} = \overline{\beta^k_{lt}}$, where the bar denotes complex conjugation.

One can show using the definition of limit functions of subdivision and (4.3) that the eigenbasis functions satisfy the following set of *scaling relations*:

$$f^i_{jr}(y/2) = \lambda_i f^i_{jr}(y) + f^i_{j\,r-1}(y) \quad \text{if } r > 0, \quad f^i_{j0}(y/2) = \lambda_i f^i_{j0}(y) \tag{4.3}$$

Real eigenbasis functions. As we consider real surfaces, it is often convenient to use real Jordan normal form of the matrix rather than the complex Jordan normal form. For any pair of the complex-conjugate eigenvalues λ_i, λ_k, we can choose the complex cyclic subspaces in such a way that they can be arranged into pairs J^i_j, J^k_j, and $b^i_{jr} = \overline{b^k_{jr}}$ for all j and r. Then we can introduce a single real subspace for each pair, with the basis c^i_{jr}, c^k_{jr}, $r = 0 \dots n^i_j$, where $c^i_{jr} = \Re b^i_{jr}$, and $c^k_{jr} = \Im b^i_{jr}$. We call such subspaces *Jordan subspaces.* Then we can introduce real eigenbasis functions $g^i_{jr}(y) = f^i_{jr}(y)$ for real λ_i, and $g^i_{jr}(y) = \Re f^i_{jr}(y)$, $g^k_{jr}(y) = \Im f^i_{jr}(y)$ for a pair of complex-conjugate eigenvalues (λ_i, λ_k). For a Jordan subspace corresponding to pairs of complex eigenvalues the order is the same as the order of one of the pair of cyclic subspaces corresponding to it.

Similar to (4.2) we can write for any surface generated by subdivision on U_1:

$$f[p](y) = \sum_{i,j,r} \alpha^i_{jr} g^i_{jr}(y) \tag{4.4}$$

Now all coefficients α^i_{jr} are real. Eigenbasis functions corresponding to the eigenvalue 0 have no effect on tangent plane continuity or C^k-continuity of the surface at zero. From now on we assume that $\lambda_i \neq 0$ for all i.

We can assume that the coordinate system in $\mathbf{R}^3$ is always chosen in such a way that the single component of $f[p]$ corresponding to eigenvalue 1 is zero. This allows us to reduce the number of terms in (4.4) to p.

4.3.1 Reduction to universal surfaces

In [106] we have shown that for surfaces without boundary the analysis of smoothness of subdivision can be reduced to analysis of *universal surfaces*. Moreover, if a subdivision scheme is C^1, almost any surface produced by subdivision is diffeomorphic to the universal surface. In this section, we introduce the universal surfaces for neighborhoods of boundary vertices, and show that a similar reduction can be performed in this case.

This fact is of considerable practical importance for design of subdivision schemes for surfaces with piecewise-smooth boundary: as we have observed in Section 4.2, convex and concave corners are not diffeomorphic; therefore, a convex and a concave corner in $\mathbf{R}^3$ cannot be diffeomorphic to the same universal surface, and cannot be generated by the same subdivision rule.

Universal map. The decomposition (4.4) can be written in vector form. Let h^i_{jr} be an orthonormal basis of $\mathbf{R}^p$. Let ψ be $\sum_{i,j,r} g^i_{jr} h^i_{jr}$; this is a map $U_1 \to \mathbf{R}^p$. Let $\alpha^1, \alpha^2, \alpha^3 \in \mathbf{R}^p$ be the vectors composed of components of coefficients α^i_{jr} from (4.4) (each of these coefficients is a vector in $\mathbf{R}^3$). Then (4.4) can be rewritten as

$$f[p](y) = \big((\psi, \alpha^1),\, (\psi, \alpha^2),\, (\psi, \alpha^3)\big) \tag{4.5}$$

This equation indicates that all surfaces generated by a subdivision scheme on U_1 can be viewed as projections of a single surface in $\mathbf{R}^p$. We call ψ the *universal map,* and the surface specified by ψ the *universal surface.* In [106], it was demonstrated that the analysis of tangent plane continuity and C^k continuity of subdivision can be reduced to analysis of the universal surface. Not surprisingly, we will see that this also holds for subdivision schemes with boundary.

In the chosen basis the matrix S is in the real Jordan normal form. Note that by definition of S for any $a \in \mathbf{R}^p$

$$(a, \psi(y/2)) = (Sa, \psi(y))$$

Using the well-known formula for inner products $(Su, v) = (u, S^T v)$, we get

$$(x, \psi(y/2)) = (x, S^T \psi(y)), \quad \text{for any } x$$

This means that the scaling relations can be jointly written as

$$\psi(y/2) = S^T \psi(y) \tag{4.6}$$

The universal map ψ is only piecewise C^k, even if we assume that subdivision produces C^k limit function on regular complexes and regular complexes with boundary: derivatives have discontinuity at the boundaries of polygons of U_1. However, one can easily construct a map κ (see [106]) such that $\varphi = \psi \circ \kappa^{-1}$ is C^1-continuous away from the center.

We will impose the following condition on the subdivision schemes that we consider: **Condition A.** For any $y \in U_1$

$$\partial_1 \psi(y) \wedge \partial_2 \psi(y) \neq 0 \quad \text{for all } y \in U_1,\ y \neq 0$$

This condition holds for all known practical schemes.

Reduction theorem. Our goal is to relate tangent plane continuity and C^k-continuity of the universal surface in $\mathbf{R}^p$ and tangent plane continuity of the subdivision scheme. The following theorem holds under our assumptions:

Theorem 2. *For a subdivision scheme satisfying Condition A to be tangent plane continuous on a k-regular complex with boundary, it is necessary and sufficient that the universal surface be tangent plane continuous; for the subdivision scheme to be C^k-continuous with p.w. C^k-continuous boundary, it is necessary and sufficient that the universal surface is C^k-continuous with p.w. C^k-continuous boundary. Almost all surfaces generated by a subdivision scheme on a k-regular complex with boundary are locally diffeomorphic to the universal surface.*

Proof. Sufficiency is clear as any surface is a linear projection of the universal surface. To prove necessity, we use Proposition 1, and show that

- if the universal surface is not tangent plane continuous then a set of subdivision surfaces of non-zero measure is not tangent plane continuous;
- if the universal surface has non-injective projection into the tangent plane same is true for a set of subdivision surfaces of non-zero measure;
- if the projection of the universal surface into the tangent plane is not C^k, same is true for a set of subdivision surfaces of non-zero measure;
- if the boundary of the universal surface is not C^k-continuous, or is not C^k-continuous with nondegenerate corner, same is true for a set of subdivision surfaces of non-zero measure.

The proof of the first three statements coincides with the proof for the surface without boundary presented in [106].

We only need to consider the fourth statement. By assumption, the boundary of the surface is C^1-continuous away from zero. Let the two pieces of the boundary be $\gamma_i : (0,1] \to \mathbf{R}^p$, $i = 1,2$, with $\gamma_1(1) = \gamma_2(1)$. We can assume both pieces to be C^1-continuous away from one. Suppose γ_1 does not have a tangent at 1; then there are at least two directions τ_1 and τ_2 which are limits of sequences of tangent directions to $\gamma_1(t)$ as t approaches 1. There is a set of three-dimensional subspaces π of measure non-zero in the space of all three-dimensional subspaces, for which the projections of both vectors τ_1 and τ_2 to the subspace are not zero. If we project the universal surface to any of these subspaces, the boundary curve of the resulting surface will not be tangent continuous. For curves tangent continuity is equivalent to C^1-continuity. For C^k-continuity the proof for curves is identical to the proof for surfaces. We conclude that the curves γ_1 and γ_2 should be C^k-continuous. Similarly, if the curves are joined with continuity less than k, then almost all curves obtained by projection into $\mathbf{R}^3$ will have the same property. Finally, if the tangents to the curves coincide, same is true for almost all projections of the curves, which means that almost all projections do not have a non-degenerate corner. □

The following important corollary immediately follows from Theorem 2:

Corollary 3. *Almost all surfaces generated by a given C^k-continuous subdivision scheme on a k-regular complex are diffeomorphic.*

Indeed, as any subdivision surface $f : U_k \to \mathbf{R}^3$ is obtained as a projection of the universal surface, for almost any choice of projection it defines a diffeomorphism of the universal surface and f.

This corollary implies in particular that the same subdivision rule cannot generate convex and concave corners simultaneously in a stable way, and separate rules are required for these cases. It should be noted that surfaces with convex and concave corners can be alternatively produced using standard rules and degenerate configurations of control points. We believe, however, that the best approach is to use special rules and not require special constraints on control points. However, it appears to be more natural to use degenerate configurations for producing surfaces with 0 and 2π corners. Analysis of the behavior of subdivision on degenerate and constrained configurations of control points is not considered in this paper and remains an open problem.

4.4 Criteria for tangent plane and C^1 continuity.

Tangent plane continuity criteria of [106] do not use the fact that only interior points of a surface are considered. Similarly, C^1-continuity criteria use only the fact that C^1-continuity is equivalent to tangent plane continuity and injectivity of the projection into the tangent plane. Therefore, C^1-continuity criteria also hold for boundary points. We only need to establish the conditions that guarantee that the boundary curves are C^1-continuous, possibly with corners.

We focus on a sufficient condition for C^1-continuity ([106] Theorem 3.6 and Theorem 4.1), which is most relevant for applications. More general necessary and sufficient conditions (e.g. [106] Theorem 3.5) can be extended in a similar way.

To state the sufficient condition, we need to define *characteristic maps*, which

are commonly used to analyze C^1 continuity of subdivision surfaces. We use a definition somewhat different from the original definition of Reif [86].

Characteristic maps.

Definition 7. *The* **characteristic map** $\Phi : U_1 \to \mathbf{R}^2$ *is defined for a pair of cyclic subspaces* J_b^a, J_d^c *of the subdivision matrix as*

1. (f_{a0}, f_{a1}) *if* $J_b^a = J_d^c$, λ_a *is real,*
2. (f_{a0}, f_{c0}) *if* $J_b^a \neq J_d^c$, λ_a, λ_c *are real,*
3. $(\Re f_{a0}, \Im f_{a0})$ *if* $\lambda_a = \bar{\lambda_c}$, $b = d$.

The domain of a characteristic map is the neighborhood U_1, consisting of k faces of the regular complex; we call these faces *segments*. We assume that the subdivision scheme generates C^1-continuous limit functions the regular complexes, and the characteristic map is C^1-continuous inside each segment and has continuous one-sided derivatives on the boundary.

Characteristic map satisfies the scaling relation $\Phi(t/2) = T\Phi(t)$, where T is one of the matrices

$$T_{\text{scale}} = \begin{pmatrix} \lambda_a & 0 \\ 0 & \lambda_c \end{pmatrix}, \quad T_{\text{skew}} = \begin{pmatrix} \lambda_a & 1 \\ 0 & \lambda_a \end{pmatrix},$$

$$T_{\text{rot}} = |\lambda_a| \begin{pmatrix} cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{pmatrix},$$

where φ is the argument of a complex λ_a.

Sufficient condition for C^1-continuity. The following sufficient condition is a special case of the condition that was proved in [106]. Although all our constructions apply in the more general case, we state the only a simplified version of the criterion to simplify the presentation. This form captures the main idea of the sufficient condition. This condition generalizes Reif's condition [86].

Define for any two cyclic subspaces $\operatorname{ord}\left(J_j^i, J_l^k\right)$ to be $n_j^i + n_l^k$, if $J_j^i \neq J_l^k$; let $\operatorname{ord}\left(J_j^i, J_j^i\right) = 2n_j^i - 2$; note that for $n_j^i = 0$, this is a negative number, and it is less than ord for any other pair. This number allows us to determine which components of the limit surface contribute to the limit normal (see [106, 104] for details). We say that a pair of cyclic subspaces J_b^a, J_d^c is *dominant* if for any other pair J_j^i, J_l^k we have either $|\lambda_a\lambda_c| > |\lambda_i\lambda_k|$, or $|\lambda_a\lambda_c| = |\lambda_i\lambda_k|$ and $\operatorname{ord}\left(J_b^a, J_d^c\right) > \operatorname{ord}\left(J_j^i, J_l^k\right)$. Note that the blocks of the dominant pair may coincide.

Theorem 4. *Let b_{jr}^i be a basis in which a subdivision matrix S has Jordan normal form. Suppose that there is a dominant pair J_b^a, J_d^c. If $\lambda_a\lambda_c$ positive real, and the Jacobian of the characteristic map of J_b^a, J_d^c has constant sign everywhere on U_1 except zero, then the subdivision scheme is tangent plane continuous on the k-regular complex.*

If the characteristic map is injective, the scheme is C^1-continuous.

In the special case when all Jordan blocks are trivial, this condition reduces to an analog of the Reif's condition.

Criterion for piecewise C^1-continuity of the boundary. Assuming that the scheme at a boundary vertex satisfies the conditions of Theorem 4, we establish additional conditions which guarantee that the scheme for almost all

control meshes generates C^1-continuous surfaces with piecewise C^1-continuous boundary with nondegenerate corners.

Define I_1, I_2 to be two segments of the boundary of the domain of a boundary characteristic map which with zero as an endpoint.

Theorem 5. *Suppose a subdivision scheme satisfies conditions of Theorem 4 for boundary vertices of valence k. Then the scheme is p.w. C^1-continuous with nondegenerate corners for boundary vertices of valence k if and only if the following conditions are satisfied.*

1. *λ_a and λ_c are positive real.*

2. *Suppose $\lambda_a > \lambda_c$, or $a = c$ and order of J_b^a is greater than order of J_d^c (diagonal scaling matrix, asymmetric scaling). Then the scheme is boundary C^1-continuous if and only if $\partial_1 f_1 \neq 0$ and has the same sign on I_1 and I_2 or $\partial_1 f_1 \equiv 0$ on I_1 and I_2.*

   The scheme is a nondegenerate corner scheme, if and only if $\partial_1 f_1 \neq 0$ on I_1 and $\partial_1 f_1 \equiv 0$ on I_2. Same is true if I_1 I_2 are exchanged.

3. *Characteristic map of Suppose $J_c^a = J_d^b$ (scaling matrix is a Jordan block of size 2), and ∂f_1 does not vanish on I_1 and I_2. Then the scheme is boundary C^1-continuous. Nondegenerate corners cannot be generated by a scheme of this type.*

4. *Suppose $a = c$ and order of J_b^a is equal to the order of J_d^c (diagonal scaling matrix, symmetric scaling). The boundary id C^1-continuous if and only if there is a nontrivial linear combination $\alpha_1 \partial_1 f_1 + \alpha_2 \partial_2 f_2$ identically vanishing on I_1 and I_2, and any other independent linear combination has the*

same sign on I_1 and I_2. The scheme is a corner scheme if and only if there is a linear combination $\alpha_1\partial_1 f_1 + \alpha_2\partial_2 f_2$ identically vanishing on I_1 and a different linear combination $\beta_1\partial_1 f_1 + \beta_2\partial_2 f_2$ identically vanishing on I_2 with $[\alpha_1, \alpha_2]$ and $[\beta_1, \beta_2]$ linearly independent.

Proof. For each of the boundary segments defined on I_1 and I_2 we need to show that the limit of the tangent exists at the common endpoint. If these limits coincide then the boundary curve of the universal surface is C^1-continuous; if the limits have different directions, then the universal surface has a nondegenerate corner.

First, we observe that by assumption the characteristic map has non-zero Jacobian on the boundary. This means that one of the components has nonzero derivative along the boundary $\partial_1 f_1(t) \neq 0$ or $\partial_1 f_2(t) \neq 0$ at any point $t \in I_1 \cup I_2$. Consider the tangent to the boundary of the surface defined by the characteristic map. It is a two-dimensional vector $v(t) = (\partial_1 f_1(t), \partial_1 f_2(t))$, where t is a point of I_1 or I_2.The tangent satisfies the scaling relation of the form $v(t/2) = 2Tv(t)$, where T is the scaling matrix for the characteristic map. The direction of the tangent has a limit if and only if T is is either T_{scale} or T_{skew} and its eigenvalues are positive (Lemma 3.1, [106]). As the projection of the universal surface is arbitrarily well approximated by the characteristic map, or coincides with it for simple Jordan structure of the subdivision matrix, we conclude that for the universal surface boundary to have well-defined tangents at zero, the eigenvalues of the characteristic map have to be positive and real. However, this condition is not sufficient for existence of tangents.

Diagonal scaling matrix, asymmetric case. In the first case that we consider the dominant cyclic subspace pair J_b^a, J_d^c $a \neq c$ (different eigenvalues) or

$a = c$, $b \neq d$, $n_b^a \neq n_d^c$ (same eigenvalue, subspaces of different sizes); These two cases are unified by the fact that the sequences $\partial_1 f_1(t_1/2^m)$ and $\partial_1 f_2(t_2/2^m)$, for $\partial_1 f_1(t_1), \partial_1 f_2(t_2) \neq 0$, change at a different rate. This can be easily seen from the scaling relation. Moreover, the ratio $\|\partial_1 f_2(t_1/2^m)\|/\|\partial_1 f_1(t_2/2^m)\|$ approaches zero as $m \to \infty$.

Suppose at some points t_1, t_2 of I $\partial_1 f_1(t_1) \neq 0$ and $\partial_1 f_1(t_2) = 0$. Then $\partial_2 f_2(t_2) \neq 0$ and the tangents at points $t/2^m$ all point in the direction $\pm e_2$, where e_2 is the unit vector along the coordinate axis corresponding to f_2. $\|\partial_1 f_2(t_1/2^m)\|/\|\partial_1 f_1(t_1/2^m)\| \to 0$ as $m \to \infty$, thus, at points $t_1/2^m$ the direction of the tangent approaches $\pm e_1$. We conclude that there is no limit, unless $\partial_1 f_1$ is either nowhere or everywhere nonzero I_1. Same applies to I_2. Conversely, if $\partial_1 f_1$ is nowhere zero, then the limit tangent direction at the center is $\pm e_1$. If it is zero everywhere, then by assumption about the characteristic map, $\partial_1 f_2$ is nowhere zero, and the limit tangent direction is $\pm e_2$. The choice of sign in each case depends on the sign of ∂f_1 or ∂f_2.

If f_1 is not zero and has the same sign on both I_1 and I_2 then the tangent is continuous, and the boundary curve is C^1-continuous. If it is zero on I_1 and nonzero on I_2, then the tangents are not parallel, and the surface defined by the characteristic map has a corner; this proves the second part of the theorem.

Scaling matrix is a Jordan block of size 2. The second condition of the theorem applies if the characteristic map components are correspond to a cyclic subspace of size 2, i.e. satisfy $f_1(t/2) = \lambda_a f_1(t) + f_2(t)$. Thus, $\partial_1 f_1 \equiv 0$ implies $\partial_1 f_2 \equiv 0$ on I_1 or I_2. If $\partial_1 f_1(t_1) = 0$ at some point t_1 on I_1 or I_2, then the tangent at $t_1/2^m$ will converge to $\pm e_2$ as $m \to \infty$, where e_2 is the unit coordinate vector for the second coordinate. However, ∂f_1 has to be nonzero at some other point

t_2, and the tangent converges to $\pm e_1$ at $t_2/2^m$. Therefore, if the limit tangent exists and is equal to $\pm e_1$ if and only if $\partial_1 f_1$ is nowhere zero. Two cases are possible now: either the limit tangents coincide on I_1 and I_2 or have opposite directions. In the latter case we have a degenerate corner, in the former a C^1-continuous boundary.

Diagonal scaling matrix, symmetric case. In the symmetric case, $a = b$, $c \neq d$ and $n_b^a = n_d^c$, the components of the characteristic map correspond to the subspaces of equal order. This means that the sequences defined above change at the same rate, and any linear combination $\alpha_1 f_1 + \alpha_2 f_2$ is also an eigenbasis function. Suppose f_1 and f_2 come from different cyclic subspaces of the same eigenvalue which have the same size. Suppose $\alpha_1 \partial_1 f_1 + \alpha_2 \partial_1 f_2$ does not vanish identically on I_1 for any nontrivial choice of α_1 and α_2. Pick two linearly independent combinations $g_1 = \alpha_1 \partial_1 f_1 + \partial_1 \alpha_2 f_2$ and $g_2 = \beta_1 \partial_1 f_1 + \beta_2 \partial_1 f_2$ which do not vanish at points t_1 and t_2 of I_1 respectively. Then the vectors $\partial_1 \Phi(t_i) = [\partial_1 f_1(t_i), \partial_1 f_2(t_i)]$ are linearly independent and the sequences $\partial_1 \Phi(t_1/2^m)$ and $\partial_1 \Phi(t_2/2^m)$ converge to different limit directions. Therefore, for the limit tangents at zero to exist, there should be a nontrivial linear combination of $\partial_1 f_1$ and $\partial_1 f_2$ which vanishes on I_1. If $\alpha_1 \partial_1 f_1 \alpha_2 \partial_1 f_2$ is such combination, it is easy to see that the limit tangent direction is, up to the sign,the direction of the vector $[-\alpha_2, \alpha_1]$. For the boundary to be C^1-continuous, the direction should be the same on two sides. Finally, the tangents on two sides exist and do not coincide if the vectors (α_1, α_2) for I_1 and I_2 are linearly independent. □

An interesting corollary of this theorem is that in the symmetric case it is necessary for p.w. C^1-continuity of the boundary that the images of $\Phi(I_1)$ and $\Phi(I_2)$ are straight line segments. Note that this is not necessary if the

eigenvalues λ_a and λ_b are different.

4.4.1 Analysis of Characteristic Maps

To verify conditions of Theorem 4 we need to establish that the characteristic map is regular and injective, and verify that it has the expected behavior on the boundary. Typically, analysis of the boundary behavior is relatively easy, as in most cases the boundary curve is independent from the interior. In this section we focus on regularity and injectivity of the characteristic map.

Regularity of the characteristic map. Just as in the case of interior points we we use self-similarity of the characteristic map to verify the regularity condition of Theorem 4: for any $t \in U_1$, the Jacobian $J[\Phi](t/2) = 4\lambda_a\lambda_b[\Phi](t)$. It is immediately clear that to prove regularity of the characteristic map it is sufficient to consider the Jacobian on a single annular portion of U_1 as shown in Figure 4.1. As all vertices of such a ring are either regular or boundary regular, we can estimate the Jacobian of the characteristic map using tools developed for analysis of subdivision on regular grids. However, there is a significant difference from the case of interior vertices: to esatblish regularity on a single ring, in general, we have to consider subdivision schemes not just on regular meshes but on regular meshes with boundary, which makes the estimates for the Jacobians somewhat more complex.

Injectivity of the characteristic map. Even if the Jacobian of a map is nonzero everywhere, only local injectivity is guaranteed. However, for interior vertices, self-similarity of the characteristic maps allows one to reduce the injectivity test to computing the index of a closed curve around zero [106]. This is

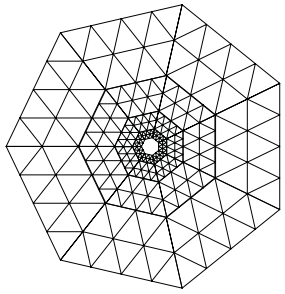


Figure 4.1: The k-gon without origin $U_1 \setminus \{0\}$ can be decomposed into similar rings, each two times smaller than the previous ring. The size of the ring is chosen in such a way that the control set of any ring does not contain the extraordinary vertex. In this figure the control set is assumed to consist out of the vertices of the triangles of the ring itself, and of a single layer of vertices outside the ring.

a relatively simple and fast operation: for example, the index can be computed counting the number of intersections of the curve with a line. This test cannot be applied for boundary points, as there are no closed curves around zero.

For boundary points, a different simple test (Theorem 6) suffices, which in all cases that we have considered is even easier to apply. However, unlike the curve index test, it does not immediately yield a general computational algorithm.

The characteristic map can be extended using scaling relations to a complete k-regular complex with boundary. In the following theorem we assume that the characteristic map is defined on the whole complex $|\mathcal{R}_k^\alpha|$.

Theorem 6. *Suppose a characteristic map $\Phi = (f_a, f_c)$ satisfies the following conditions:*

1. *the preimage $\Phi^{-1}(0)$ contains only one element, 0;*

2. *the characteristic map has a Jacobian of constant sign at all points where it is defined.*

3. *The boundary of the image of the characteristic map does not intersect itself;*

4. *the image of the characteristic map is not the whole plane.*

Then this characteristic map is injective.

Proof. As in [105] we can show that the the characteristic map is continuous at infinity, and if the P is the stereographic projection of the sphere to the plane, $\tilde{\Phi} = P^{-1}\Phi P$ is a continuous mapping of a subset $D = P^{-1}(|\mathcal{R}_k^\alpha|)$ of the sphere into the sphere, with poles mapped to poles; $\tilde{\Phi}$ is a local homeomorphism away from poles.

We observe that the points of the boundary of the image $\tilde{\Phi}(D)$ can be images only of the boundary of D. Suppose the boundary of the image is not empty; we show that the image of the boundary curve $\tilde{\Phi}(\partial D)$ coincides with the boundary of the image $\partial\left(\tilde{\Phi}(D)\right)$. Let y_1 be a point of the boundary of the image. We already know that $\partial\left(\tilde{\Phi}(D)\right) \subset \tilde{\Phi}(\partial D)$. The image of the boundary has no self intersections. It is easy to see that the boundary of the domain is a simple closed Jordan curve, and so is its image $\tilde{\Phi}(\partial D)$. Suppose $\partial\left(\tilde{\Phi}(D)\right) \neq \tilde{\Phi}(\partial D)$. Then there is a point y on the image of the bounadry $\tilde{\Phi}(\partial D)$ which is an interior point of $\tilde{\Phi}(D)$. As $\tilde{\Phi}(\partial D)$ separates the sphere into two linearly connected domains, we can connect each point in either domain to point y with a continuous curve which does not intersect $\tilde{\Phi}(D)$. Thus, any two points on the sphere can be connected by a continuous curve which does not intersect $\partial\tilde{\Phi}(D)$. We conclude that the image $\tilde{\Phi}(D)$ is the whole sphere. Therefore, either $\partial\left(\tilde{\Phi}(D)\right) = \tilde{\Phi}(\partial D)$,

or the image is the whole sphere. The latter option contradicts the last condition of the theorem. In the former case, If we exclude the poles of the sphere, the mapping is a local homeomorphism of one simply connected domain to another. We can easily prove it is a covering: consider an interior point y of the domain, and the set $\tilde{\Phi}^{-1}(y)$. Suppose it is infinite. Then it has a limit point, which cannot be an interior point of D (otherwise, $\tilde{\Phi}$ is not a local homeomorphism at that point). Similarly, it cannot be a boundary point, unless it is one of the poles. It cannot be the south pole x_s for which $P(x_s) = 0$, because then $\Phi(\tilde{x}_s)$ has to be y which contradicts the assuption $\Phi(0) = 0$. Similar fact holds for the opposite pole. We conclude that $\tilde{\Phi}^{-1}(y)$ is finite for each point. Similar is true for boundary points away from the poles. $\tilde{\Phi}$ is a local homeomorphism and maps the boundary exactly to the boundary. Let y be a point of the image away from poles, and let $x_1, x_2, \ldots x_n$ be points of $\tilde{\Phi}^{-1}(y)$. Then for each x_i there is a sufficiently small neighborhood U_i which maps homeomorphically to a neighborhood of $x_i in \tilde{\Phi}(D)$. Then the inverse image of $\cap_i \tilde{\Phi}(U_i)$ is a finite union of disjoint diffeomorphic subsets of D. We conclude that $\tilde{\Phi}$ is a covering on D with poles excluded. However, we have observed that the image of D is simply connected. Therefore, the covering has to be injective. We conclude that the characteristic map is injective. □

4.5 Smoothness criteria

In [106] we have derived general necessary and suficient conditions for tangent plane continuity and C^k-continuity of subdivision surfaces. The conditions for tangent plane continuity require practically no modification: we have to exend the definition of the universal surfaces, characteristic and parameteric maps in a

straighforward manner to handle the boundary case. e do not need to make any distinction between the 3 possible cases of boundary vertices (smooth, concave and convex corners).

Criteria for C^k continuity for boundary vertices are more or less straightforward extension of the criteria of [106]. The criteria are based on the following propositions.

Proposition 7. *A function $f : X \to \mathbf{R}^p$ where $X = Q_1$, Q_2 is C^k-continuous and regular if and only if there is an*

Theorem 8. *For a subdivision surface to be C^1-continuous at a vertex v on the boundary it is sufficient that the characteristic map is regular and injective and the image of the of the closed boundary under the characteristic map consists of two C^1 segements, joined at the point 0 which do not form a cusp. If the tangents two the two segments coincide, the vertex is a smooth boundary vertex. Otherwise, it is a convex or concave corner, dependingon the angle spanned by the image of the half k-gon under the chaacteristic map.*

4.6 Verification of C^1-continuity

To apply the criteria we have established for C^1-continuity on the boundary to specific schemes, we need to analyze the eigenstructure of the subdivision matrices and then verify the assumptions of the theorems for suitable characteristic maps. The first step is the only one that needs to be performed for each scheme individually. The second step is identical to the process described in [105] and is identical for all schemes, so we do not describe the details of the computation here. For the two schemes that we consider additional conditions on the bound-

ary hold trivially as the boundary rules coincide with B-spline rules. Thus we only focus on the analysis of the structure of the subdivision matrices which is in this case is substantially more complex than the analysis for interior rules.

4.6.1 Loop scheme

In this section we describe the structure of the boundary subdivision matrices for the Loop scheme. Some parts of our analysis are similar to the analysis performed by Jean Schweitzer [88].

The control mesh for a boundary patch surrounding an extraordinary vertex is shown in Figure 4.2. There are 3 different types of vertices in the control mesh, shown in the same figure. A different subdivision mask is used for each type. There are two masks for the vertices of types 1 and 2, one for boundary vertices and one for interior vertices. We consider these vertices to have the same type for notational convenience.

The figure also shows the masks of the rules that we consider. Our family of schemes includes all schemes satisfying the following conditions:

1. The support for each mask is the same as for the the Loop scheme or for the cubic B-spline on the boundary;

2. The only masks that are modified are the masks for odd vertices adjacent to the central vertex, and for the central vertex itself (types 0,1).

3. The masks for interior edge vertices of type 1 are all identical and symmetric with respect to the edge connecting the vertex with the central vertex. The masks for two boundary vertices of type 1 are also identical.

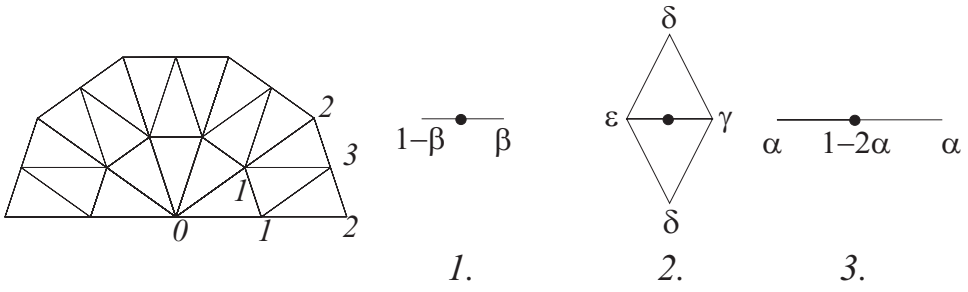


Figure 4.2: Control mesh for a boundary patch of a Loop subdivision surface and masks of the subdivision rules. 1. The rule for the odd vertices on the boundary adjacent to the central vertex (type 1). 2. The rule for the interior odd vertices adjacent to the central vertex (type 1). 3. The rule for the central vertex (type 0). The rules for vertices of type 2 (interior) and 3 are the standard Loop rules; the rule for the vertex of type 2 (boundary)is the standard one-dimensional cubic spline rule.

We assume that all coefficients in the masks are positive. This choice is sufficiently general to construct a variety of schemes; on the other hand, complete eigenanalysis can be performed for all schemes from this family. We show that no scheme from this family can produce a rule for a convex corner. There are reasons to believe that this is true for any scheme with positive coefficients or small support.

For the specific schemes that we consider the boundaries do not depend on the control points in the interior. Potentially, the boundary can depend on the valence of a boundary vertex, this is the case with the scheme presented in [36]. However, we believe that this is best avoided, and present a set of schemes for which the boundary rules are simply cubic spline rules, except at vertices marked as corners, where the interpolation is forced.

Subdivision matrix. We assume that $k > 1$; we will consider the case $k = 1$ separately. The subdivision matrix for a boundary vertex with k adjacent triangles has the form shown in Figure 4.3

In block form this matrix can be written as

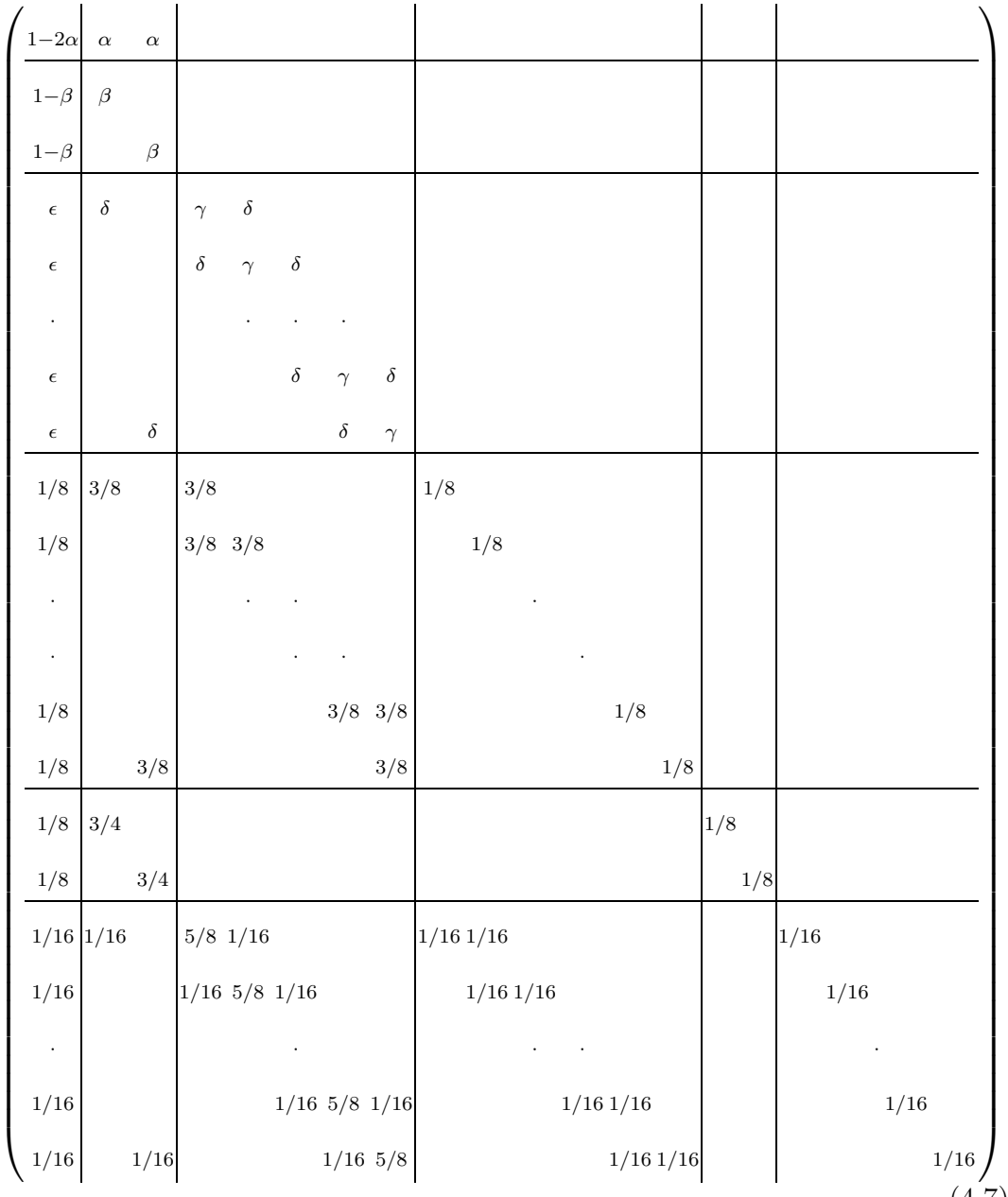
$$\left(\begin{array}{c|cc|c|c|c|c} 1-2\alpha & \alpha & \alpha & & & & \\ \hline 1-\beta & \beta & & & & & \\ 1-\beta & & \beta & & & & \\ \hline a_1 & A_{10} & & A_{11} & & & \\ \hline a_2 & A_{20} & & A_{21} & \frac{1}{8}I_k & & \\ \hline 1/8 & 3/4 & & & & 1/8 & \\ 1/8 & & 3/4 & & & 1/8 & \\ \hline a_3 & & & A_{31} & A_{32} & & \frac{1}{16}I_{k-1} \end{array}\right) \tag{4.8}$$

The vectors a_1 and a_3 have length $k-1$, the vector a_2 has length k, I_k and I_{k-1} are unit matrices of sizes k and $k-1$. Note that the eigenvalues of the matrix are 1/8 1/16, the eigenvalues of the upper-left 3×3 block A_{00} and the eigenvalues of the matrix A_{11}. The matrix A_{11} is tridiagonal, of size $k-1 \times k-1$. The eigenvalues of a_{00} are A_{11}, and two more eigenvectors that correspond to eigenvalues β and $\beta - 2\alpha$, with the first entry equal to zero.

Following [88], we observe that $k-1 \times k-1$ tridiagonal symmetric matrices have the following eigenvectors, independent of the matrix, $j = 1 \ldots k-1$:

$$v^j = [\sin j\theta_k, \sin 2j\theta_k, \ldots \sin (k-1)j\theta_k] \tag{4.9}$$

where $\theta_k = \pi/k$. Multiplying the matrix A_{11} by the vectors, we see that the



(4.7)

Figure 4.3: We use ϵ to denote $1 - 2\delta - \gamma$.

eigenvalues are $\lambda_j = 2\delta \cos j\theta_k + \gamma$.

If $\alpha \neq 0$, Out of two remaining eigenvectors, only the eigenvector v^β corresponding to β is typically of interest to us. It has the form $[0, 8C, -8C, \left(\beta I - A_{11}^{-1}\right)[C, 0 \ldots -C]]$, where C is a constant, if $\beta I - A_{11}$ is nondegenerate.

A more revealing expression for the components can be found if we regard the eigenvector as a solution to the recurrence

$$\delta\left(v_{i-1}^\beta + v_{i+1}^\beta\right) + (\gamma - \lambda)v_i^\beta = 0, \quad i = 1 \ldots k-2$$

(the numbering of entries in v_β was chosen to be $0, k, 1, 2, \ldots k-1$ to make the equations uniform equations). In addition, we have an additional condition $v_0^\beta = -v_k^\beta$, to ensure that $\left[0, v_\beta^0, v_\beta^1\right]$ is the eigenvector of A_{00}.

The behavior of the solution of the recurrence depends on the ratio $r = (\gamma - \lambda)/\delta$, assuming $\delta \neq 0$ (otherwise, the matrix is diagonal with all eigenvalues equal to γ). The additional condition $v_\beta^0 = -v_\beta^1$ determines a unique solution up to a constant multiplier, even if the matrix $\beta I - A_{11}$ is degenerate. Solutions are listed in Table 4.1.

If $\alpha = 0$, the eigenvalue β has a two-dimensional eigenspace. Two eigenvectors v^β and v'^β satisfying conditions $v_0^\beta = 0$ and $v_k^\beta = 0$ are shown in Table 4.2, for the cases when the matrix $\beta I - A_{11}$ is not degenerate, i.e. when for all $1 \leq j \leq k-1$, $r \neq 2\cos j\theta_k$.

Finally, suppose $\alpha = 0$ and $r = -2\cos(j\theta_k)$ for some j. In this case $\beta = \gamma - \delta r$ is an also an eigenvalue of A_{11}, and, therefore, has multiplicity 3. In this case it has a Jordan block of size 2, and only 2 eigenvectors which can be taken to be $v_j^\beta = \sin j\theta_k$ and $v'^\beta_j = \cos j\theta_k$, $j = 0 \ldots k$.

$r > 2,\ k$ odd	$(-1)^i \cosh\left(i - \frac{k}{2}\right)\theta, \quad r = 2\cosh\theta$
$r > 2,\ k$ even	$(-1)^i \sinh\left(i - \frac{k}{2}\right)\theta, \quad r = 2\cosh\theta$
$r = 2,\ k$ odd	$(-1)^i$
$r = 2,\ k$ even	$(-1)^i\left(n - \frac{k}{2}\right),$
$-2 < r < 2$	$\sin\left(i - \frac{k}{2}\right)\theta, \quad r = -2\cos\theta$
$r = -2$	$i - \frac{k}{2}$
$r < -2$	$\sinh\left(i - \frac{k}{2}\right)\theta, \quad r = -2\cosh\theta$

Table 4.1: The eigenvector corresponding to the eigenvalue β for $\alpha \neq 0$.

$r > 2,$	$(-1)^i \sinh i\theta,\ (-1)^i \sinh(i-k)\theta, \quad r = 2\cosh\theta$
$r = 2,$	$(-1)^i i,\ (-1)^i(i-k)$
$-2 < r < 2$	$\sin i\theta,\ \sin(i-k)\theta, \quad r = -2\cos\theta$
$r = -2$	$i,\ i-k$
$r < -2$	$\sinh i\theta,\ \sinh(i-k)\theta, \quad r = -2\cosh\theta$

Table 4.2: The pair of eigenvectors corresponding to the eigenvalue β for $\alpha = 0$.

Summary of the eigenstructure. We have determined that the eigenvalues of the subdivision matrix are 1,β,$\beta - 2\alpha$, 1/8, 1/16, and $\lambda_j = 2\delta \cos j\theta_k + \gamma$, $j = 1 \ldots k-1$. The eigenvectors corresponding to the eigenvalues λ_j do not depend on the matrix and are given by (4.9). The eigenvectors corresponding to the eigenvalue β depends on the ratio $r = (\gamma - \beta)/\delta$; its entries are given by the formulas in Table 4.1. For $\alpha \neq 0$, there is a single eigenvector. For $\alpha = 0$, there is a pair of eigenvectors (Table 4.2) for the case when β is not an eigenvalue of A_{11}. If β is an eigenvalue of A_{11}, it has a nontrivial Jordan block of size 2.

The case $k = 1$. The matrix in this case has eigenvalues β, $\beta - 2\alpha$, and a triple eigenvalue 1/8. The eigenvectors can be trivially computed.

Coefficients for smooth boundary vertices. One possible choice was given by Hoppe et al. [36] and examined in detail in [88]. In our notation, this choice corresponds to $\beta = 5/8$, $\alpha = 1/8$, $\gamma = 3/8$, $\delta = 1/8$. For extraordinary vertices, and $\beta = 1/2$ for other vertices. Remarkably, the ratio r is -2. The disadvantage of this choice is that the shape of the boundary curve depends on the valence of the vertices on the boundary, hence it becomes impossible to join two meshes continuously along a boundary if extraordinary vertices on two sides do not match.

If we require the boundary curve to be a cubic spline, β has to be 1/2 and α has to be 1/8. We have two degrees of freedom left: γ and δ. It turns out to be sufficient to use only one, and we fix δ at the value corresponding to the regular valence, i.e. 1/8.

We consider the cases $k > 2$, $k = 2$ and $k = 1$ separately.

Case $k > 2$. Once α, β and δ are fixed, the eigenvalues of the subdivision matrix

become 1, $\beta = 1/2$, $\beta - 2\alpha = 1/4$, 1/8, 1/16, and $\lambda_j = (1/4)\cos j\theta_k + \gamma$.

The tangent vector on the boundary of the surface corresponds to the eigenvector of the subdivision matrix with eigenvalue $\beta = 1/2$. This vector should be one of the subdominant eigenvectors. The second subdominant eigenvector is likely to correspond to the largest of the eigenvalues λ_j, i.e. to the eigenvalue $\lambda_1 = \gamma + (1/4)\cos\theta_k$. In order for the eigenvalue 1/2 to be subdominant, we choose γ in such a way that $|\lambda_j| < 1/2$ for $j > 1$, i.e. $\lambda_2 < 1/2$ and $\lambda_{k-1} > -1/2$. For positive γ, the second condition is satisfied automatically. We also would like $\lambda_1 > \beta - 2\alpha = 1/4$. This leads to the following range for γ:

$$\frac{1}{4}(1 - \cos\theta_k) < \gamma < \frac{1}{2} - \frac{1}{4}\cos 2\theta_k \tag{4.10}$$

In this range we also have $|\lambda_1| > |\lambda_j|$ for $j > 1$. There are two choices of γ that we find particularly interesting: $\gamma = 1/4$ and $\gamma = 1/2 - 1/4\cos\theta_k$.

The first choice, $\gamma = 1/4$, is the maximal value of γ independent of k for which it is in the correct range for all $k > 2$. Note that in this case $r = -2$ again. The second choice, leads to equal subdominant eigenvalues $\beta = \lambda_1 = 1/2$. In this case, $r = -2\cos\theta_k$, that is, we can choose θ to be θ_k. The expressions for the subdominant eigenvectors are $v_j^1 = \sin\theta_k$ and $v_j^\beta = \cos\theta_k$, i.e. form a half of a regular $2k$-gon.

In both cases, the characteristic map defined by these eigenvectors is regular. By convex hull property of subdivision with positive coefficients, the image of the characteristic map is entirely contained in one half-plane, so the characteristic map has to be injective.

The choice of $\gamma = 1/2 - 1/4\cos\theta_k$, although being slightly more complex, appears to be more natural. It has the additional advantage of coinciding with

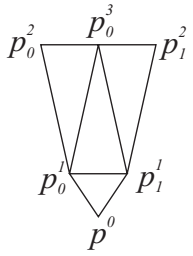


Figure 4.4: The control mesh for the characteristic map in the case $k = 1$.

the regular value $\gamma = 3/8$ for $k = 3$.

Case $k = 2$. In this case, the eigenvalues are 1, 1/2, 1/4, 1/8, 1/16, and $\lambda_1 = \gamma$. Thus, we need to pick $1 > \gamma > 1/4$, to get the same eigenvectors as in the case $k > 2$. It is interesting to note however, that the choice of $\gamma = 1/4$ also results in C^1 surface, although the behavior of the scheme becomes less desirable.

Case $k = 1$. The subdominant eigenvalues are 1/2 and 1/4. They define a configuration of eigenvectors shown in Figure 4.4. The characteristic map is also regular and injective in this case.

Coefficients for corner vertices. Separate rules have to be defined for corners. The interpolation conditions for corners require $\alpha = 0$. Therefore, the block A_{00} has a double eigenvalue β. For a corner, the tangent plane is defined by the two tangents at the non-C^1-continuous point of the boundary. Unlike the case of the smooth boundary points, there is no need to fix all rules on the boundary – parameter β still can be used to ensure smoothness of the limit surface. Hence the rules of Hoppe et al. [36] can be used. One can see [88] that the characteristic map has a convex corner. Therefore, this scheme cannot produce concave corners. *It turns out that in fact no scheme from the class that we have defined can produce smooth concave corners.*

The explicit knowledge of eigenvectors and the convex hull property allows us to determine quickly if a scheme can possibly produce convex or concave corner. If β has multiplicity 3 with Jordan blocks of size 2 and 1 which happens when it is an eigenvalue of A_{11}, the scheme is likely to be non tangent plane continuous; we assume that this is not the case. Then the eigenvectors of interest can be found in Table 4.2 for various values of $r = (\gamma - \beta)/\delta$.

It is easy to see that positive values of r are of little interest to us, because the components of the vectors alternate signs in these cases, and are likely to produce nonregular characteristic maps. Also, for $r \leq -2$ we are guaranteed to get a convex configuration of control points for the characteristic map. As the characteristic map interpolates the boundary curve, it cannot have a concave corner. We conclude that we have to use r from the range $(-2, 0)$. We have seen that in this case the eigenvectors corresponding to the eigenvalue β can be taken to be $\sin i\theta$, $\sin(i-k)\,\theta$, $r = -2\cos\theta$. The triangular fan generated by these two vectors spans the angle $k\theta$. *This means that the corner is convex if $\theta < \theta_k$, and concave otherwise.* In other words, $r = -2\cos\theta < -2\cos\theta_k$, or

$$\gamma < \beta - 2\delta\cos\theta_k \tag{4.11}$$

In addition, we need to ensure that the double eigenvalue β is actually subdominant. To achieve this, we choose δ and γ large enough so that $2\delta\cos j\theta_k + \gamma < \beta$, $j = 1 \ldots k-1$. As $2\delta\cos j\theta_k + \gamma$ decreases as a function if j, and we assume that $\gamma > 0$, it is sufficient to require that $2\delta\cos\theta_k + \gamma < \beta$, which coincides with the convexity condition. We conclude that for $r < 0$ the subdivision scheme can generate only convex smooth corners).

One can show that this is true even if we do not assume that $\alpha = 0$.

In the case $k = 1$, one can also immediately see that the corner produced by subdivision is convex.

Coefficients for concave corner vertices. We assume that $k > 1$. It is impossible to have stationary subdivision rules for a triangular mesh producing a concave corner for $k = 1$. As we have observed, concave corners cannot be produced simply by changing some of the coefficients using the same stencil. One can also show that no scheme with positive coefficients can produce interpolating smooth concave corners. It is possible to construct rules to produce C^1-continuous surface with concave corners, but negative coefficients and larger support have to be used.

Our approach to deriving the rules is based on the idea of reduction of the magnitudes of all eigenvalues, excluding 1 and $\beta = 1/2$. It turns out that this approach leads to a particularly simple rules for subdivision.

For the scheme to produce smooth surfaces at a corner vertex the eigenvectors x^β, x'^β of the eigenvalue $\beta = 1/2$ should be subdominant. If we choose these eigenvectors to be $x^\beta = [0, 0, 1, v^\beta / \sin k\theta, \ldots]$, $x'^\beta = [0, 0, 1, v'^\beta / \sin k\theta \ldots]$ (cf. Table 4.2), corresponding left eigenvectors are very simple: $l = [-1, 0, 1, 0, \ldots]$, $l' = [-1, 1, 0, 0, \ldots 0]$. The left eigenvector l^0 for the the eigenvalue 1 is $[1, 0, \ldots 0]$. Consider the following modification of the vector of control points

$$\tilde{p} = (1 - s)p + s\left((l^0, p)x^0 + (l, p)x^\beta + (l', p)x'^\beta\right)$$

where x_0 is the eigenvector $[1, \ldots 1]$ of the eigenvalue 1. Substituting expressions for the left eigenvectors we get

$$\tilde{p} = (1 - s)p + s\left(p^0 x^0 + \left(p_0^1 - p^0\right)x^\beta + \left(p_k^1 - p^0\right)x'^\beta\right) \tag{4.12}$$

The effect of this transformation is to scale all components of p in the eigenbasis of the subdivision matrix by $(1-s)$ except those corresponding to the eigenvalues 1 and β. If repeated at each subdivision step, it is equivalent to scaling all eigenvalues except 1 and β by $(1-s)$.

To simplify the rules, we observe that it is unnecessary to scale multiple eigenvalues 1/16 and 1/8 of the lower-right blocks of the subdivision matrix. If we apply the rules (4.12), not to the whole vectors of control points p, but to a truncated part, modifying only control points of type 1, as a result, the eigenvalues 1/8 and 1/16 will not change. This observation leads us to the following choice of rules:

$$\tilde{p}_i^1 = (1-s)p_i^b + s\left(p^0 + \left(p_0^1 - p^0\right)\frac{\sin(k-i)\theta}{\sin k\theta} + \left(p_0^1 - p^0\right)\frac{\sin i\theta}{\sin k\theta}\right) \qquad (4.13)$$

In the matrix form, this transformation can be written as

$$T = \left(\begin{array}{c|c} M & 0 \\ \hline 0 & I \end{array}\right)$$

Multiplying this matrix by the subdivision matrix on the left, we see that the eigenvalues of the product ST are eigenvalues of the blocks $B_{00}M$ and B_{11}. By construction. eigenvalues of $B_{00}M$ are 1, 1/2, $(1-s)\left(2\delta\cos j\theta_k + \gamma\right)$, $j = 1\ldots k-1$. As we have seen before, the eigenvalues of B_{11} are 1/8 and 1/16.

By choosing the value of s so that $(1-s)\left(2\delta\cos\theta_k + \gamma\right) < 1/2$, we can ensure that the $\beta = 1/2$ is the subdominant eigenvalue. The parameter s can be viewed as a tension parameter for the corner, which determines how flat the surface is near the corner.

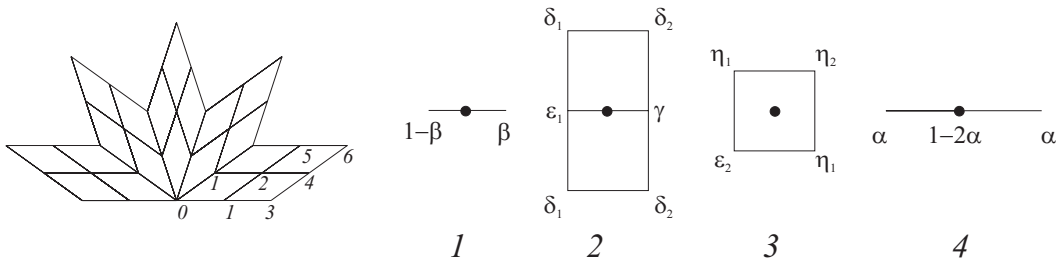


Figure 4.5: Control mesh for a boundary patch of a Catmull-Clark subdivision surface and masks of the subdivision rules. 1. The rule for the boundary odd edge vertices adjacent to the central vertex (type 1). 2. The rule for the interior odd edge vertices adjacent to the central vertex (type 1). 3. The rule for the central vertex (type 0). 2. The rule for the odd face vertices adjacent to the central vertex (type 2). The rules for vertices of type 4, 5 and 6 are the standard Loop rules; the rule for the vertex of type 3 is the standard one-dimensional cubic spline rule.

4.6.2 Catmull-Clark scheme

The analysis of the eigenstructure of the boundary subdivision matrices becomes more complex in the case of the Catmull-Clark scheme. Using the Catmull-Clark scheme as an example, we describe a technique that can be used to analyze schemes with larger support.

The control mesh for a boundary patch surrounding an extraordinary vertex is shown in Figure 4.5.

There are 6 different types of vertices in the control mesh, shown in the same figure. For two types (1 and 3) there are two different masks that are used for boundary and interior vertices respectively. As we did in the case of the Loop scheme, we introduce a number of undefined coefficients into the masks

and find eigenvalues and eigenvectors of the subdivision matrix as functions of coefficients. The choice of the parameters is guided by the same considerations as for the Loop scheme.

Various types of boundary behavior (smooth convex corner, smooth boundary) can be obtained by choosing appropriate values of the parameters. Again, we can show that no scheme from this class can generate surfaces with smooth concave corners.

Subdivision matrix. Subdivision matrix has somewhat more complex structure for the Catmull-Clark scheme. The general form is shown in Figure 4.6. Note that a few of the blocks of the matrix are not symmetric, or even square, and it is not immediately clear how to diagonalize the matrix.

In the block form, the matrix can be written as

$$\left(\begin{array}{c|c|c|c} A_{00} & & & \\ \hline A_{10} & \frac{1}{8}I_2 & & \\ \hline A_{20} & A_{21} & A_{22} & \\ \hline A_{30} & A_{31} & A_{32} & \frac{1}{64}I_k \end{array}\right)$$

where the diagonal blocks are

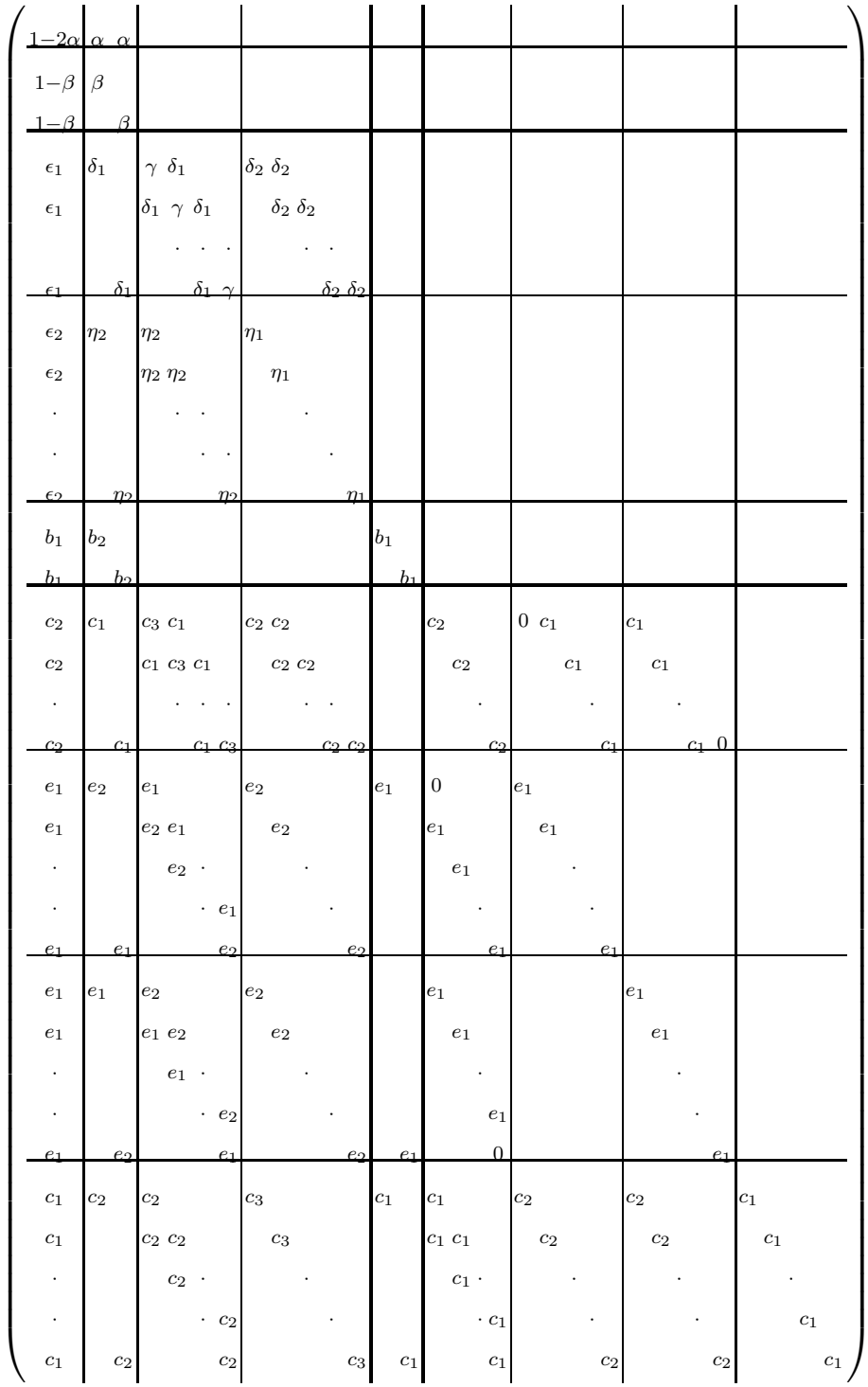


Figure 4.6: The subdivision matrix for the Catmull-Clark scheme.

$$A_{00} = \left(\begin{array}{c|c|c|c} 1-2\alpha & \alpha\ \alpha & & \\ 1-\beta & \beta & & \\ 1-\beta & \beta & & \\ \hline \epsilon_1 & \delta_1 & \gamma\,\delta_1 & \delta_2\delta_2 \\ \epsilon_1 & & \delta_1\,\gamma\,\delta_1 & \delta_2\delta_2 \\ & & \cdot\ \cdot\ \cdot & \cdot\ \cdot \\ \epsilon_1 & \delta_1 & \delta_1\,\gamma & \delta_2\delta_2 \\ \hline \epsilon_2 & \eta_2 & \eta_2 & \eta_1 \\ \epsilon_2 & & \eta_2\eta_2 & \eta_1 \\ \cdot & & \cdot\ \cdot & \cdot \\ \cdot & & \cdot\ \cdot & \cdot \\ \epsilon_2 & \eta_2 & \eta_2 & \eta_1 \end{array}\right) \qquad A_{22} = \left(\begin{array}{c|c|c} c_2 & 0\,c_1 & c_1 \\ c_2 & c_1 & c_1 \\ \cdot & \cdot & \cdot \\ c_2 & c_1 & c_1\,0 \\ \hline 0 & e_1 & \\ e_1 & e_1 & \\ e_1 & \cdot & \\ \cdot & \cdot & \\ e_1 & e_1 & \\ \hline e_1 & & e_1 \\ e_1 & & e_1 \\ \cdot & & \cdot \\ e_1 & & \cdot \\ 0 & & e_1 \end{array}\right) \tag{4.14}$$

Note that all eigenvalues of A_{22} are guaranteed to be less than 1/8 (the sum of the magnitudes of the entries on any line does not exceed 1/8). Thus, only the eigenvalues of A_{00} are of interest to us. Next, we observe that the matrix A_{00} itself has two blocks on the diagonal; the first 3×3 block is identical to the block that we have considered for the Loop scheme; it has eigenvalues 1, β and $\beta - 2\alpha$. The only remaining block that we have to consider is

$$\bar{A}_{00} = \left(\begin{array}{ccccc|ccccc} \gamma & \delta_1 & & & & \delta_2 & \delta_2 & & & \\ \delta_1 & \gamma & \delta_1 & & & & \delta_2 & \delta_2 & & \\ & \cdot & \cdot & \cdot & & & & \cdot & \cdot & \\ & & \delta_1 & \gamma & & & & & \delta_2 & \delta_2 \\ \hline \eta_2 & & & & & \eta_1 & & & & \\ \eta_2 & \eta_2 & & & & & \eta_1 & & & \\ & \cdot & \cdot & & & & & \cdot & & \\ & & \cdot & \cdot & & & & & \cdot & \\ & & & \eta_2 & & & & & & \eta_1 \end{array}\right)$$

This matrix acts on control points of types 1 and 2, excluding boundary control points of type 1.

Transformation of the subdivision matrix. Assume $k > 1$ (we will consider the case $k = 1$ separately). The eigenvalues and eigenvectors of $\bar{A}_0 0$ can be found directly from the recurrences derived from the subdivision rules. We take a somewhat different approach, similar to the DFT analysis used for interior extraordinary vertices. This approach have somewhat greater generality and potentially can be applied to analyze subdivision schemes with larger supports. To find the eigenvalues of $\bar{A}_0 0$, we introduce a new set of control points. We replace control points p_i^2, $i = 0 \ldots k-1$, with $k+1$ control points $\tilde{p}_i^2$ satisfying

$$p_i^2 = \frac{1}{2}\left(\tilde{p}_i^2 + \tilde{p}_{i+1}^2\right) \tag{4.15}$$

for $i = 0 \ldots k-1$. Also, let $\tilde{p}_i^1 = p_i^1$. Note that we increase the number of control points. These equations clearly do not define the new control points

uniquely. However, it is not relevant for our purposes. In the matrix form, the relation between the original vector of control points of types 1 and 2 and the transformed vector $\tilde{p}$ can be written as $p = T\tilde{p}$, where T is a $2k+1 \times 2k+2$ matrix.

In addition, we define the subdivision rules for the new control points. We choose the rules for $\tilde{p}$ in such a way that the relations 4.15 also hold after the subdivision rules are applied to p and $\tilde{p}$. Let $\tilde{S}$ be the subdivision matrix for $\tilde{p}$. Then our choice of rules means that

$$STp = T\tilde{S}\tilde{p}$$

If λ is an eigenvalue of $\tilde{S}$, then $\tilde{S}\tilde{p}^\lambda = \lambda\tilde{p}^\lambda$ where $\tilde{p}^\lambda$ is the corresponding eigenvector, and

$$ST\tilde{p}^\lambda = T\tilde{S}\tilde{p}^\lambda = \lambda T\tilde{p}^\lambda$$

Therefore, λ is also an eigenvalue of S, unless $Tp^\lambda = 0$. Note that the nullspace of T has dimension 1 and and contains the vector $p_i^1 = 0$, $\tilde{p}_i^2 = (-1)^i$. Hence a complete set of eigenvalues and of S can be obtained from eigenvalues and eigenvectors of $\tilde{S}$ once we exclude the eigenvalue corresponding to this vector, if it happens to be an eigenvector.

We choose the subdivision rule for $\tilde{p}_i^2$ as follows:

$$[\tilde{S}\tilde{p}]_i^2 = \epsilon_2 p^0 + 2\eta_2 p_i^1 + \eta_1 \tilde{p}_i^2 \tag{4.16}$$

In terms of new control points, the rule for control points of type 1 becomes

$$[Sp]_i^1 = \epsilon_1 p^0 + \delta_1 \left(p_{i-1}^1 + p_{i+1}^1\right) + \gamma p_i^1 + \frac{\delta_2}{2}\left(\tilde{p}_{i-1}^2 + 2\tilde{p}_i^2 + \tilde{p}_{i+1}^2\right)$$

The matrix $\bar{A}_0 0$ is transformed into

$$\tilde{A}_{00} = \left(\begin{array}{ccccc|ccccc} \gamma & \delta_1 & & & & \frac{\delta_2}{2} & \delta_2 & \frac{\delta_2}{2} & & \\ \delta_1 & \gamma & \delta_1 & & & & \frac{\delta_2}{2} & \delta_2 & \frac{\delta_2}{2} & \\ & \cdot & \cdot & \cdot & & & & \cdot & \cdot & \cdot \\ & & \delta_1 & \gamma & & & & & \frac{\delta_2}{2} & \delta_2 & \frac{\delta_2}{2} \\ \hline & & & & & \eta_1 & & & & \\ 2\eta_2 & & & & & & \eta_1 & & & \\ & 2\eta_2 & & & & & & \eta_1 & & \\ & & \cdot & & & & & & \eta_1 & \\ & & & 2\eta_2 & & & & & & \eta_1 \end{array}\right) \tag{4.17}$$

Note that $\tilde{p}_0^2$ and $\tilde{p}_k^1$ depend on p_0^1 and p_k^1 which are outside this matrix. Rearranging the entries, we get the matrix

$$\left(\begin{array}{cc|ccccc|ccccc} \eta_1 & & & & & & & & & & \\ & \eta_1 & & & & & & & & & \\ \hline \frac{\delta_2}{2} & & \gamma & \delta_1 & & & & \delta_2 & \frac{\delta_2}{2} & & \\ & & \delta_1 & \gamma & \delta_1 & & & \frac{\delta_2}{2} & \delta_2 & \frac{\delta_2}{2} & \\ & & & \cdot & \cdot & \cdot & & & \cdot & \cdot & \cdot \\ & \delta_2 & & & \delta_1 & \gamma & & & & \frac{\delta_2}{2} & \delta_2 \\ \hline & & 2\eta_2 & & & & & \eta_1 & & & \\ & & & 2\eta_2 & & & & & \eta_1 & & \\ & & & & \cdot & & & & & \eta_1 & \\ & & & & & 2\eta_2 & & & & & \eta_1 \end{array}\right)$$

which has 4 diagonal or tridiagonal subblocks of size $k-1 \times k-1$. This

matrix has a double eigenvalue η_1. The rest of the eigenvalues are eigenvalues of the matrix A consisting only of the 4 tridiagonal subblocks. We have already observed that three diagonal matrices have eigenvectors independent from the entries of the matrix. Denote H the matrix with entries $\sin ij\theta_k$, with $\theta_k = \pi/k$ as before, $i, j = 1 \ldots k-1$. This matrix to some extent has the same role in the analysis of subdivision matrices of boundary vertices as the DFT matrix has in the analysis of subdivision matrices of interior vertices. The transform $\mathcal{H}$ is defined as $\mathrm{diag}\,(H, H)$. The inverse of this matrix is $\mathcal{H}^{-1} = \mathrm{diag}\,((2/k)H, (2/k)H)$.

$$\begin{aligned} \mathcal{H}A\mathcal{H}^{-1} &= \frac{2}{k}\begin{pmatrix} H & 0 \\ 0 & H \end{pmatrix}\begin{pmatrix} B_{00} & B_{01} \\ B_{10} & B_{11} \end{pmatrix}\begin{pmatrix} H & 0 \\ 0 & H \end{pmatrix} \\ &= \frac{2}{k}\begin{pmatrix} HB_{00}H & HB_{01}H \\ HB_{10}H & HB_{11}H \end{pmatrix} \end{aligned} \tag{4.18}$$

each block $HB_{ij}H$ is a diagonal matrix. Finally, we apply a the following permutation to the components of the vector: $\left[p_1^1, p_2^1, \ldots p_{k-1}^1, \tilde{p}_1^2, \tilde{p}_2^2, \ldots \tilde{p}_{k-1}^2\right] \rightarrow \left[p_1^1, \tilde{p}_1^2, p_2^1, \tilde{p}_2^2 \ldots p_{k-1}^1, \tilde{p}_{k-1}^2\right]$. Let P be the corresponding permutation matrix. The matrix A is reduced to the block diagonal form

$$P\mathcal{H}A\mathcal{H}^{-1}P^{-1} = \begin{pmatrix} B(1) & & & \\ & B(2) & & \\ & & . & \\ & & & B(k-1) \end{pmatrix} \tag{4.19}$$

where the blocks $B(i)$, $i = 1 \ldots k-1$, are 2×2 matrices

$$B(i) = \begin{pmatrix} \gamma + 2\delta_1 \cos\dfrac{i\pi}{k} & \delta\left(1 + \cos\dfrac{i\pi}{k}\right) \\ 2\eta_2 & \eta_1 \end{pmatrix} \tag{4.20}$$

The explicit expressions for the eigenvalues are not particularly enlightening in the general case and we omit them here.

Case $k = 1$. In this case, the eigenvalues and eigenvectors can be computed directly. The eigenvalues are β, $\beta - 2\alpha$, η_2, 1/8, 1/16 and 1/64.

Eigenvectors. We start with eigenvectors of the matrix A_{00}. We assume that $\eta_1 \neq 0$ and $\delta_1 \neq 0$ and none of the eigenvalues of the blocks $B(i)$ coincide with η_1. In this case, the eigenvectors corresponding to each block $B(i)$ can be taken to be $[0, \ldots 0, 1, r, 0, \ldots 0]$, where the only two nonzero entries are in positions $2i-1$ and $2i$, $r = -2\eta_2/(\eta_1 - \lambda)$, and λ is the eigenvalue. Applying the inverse permutation and transform H, we get eigenvectors of the form $[v^i, rv^i]$, with v^i being a vector of length $k-1$ with entries $\sin(j\theta_k)$, $j = 1 \ldots k-1$. The entries of the eigenvector of A_{00} corresponding to $\tilde{p}_0^2$ and $\tilde{p}_k$ are zero. The remaining possible eigenvalues of A_{00} are β, $\beta - 2\alpha$ and η_1. Once the eigenvalue is known, the expressions for the eigenvectors can be found directly from the subdivision rules. Keeping in mind that for all eigenvectors except the eigenvector of the eigenvalue 1 $p^0 = 0$, an interior control point of type 1 p_i^1 and for a control point of type 2 p_i^2 from an eigenvector p with eigenvalue λ should satisfy

$$\begin{aligned}
&\lambda p_i^1 = \delta_2\left(p_{i-1}^1 + p_{i+1}^1\right) + \delta_1\left(p_i^2 + p_{i-1}^2\right)|\gamma_i^1 \quad i = 1 \ldots k-1 \\
&\lambda p_i^2 = \eta_2\left(p_i^1 + p_{i+1}^1\right) + \eta_1 p_i^2 \quad i = 0 \ldots k-1 \\
&\lambda p_0^1 = \beta p_0^1 \\
&\lambda p_k^1 = \beta p_k^1
\end{aligned} \tag{4.21}$$

For $\lambda \neq \eta_1$, this leads to the following system of equations for p_i^1, $i = 1 \dots k-1$,

$$\left(\delta_2 + \delta_1 \frac{\eta_2}{\lambda - \eta_1}\right)\left(p_{i-1}^1 + p_{i+1}^i\right) + \left(\gamma - \lambda + \frac{2\delta_1\eta_2}{\lambda - \eta_1}\right) p_i^1 \tag{4.22}$$

Denote $\tilde{\eta} = \delta_1\eta_2/\delta_1$. Then, if $\lambda = \eta_1 - \tilde{\eta}_1$, The equation is reduced to $p_i^1(\gamma - \eta_1 + \tilde{\eta}_1 - 2\delta_2) = 0$, which has nontrivial solutions only if $(\gamma - \eta_1 + \tilde{\eta}_1 - 2\delta_2) \neq 0$.

Now we can find expressions for the eigenvectors. We start with the eigenvector of the eigenvalue η_1. Two cases are possible:

1. $\beta = \eta_1$. Then there are two eigenvectors which both have $p_i^1 = (-1)^i$, and for the first one $p_i^2 = (\lambda - \gamma + 2\delta_2)(-1)^i i/\delta_1$, and for the second one $p_i^2 = (\lambda - \gamma + 2\delta_2)(-1)^i(i+1)/\delta_1$.

2. $\beta \neq \eta_1$. In this case, $p_i^1 = 0$, and $p_i^2 = (-1)^i$.

If one of the eigenvalues β or $\beta - 2\alpha$ coincides with η_1, its eigenvectors are described by the same formulas. Suppose $\beta \neq \eta_1$. Then three cases are possible for the eigenvector of β.

1. $\beta = \eta_1 - \tilde{\eta}_1$, $\gamma + \beta - 2\delta_2 = 0$. In this case, the eigenvalue β has multiplicity $k+1$, and the components p_i^1, $i = 0 \dots k$ can be chosen arbitrarily.

2. $\beta = \eta_1 - \tilde{\eta}_1$, $\gamma + \beta - 2\delta_2 \neq 0$. In this case, the eigenvalue β has multiplicity 2, the components p_i^1, $i = 1 \dots k-1$ are zero, and p_0^1, p_k^1 can be chosen arbitrarily.

3. $\beta \neq \eta_1 - \tilde{\eta}_1$, $\gamma + \beta - 2\delta_2 \neq 0$. This is the most useful case. Let

$$r(\lambda) = \frac{\gamma - \lambda + \frac{2\delta_1\eta_2}{\lambda - \eta_1}}{\delta_2 + \delta_1 \frac{\eta_2}{\lambda - \eta_1}} \tag{4.23}$$

then (4.22) reduces to $p^1_{i-1} + p^1_{i+1} + r(\beta)p^1_i = 0$. We have already explored the possible solutions of these equations in Section 4.6.1. The most useful range of $r(\beta)$ is $(-2, 0)$, in which case the eigenvector can be chosen to be $\sin((i - k/2)\theta)$, with $r(\beta) = -2\cos\theta$.

Finally, for $\beta - 2\alpha$ there are two possibilities.

1. $\beta - 2\alpha = \eta_1 - \tilde{\eta}_1$, $\gamma + \beta - 2\alpha - 2\delta_2 \neq 0$. In this case, the eigenvalue β has multiplicity $k - 1$, the components p^1_i, $i = 1 \ldots k - 1$ can be chosen arbitrarily, $p^1_0 = p^1_k = 0$.

2. $\beta - 2\alpha \neq \eta_1 - \tilde{\eta}_1$, $\gamma + \beta - 2\alpha - 2\delta_2 \neq 0$. This case is similar to the third case for the eigenvalue β, with $r(\beta)$ replaced with $r(\beta - 2\alpha)$.

If $\alpha = 0$, then in the case $\beta \neq \eta_1 - \tilde{\eta}_1$, $\gamma + \beta - 2\delta_2 \neq 0$, the eigenvalue β has two eigenvectors that can be chosen to be $\sin i\theta$ and $\sin(i - k)\theta$ (see Table 4.1).

Coefficients for smooth boundary vertices. As it was discussed in Section 4.6.1, it is desirable to use $\beta = 1/2$ and $\alpha = 1/8$ for smooth boundary vertices. This choice of coefficients leads to a cubic spline boundary curve. It is easy to see that we need only a single parameter in this case to ensure C^1-continuity. We choose the parameter γ, using the standard values for all other parameters: $\eta_1 = \eta_2 = 1/4$, $\delta_1 = \delta_2 = 1/16$. In this case, the expression for the eigenvalues λ_j, λ'_j simplifies to

$$\lambda_j, \lambda'_j = \frac{1}{2}\tilde{\eta} + \frac{1}{8} \pm \frac{1}{8}\sqrt{16\tilde{\eta}^2 - 8\tilde{\eta} + 1 + 2\left(1 + \cos j\theta_k\right)} \quad j = 1 \ldots k - 1$$

Note that for any k, j and any $0 < \gamma < 1$, $|\lambda_j| < \lambda_1$ and $|\lambda'_j| < \lambda_1$. From the formulas for the eigenvectors we can tell that it is desirable to have

subdominant eigenvalues $\beta = 1/2$ and λ_1. For λ_1 to be equal to $1/2$, we can take $\gamma = 3/8 - (1/4)\cos\theta_k$. Note that for the regular case $k = 2$ we get the standard value $\gamma = 3/8$. In general, for $1/2$ to be one of the subdominant eigenvalues, it is necessary that $\gamma < 3/8 - (1/4)\cos 2\theta_k$. If one wishes to use a single value of γ for all valences, then the maximal possible choice of γ is $1/8$.

Case k = 1. For the regular choices of parameters, the subdominant eigenvalues are 1/2 1/4, and 1/4 has a Jordan block of size 2. The resulting scheme is C^1, although the normals converge to the limit slower than in other cases due to the presence of the Jordan block. In this case the *parametric map* does not coincide with the characteristic map. The parametric map can be informally characterized as the map approximating, up to affine invariance, any subdivision surface generated near the central control point. Typically, it coincides with the characteristic map, but in the case when one of the subdominant eigenvalues has a nontrivial Jordan block, these maps can be different. The tangent vectors are actually determined by the control vectors of the parametric map. The control net of the characteristic and parametric maps for $k = 1$ and the standard choice of coefficients is shown in Figure 4.7. Assuming the ordering of components $x^1 = [p^0, p_0^1, p_1^1, p^2, p_0^3, p_1^3, p_0^4, p_0^5, p_0^6]$, the eigenvectors defining the maps are $x'^2 = [0, 1, -1, 0, 2, -2, 1, -1, 0]$ (eigenvalue 1/2), $x^2 = [1, 2, 2, 5, 11, 11, 1, 10, 10, 51/5]$ (eigenvalue 1/4, generalized eigenvector), and $[0, 0, 0, 1, 0, 0, 2, 2, 4]$. The characteristic map is defined by the pair (x^1, x^2), the parametric map is defined by the pair (x^1, x'^2).

Coefficients for convex corner vertices. For the corner vertices we choose $\alpha = 1$, $\beta = 1/2$. In this case, we have to ensure that the two eigenvectors of the double eigenvalue β are the subdominant eigenvectors. The necessary condition

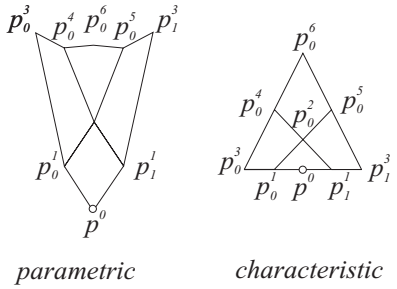


Figure 4.7: The control mesh for the parametric and characteristic maps in the case $k = 1$.

for this is $\lambda_1 < \beta$. In addition, we have to verify that the resulting corner is indeed convex. As it was the case for the Loop scheme, if the characteristic map is regular, for convexity it is sufficient that the control mesh of the characteristic map has a convex corner at the central vertex. As the subdominant eigenvectors for the eigenvalue β can be chosen to have components $p_i^1 \sin i\theta$ and $\cos i\theta$, with $-2\cos\theta = r(\beta)$ with $r(\beta)$ defined by (4.23), the condition for convexity is $r < -2\cos\theta_k$. As it was the case for the Loop scheme, this condition turns out to be exactly equivalent to the condition for the eigenvalue β to be subdominant. We arrive at the same conclusion: *no scheme from the class that we have defined can produce smooth concave corners.*

Coefficients for concave corner vertices. To obtain coefficients that would allow us to generate surfaces with smooth concave corners, we use the same approach that we used for the Loop scheme: we modify the coefficients in such a way that all eigenvalues of the matrix A_{00} except 1 and $\beta = 1/2$ are scaled by constant $s < 1$. Recall that the idea is to use subdivision rules with γ chosen in such a way that the eigenvectors of the eigenvalue $\beta = 1/2$ produce a concave

configuration, and use additional modification of control points to ensure that β are subdominant. The additional rules were derived from the expression

$$\tilde{p} = (1-s)p + s\left((l^0, p)x^0 + (l, p)x^\beta + (l', p)x'^\beta\right)$$

where x_0 is the eigenvector $[1, \dots 1]$ of the eigenvalue 1, x^β and x'^β are eigenvectors of the eigenvalue β, and l^0, l and l' are corresponding left eigenvectors. The left eigenvectors l, l and l' are exactly the same as for the Loop scheme: $l = [-1, 0, 1, 0, \dots]$, $l' = [-1, 1, 0, 0, \dots 0]$ and $l = [1, 0, \dots 0]$. The eigenvectors x^β and x'^β coincide with the eigenvectors for the Loop scheme when restricted to the vertices of type 1. To obtain the desired scaling of eigenvalues we also need to modify vertices of type 2. The components of the eigenvectors corresponding to the vertices of type 2 are easily computed using subdivision rules (cf. (4.21)):

$$p_i^2 = \frac{\eta_2}{\lambda - \eta_1}\left(p_i^1 + p_{i+1}^1\right) = \left(p_i^1 + p_{i+1}^1\right)$$

Therefore, the analog of rules (4.13) for the Catmull-Clark subdivision is

$$\begin{aligned}
[Sp]_i^1 &= (1-s)p_i^1 + s\left(p^0 + \left(p_0^1 - p^0\right)\frac{\sin(k-i)\theta}{\sin k\theta} + \left(p_k^1 - p^0\right)\frac{\sin i\theta}{\sin k\theta}\right) \\
[Sp]_i^2 &= (1-s)p_i^2 + s\Big(p^0 + \left(p_0^1 - p^0\right)\frac{\sin(k-i)\theta + \sin(k-i+1)\theta}{\sin k\theta} \\
&\quad + \left(p_k^1 - p^0\right)\frac{\sin i\theta + \sin(i+1)\theta}{\sin k\theta}\Big)
\end{aligned} \tag{4.24}$$

4.7 Conclusions

The type of subdivision schemes considered in this paper is quite broad, and covers most important cases. It is important to note that our analysis applies

not only to surfaces with piecewise smotoh boundary, but to a more general class of surfaces with creases, which can be obtained by stiching together patches with p.w. smooth boundary. However, as it was pointed out in Section 4.2 a few useful cases are left unconsidered. Practically most important are the cases of higher-order corners (cusps), cone-like surface features. Because of diversity of cases, we believe that step-by-step approach is appropriate for analysis.

One important aspect of theory, only briefly covered in Section 4.6 is the general analysis of the convergence rates for the regular case. This analysis and its applications to interpolating subdivision will be considered in greater detail in a future paper.

Finally, while the techniques described in the paper allow one to analyze a broad class of schemes in a uniform manner, substantial effort is still required for each scheme, including apporximation of the characteristic map. An ideal smoothness criterion would link the subdivision scheme coefficients and smoothness of the limit surfaces more directly.

Chapter 5

Boolean Operations On Multiresolution Surfaces

The contents of this chapter were published in the SIGGRAPH 2001 Conference Proceedings [5]. It presents algorithms for performing approximate boolean operations on solids bounded by piecewise-smooth multiresolution subdivision surfaces introduced in Chapters 2. The result of the operation is represented in the same form, which is one of the major advantages of the proposed approach.

5.1 Introduction

Boolean operations are a natural way of constructing complex solid objects out of simpler primitives. This approach is very common in computer-aided geometric design, as many artificial objects can be constructed out of simple parts, such as cylinders, rectangular blocks, and spheres.

Few computational representations of solids are closed with respect to boolean operations. This means that the result of a boolean operation can-

not be represented precisely in most cases. One way to avoid this problem is to use a tree of boolean operations as the object representation and implement various algorithms directly on such representation. This approach is referred to as constructive solid geometry (CSG) [34]. However, for many applications CSG is not the most efficient or appropriate. Most commonly, boundary representations (B-Reps) of solids are used and boolean operations have to be implemented in B-Rep framework. Such an implementation is quite difficult for higher-order B-Reps as it requires intersecting parametric surfaces, separating them into pieces and constructing new surfaces out of these pieces.

Existing systems typically treat a B-Rep as a collection of trimmed spline patches, sharing the boundaries. The boundaries of individual patches are often matched only approximately, as it is difficult to ensure that two trimming curves in different parametric domains are identical in space. Each intersection operation leads to increasingly complex and difficult to handle trimming curves. It is difficult to apply smooth deformations to the resulting models, since special care must be taken to avoid cracks, etc. An elementary operation required for this surface representation is to intersect two trimmed NURBS patches, which is a difficult problem by itself. As a result, boolean operations are often slow and not fully robust, although excellent results are achieved by some solid modeling cores.

For many computer graphics and animation applications such high-precision and complex techniques are not essential. The most difficult cases such as the case of two identical, but slightly rotated intersecting objects are often of little relevance. At the same time, keeping the calculations efficient and robust is important, as well as ensuring the complexity of the model is manageable.

In this paper, we present a new approach to computing the result of boolean

Figure 5.1: "Venus with drawers" (after S. Dalí) is created using union, intersection and difference operations.

operations on B-Rep solids. We represent the boundary surfaces as piecewise-smooth subdivision surfaces, described in greater detail in Section 5.2. For brevity, we are going to call such solids *free-form* solids. The advantage of this representation is its simplicity: the surface is defined by a control mesh with tagged creases and corners, as well as sets of details added at finer levels. Continuity and smoothness of the surface are guaranteed automatically. Representations of this type are increasingly popular, as they considerably simplify modeling complex free-form objects.

While the problem we are solving is similar to the traditional CAGD problem, our work is primarily motivated by requirements of applications in computer animation and conceptual design. We aim at fast and robust *approximate* calculations; it should be possible to mix boolean operations with free-form deformations and other types of surface manipulation.

These goals radically change the set of problems that we need to solve:

surface-surface intersection, usually regarded as the central part of an implementation of boolean operations, becomes secondary. In particular, we relax the requirement that the topology of the intersection curve is computed precisely. Our primary emphasis is on algorithms for construction of the approximating multiresolution surface for the result. Rather than adjusting our representation to the needs of boolean operations, e.g. by introducing trimming curves, we develop algorithms that allow us to keep the representation of the results simple.

Our main contributions include:

- an algorithm for constructing a coarse control mesh for the result of a boolean operation;
- algorithms for defining and optimizing a parameterization of one multiresolution subdivision surface (result) over another (one of the original surfaces).
- a hierarchical fitting procedure for a surface parameterized over another surface.

5.1.1 Previous Work

Our work is most closely related to, and was done in parallel with, the work of Litke et al. [59] on trimming subdivision surfaces. A few similar issues have to be addressed in both cases. In particular, the connectivity of the control mesh for the trimmed subdivision surface has to be changed, and the new surface needs to be parameterized over the original. However, for trimming there is no need to merge the control meshes of two separate surfaces, and in our case there is no need to use a special combined subdivision scheme [52] for representing the intersection curve. [59] does not optimize parameterizations and surface fitting

issues are resolved differently.

Linsen [57] has developed a technique for blending of subdivision surfaces. While [57] presents a construction of a combined control mesh for a blend of two subdivision surfaces, the issues of matching the geometry of the intersection curve and approximation of the result of boolean operations are not considered.

Multiresolution subdivision surfaces were introduced in [61, 84, 108]; we use piecewise smooth subdivision rules of [6] to be able to represent sharp features on multiresolution surfaces. Fitting of subdivision surfaces is discussed in [33, 36].

Our work can be contrasted with the work of Rappoport et al. [85] and earlier work of Goldfeather et al. [28] and Rossignac [24] on efficient rendering CSG objects. While boolean operations on CSG objects are straightforward, substantial effort is required to render them and interactive rendering is possible only for simple objects. On the other hand, it is much more difficult to implement boolean operations on multiresolution surfaces, but interactive rendering is straightforward for surfaces of substantial complexity.

Extensive literature exists on solid modeling with B-Reps (surveys can be found in [1, 87]). The emphasis there is on accuracy and correct and consistent handling of degenerate cases, issues that we avoid by replacing the requirement of topological correctness of the result with the weaker requirement of topological consistency.

To compute approximate intersection curves we use perturbation techniques of [92].

The part of our work on parameterization optimization builds on the techniques used in mesh optimization community [25, 27]. Different methods to solve similar problems were proposed in [64] and [49].

5.1.2 Overview of the algorithm

We apply a boolean operation (intersection, difference or union) to two free-form solids bounded by parametric surfaces (Figure 5.2). The details of our surface representation are discussed in Section 5.2. We assume that each bounding surface is an orientable closed surface embedded in $\mathbf{R}^3$. A surface M of this type separates the space into a bounded and an unbounded connected volume. The free-form solid defined by M is the bounded volume.

The main steps of our procedure are illustrated in Figure 5.3.

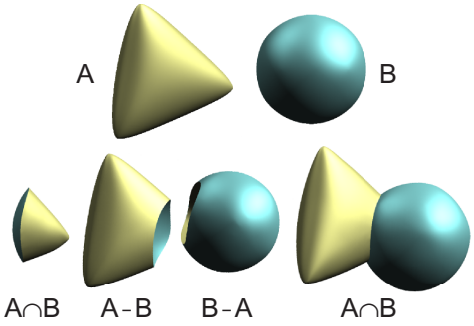


Figure 5.2: Elementary boolean operations on simple subdivision surfaces.

Step 1. Compute an approximate intersection curve, finding its images in each of the two parametric domains of the original surfaces.

Step 2. Construct the connectivity of the control mesh for the result, and an initial parameterization of parts of the resulting surface over the domains of the original surfaces.

Step 3. Optimize the parameterization of the result over the original domains.

Step 4. Determine geometric positions for the control points of the result using hierarchical fitting.

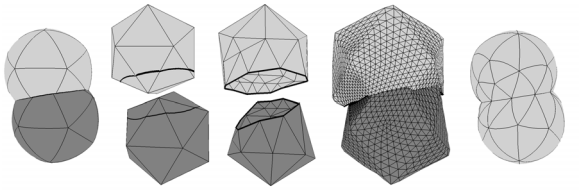


Figure 5.3: Left to right: Two separate surfaces intersected. Images of the intersection curve in the parametric domains. Parametric domains after cutting. Parametric domain for the result after optimization. Final result after fitting.

Notation. All quantities with index 1 refer to the first solid, and all quantities with index 2 refer to the second solid. The parametric domains of the surfaces of the solids are denoted M_1 and M_2, the maps defining the surfaces are $f_i : M_i \to \mathbf{R}^3$ (Figure 5.4). Greek letters are reserved for arbitrary points in the parametric domains; e.g. $\alpha = (u, v, w, i)$ where i is the triangle index and (u, v, w) are the barycentric coordinates, $u + v + w = 1$.

M_i	domains for original surfaces
$f_i : M_i \to \mathbf{R}^3$	evaluation maps for the original surfaces
$c(t) : I \to \mathbf{R}^3$	spatial intersection curve
$c_i(t) : I \to M_i$	images of the intersection curve in the domains of original surfaces
M_i'	parts of the domains M_i to be combined into a new domain
$p_i' : M_i' \to M_i$	parameterization of M_i' over the domain of the original surface M_i
$c_i'(t) : I \to M_i'$	images of the intersection curve in domains M_i'
$\widetilde{M}$	domain for the result formed by merging M_i'
$\widetilde{M_i}$	subdomains of $\widetilde{M}$ parametrized over M_i
$\widetilde{p}_i : \widetilde{M_i} \to M_i$	parameterizations of $\widetilde{M_i}$
$\widetilde{c}(t) : I \to \widetilde{M}$	image of the intersection curve in $\widetilde{M}$
p_v^j	control point at a vertex v of a parameter domain at refinement level j

5.2 Multiresolution Subdivision Surfaces

Before describing the algorithm in greater detail, we briefly review subdivision surfaces with a special focus on parameterization. Subdivision surfaces are defined by an initial control mesh. We use a variant of Loop's subdivision scheme for triangular meshes with rules for corners and creases [6].

Multiresolution surfaces extend subdivision surfaces by introducing *details* at each level. Each time a finer mesh is computed, it is obtained by adding detail offsets to the subdivided coarser mesh. The process of reconstructing a

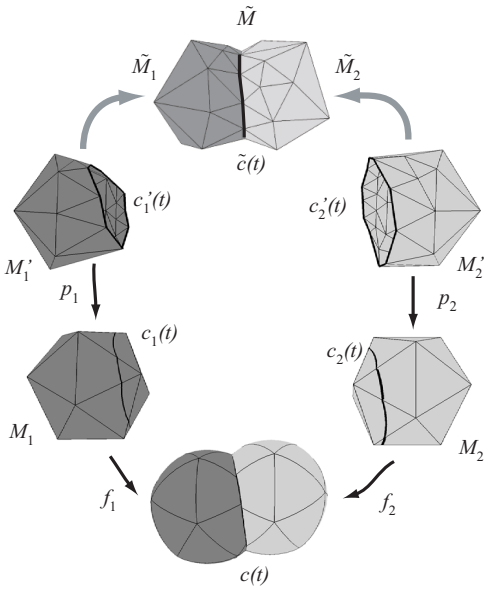


Figure 5.4: Maps used in the algorithm.

surface from the coarse mesh and details is called *synthesis*. Formally, let S be the subdivision operator (a matrix mapping control points on a coarser level to a finer level) let p^l be the vector of control points on level l. Given the detail coefficients d^{l+1} for the next subdivision level, the rule for computing the control points on the finer level is $p^{l+1} = Sp^l + d^{l+1}$.

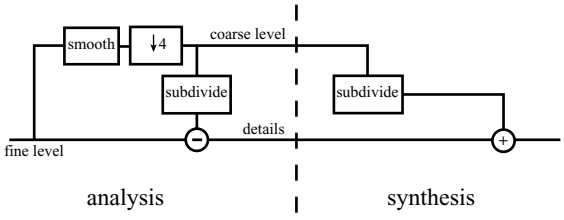


Figure 5.5: Synthesis and analysis diagrams for multiresolution surfaces.

The inverse process of converting the data specified on a fine resolution level to the sequence of detail sets and the coarsest level mesh is called *analysis*.

For analysis, we need a way of obtaining the coarse mesh from the fine mesh. This can be done in a number of ways: simple Laplacian smoothing or Taubin's smoothing[100], quasi-interpolation [59] or least-squares fitting. The synthesis and analysis diagrams are shown in Figure 5.5. Figure 5.6 shows smooth surfaces corresponding to different levels of resolution.

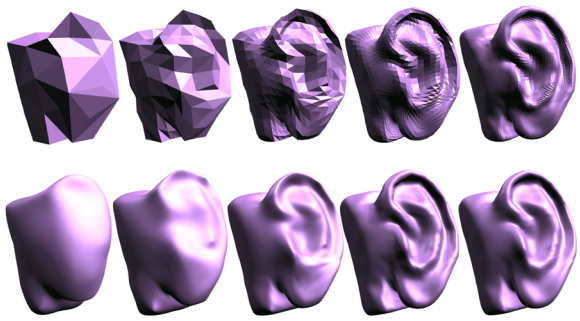


Figure 5.6: Multiresolution surface. Upper row: coarse-to-fine hierarchy of control meshes. Lower row: corresponding surfaces.

For the purposes of this work, a specific choice of analysis method is irrelevant; it is important to note that given a multiresolution mesh represented as the coarsest mesh and details on finer levels one can reconstruct the surface uniquely, without knowing what analysis method was used. In the areas on the surface resulting from a boolean operation where the details need to be recomputed we use fitting and quasi-interpolation to obtain the details, as described in Section 5.5.

Parameterization over the initial control mesh. Suppose the initial control mesh is a simple polyhedron, i.e., it does not have self-intersections. (We do not need this assumption, but it simplifies the presentation.) Suppose each time we apply the subdivision rules to compute the finer control mesh, we also apply midpoint subdivision to a copy of the initial control polyhedron.

Note that each control point that we insert in the mesh using subdivision corresponds to a point in the midpoint-subdivided polyhedron. Another important fact is that midpoint subdivision does not alter the control polyhedron regarded as a set of points; and no new vertices inserted by midpoint subdivision can possibly coincide.

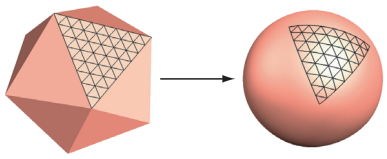


Figure 5.7: Dyadic parameterization: the surface is parameterized over the coarsest level control mesh.

As we repeatedly subdivide, we get a mapping from a denser and denser subset of the control polygon to the control points of a finer and finer control mesh. In the limit, we get a map from the control polygon to the surface (Figure 5.7). This parameterization permits direct evaluation at arbitrary parameter position following the approach of [96], which trivially generalizes to multiresolution setting.

We use the term *parametric domain of a surface* for the top-level control mesh when we discuss parameterizations. The triangles of the top-level control mesh are referred to as *parametric triangles*. *Parametric position* of a point is defined by an index of the triangle in which it is located, together with barycentric coordinates in the triangle.

We reserve the term *vertices* for the vertices of triangles in the parametric domain; the vertices of the three-dimensional control meshes are referred to as *control points*. Multiple control points (one for each level $l \geq i$) correspond to the vertices on levels finer than i.

Linear vertex charts. Quite often, we need to move points continuously in the parametric domain of a subdivision surface. In this case, it is convenient to use local charts, i.e. maps of parts of the parametric domain (the control mesh) to the plane. We use one of the simplest types of charts, piecewise linear charts. A piecewise linear chart maps one ring of triangles $N_1(v)$ around a vertex v of valence k of the control mesh to a regular k-gon Π_k in the plane. Let $g : N_1(v) \to \Pi_k$ be the map from the neighborhood of a vertex to the chart. We can move a point p in one of the triangles in $N_1(v)$ anywhere within the neighborhood: we map it to the plane using the map g, apply a transformation T in the plane, and map it back to get the new position p': $p' = g^{-1} \circ T \circ g(p)$. The point p' can end up in any triangle of $N_1(v)$. We use this simple procedure to move points in the parametric domains in two cases: intersection curve snapping (Section 5.4) and parameterization optimization (Section 5.5).

5.3 Intersection Curve

For simplicity, we assume in the exposition that the objects intersect along a single curve; in the case of multiple intersection curves, all considerations apply to each curve individually.

The goal of the first step of our algorithm is to find a piecewise-linear approximation to the intersection curve $c : I \to \mathbf{R}^3$, where I is an interval, along with its images in the domains M_1 and M_2, $c_i : I \to M_i$, $i = 1, 2$.

The problem of intersecting two surfaces has received a lot of attention (e.g. [4], [37], [89], [46]). The main difficulty is that the topology of the intersection is unknown in general and may be unstable with respect to small perturbations of the surfaces.

However, we observe that the cases where the problem is ill-conditioned are typically of least interest to us. It is relatively unlikely that it is necessary to find intersections of two slightly touching objects precisely. Thus, we can weaken the requirements of our algorithm and only require that it produces intersections with *valid* topology rather than correct topology. In other words, we require that there are small perturbations of the original surfaces such that the intersection curve has that topology. This allows us to replace the problem of intersecting smooth surfaces with the problem of intersecting approximating meshes. Intersecting polyhedra is a substantially simpler problem, although some effort has to be invested to obtain a fast and robust algorithm.

It should be noted that one can theoretically obtain true intersection topology in all cases excluding degenerate (e.g. single point of contact) by using adaptive refinement following an approach similar to [44]. However, as the topology of the intersection curve can be highly complex even for simple surfaces, many levels of refinement can be required in certain areas, which is contrary to our goals of efficiency and robustness.

The naive mesh intersection algorithm (intersect each pair of triangles from two meshes; construct intersection curves as connected sequence of triangle pair intersections) is inefficient and is not robust. We address these problems with bounding box hierarchies and control point perturbation.

Bounding box hierarchies. To accelerate the algorithm we use axis-aligned bounding box hierarchies for each mesh. This appears to be the most efficient approach for our data. Using tighter bounding volumes, such as nonaligned bounding boxes or higher order volumes which are useful for collision detection (see [56]) does not necessarily lead to major improvements in performance for

computing intersections. Most collision algorithms are optimized for quick exclusion testing, but in our case we are expecting collisions. Using axis-aligned bounding boxes allows each collision test to be executed quickly, each one localizing the collision further.

Perturbation method for computing intersections. The polyhedral intersection algorithm relies crucially on the test whether an edge is intersected by a triangle. Usually, this test is implemented with the above-predicate, which determines whether a point p_0 is above or below the plane of a triangle. Consider a triangle with points $p_1, p_2, p_3 \in \mathbf{R}^3$. A point is above the triangle if the determinant $\det[p_0, p_1, p_2, p_3]$ is positive, where the points p_i are represented in homogeneous form. The evaluation of this determinant is error-prone due to rounding errors in floating point arithmetic.

One can find a tight bound for the error [94], and if it leads to an undetermined result, resort to exact or arbitrary precision arithmetic [9, 94].

We use a simpler approach based on the perturbation scheme of [92]. In the case of a determinant sign uncertainty, we abort the intersection computation, perturb the input points by a small amount (depending on the triangle size) and perform the test again for all triangles affected by the perturbation. This is consistent with our goal of finding a consistent approximate intersection reliably and efficiently. Nevertheless, we note the concern about perturbation schemes in geometric modeling: parallel edges and other degeneracies are often intentional design decisions [19, 10, 98, 92].

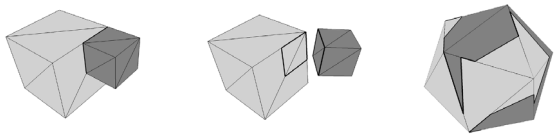
Specifically we replace each point p_i with a linear function $p_i(\varepsilon) = p_i + \varepsilon r_i$, where r_i is a random direction. To determine the sign for the original data, we

use

$$\lim_{\varepsilon \to 0^+} \operatorname{sign} \det[p_i(\varepsilon), p_j(\varepsilon), p_k(\varepsilon), p_l(\varepsilon)].$$

We can easily determine the sign of the expression: the determinant is a cubic polynomial in ε, and the sign is determined by coefficient of the linear term, if its sign can be computed reliably. If it can not, we choose a different perturbation, and recompute affected points. A more satisfying way of dealing with the problem is to use accurate calculations on the determinant.

Figure 5.8 shows some examples of intersections where degenerate cases are resolved with perturbation.



(a) (b) (c)

Figure 5.8: Degeneracies resolved by perturbation. (a) Two cubes with incident edges and vertices. (b) Intersection curves. (c) Intersection of two identical icosahedra. The polyhedra are rendered with perturbation to illustrate the topologically valid intersection.

5.4 Cutting and Merging Parametric Domains

Once the intersection topology is determined, we proceed to cut the parametric domains of the original solids and combine them into a single parametric domain for the resulting object. At this stage we do *not* determine the positions of the control points for the new object; nor do we establish a final correspondence between the points of the new parametric domain and the original domain. We do assign initial values for this parameterization to provide initialization

for optimization later. We used three main considerations when choosing the algorithm for cutting the domains:

- the topology of the cut should be the same as the topology of the intersection curve;
- as few as possible vertices should be added to the mesh representing the domain;
- the valence of the inserted vertices should be kept small.

The last two requirements often conflict with each other. For example, it might be necessary to split a single edge of the original domain into many pieces to capture the topology of the intersection curve. If we do not insert any additional vertices and simply connect all the points on the edge to the opposite vertex, this will considerably increase its valence. We prevent such situations by inserting vertices using only quadrisection of a triangle. This operation has to be followed by bisections of adjacent triangles to keep the mesh of the domain conforming; as a result, the valence of vertices may still increase, but much more slowly.

It should be noted that our algorithm intentionally ignores the geometry of the intersection curve and of the original surfaces to the extent it is possible without violating the first requirement.

There are two steps in constructing the domain for the result of a boolean operation: cutting each of the original domains M_1 and M_2 using the intersection curve, and merging relevant parts into a single domain. At the first stage, several pieces are produced for each of the initial meshes; depending on the operation, one or the other piece has to be discarded. We assume that the normals of the surfaces bounding the solids are oriented outwards, which allows us to identify

the part of the mesh to be discarded locally.

More formally, the output of this algorithm is a new parametric domain $\widetilde{M}$, separated into two parts $\widetilde{M_1}$ and $\widetilde{M_2}$, such that $\widetilde{M_1} \cap \widetilde{M_2}$ is a single chain of crease edges forming a piecewise-linear curve $\widetilde{c}(t)$ along with maps p_i, $i = 1, 2$ from vertices of $\widetilde{M_i}$ to M_i. Note that on the curve $\widetilde{c}(t)$ both maps p_i are defined. The additional property that we require is that $p_i(\widetilde{c})$ is in the image of the intersection curve c_i in the parametric domain M_i, $i = 1, 2$, and for every point α on the curve $\widetilde{c}$ $p_i(\alpha) = c_i(t)$, $i = 1, 2$, for some t. The last condition ensures that the common curve $\widetilde{c}$ of subdomains M_1 and M_2 is mapped one-to-one to the spatial intersection curve $c(t)$.

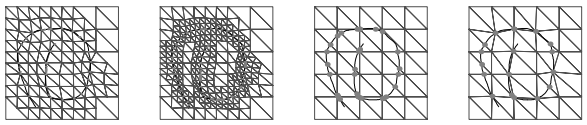
5.4.1 Cutting

The result of cutting is a set of two domains M_1' and M_2' that are combined into the resulting domain $\widetilde{M}$ in the next stage. Simultaneously, a map p_i' is constructed for each of the domains that maps it to the original domain M_i, and the image of the boundary $p_i'(\partial M_i')$ is contained in the image of the intersection curve $c_i(t)$.

Each of the domains is constructed gradually by refining a copy of one of the original domains M_i. Initially, M_i' is just a copy of M_i, and p_i' maps vertices of M_i' to the corresponding vertices of M_i. For the intersection curve we also maintain two temporary images $c_i'(t)$ which define the position of the intersection curve in the new domains. Again, these images are initialized to copies of $c_i(t)$.

The cutting algorithm has two alternating steps: refinement and snapping (Figure 5.9). The goal of snapping is to identify points of the intersection curve $c_i'(t)$ with nearby vertices of the parametric domain. Snapping is optional, but

important for obtaining domains of low complexity. The goal of refinement is to reduce the size of triangles to simplify the shape of the intersection of the curve $c_i'(t)$.



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

Figure 5.9: Refinement and snapping. (a) and (b) Refinement steps. (c) Curve in one of the domains $\widetilde{M}_i$, i=1,2. (d) Snapping the curve to vertices.

Refinement. A triangle containing a part of the intersection curve (*a curve triangle*) is refined if the curve intersects the triangle boundary more than twice, does not intersect it at all, or intersects it twice but on the same side. We call such a triangle *bad*. When a triangle is refined, the maps p_i' are assigned values in the domain M_i for new vertices using midpoint subdivision of barycentric coordinates in the corresponding triangle of M_i. On each refinement step the positions of the images of the intersection curve c_i' in the new domain M_i' are recomputed by converting the barycentric coordinates in the parent triangle to the coordinates in the new triangles.

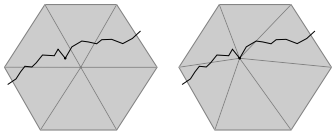


Figure 5.10: Left: Before snapping, all parameter values for curve vertices are mapped to chart; Right: After snapping, all chart coordinates are mapped back to triangles and barycentric coordinates for a new parameterization of the center vertex.

Snapping. We snap the curve c_i' to a vertex of M_i' if there is a point α on the curve for which one of the barycentric coordinates is larger than a *snapping constant* in the range 0 to 1. For most of the organic surfaces we used a snapping constant of 0.8, snapping to the curve if one of its sample's barycentric coordinates was 0.8 or greater. For the more mechanical shapes we effectively turned off snapping by setting this constant to 1.0, creating a closer match to the curve at the cost of a finer triangulation.

Snapping moves the curve to the vertex in two steps (Figure 5.10). First, we move the image of the vertex in the vertex chart $g_v(v)$ to the image of the curve $g_v(c_i'(t))$; a piecewise linear map r maps the k-gon Π_k to itself, with the point $g_v(v)$ mapped to a point $g(\alpha)$ on the curve. Then we apply the inverse transformation r^{-1} to each point of the curve to move it inside Π_k. Finally, we apply the inverse of the parametric map g_v to obtain new positions of the curve points in the parametric domain M_i. As a result, the point α of the curve c_i' is mapped to v, and the curve is continuously shifted in the domain. Snapping may produce new bad triangles, forcing us to refine again. The complete cutting algorithm is given by the following pseudo-code.

Algorithm Snap and Refine

do

    foreach triangle vertex v on the curve

        find closest curve point α to v

        snap v to α if possible

    if there are bad triangles

        refine bad triangles

while there are bad triangles

Upon termination, one can easily find a sequence of edges which is topologically equivalent to the intersection curve: one simply has to split the triangles intersected by the curve in two (Figure 5.11). The values in the intersection curve images $c_i'(t)$ are used to set the initial parametric positions $p_i'(t)$ for the vertices on the edges along which we cut the domain M_i'.

Once the domain is cut, the part which is not required to construct the result of the boolean operation is removed. Note that in the resulting domain M_i' the parametric positions in M_i of all interior vertices are determined by midpoint subdivision of barycentric coordinates. Only the positions on the boundary are shifted towards the image of the intersection curve; this may create folds in the initial parameterization, which are removed at later stages.

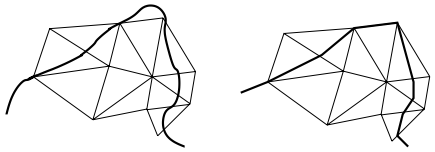


Figure 5.11: Left: curve in the parametric domain. Right: topologically equivalent strip of edges.

The procedure we have described is used for all vertices except for crease vertices. These vertices are constrained to snap only to the points on $c_i'(t)$ which are also on the same crease.

5.4.2 Merging

Next, the domains M_1' and M_2' are joined along their boundaries (Figure 5.12). The output of this stage is the domain $\widetilde{M}$ for the result, with the intersection

curve corresponding to a sequence of crease edges. We describe the algorithm for a single connected intersection curve to simplify the presentation, but it can be applied to multiple intersection curves without any changes.

Initially, correspondence between points of the boundaries of the domains M_1 and M_2 is specified indirectly: points α_1 and α_2 on the boundaries of M_1 and M_2 are identical, if $p_i'(\alpha_i) = c_i(t)$ for some t and $i = 1, 2$, i.e. they correspond to the same position on the intersection curve. However, it is not true in general that if α_1 is a vertex, corresponding α_2 is a vertex.

In order to match the vertices on corresponding boundaries, we will use similar snapping and refinement steps as before, modifying the domains M_1' and M_2', and the parameterizations p_1' and p_2'. If one domain is lacking a boundary vertex, we can create a new one using refinement. Boundary vertices which almost coincide can be snapped together.

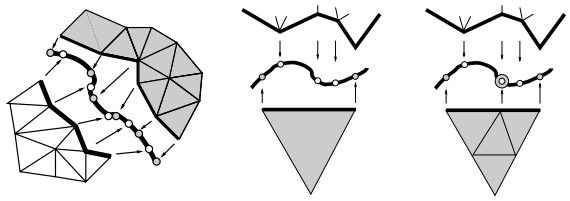


Figure 5.12: Domain merging. Left: Boundary vertices do not match up. Middle and Right: Triangle split matching a vertex on the other domain.

The algorithm has two phases and is entirely symmetric for both domains M_1' and M_2'. The first phase is an iteration where close pairs of unmatched boundary vertices are snapped together. Suppose the domain M_1' has a vertex v_1 with $p_1'(v_1) = c_1(t_1)$, and the other domain M_2' a vertex v_2 with $p_2'(v_2) = c_2(t_2)$, and $|c(t_1) - c(t_2)| < \varepsilon$ for a choice of snapping constant ε. Then we adjust $p'(v_1)$

and $p'(v_2)$ so that $p'_i(v_i) = c_i((t_1 + t_2)/2)$. ε is chosen to be a fraction of the minimal spatial distance between sequential vertices on the curve taken from the same side.

The second phase creates new vertices by refinement. Consider a boundary edge e of M'_1 that corresponds to an intersection curve segment from t_0 to t_1. Assume that this segment contains a boundary vertex of the other domain at curve parameter t for $t_0 < t < t_1$. In this case we split edge e to get a new vertex v and assign a parametric value $p'(v_1)$ to be $c_1(t)$. We repeat the steps until all vertices are matched.

Finally, we are in a situation when the boundaries of M'_1 and M'_2 can be trivially identified, to produce the domain $\widetilde{M}$ for the resulting surface. The subdomains $\widetilde{M_1}$ and $\widetilde{M_2}$ are simply images of M'_1 and M'_2 in the joined domain, and parameterizations $\widetilde{p}_i$ over the original domains are given by reassigning values of p'_i on M'_i to corresponding vertices in $\widetilde{M}$.

During the merging process, we also take care to mark the intersection as a crease, and mark as corner vertices any vertex formed by the intersection of a crease with the intersection curve. We further mark concave and convex sectors for subdivision rules of [6] based on the angle of a sector on the new surface in the limit, evaluated on the original surfaces.

5.5 Parameter Optimization

The snapping and merging steps guarantee that every vertex of the newly constructed domain $\widetilde{M}$ can be located in one of the original domains M_1 and M_2. This allows us to evaluate the corresponding surface positions for any point α in $\widetilde{M}$ as $f_i(p_i(\alpha))$ for $i = 1$ or 2. However, the maps p_i are not one-to-one and

can introduce substantial distortion into the surface shape.

As the next step of our algorithm, we optimize the parameterizations $\widetilde{p}_i$ of $\widetilde{M}_i$. No new maps or domains are created. This step has two goals:

- ensure that the parameterization is one-to-one;
- and that images of the triangles $\widetilde{M}$ have aspect ratios not too far from one.

We used two methods to optimize the parameterization, widely used Laplacian smoothing (e.g. [25]) and area-to-perimeter ratio maximization [2], combined with the optimization technique of [27].

The advantage of Laplacian smoothing is that it is relatively easy to evaluate and accelerate. However, it is known to produce results with flipped or extremely thin triangles, especially near boundaries with concave corners. Such boundaries are common in the meshes produced by taking differences of free-form solids. The second, slower, method is used to improve the parameterization and eliminate flipped triangles.

Optimization functionals . We define the distortion measures that we minimize for a vertex of a planar mesh, and then explain how we compute these quantities for a vertex of a parametric domain mapped into another parametric domain.

Laplacian smoothing minimizes the difference between the position of a vertex $q(v)$ and the barycenter of surrounding vertices:

$$E_{\text{Laplace}}(v) = \sum_{w \in N_1(v)} \|q(v) - q(w)\|^2$$

To obtain the functional to minimize we simply sum $E_{\text{Laplace}}(v)$ over all vertices.

The second distortion measure that we use is equivalent to *area-to-perimeter ratio*, which favors equilateral triangles [2]. Instead of computing the perimeter, we use the sum of squares of edges, to make it a smooth function of point positions.

$$E_{\text{ap}}(v) = \\ -\min_{[u,v,w]} \frac{\text{Area}([q(u), q(v), q(w)])}{\|q(u) - q(v)\|^2 + \|q(u) - q(w)\|^2 + \|q(v) - q(w)\|^2}$$

where $[a, b, c]$ denotes a triangle with vertices a, b, c and $[u, v, w]$ ranges over the triangles of $N_1(v)$. It is easy to show that the minimal value of the functional is attained for an equilateral triangle. In this case, to obtain the functional we take the maximal value over all vertices, which amounts to taking the maximal value over all triangles. The two distortion measures are compared in Figure 5.13.

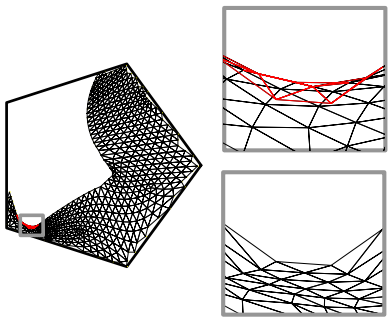


Figure 5.13: Left: Parameter optimization in a single chart. The boundary of the parameterization has concavities. Upper right: Laplacian smoothing produces a fold on the boundary (red). Lower right: Area-to-perimeter ratio optimization punishes for flipped triangles and produces a one-to-one parameterization.

Computing the distortion measures in a parametric domain. The simplest approach to compute one of the distortion measures for a vertex v of the

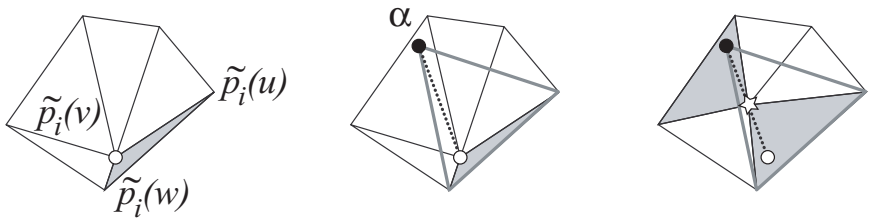


Figure 5.14: Area-to-perimeter ratio optimization. Left: initial ring; the triangle with worst aspect ratio is highlighted. Middle: position $\widetilde{p}_i(v)$ of vertex v is moved along the dotted line segment towards optimal position α. Right: the optimal position is marked with star. For this position distortion of two highlighted triangles is the same.

domain $\widetilde{M}$ is to map the positions $\widetilde{p}_i(w)$ of the vertices w of $\widetilde{M}$ adjacent to v to the plane, and evaluate the functional here. As long as the distortion introduced by the map is small, we can safely use the distortion measure computed in this way for optimization. Linear charts described in Section 5.2 can be used to map points to the plane if we assume that all points $\widetilde{p}_i(w)$ involved in computing a single term in the functional are contained in a single triangle ring of one of the original domains M_i. In other words, the following condition holds:

Single-ring condition. *The parametric image $\widetilde{p}_i(N_1(v))$ of a ring of triangles centered at a vertex v of $\widetilde{M}_i \subset \widetilde{M}$ is contained in a ring of triangles $N_1(w)$ for some vertex w.*

However, after the initial step (cutting and merging of the original domains) one cannot guarantee that images of all triangles of $\widetilde{M}$ are contained in a single ring of triangles in one of the domains M_i. Snapping may spread a few triangles between rings (Figure 5.15). We use adaptive subdivision of positions $\widetilde{p}_i$ of vertices of $\widetilde{M}$ in the parametric domain to refine the mesh $\widetilde{M}$ until the condition

is satisfied.

Subdivision in parametric domain. As the vertices of a triangle of $\widetilde{M}_i$ may map to different triangles of M_i it is not immediately clear how to subdivide parametric values at these vertices. We use the following approach:

Suppose v_1 and v_2 are two vertices in $\widetilde{M}_i$ such that $\widetilde{p}_i(v_1)$ and $\widetilde{p}_i(v_2)$ are in different nonadjacent triangles of M_i. Dijkstra's shortest path algorithm is used to find the chain of triangles between v_1 and v_2. The vertex w in $\widetilde{M}_i$ inserted on the edge connecting v_1 v_2 is assigned position $\widetilde{p}_i(w)$ which is the center of the middle triangle on the path, if there is one. We apply subdivision until all vertices in $\widetilde{M}$ satisfy the single-ring condition.

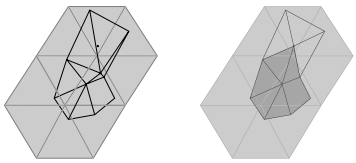


Figure 5.15: Left: Each one-ring is contained within a single chart from original mesh. Right: The dark one-ring is no longer contained in any chart.

Optimization procedures. A single step of Laplacian smoothing consists of moving the position $\widetilde{p}_i(v)$ of a vertex v of $\widetilde{M}_i$ to the average of the positions of surrounding vertices. This can be done using the linear charts as described above; we move the position of each vertex as far as possible towards the barycenter without violating the single-ring condition for any of the rings depending on it.

For the area-to-perimeter ratio, the distortion is minimized by the following procedure (Figure 5.14). We pick the triangle $[v, w, u]$ in the ring $N_1(v)$ with maximal distortion and move the position $\widetilde{p}_i(v)$ of the vertex v along the

segment connecting the old position with a point α, such that the triangle $[\alpha, \widetilde{p}_i(w), \widetilde{p}_i(u)]$ is equilateral. We do a search on the segment for the position that would minimize the area-to-perimeter distortion for the triangle, while keeping distortion of all other triangles lower, and respecting the single-ring condition for surrounding triangles (Figure 5.13).

5.6 Fitting

The previous stages of the algorithm yield the domain $\widetilde{M}$ for the resulting surface and parameterizations $\widetilde{p}_i$ of all vertices in subdomains $\widetilde{M}_i$ $i = 1, 2$ over the domains of the original surfaces. Furthermore, these parameterizations have the single-ring property. This allows us to compute the positions in M_i not only for top-level vertices of $\widetilde{M}$, but also for any vertex added to $\widetilde{M}$ by subdivision. For this we only need to be able to assign parametric coordinates to the new vertices, which we do using linear charts as in Section 5.5.

However, no geometry is computed for the resulting surface. We fix the parameterizations and regard the new surface and parts of the old surfaces as functions on the newly constructed domain (Figure 5.16). Our goal is to compute the positions of control points for an approximation to the result, avoiding introducing details on fine levels. This is achieved by fitting surfaces defined by the control points to the original surfaces. The fitting procedure can be performed adaptively, increasing resolution where necessary. We consider the simplest version: we fit a subdivision surface with the control mesh obtained by subdividing $\widetilde{M}$ m times, to the original surface. We perform the fit in a hierarchical top-down manner, to obtain a multiresolution representation in the process.

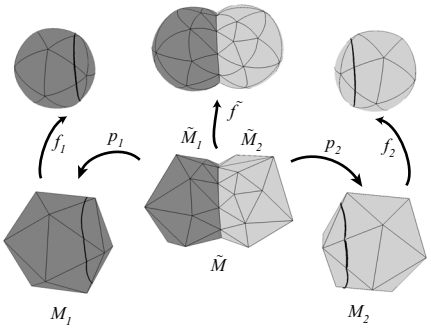


Figure 5.16: Surface fitting: We minimize the difference between the new and original surfaces.

Observe that we can evaluate the old surface f_i at any vertex at any subdivision level of the result domain $\widetilde{M}$ using the composition $f_i \circ \widetilde{p}_i$. We use a generalization of Stam's technique [96] to piecewise smooth subdivision surfaces to evaluate f_i at arbitrary points of the domains M_i, $i = 1, 2$.

Let V_m be the set of vertices of the $\widetilde{M}$ after m subdivision steps. Then the difference between two surfaces can be measured by the following functional:

$$\sum_{i=1,2} \int_{M_i} \| f_i(\widetilde{p}_i(\alpha)) - \sum_{v \in V_m} p_v^m B_v^m(\alpha) \|^2 d\alpha$$

where p_v are the control points for the resulting surface on subdivision level m and $B_v^m(\alpha)$ are the basis functions at vertices of level m. This functional is quadratic in p_v^m. The integrals can be computed explicitly, but we found that the results are not substantially different from replacing the continuous integrals with differences of control points n levels below the level being fitted (we use $n = 3$). As a result, (5.6) is replaced with a different functional:

$$\sum_{i=1,2} \sum_{v \in V^{m+n}} \| f_i(\widetilde{p}_i(v)) - \sum_{v \in V_{m+n}} [S^n p^m]_v \|^2$$

where $[S^n p^m]_v$ is the control point at vertex v obtained as a result of subdividing control mesh p^m n times. In vector notation, the expression above can be written as $\sum_{i=1,2} \sum_{v \in V^{m+n}} \|q^{n+m} - S^n p^m\|^2$, with $q_v = f_i(\widetilde{p}_i(v))$.

It is possible to show that the relative difference between the continuous and discrete functionals is bounded and derive accurate bounds using estimates on the magnitude of subdivision basis functions. The discrete form is a standard least-squares fit problem, which can be solved by a number of efficient methods (e.g. conjugate gradient). However, we have found that visually better results are obtained by imposing additional constraints on movement of the control points: on level m we allow the points to move only in normal direction to the surface constructed by the fit on level $m - 1$. In this way, we obtain a mesh similar to the normal mesh of [31]. While the accuracy of the fit in the mean square sense decreases, the visual surface quality improves, as this approach prevents forming folds and ripples. At this time, we have no formal justification for imposing such constraints. It should be noted that the area that needs to be fitted grows when fitting finer levels of the multi-resolution mesh. This area grows by the size of the subdivision mask on the previous level, but this is not a large concern since it only adds a layer of vertices around the perimeter while the number of vertices internal to the optimized area grows exponentially at each subdivision level.

If high-accuracy approximation of the result is desired, once a good approximation is achieved by the fit, we switch to *quasi-interpolation* to compute further details on the surface ([59]). This is done solely for efficiency.

A careful examination of (5.6) reveals that it is possible to optimize not only the positions of the control points p^m, but also the parameterizations $\widetilde{p}$ to obtain a better approximation; our optimization of the parametric maps described in Section 5.5 tends to work in this direction. However, we make no attempt to optimize (5.6) directly; this is a possible direction for future research.

5.7 Results

We have tested our algorithm on various closed multiresolution surfaces. Figure 5.2 shows all the operations possible with two objects A and B. Figure 5.22 shows the coarsest level triangulation. Note the low valence of the new vertices near the curve. Figure 5.1 also used all three boolean operations in its construction. The remaining figures show useful operations possible with boolean operations on free-form solids.

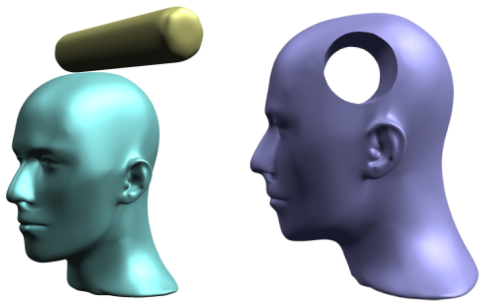


Figure 5.17: Subtracting a cylinder from the mannequin head.

5.8 Conclusion and Future Work

While our work addresses a classical problem in geometric modeling, our emphasis is quite different from most of the work we are familiar with. The algorithms

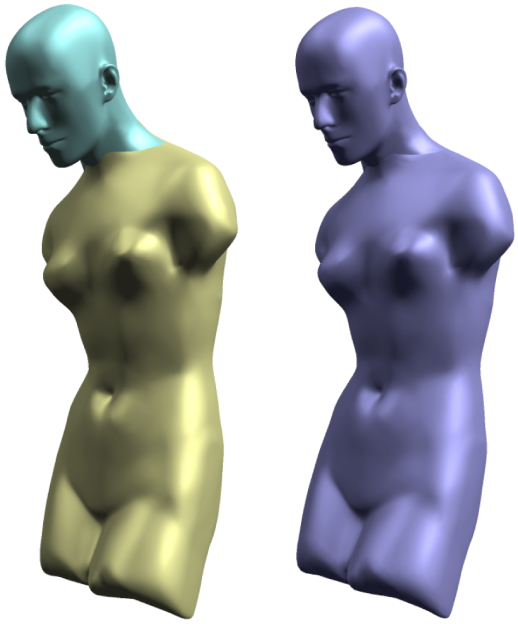


Figure 5.18: Union of the head and the body. Left: Original solids. Right: New solid obtained by union.

that we have developed primarily address the problem of constructing a valid and usable model for the result of the boolean operation, rather than computing precisely all geometric objects characterizing the result (i.e. the intersection curve and parametric images of the intersection curve in the domains of the objects). Thus our algorithms can be viewed as complimentary to work on surface-surface intersections. Any accurate algorithm can be used to compute the intersection curve instead of our approximate algorithm. As future work, we plan to explore integration of precise surface-surface intersection algorithms into our framework. The algorithms described in Section 5.5 typically improve parameterizations. However, even defining rigorously measures of quality of the parameterization of one surface over another requires additional research. In Euclidean domains efficient techniques such as multigrid dramatically accelerate convergence of linear methods such as Laplacian smoothing. It is unclear

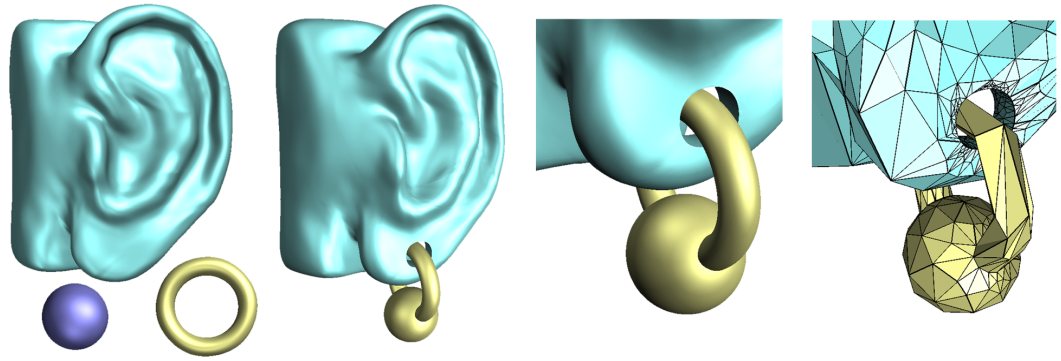


Figure 5.19: Modeling with multiresolution surfaces. The earring is assembled from a sphere and a torus. The ear is pierced with an enlarged version of the torus. The ear and the pierced ear are represented as multiresolution surfaces.

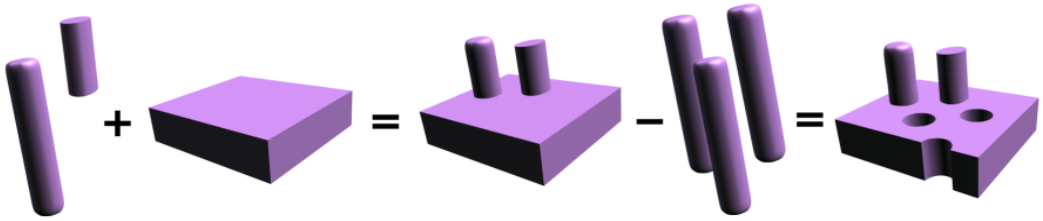


Figure 5.20: Unions and differences of piecewise-smooth surfaces. The resulting surfaces have creases and corners.

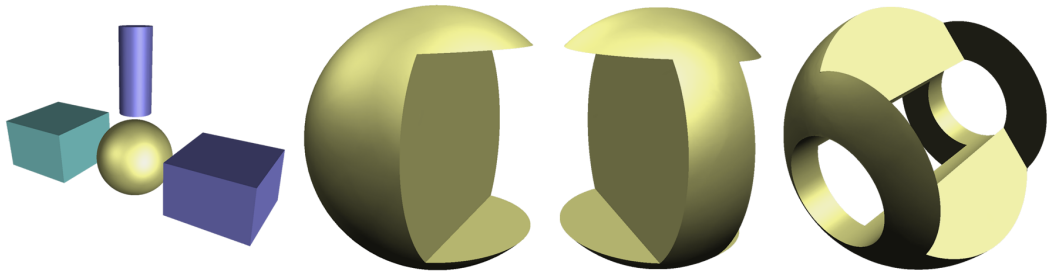


Figure 5.21: Sequence of difference operations: Subtracting two boxes and a cylinder from a sphere. The result has convex and concave corners. The subdivision scheme that we use [6] represents these features explicitely.

how to apply similar techniques to functions with values in parametric domains.

Boolean operations on meshes with significantly different complexity can result in high valence vertices on the resulting mesh. Any algorithm that significantly increases the number of regular vertices would result in a better surface.

It appears that we are able to approximate the results of boolean operations arbitrarily well, assuming that the topology of the intersection curve was resolved correctly. However, there is no guarantee that this is the case, and our algorithms require further analysis.

Our current implementation is not optimized for speed; the time required for operations is typically short: from real time to about five seconds for objects with larger control meshes such as the head/cylinder difference. We believe that for simple objects, boolean operations can be performed instantaneously.

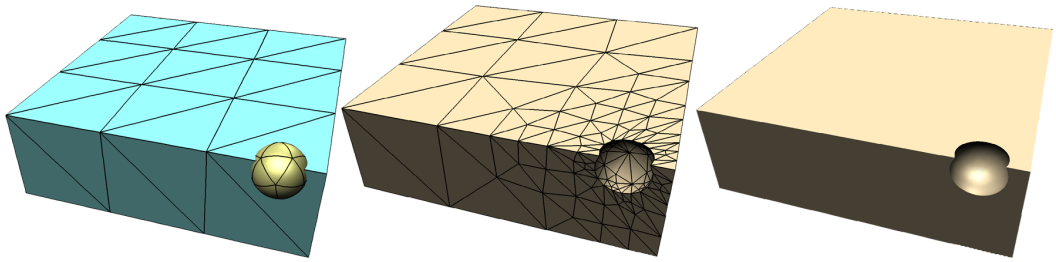


Figure 5.22: Difference of objects of different scale. Left: input surfaces. The patches of the box are much larger than the ones of the sphere. Middle: control mesh for the difference. The patch size changes gradually. Right: resulting surface.

Chapter 6

Cut-and-Paste Editing of Multiresolution Surfaces

The contents of this chapter were published in the SIGGRAPH 2002 Conference Proceeedings [7]. It describes a set of algorithms based on multiresolution subdivision surfaces that perform at interactive rates and enable intuitive cut-and-paste operations. The algorithms allow one to select an arbitrary geometric feature on one object and transfer it to another object.

6.1 Introduction

Pasting and blending of images are among the most common operations implemented by image manipulation systems. Such operations are a natural way to build complex images out of individual pieces coming from different sources. For example, photographs can be easily combined with hand-drawn and computer-generated images. In contrast, pasting and blending tools are hardly available for surfaces. Most geometric modeling systems expect the user to manipu-

late control points of NURBS, individual mesh vertices and polygons, or use conventional, higher-level operations such as volume deformations and boolean operations. In an image processing system, vertex and control point manipulation would be equivalent to painting an image pixel-by-pixel. While it may be useful to have access to such low-level operations in certain cases, most image manipulations are done using higher-level tools.

In this paper we describe a technique for interactive cut-and-paste editing of surfaces, an important instance of a natural operation on a surface (see Figure 6.1 for an example). The algorithms we propose enable a number of useful design scenarios which are difficult to perform using existing technology. For example, in the design of automobile body parts, it is common to work in parallel on a digital mock-up and on a clay model. Using the cut-and-paste technique, a designer can paste a logo obtained by 3D scanning onto a digitally-modeled surface, import features from a library of predefined shapes, or copy parts of a design from a different project.

The basic idea of pasting is quite simple. The user selects an area of interest on the source surface. Both the source and the target surfaces are separated into *base* and *detail*, such that the detail surface represents a vector offset over the base surface. Next, the user specifies a location and an orientation on the target surface where the source feature is to be pasted and interactively adjusts the position, orientation, and size of the pasted feature. The main questions we address in this paper are:

- How to separate a surface into base and detail ?
- How to identify an area on the target surface where the feature should be pasted and how to establish the necessary mappings between the source

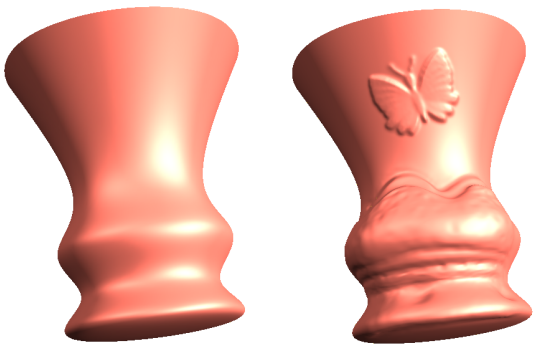


Figure 6.1: An example of a pasting operation.

and the target ?

- How to implement the process efficiently to allow for interactive pasting of complex features ?

We use multiresolution subdivision surfaces as our underlying representation [61, 107]. The actual computer representation is a semi-regular control mesh for the surface and most operations are performed on this mesh. The associated limit surface is used for computing quantities such as tangents and normals, as well as for additional refinement when necessary for antialiasing. This is similar to pixel representations of images: when images are scaled or rotated, they are typically assumed to be sampled representations of smoothly varying continuous images (e.g., obtained by cubic interpolation).

The regular and hierarchical structure of this surface representation makes it possible to perform operations on detailed surfaces at interactive rates as discussed in Section 5.2. In many ways, using this representation makes surface manipulation similar to image manipulation: almost everywhere the connec-

tivity of the mesh approximating the surface is locally regular. At the same time, many common problems specific to geometry have to be addressed: the lack of a common parameterization domain for separate surfaces, the lack of a unique best parameterization domain for the surfaces, the separation of surface features. While our algorithms can be applied to a broad class of surfaces, there are limitations on the source and target geometry for which the pasting paradigm is appropriate (see Section 6.9 for a discussion).

6.2 Previous Work

The concept of surface pasting was introduced in the work of Bartels, Mann and co-workers in the context of hierarchical splines [3, 13, 16, 62, 101]. We are using similar ideas, but most of the technical details are different. Most importantly, we consider more general surface types and we do not assume that separate detail and base surfaces are given: they need to be extracted from the input surfaces.

Moving existing features on a mesh was explored by Suzuki et al. [99]. The advantage of their approach is that no resampling of the repositioned feature is performed. However, continuous remeshing is required, which limits the complexity of the objects and features that can be handled. The issues of pasting features between surfaces and the separation into base and detail surfaces are not considered by these authors.

The task of base/detail separation is similar to the construction of displaced subdivision surfaces [48]. One of the elements of our approach, i.e., mesh smoothing to extract a base surface, was described by Kobbelt et al. [42] in the more general context of arbitrary meshes. An alternative approach was

proposed by Guskov et al.[30]. We discuss the advantages and disadvantage of restricting the class of surfaces to semiregular meshes in Section 6.3.2. Part of our construction of base surfaces is closely related to the work of Kobbelt et al. on variational subdivision [43, 41]. It also draws upon the work of Polthier et al. [78, 73].

Parameterization techniques are important in many geometric modeling and texturing applications and a variety of algorithms have been proposed, including general parameterization methods [26, 27] for reparameterization (i.e., changing connectivity to semi-regular) [23, 31, 45, 49] and texture mapping [55, 80, 63]. In [47], Kuriyama and Koneko use local parameterizations to add offsets to a surface. The work of Pedersen [74, 75] on interactively placing textures on implicit surfaces is also relevant as it requires dynamic reparameterization of surface areas similar to pasting.

However, the problem of parameterizing a surface area over a plane with minimal visual distortion is far from solved. As explained in Section 6.6, until recently no algorithms combining several crucial properties for our application were available. We use a variation of the remarkable algorithm by Sheffer and Sturler [93] which satisfies our requirements.

6.3 Pasting Surfaces

We begin with a formalized description of pasting operations on surfaces. At this point we discuss continuous surfaces and mappings without considering their discrete representations. This framework applies to a wide class of manifold surfaces, ranging from splines to implicit surfaces. Precise descriptions of all basic mathematical concepts that we use can be found in any standard textbook

(e.g., [102]).

6.3.1 Formulation of the Problem

For simplicity, we restrict our attention to parts of surfaces parameterized over planar domains: $M \subset \mathbf{R}^2$. Furthermore, we assume that these parameterizations are sufficiently smooth. Given two surfaces $(M_1, \mathbf{f}_1)$ and $(M_2, \mathbf{f}_2)$, where $\mathbf{f}_1$ and $\mathbf{f}_2$ are their parameterizations, we would like to paste a feature from one surface to the other (see Figure 6.1). Such an operation requires separating each surface into two parts: the *base* surface and the *detail* surface. The goal is to replace the detail part of the second surface with the detail part of the first. The key question is how to transfer correctly the details from one surface to the other.

Base and detail surfaces. The base surface $\mathbf{b}(x)$ is typically a smoothed or flattened version of the original surface (we discuss appropriate choices in Section 6.5). The detail surface $\mathbf{d}(x)$ can be defined as $\mathbf{f}(x) - \mathbf{b}(x)$. However, to ensure that the offset direction is at least invariant with respect to rigid transformations of the base, it must be represented in a local frame. The local frame is a triple of vectors $(\mathbf{n_b}, \partial_1\mathbf{b}, \partial_2\mathbf{b})$, including the normal and two tangents (two partial derivatives of the parameterization). It is convenient to think about these derivatives together as a map $D\mathbf{b}$ (differential of $\mathbf{b}$) which maps vectors in the plane to vectors in the tangent plane of the surface. The detail surface is thus defined by the triple d^n, d^{t1}, d^{t2}, which can also be thought of as a scalar displacement along the normal d_n and a tangential displacement in parametric coordinates $\mathbf{d}^t = (d^{t1}, d^{t2})$. The equation relating the original surface, the base, and the details can be written as: $\mathbf{f}(x) = \mathbf{b}(x) + D\mathbf{b}(x)\mathbf{d}^t(x) + \mathbf{n_b}(x)d^n(x)$,

where x is a point in the domain.

Surface pasting. With both surfaces separated into base and detail parts, we can formulate a precise definition of pasting. All quantities with index 1 refer to the source surface from which we extract the details and all quantities with index 2 refer to the target surface on which the details are pasted.

Suppose the part of the surface we want to paste is defined over $G_1 \subset M_1$. Let p be a map from G_1 to M_2, which defines how the surface is pasted. We discuss separately how p is chosen (Section 6.6).

The result of a simple pasting operation is a new surface coinciding with $\mathbf{f}_2$ outside $p(G_1)$, which has the same base as $\mathbf{f}_2$ but for which the details are taken from the source surface:

$$\mathbf{f}^{pasted} = \mathbf{b}_2 + \left(D\mathbf{b}_2 \, Dp \, \mathbf{d}_1^t + n_{\mathbf{b}_2} d_1^n \right) \circ p^{-1}$$

where all functions are evaluated at a point $x_2 \in p(G_1)$, and $\circ$ denotes function composition.

Note that we use the composition of differentials $D\mathbf{b}_2 \circ Dp$ to transform the tangential component of details. This establishes the natural map between the local frames on the source and target surfaces. Figure 6.2 illustrates the different maps involved.

Using this formulation, there are two main choices to be made: the separation of both source and target surfaces into base and detail and the definition of a pasting mapping p, identifying the domain G_1 with a part of the domain M_2.

The map p has to satisfy two conditions: it has to be one-to-one and it should minimize distortion of the mapped feature. An important consideration

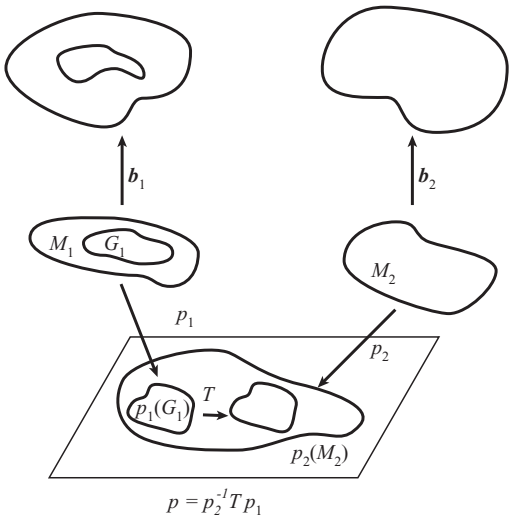


Figure 6.2: A diagram of surface maps involved in pasting.

is whether the mapping p from the domain of one surface to the domain of the other surface is constructed directly or by using an intermediate planar domain. We favor the latter approach as it considerably simplifies three tasks: making sure that the mapping is visually smooth, minimizing distortion, and resampling the source over the target sampling pattern. To explain our choices, we need to be more specific about the surface representation we are using.

6.3.2 Multiresolution Subdivision Surfaces

The representation that we use was introduced in various forms in [61, 84, 108]. Subdivision defines a smooth surface recursively as the limit of a sequence of meshes.[1] Each finer mesh is obtained from a coarse mesh by using a set of fixed refinement rules, e.g., Loop [60] or Catmull-Clark [11] subdivision rules. In our implementation we use Catmull-Clark rules. Multiresolution surfaces

[1]To be more accurate, we should say that the limit surface is the pointwise limit of a sequence of piecewise linear functions defined on the initial control mesh.

extend subdivision surfaces by introducing *details* at each level. Each time a finer mesh is computed, it is obtained by adding detail offsets to the subdivided coarse mesh. If we are given a *semi-regular mesh*, i.e., a mesh with subdivision connectivity, we can easily convert it to a multiresolution surface if we define a smoothing operation to compute vertices on a coarse level from a finer level. The details are then computed as differences between levels (see Section 6.5).

An aspect of multiresolution surfaces important for modification operations is that details are represented in local coordinate frames, which are computed from the coarser level. This is analogous to representing the detail surface in the frame computed from the base surface.

For our purposes, it is important to interpret the multiresolution surface as a function on a domain. A multiresolution surface can be naturally viewed as a function on the initial mesh as shown in Figure 6.3.

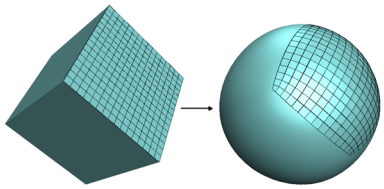


Figure 6.3: Natural parameterization of the subdivision surface. Each time we apply the subdivision rules to compute the finer control mesh we also apply midpoint subdivision to a copy of the initial control mesh. As we repeatedly subdivide, we get a mapping from a denser and denser subset of the control polygon to the control points of a finer and finer control mesh. In the limit we get a map from the control polygon to the surface.

Advantages and disadvantages of the representation. The main reason behind our choice of representation is efficiency. There are a number of reasons why semiregular meshes allow for highly efficient algorithms:

- Connectivity information only needs to be stored for the coarsest level. Geometric data is stored in a regular and space-coherent manner. Both factors are important for computer architectures for which bad cache behavior results in poor performance. In addition, regularly sampled patches can be rendered very efficiently.

- Our meshes have a built-in natural hierarchy that can be exploited by numerical solvers. These are used, for example, to define a family of smooth surfaces for base surface selection by hierarchical fitting and for parameterization, when an initial approximation of the fine level solution can be obtained by solving the system on a coarse level, followed by refining the solution by subdivision.

- Compact representation for smooth surfaces: for example, the initial vase in Figure 6.1 is completely defined by the initial mesh, and the smooth surface can be recomputed on the fly. When details are added, additional refinment is needed only in the region with details.

- Local frames can be computed in a simple, fast, and reliable way, consistently across resolution levels (i.e., refining the mesh for the base surface does not change the frames computed for the surface at existing vertices).

Our experiments with solvers that take advantage of the regular structure and generic solvers that use a sparse matrix representation suitable for arbitrary meshes show that the former yield speedups by a factor of 2 to 4. Furthermore,

using a hierarchical solver for an arbitrary mesh would require building a hierarchy by simplification, a step entirely omitted in our construction.

The main disadvantage of representing surfaces using semiregular meshes is having to convert surfaces represented by arbitrary meshes to this format. Fortunately, considerably progress has been made in this area [45, 49, 31] and commercial software (e.g., Paraform, Raindrop Geomagic) typically includes such conversion tools. All of the scanned models used in this paper were converted to semiregular meshes using Raindrop Geomagic. We believe that, in all cases when surface data is extensively modified, conversion is the best approach, as reparameterization is almost inevitable if the surface is texture-mapped. Hoppe et al. provide a detailed study of the benefits of a conversion to a similar representation (i.e., the geometric image).

6.3.3 Pasting with an Intermediate Plane

A direct construction of the pasting mapping p is difficult to make efficient. Visual smoothness and minimization of distortion are typically achieved by minimizing appropriate functionals. In the case of a direct mapping of the source region to the target surface domain, the values of the mapping are not a part of any affine space. Indeed the domain of the surface is a collection of faces of the coarse-level control mesh, so each point needs to be characterized as (i, u, v) where i is the face id, and (u, v) are coordinates within the face. Unless the whole surface can be reparameterized on a plane, there is no simple way to compute linear combinations of two arbitrary points (e.g., the midpoint of the interval connecting the points), which makes the application of most common computational techniques very difficult. Even a simple operation such

as computing angles of a triangle given three vertices becomes a complicated task, an important consideration for the angle-based flattening technique we consider.

To avoid these difficulties, we parameterize the corresponding areas of the source and target over the plane. The idea is to map each surface onto the plane as isometrically as possible and then align the two planar parameterizations, using a linear transformation to compensate for the first-order distortion. In this case, the pasting map is restricted to a simple class of maps (i.e., linear transformations), but new parameterizations p^1 and p^2 are constructed for the parts of surfaces of interest for every pasting operation.

There is a similarity between the idea of our approach and the method of Praun et al. [81] for establishing correspondences between different meshes. In [81] the correspondence is established by reparameterizing each mesh on the same base domain. Given our disk topology assumption, we can use the plane as the common domain.

Our approach has two main disadvantages. First, it makes it difficult to generalize our technique to pasting regions with topology different from that of a subset of a plane (e.g., pasting all details from one sphere to another). Second, it may result in higher distortion than a direct mapping from one surface to the other. The higher is the Gaussian curvature of the base surface, the more likely it is that additional distortion is introduced. A direct mapping method similar to the one used in [5] might produce better results in this case, but it would make pasting of complex surfaces at interactive rates difficult, if at all possible.

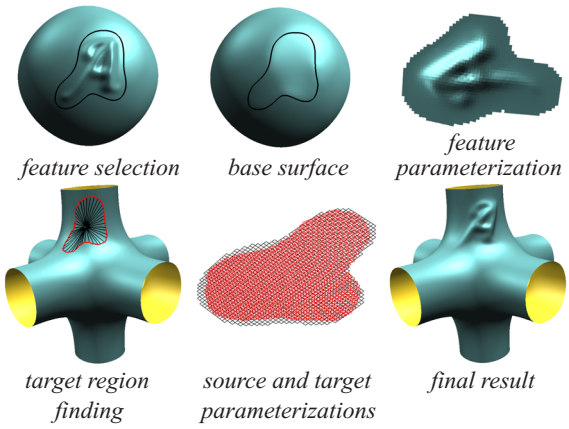


Figure 6.4: Main steps of the pasting algorithm, row-wise from top left: (a) selected feature on the source surface, (b) base source surface, (c) source parameterization onto the plane, (d) target region finding by geodesic walking, (e) source and target parameterizations superimposed in the plane, (f) source feature pasted onto the target surface.

6.4 Overview of the Algorithm

The main steps of our algorithm are illustrated in Figure 6.4:

1. The user marks a region on the source surface and optionally specifies a *spine*. A spine of a region is a collection of curves which capture the general shape of the region. It approximates the medial axis of the region and it can be used by the system for mapping the source to the target (see also Figure 6.8).

2. The details are separated from the base for the source surface. The user interactively selects a base surface from a continuous range interpolating between a zero level given by a membrane surface and the actual surface (Section 6.5).

3. The source region is parameterized over the plane (Section 6.6).

4. The boundary of the source region is parameterized by distance and direction from the spine and a covering by disks is computed.

5. The user positions at least one point of the spine on the target, and specifies an orientation.

6. A target region for pasting is determined on the surface using geodesic disks (Section 6.7).

7. The target area is mapped to a common plane with the source and the source is resampled over the target sampling pattern (Section 6.8).

8. The resulting surface is computed by blending the target base surface, the source surface resampled details, and the target details. The user can

specify different blending modes.

6.5 Separating Base Surface from Detail

An important step in the pasting process is the definition of which features of the source surface region constitute details that the user wants to paste over the target surface, as opposed to the larger-scale surface shape that should be ignored. Separating the base surface from the details depends on the semantics of the operation and has to be user-guided. For example, one may want to extract only the texture-like geometry of the skin on the nose of a head model, or to paste the entire nose onto a different model. Different choices for base-detail separation result in different pasting effects (Figure 6.6).

Our approach is to provide a continuum of base surface choices guided by a single parameter which can be thought of as the flatness of the base surface. A natural way to obtain a smooth base surface given our multiresolution data representation is to remove or reduce the multiresolution details present in the multiresolution hierarchy on the finer levels. The degree to which this approach works depends on the way the coarser levels were obtained when the hierarchy was constructed. By comparing several approaches (Taubin's smoothing, quasi-interpolation, and fitting), we found that fitting works best for pasting.

Least-Squares Fitting. The fitting procedure minimizes a functional that measures how well the smooth surface fits the vertices of the original mesh subdivided to the finest level M. While fitting of subdivision surfaces is not new (e.g., [58]), there appears to be no detailed description of it in the literature and we present it here for completeness. The minimization problem for level m

of the smooth surface hierarchy can be stated as:

$$\min_{p} \sum_{w \in V^M} ||p_w^M - [S^{M-m} p^m]_w||^2 \tag{6.1}$$

where the minimum is computed over all possible choices of control points p^m for the smooth mesh, V^M is the set of vertices of the finest-level mesh, p^M are the corresponding control points, S^{M-m} is the subdivision matrix for $M-m$ subdivision steps, and $[]_w$ means that the resulting smooth surface is evaluated at parameter values corresponding to vertices w of the control mesh. The minimization problem is equivalent to finding solutions for the linear system $A^T Ax = A^T b$, with $A = S^{M-m}$, $b = p^M$ and $x = p^m$, and can be solved by using the Conjugate Gradient method. To apply this method, the only operations needed aside from linear combinations of vectors and dot products, are matrix-vector multiplications for the S^{M-m} matrix and its transpose. As the matrix is obtained by iterative application of the subdivision matrix, there i sno need to represent or store it explicitly: applying A corresponds to the application of $M-m$ subdivision steps. Applying A^T to a vector can be interpreted in similar terms. More specifically, as shown in Figure 6.5, the mask for each vertex v on level $m-1$ contains all vertices on level m which are affected by v when subdivision is performed. If vertex v has coefficient α in the subdivision rule used to compute the control point for vertex w, then vertex w has coefficient α in the transpose averaging rule for v.

Once the sequence of levels is computed, a continuum of base surfaces can be obtained by interpolation as shown in Figure 6.6. The user can select one interactively by moving a slider.

An alternative approach to fitting is to use the quasi-interpolation approach

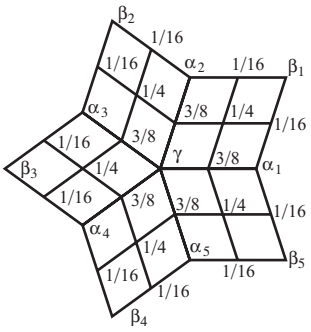


Figure 6.5: The mask for local averaging used for transpose subdivision. Coefficients α_i and β_i are the Catmull-Clark vertex rule coefficients for corresponding vertices. $\gamma = 1 - n\alpha - n\beta$, where n is the valence of the central vertex and α and β are the vertex rule coefficients.

of Litke et al.[58]. This approach is somewhat faster, but results in larger errors. We use the more accurate fitting approach as it does not constitute a bottleneck in our system.

Boundary constraints. The technique previously described explains how to produce smoother approximations of the surface globally. This approach is quite fast as the base surfaces on different levels can be precomputed and only interpolation is required after that. However, when we separate the feature from the surface, we need a base surface only near the feature. Even more importantly, in most cases the details should gradually decay in magnitude as we approach the boundary of the feature. To adapt the global base surfaces to our needs, we use the following simple blending approach: the local base surface is computed as a blend of the source surface and a global base surface. The region in the interior of the feature is assigned alpha values 0 and all vertices outside the region are given values 1. Next, relaxation is applied for values in

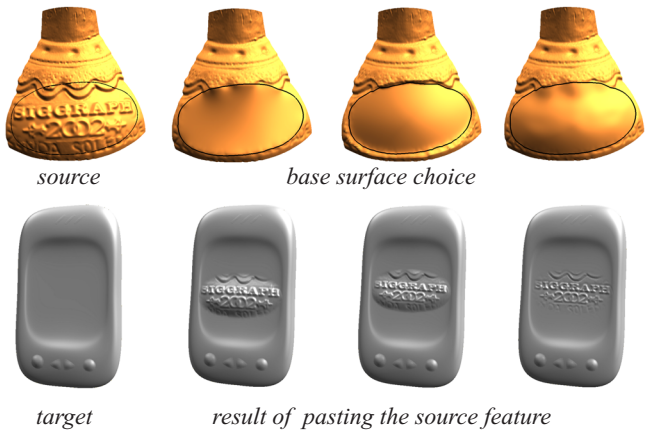


Figure 6.6: Depending on the choice of base surface, different scales of shape details are transferred to the target.

the interior while keeping the values outside constant. The amount of relaxation is user-controllable and allows to change the way features blend into the target. The resulting alpha values are used to interpolate between the global base and the source surface (Figure 6.10).

Minimal base surface. Base surfaces defined by fitting and blending cannot be flatter in the area of the feature than the base surface obtained by fitting on the coarsest level. This might be not appropriate for some applications where it is necessary to retain more of the feature shape (e.g., Figure 6.13). In such cases, it is best to define the base surface as a smooth, relatively flat surface that fills the hole remaining after the feature is cut off. To obtain such a surface, we optimize the membrane energy of the surface inside the feature curve while constraining its boundary to remain fixed. We use a multigrid-type approach [43], which is a natural choice in the context of our multiresolution representation. Similarly, the transition between the feature and the base is

handled by assigning alpha values. This allows us to extend the range of possible base surfaces beyond the coarsest-level fitted surface. As a result the user has a choice of base surfaces varying from the minimal surface spanning the outline of the feature, all the way to the source surface.

6.6 Parameterization

Once we have separated the details from the base surface, we need to find a map from the source base surface to the target, to be able to transfer the details. As it was discussed in Section 6.3.3, we construct the map in two steps. First, we map the source surface to the plane. Second, we determine the region on the target surface where the feature will be pasted and we parameterize it onto the same plane.

Parameterization is needed both for the source and target base surfaces. The type of surface patches that we need to parameterize is relatively uncommon: while the surface is likely to be quite smooth, the shape of the patch can be relatively complex. The parameterization we construct should satisfy the following requirements:

- The parameterization region should not be chosen a priori. The need for this can be seen from the following simple example: the outline of a feature selected on the plane base surface can be arbitrarily complex, however the parameterization should not be different from the surface itself. Any algorithm that requires a fixed domain is not likely to perform well in this situation.

- The parameterization should be guaranteed to be one-to-one. As we need

to resample, for each vertex on the target we need to identify a unique position on the source. This means that at least the map from the source to the plane has to be one-to-one.

- The parameterization should minimize a reasonable measure of distortion. Ideally, for developable surfaces it should be an isometry up to a scale factor. The algorithm that we use does not explicitly minimize a measure, but it appears to produce results with close to minimal shape distortion, as discussed below. It tends to produce better results than all other algorithms that we have tried in situations relevant for us.

Most of the existing parameterization algorithms do not determine the domain automatically; it is either determined using a heuristic approach or it has to be prescribed by the user. The parameterization described in [80] allows for free boundary evolution, but it requires a vector field defined over the surface and it does not provide a one-to-one guarantee. Until recently, the algorithms that guaranteed a one-to-one parameterization required convex domains, like the many variations of Floater's algorithm ([26]). We use the algorithm of Sheffer and Sturler [93] which meets our requirements most closely.

Angle-based flattening. For a mesh, a parameterization is defined by specifying the positions (parametric coordinates) of all vertices of the mesh in the plane. Without the loss of generality, we can assume a triangular mesh (we use quad subdivision surfaces, but each quad can be easily split into two triangles). The idea of angle-based flattening is to compute the parametric coordinates of the vertices indirectly: first, all angles are computed using an optimization procedure, then the planar mesh is reconstructed by fixing the length of one of the

edges. The reason for computing the angles rather than vertex positions directly is that the one-to-one condition can be easily enforced and aspect ratios can be controlled explicitly. The disadvantage is that the reconstruction procedure is relatively unstable, as positions of vertices depend sequentially on each other. However, we found that for the relatively small numbers of triangles that we use (at most thousands), this is never a problem.

Next we describe the formulation of the optimization problem for angles mostly following [93]. Let t denote a triangle of the mesh, T the set of all triangles; let v be a vertex and V the set of all vertices in mesh. If v is a vertex of t then the angle α_t^v is the corresponding angle in the triangle t. Let $N(v)$ be the set of all triangles sharing a vertex v. The target value for angle α_t^v is defined as follows: $\varphi_t^v = 2\pi\alpha_s^v / \sum_{s\in N(v)} \alpha_s^v$, i.e., ideally all angles at a vertex should be rescaled by the same amount, so that their sum is equal to 2π.

The resulting functional is

$$F(\alpha) = \sum_{t,v\in t} w_t^v (\alpha_t^v - \varphi_t^v)^2,$$

where the weights are chosen to be $1/(\varphi_t^v)^2$.

Minimization of this functional needs to be constrained for the results to correspond to a valid planar triangulation. The necessary constraints are as follows: (a) the angles should stay above some minimal value ϵ; (b) the sum of all angles at a vertex should be 2π; (c) the sum of all angles of each triangle should be π; (d) each 1-neighborhood of a vertex should be consistent. This means that if we reconstruct a neighborhood triangle-by-triangle going around the vertex, the last edge of the last triangle should coincide with the first edge of the first triangle as shown in Figure 6.7.

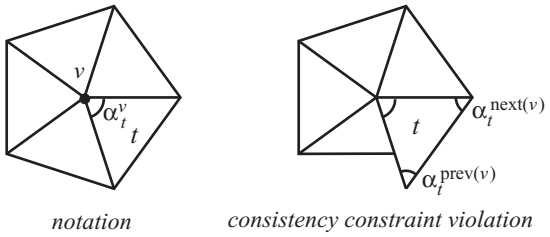


Figure 6.7: The image on the right shows the situation prevented by the constraint $g_4(t)$ in the parameterization algorithm.

The mathematical expressions for these four constraints are:

$$g_1(v,t) = \alpha_t^v - \epsilon \geq 0; \quad g_2(v) = \sum_{t \in N(v)} \alpha_t^v - 2\pi = 0;$$

$$g_3(t) = \sum_{v \in t} \alpha_t^v - \pi = 0$$

$$g_4(v) = \quad \Pi_{t \in N(v)} \frac{\sin \alpha_t^{\text{prev}(t)}}{\sin \alpha_t^{\text{next}(v)}} - 1 = 0.$$

The inequality constraints are best enforced in an iterative procedure for minimizing the functional by rejecting values that violate the constraints. The equality constraints are included into the functional by means of Lagrange multipliers: $L(\alpha) = F(\alpha) + \sum_v \lambda(v) g_2(v) + \sum_t \mu(t) g_3(v) + \sum_v g_4(t)$.

This is a nonlinear optimization problem which is solved using Newton's method: $x_{n+1} = x_n - H(L)^{-1} \nabla L$, where ∇L and $H(L)$ are the gradient and the hessian of L, and x is the vector of all angles and Lagrange multipliers. At each step, we need to solve a linear system to invert $H(L)$. In [93] a direct solver is used. We observe that the system does not change much from one iteration of the Newton method to the next. Furthermore, for smooth surfaces it is likely that the initial guess for the angles is quite close to the solution (e.g. for the developable surface it is already a solution). This indicates that iterative

solvers are likely to perform quite well. Although the system is symmetric, it is not positive definite, and the Conjugate Gradient method cannot be used. However, the Conjugate Residuals method applies. For fast convergence rates, preconditioning is required, i.e., an approximate sparse inverse of the matrix needs to be computed. To avoid an expensive preconditioner computation, we add a small negative constant equal to the inverse of the number of triangles on the diagonal which makes it possible to avoid pivoting when calculating an incomplete factorization (ILU) preconditioner. Our experience was that a small number of iterations of the solver were sufficient to obtain a reasonable parameterization.

It should be noted that, while the constraints guarantee that the resulting mesh is one-to-one locally (no flipped triangles), the boundary of the image may self-intersect and globally the map is still not one-to-one. A technique for eliminating such self intersections is described in [93].

6.7 Determining a Target Region

Before pasting can be performed, an area on the target surface corresponding to the feature has to be identified and parameterized. It is a chicken-and-egg problem: to determine the region covered by the pasted feature, we need to map it to the target; however, mapping the feature to the target requires parameterizing the corresponding part of the target surface over the plane. As parameterizing the whole target is generally not an option, we use the following approach: we observe that initially we need to identify only an approximate boundary region where the feature will fit, rather than to establish a one-to-one mapping of the interior. Once the region is identified, it can be parameterized over the plane

and a mapping is computed as the composition of the two parameterizations.

The algorithm that we use for identifying the region proceeds in several steps: first, we represent the boundary of the source region in a generalized radial form, constructing line segments (planar geodesics) connecting the spine to the boundary. Then we map the one-dimensional spine to the target, and use geodesics on the target to map the boundary points to the target. Finally, we connect the points on the target and fill in the interior region (Figure 6.8). The computation of geodesics passing through a point is a central tool in the algorithm and is discussed in greater detail.

Parameterizing the source boundary. The user has the option to draw a curve on the surface, possibly with several branches, which serves as the spine of the feature. It is our main intention to help the system map the feature to the target surface with the least distortion. If the user does not define a spine, a single point (the centroid of the boundary of the parameterization) is automatically selected to serve as the spine.

The following algorithm is used to parameterize the source boundary. First, the spine is mapped to a curve in the plane by the parameterization. Let $c_0, \ldots c_{m-1}$ be equispaced points on the spine in the parametric domain. The number of points can be adjusted to trade speed for quality.

For each vertex w_j on the boundary of the source parameterization find the closest point c_i. Let n_i be the number of points closest to the point c_i, d_j be the distance from c_i to w_j and γ_j be the angle between the direction from c_i to w_i and the spine. If the spine consists of a single point, an arbitrary fixed direction is used as the direction of the spine.

The boundary of the source region can be characterized by the set of triples

(c_i, d_j, γ_j), where $i = 0 \ldots m-1$, and $j = 0 \ldots n_i - 1$. This collection of triples can be thought of as a discrete parameterization of the boundary with respect to the spine generalizing the radial parameterization. In the case of a single-point spine, this is just the radial parameterization.

Mapping the spine to the target. Mapping the spine to the target is straightforward: the user specifies an initial position and orientation for a point on the spine. The other points on the spine are obtained sequentially by walking as follows. Suppose the positions $T(c_0) \ldots T(c_i)$ are known. If the angle between the intervals (c_{i-1}, c_i) and (c_i, c_{i+1}) is β_i, then the next point on the spine is obtained by walking on the target surface at an angle β_i to the previous segment for a distance equal to $|c_i, c_{i+1}|$.

Finding the target region. Once the positions of all points $T(c_0) \ldots T(c_{m-1})$ are found on the target, we find the positions of each boundary point $T(w_j)$ using the corresponding triple (c_i, d_j, γ_j). Specifically, we walk starting from c_i along a geodesic direction forming the angle γ_j on the target for a distance d_j to obtain $T(w_j)$.

Once all the points on the boundary are found, they need to be connected. We do that by using a plane which passes through the two points $T(w_j)$ and $T(w_{j+1})$ and the normal at one of the points. If both normals happen to be aligned with the direction between the points, an additional point is inserted between them by adding a boundary point on the source midway, generating a radial representation for it, and adding an extra geodesic path.

We traverse the triangles along the intersection of the plane with the surface starting from $T(w_j)$ in the direction of $T(w_{j+1})$. There are three possible

outcomes: either we reach $T(w_{j+1})$ (both points are on the same continuous segment of the plane surface intersection), we return to $T(w_j)$, or we reach a boundary. In the last two cases, we add a new point on the boundary of the source region and repeat the procedure for each pair of points.

Once all sequential points $T(w_j)$ on the target surface are connected, we use a fill algorithm to mark the complete region.

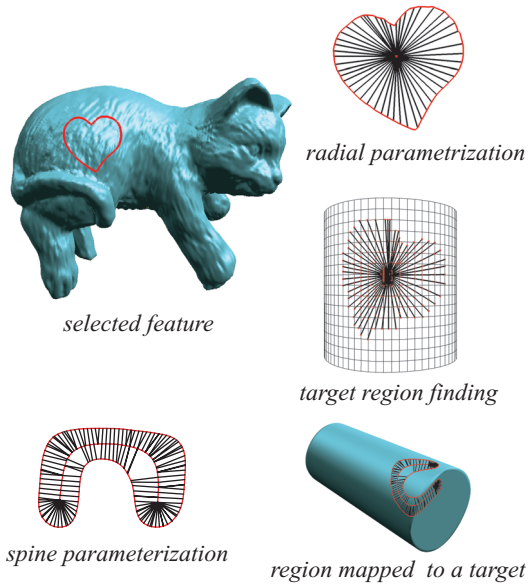


Figure 6.8: Finding the target region.

It is important to note that the algorithm may produce an area which is not topologically equivalent to a disk. For example, any large enough region mapped to a sphere can cover the entire sphere. The algorithm described above should be followed by a test checking the topology of the resulting area. This can be done by computing the genus assuming that there is a face attached to the boundary loop of the region. The genus computed in this way should be zero, and the region should have exactly one boundary loop. If the test fails,

pasting at this scale is not possible. The user should decrease the scale for the pasted feature, or place it at a different location.

Target parameterization. Once the target is determined, it is mapped to the plane. Generally, we use the same relatively expensive angle-based flattening algorithm for target parameterization each time a pasting operation is performed. This allows us to achieve maximum flexibility in feature placement and lowest distortion. While this approach still permits interactive manipulation rates, the frame rate is much better if a larger area of the target can be parameterized and the feature is moved inside this area. In this case, the most expensive part, i.e., target area finding and reparameterization is completely excluded, and only resampling has to be done at most steps.

Geodesic walking. One of the key ingredients of the algorithm for determining the target region is the algorithm for computing a geodesic emanating from a given point in a specified direction. While a number of algorithms for this or similar problems have been proposed [79, 40, 50], our application has specific requirements.

- We need the algorithm to be fast, as the target region has to be found at interactive rates. This makes it difficult to use methods based on front propagation.

- Even more importantly, we need a *continuity property*. Note that termination of the algorithm for finding the target region depends on our ability to make the distance between points $T(w_j)$ on the target arbitrarily small by increasing the density of the points w_j on the source boundary. Such

continuity means that as we decrease the angle between two outgoing geodesics for a point, the distance between their endpoints can be made arbitrarily small. It is known however that straightest geodesics on meshes may violate this condition ("the saddle point problem").

- Accuracy of the result is of secondary importance, as the mapping process is approximate. Also "the swallow tail problem", i.e., the fact that geodesics may intersect near an elliptic point, is not relevant for us as we only determine the the boundary of the region and we do not construct a one-to-one map.

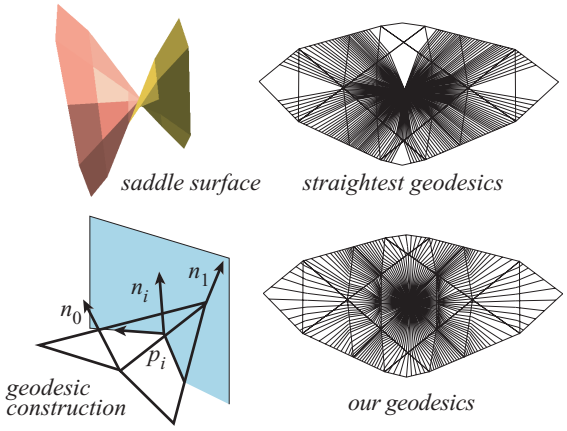


Figure 6.9: Comparison of straightest geodesics and our geodesics. Note the empty regions for the straightest geodesics: no matter how densely the directions are sampled, no geodesic passes through a part of the region.

Our procedure is based on the fact that the geodesic $g(t)$ is always a locally normal curve, i.e., its second derivative $g''(t)$ is pointing along the normal to the surface. By interpolating the normals, we approximate a smooth surface

with continuously changing normal. The elementary step remains going from triangle to triangle, but the angles are computed differently.

Suppose we start at a point p_i at the edge e_0 of a triangle T_i, and v_0 and v_1 are the vertices of the edge e_0. Let n_0 and n_1 be the normals at the vertices v_0 and v_1. We compute the normal n_i at the point p_i as the average of the normals n_0 and n_1. Suppose there is an initial direction vector t_i defined. The procedure described next defines the direction vector at any point of the discrete geodesic to be perpendicular to the normal vector at the point. (If the initial one is not, we project it to the plane perpendicular to the normal). To obtain the point p_{i+1} and the new direction t_{i+1} in triangle T_{i+1} we perform the following steps:

1. Intersect the plane spanned by n_i and t_i with T_i to get a direction $P(t_i)$. If the plane coincides with the plane of T_i, t_i itself is used.

2. Intersect the line along $P(t_i)$ in the triangle T_i with its edges to get the point p_{i+1}. Suppose the intersected edge is e_1 with endpoints v_1 and v_2. The next triangle T_{i+1} is the triangle across the edge e_1.

3. Compute the normal n_{i+1} at p_{i+1} as average of the normals n_1 and n_2 at vertices v_1 and v_2. Project the direction $P(t_i)$ onto the plane perpendicular to n_{i+1} to obtain t_{i+1}. If $P(t_i)$ is parallel to n_{i+1}, we use the average of the projections obtained for two small perturbations of position of the point p_{i+1}.

It can be proven that this procedure satisfies the continuity requirement if the mesh approximating the surface is smooth enough, i.e., the projection of the ring of triangles around any vertex onto the plane perpendicular to the normal is one-to-one.

6.8 Mapping and Resampling

Once the mappings from the source and target to the plane are established, their planar images are aligned using the point and orientation correspondences specified by the user when the target area was chosen. The final step in the pasting algorithm is resampling and combining the details from the source with the details and base surface of the target.

For every vertex v of the parameterization of the target which is inside the parameterization domain of the source, we find the corresponding quad of the source parameterization. Then u, v coordinates are computed in this quad, and the source is evaluated. Evaluation can be done in two ways: for fast resampling, the values of the source at the vertices of the quad are interpolated. For higher quality, subdivision surface evaluation [96] should be used. This is similar to using bilinear filters for fast image editing and bi-cubic filtering for a higher-quality final result.

Adaptive refinement and sampling. The further away the geometry of the feature is from a displacement map, the less suitable pasting for surface operations is. However, in some cases it is desirable to use the pasting paradigm to place objects which cannot be reparameterized over the plane without considerable distortion (Figure 6.13 left). In other cases, the resolution of the source surface is substantially higher than the resolution of the target. In these cases, uniform sampling of the target is not adequate and a form of adaptivity is needed. Hybrid meshes [29] offer the maximal degree of flexibility, as it is possible to perform irregular refinement in some spots and align mesh edges exactly with pasted feature edges. We use a more conventional approach where

only regular refinement of individual faces is allowed. However, rather than quadrisecting individual faces recursively according to a criterion, we estimate the local density of source samples on target face, and directly estimate the subdivision level required for a given face, refining faces to that level uniformly.

6.9 Results

A number of models created using our system are shown in Figures 6.11 to 6.13. Figure 6.11 shows how details from a scanned object are pasted on a simple vase model. In this case, the object itself serves as the base surface. Similarly, Figure 6.12 demonstrates how details from a scanned model can be combined with a different computer model. Figure 6.13 (right) demonstrates how a medium scale detail can be pasted on a surface while preserving small-scale surface details. Figure 6.13 (left) also shows examples of feature manipulation on the surface.

In all cases the operations were performed interactively, but the frame rate varied greatly depending on the complexity of the feature, the complexity of the target region, and the sampling density in the target region. If the target is a simple smooth object, a large area can be parameterized at once without significant distortion, and no dynamic parameterization is required. Then sufficiently complex models still permit high frame rates. At the same time, if no large region can be parameterized without distortion, the frame rate varies in the range 5-0.5 frames per second.

Limitations of the approach . The principal limitations of our approach include:

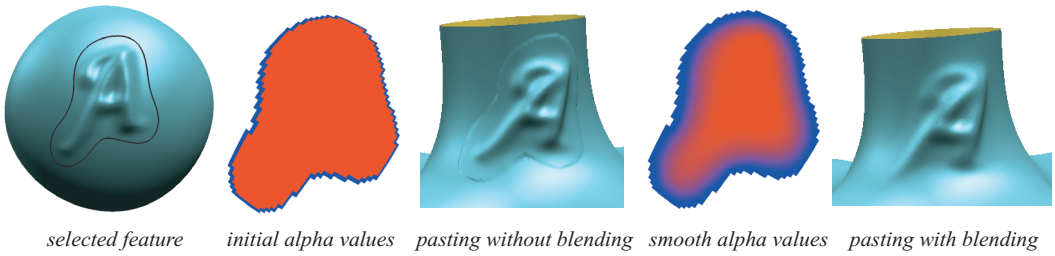


Figure 6.10: Blend region computation for eliminating boundary artifacts.

- The algorithm fails to produce a valid surface when the identified target region is not homeomorphic to a disk. This may occur, for example, if it completely covers a handle.

- The approach is useful for transferring features from one surface to another when the curvature of the chosen target base surface does not deviate radically from the curvature of the source base surface at corresponding points. While the algorithm will produce a valid surface for any situation when the identified target region has disc topology, when the target and source base surfaces are radically different the resulting surface may exhibit distortion of features and self-intersections.

- The resulting surfaces may exhibit geometric aliasing near sharp features as the sampling pattern of the target is used to resample the source. Possible straightforward solutions include adaptive refinement near sharp features which does not eliminate the problem but reduces the scale of artifacts, and smoothing which eliminates the artifacts at the expense of detail. A more promising approach is mentioned in Section 6.10.

Except for the first one, all of the above limitations are "soft" in the sense that the algorithms we have described still produce a formally valid result.

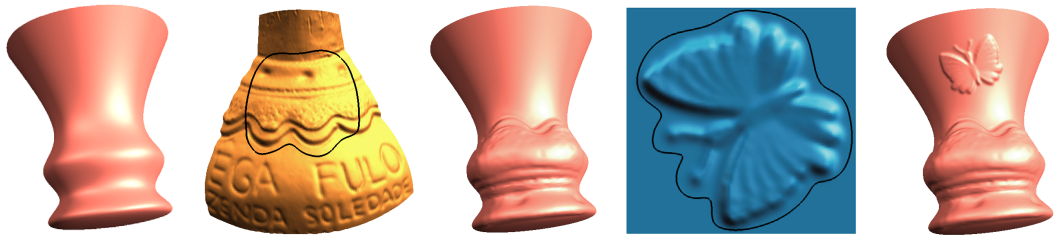


Figure 6.11: Combining a feature obtained by scanning a wine bottle with a displacement map created from a photograph onto a simple vase model.

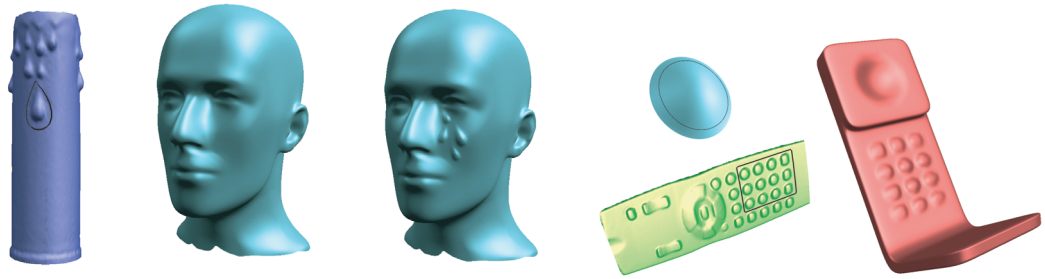


Figure 6.12: Left: details from a scanned candle pasted on the mannequin head. Right: features of different scales from other models added to a phone model.

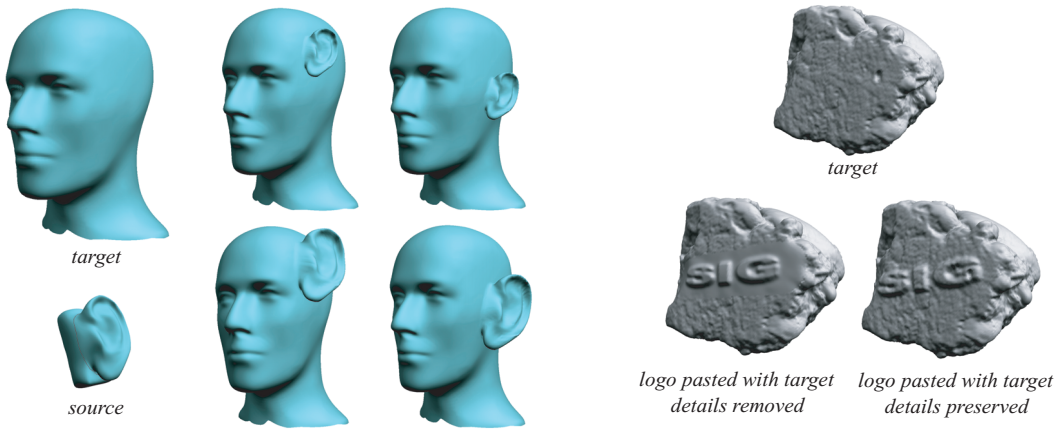


Figure 6.13: Left: pasting a complex feature (an ear) onto the mannequin head. Right: blending of source and target details. The object is a scanned rock.

6.10 Conclusion and Future Work

We have described an approach to surface editing that can be extended in many ways. One can imagine a variety of blending modes, combinations of pasting and texture generation, as well as other enhancements. One of the important advantages of the approach is that the structure of the target mesh is not changed by pasting (except for possible adaptive refinement). This means that complexity of the object is not likely to increase quickly each time a feature is added, as it is the case with boolean operations. This is also a disadvantage, as pasting features with complex shapes may result in strong mesh distortion.

While applicable to a broad range of surfaces, pasting is primarily intended for displacement-map-like features. In its current implementation, the further away a feature is from a displacement map, the more likely self-intersections are to appear especially when a feature is pasted on a highly curved surface. We believe that the applicability of the approach can be extended if hierarchal pasting is used, i.e., the feature is decomposed into details and each level is pasted onto the previous. In this case, more complex features can be pasted more robustly.

Many CAD models have sharp creases. While a multiresolution surface can approximate sharp creases arbitrarily well, the approximate creases are never perfectly sharp and often exhibit aliasing. Furthermore, using details on all levels to introduce a simple corner is wasteful. The representation can be extended [8] to introduce such features by tagging some of the edges but without changing connectivity. Note that the parameterization in this approach (Figure 6.14) has to conform to the sharp feature. An important future enhancement of our system is ability to paste sharp features.

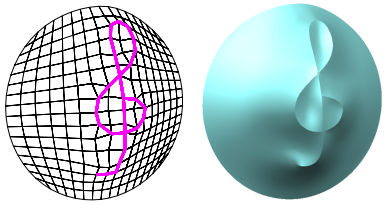


Figure 6.14: Adding a sharp crease to a multiresolution surface without changing the connectivity.

Acknowledgements. The authors thank the staff and students of NYU Media Research Lab for their help. Special thanks go to Xin Zhang for his help with writing and debugging parts of the code. This work was partially supported by funds from the NYU Center for Advanced Technology. IBM Faculty Partnership award, Sloan Foundation Fellowship, NSF award ACI-9978147, CCR-9900528, CCR-0093390 and NYU Dean's fellowship.

Bibliography

[1] A. Agrawal and A. Requicha. A paradigm for the robust design of algorithms for geometric modeling". *Computer Graphics Forum*, 13(3):33–44, 1994.

[2] R. E. Bank. *PLTMG: A Software Package for Solving Elliptic Partial Differential Equations – Users' Guide 7.0*, volume 15 of *Frontiers in Applied Mathematics*. SIAM Books, Philadelphia, 1994.

[3] C. Barghiel, R. Bartels, and D. Forsey. Pasting spline surfaces. In M. Daehlen, T. Lyche, and L. L. Schumaker, editors, *Mathematical Methods for Curves and Surfaces: Ulvik, Norway*, pages 31–40, Nashville, TN., 1994. Vanderbilt University Press. Available on WWW as ftp://cgl.uwaterloo.ca/pub/users/rhbartel/Paste.ps.gz.

[4] R. E. Barnhill, G. Farin, M. Jordan, and B. R. Piper. Surface/surface intersection. *Computer Aided Geometric Design*, 4(1-2):3–16, July 1987.

[5] H. Biermann, D. Kristjansson, and D. Zorin. Approximate boolean operations on free-form solids. In *Proceedings of ACM SIGGRAPH 2001*, Computer Graphics Proceedings, Annual Conference Series, pages 185–194. ACM Press / ACM SIGGRAPH, August 2001. ISBN 1-58113-292-1.

[6] H. Biermann, A. Levin, and D. Zorin. Piecewise-smooth subdivision surfaces with normal control. *Proceedings of SIGGRAPH 2000*, pages 113–120, July 2000.

[7] H. Biermann, I. Martin, F. Bernardini, and D. Zorin. Cut-and-paste editing of multiresolution surfaces. *ACM Transactions on Graphics*, 21(3):312–321, July 2002.

[8] H. Biermann, I. M. Martin, D. Zorin, and F. Bernardini. Sharp features on multiresolution subdivision surfaces. In *9th Pacific Conference on Computer Graphics and Applications*. IEEE, October 2001.

[9] C. Burnikel, S. Funke, and M. Seel. Exact arithmetic using cascaded computation. *ACM Symposium on Computational Geometry*, (14):175–193, 1998.

[10] C. Burnikel, K. Mehlhorn, and S. Schirra. On degeneracy in geometric computations. In *Proceedings of the Fifth Annual ACM-SIAM Symposium on Discrete Algorithms (Arlington, VA, 1994)*, pages 16–23, New York, 1994. ACM.

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

[12] A. S. Cavaretta, W. Dahmen, and C. A. Micchelli. Stationary subdivision. *Memoirs Amer. Math. Soc.*, 93(453), 1991.

[13] L. K. Y. Chan, S. Mann, and R. Bartels. World space surface pasting. In W. Davis, M. Mantei, and V. Klassen, editors, *Graphics Interface*, pages 146–154, May 1997.

[14] F. Cirak, M. Ortiz, and P. Schröder. Subdivision surfaces: A new paradigm for thin-shell finite element analysis. *Internat. J. Numer. Methods Engrg.*, pages 2039–207, 2000.

[15] A. Cohen, N. Dyn, and D. Levin. Matrix subdivision schemes. Technical report, Université Pierre et Marie Curie, 1997.

[16] B. Conrad and S. Mann. Better pasting via quasi-interpolation. In P.-J. Laurent, P. Sablonnière, and L. L. Schumaker, editors, *Curve and Surface Design: Saint-Malo, 1999*, pages 27–36, Nashville, TN., 2000. Vanderbilt University Press.

[17] T. D. D. Zorin, P. Schröder and W. S. L. Kobbelt, A. Levin. Subdivision for modeling and animation. SIGGRAPH 2000 Course Notes, 2000.

[18] T. DeRose, M. Kass, and T. Truong. Subdivision surfaces in character animation. *Proceedings of SIGGRAPH 98*, pages 85–94, July 1998. ISBN 0-89791-999-8. Held in Orlando, Florida.

[19] K. Dobrindt, K. Mehlhorn, and M. Yvinec. A complete and efficient algorithm for the intersection of a general and a convex polyhedron. In *Algorithms and data structures (Montreal, PQ, 1993)*, pages 314–324. Springer, Berlin, 1993.

[20] D. Doo. A subdivision algorithm for smoothing down irregularly shaped polyhedrons. In *Proceedings on Interactive Techniques in Computer Aided Design*, pages 157–165, Bologna, 1978.

[21] D. Doo and M. Sabin. Analysis of the behaviour of recursive division surfaces near extraordinary points. *Computer Aided Design*, 10(6):356–360, 1978.

[22] B. A. Dubrovin, A. T. Fomenko, and S. P. Novikov. *Modern geometry—methods and applications. Part I.* Springer-Verlag, New York, second edition, 1992. The geometry of surfaces, transformation groups, and fields, Translated from the Russian by Robert G. Burns.

[23] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle. Multiresolution analysis of arbitrary meshes. *Proceedings of SIGGRAPH 95*, pages 173–182, August 1995. ISBN 0-201-84776-0. Held in Los Angeles, California.

[24] D. Epstein, N. Gharachorloo, F. Jansen, J. Rossignac, , and C. Zoulos. Multiple depth-buffer rendering of csg. Technical report, IBM Research Report, 1989.

[25] D. A. Field. Laplacian smoothing and delaunay triangulations. *Comm. Applications: Numerical Methods*, (4):709–712, 1988.

[26] M. S. Floater. Parametrization and smooth approximation of surface triangulations. *Computer Aided Geometric Design*, 14(3):231–250, 1997. ISSN 0167-8396.

[27] L. Freitag, M. Jones, and P. Plassmann. A parallel algorithm for mesh smoothing. *SIAM J. Sci. Comput.*, 20(6):2023–2040 (electronic), 1999.

[28] J. Goldfeather, J. P. M. Hultquist, and H. Fuchs. Fast constructive-solid geometry display in the pixel-powers graphics system. *Computer Graphics*

(Proceedings of SIGGRAPH 86), 20(4):107–116, August 1986. Held in Dallas, Texas.

[29] I. Guskov, A. Khodakovsky, and P. Schröoder. Hybrid meshes. In *Proceedings of SoCG 2002*, 2002.

[30] I. Guskov, W. Sweldens, and P. Schröder. Multiresolution signal processing for meshes. In *Proceedings of SIGGRAPH 99*, Computer Graphics Proceedings, Annual Conference Series, pages 325–334, Los Angeles, California, August 1999. ACM SIGGRAPH / Addison Wesley Longman. ISBN 0-20148-560-5.

[31] I. Guskov, K. Vidimce, W. Sweldens, and P. Schrder. Normal meshes. *Proceedings of SIGGRAPH 2000*, pages 95–102, July 2000. ISBN 1-58113-208-5.

[32] A. Habib and J. Warren. Edge and vertex insertion for a class of C^1 subdivision surfaces. *Computer Aided Geometric Design*, 16(4):223–247, 1999.

[33] M. Halstead, M. Kass, and T. DeRose. Efficient, fair interpolation using Catmull-Clark surfaces. In *Computer Graphics Proceedings*, Annual Conference Series, pages 35–44. ACM Siggraph, 1993.

[34] C. M. Hoffmann. *Geometric and solid modeling: a introduction.* San Mateo, California: Morgan Kaufmann, 1989.

[35] H. Hoppe. Progressive meshes. In H. Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 99–108.

ACM SIGGRAPH, Addison Wesley, August 1996. held in New Orleans, Louisiana, 04-09 August 1996.

[36] H. Hoppe, T. DeRose, T. Duchamp, M. Halstead, H. Jin, J. McDonald, J. Schweitzer, and W. Stuetzle. Piecewise smooth surface reconsruction. In *Computer Graphics Proceedings*, Annual Conference Series, pages 295–302. ACM Siggraph, 1994.

[37] J. Hoschek and D. Lasser. *Fundamentals of computer aided geometric design*. A K Peters Ltd., Wellesley, MA, 1993. Translated from the 1992 German edition by Larry L. Schumaker.

[38] A. Khodakovsky and P. Schröder. Fine level feature editing for subdivision surfaces. In *Proceedings of ACM Solid Modeling*, 1999.

[39] A. Khodakovsky, P. Schröder, and W. Sweldens. Progressive geometry compression. *Proceedings of SIGGRAPH 2000*, July 2000.

[40] R. Kimmel and J. A. Sethian. Computing geodesic paths on manifolds. *Proc. Natl. Acad. Sci. USA*, 95(15):8431–8435 (electronic), 1998.

[41] L. Kobbelt. A variational approach to subdivision. *Comput. Aided Geom. Design*, 13(8):743–761, 1996.

[42] L. Kobbelt, S. Campagna, J. Vorsatz, and H.-P. Seidel. Interactive multi-resolution modeling on arbitrary meshes. *Proceedings of SIGGRAPH 98*, pages 105–114, July 1998. ISBN 0-89791-999-8. Held in Orlando, Florida.

[43] L. P. Kobbelt. Discrete fairing and variational subdivision for freeform surface design. *The Visual Computer*, 16(3-4):142–150, 2000. ISSN 0178-2789.

[44] L. P. Kobbelt, K. Daubert, and H.-P. Seidel. Ray tracing of subdivision surfaces. *Eurographics Rendering Workshop 1998*, pages 69–80, June 1998. Held in Vienna, Austria.

[45] V. Krishnamurthy and M. Levoy. Fitting smooth surfaces to dense polygon meshes. In H. Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 313–324. ACM SIGGRAPH, Addison Wesley, August 1996. held in New Orleans, Louisiana, 04-09 August 1996.

[46] S. Krishnan and D. Manocha. An efficient surface intersection algorithm based on lower-dimensional formulation. *ACM Transactions on Graphics*, 16(1):74–106, January 1997. ISSN 0730-0301.

[47] S. Kuriyama and T. Kaneko. Discrete parameterization for deforming arbitrary meshes. *Graphics Interface '99*, pages 132–139, June 1999. ISBN 1-55860-632-7.

[48] A. Lee, H. Moreton, and H. Hoppe. Displaced subdivision surfaces. *Proceedings of SIGGRAPH 2000*, pages 85–94, July 2000. ISBN 1-58113-208-5.

[49] A. W. F. Lee, W. Sweldens, P. Schröder, L. Cowsar, and D. Dobkin. MAPS: Multiresolution adaptive parameterization of surfaces. *Proceedings of SIGGRAPH 98*, pages 95–104, July 1998. ISBN 0-89791-999-8. Held in Orlando, Florida.

[50] H. Lee, L. Kim, M. Meyer, and M. Desbrun. Meshes on fire. In *EG Workshop on Computer Animation and Simulation*, 2001.

[51] A. Levin. Analysis of quasi-uniform subdivision schemes. In preparation, 1999.

[52] A. Levin. Combined subdivision schemes for the design of surfaces satisfying boundary conditions. *Computer Aided Geometric Design*, 16(5):345–354, 1999.

[53] A. Levin. Interpolating nets of curves by smooth subdivision surfaces. In A. Rockwood, editor, *SIGGRAPH 99 Conference Proceedings*, Annual Conference Series, pages 57–64. Addison Wesley, 1999.

[54] A. Levin. Interpolating nets of curves by smooth subdivision surfaces. *Proceedings of SIGGRAPH 99*, pages 57–64, August 1999.

[55] B. Lévy and J.-L. Mallet. Non-distorted texture mapping for sheared triangulated meshes. *Proceedings of SIGGRAPH 98*, pages 343–352, July 1998.

[56] M. Lin and S. Gottschalk. Collision detection between geometric models: A survey. *Proceedings of IMA Conference on Mathematics of Surfaces*, 1998.

[57] L. Linsen. Schneiden und vereinen von kontrollnetzen (intersection and merging of control meshes.). Master's thesis, Universität Karlsruhe, 1997. in German.

[58] N. Litke, A. Levin, and P. Schröder. Fitting subdivision surfaces. In *IEEE Visualization 2001*, pages 319–324, October 2001. ISBN 0-7803-7200-x.

[59] N. Litke, A. Levin, and P. Schroeder. Trimming for subdivision surfaces. *Computer Aided Geometric Design*, 18(5):463–481, 2001.

[60] C. Loop. Smooth subdivision surfaces based on triangles. Master's thesis, University of Utah, Department of Mathematics, 1987.

[61] M. Lounsbery, T. DeRose, and J. Warren. Multiresolution analysis for surfaces of arbitrary topological type. *Transactions on Graphics*, 16(1):34–73, January 1997.

[62] M. Ma. The direct manipulation of pasted surfaces. Master's thesis, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1, 2000. Available on WWW as ftp://cs-archive.uwaterloo.ca/cs-archive/CS-2000-15/.

[63] J. Maillot, H. Yahia, and A. Verroust. Interactive texture mapping. *Proceedings of SIGGRAPH 93*, pages 27–34, August 1993. ISBN 0-201-58889-7. Held in Anaheim, California.

[64] L. Markosian, J. M. Cohen, T. Crulli, and J. F. Hughes. Skin: A constructive approach to modeling free-form shapes. *Proceedings of SIGGRAPH 99*, pages 393–400, August 1999. Held in Los Angeles, California.

[65] J. Mitchell. Geometric shortest paths and network optimization. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*. Elsevier Science, 2000.

[66] R. E. Moore. *Methods And Applications Of Interval Analysis*. SIAM, Philadelphia, 1979.

[67] J. Munkres. *Elementary Differential Topology*. Princeton University Press, 1966.

[68] A. Nasri. Interpolation of open B-spline curves by recursive subdivision surfaces. In T. Goodman and R. Martin, editors, *Mathematics of Sur-*

faces VII, pages 173 –188. Institute of mathematics and its applications, Information Geometers, 1997.

[69] A. H. Nasri. Polyhedral subdivision methods for free-form surfaces. *ACM Transactions on Graphics*, 6(1):29–73, January 1987.

[70] A. H. Nasri. Boundary corner control in recursive subdivision surfaces. *Computer Aided Design*, 23(6):405–410, 1991.

[71] A. H. Nasri. Surface interpolation on irregular networks with normal conditions. *Computer Aided Geometric Design*, 8:89–96, 1991.

[72] A. H. Nasri. Surface interpolation on irregular networks with normal conditions. *CAGD*, 8:89–96, 1991.

[73] B. Oberknapp and K. Polthier. An algorithm for discrete constant mean curvature surfaces. In *Visualization and mathematics (Berlin-Dahlem, 1995)*, pages 141–161. Springer, Berlin, 1997.

[74] H. K. Pedersen. Decorating implicit surfaces. *Proceedings of SIGGRAPH 95*, pages 291–300, August 1995. ISBN 0-201-84776-0. Held in Los Angeles, California.

[75] H. K. Pedersen. A framework for interactive texturing operations on curved surfaces. *Proceedings of SIGGRAPH 96*, pages 295–302, August 1996. ISBN 0-201-94800-1. Held in New Orleans, Louisiana.

[76] K. Perlin and L. Velho. Live paint: Painting with procedural multiscale textures. *Proceedings of SIGGRAPH 95*, pages 153–160, August 1995.

[77] J. Peters and U. Reif. Analysis of generalized B-spline subdivision algorithms. *SIAM Jornal of Numerical Analysis*, 1997.

[78] U. Pinkall and K. Polthier. Computing discrete minimal surfaces and their conjugates. *Experiment. Math.*, 2(1):15–36, 1993.

[79] K. Polthier and M. Schmies. Straightest geodesics on polyhedral surfaces. In H. Hege and K. Polthier, editors, *Mathematical Visualization*. Springer Verlag, 1998.

[80] E. Praun, A. Finkelstein, and H. Hoppe. Lapped textures. In *Proceedings of ACM SIGGRAPH 2000*, Computer Graphics Proceedings, Annual Conference Series, pages 465–470. ACM Press / ACM SIGGRAPH / Addison Wesley Longman, July 2000. ISBN 1-58113-208-5.

[81] E. Praun, W. Sweldens, and P. Schröder. Consistent mesh parameterizations. In *Proceedings of ACM SIGGRAPH 2001*, Computer Graphics Proceedings, Annual Conference Series, pages 179–184. ACM Press / ACM SIGGRAPH, August 2001. ISBN 1-58113-292-1.

[82] H. Prautzsch and U. Reif. Degree estimates for Cspan$k$-piecewise polynomial subdivision surfaces. *Adv. Comput. Math.*, 10(2):209–217, 1999.

[83] H. Prautzsch and G. Umlauf. A Gspan2-subdivision algorithm. In *Geometric modelling (Dagstuhl, 1996)*, pages 217–224. Springer, Vienna, 1998.

[84] K. Pulli and M. Lounsbery. Hierarchical editing and rendering of subdivision surfaces. Technical Report UW-CSE-97-04-07, Dept. of CS&E, University of Washington, Seattle, WA, 1997.

[85] A. Rappoport and S. Spitz. Interactive boolean operations for conceptual design of 3-d solids. *Proceedings of SIGGRAPH 97*, pages 269–278, August 1997. Held in Los Angeles, California.

[86] U. Reif. A unified approach to subdivision algorithms near extraordinary points. *Computer Aided Geometric Design*, 12:153–174, 1995.

[87] J. Rossignac and A. Requicha. Solid modeling. In J. Webster, editor, *Encyclopedia of Electrical and Electronics Engineering*. John Wiley and Sons, 1999.

[88] J. E. Schweitzer. *Analysis and Application of Subdivision Surfaces*. PhD thesis, University of Washington, Seattle, 1996.

[89] T. Sederberg and T. Nishita. Geometric Hermite approximation of surface patch intersection curves. *Computer Aided Geometric Design*, 8(2):97–114, 1991.

[90] T. Sederberg and S. Parry. Free-form deformation of solid geometric models. *Computer Graphics*, 20(4):151–160, 1986.

[91] T. W. Sederberg, J. Zheng, D. Sewell, and M. Sabin. Non-uniform recursive subdivision surfaces. In M. Cohen, editor, *SIGGRAPH 98 Conference Proceedings*, Annual Conference Series, pages 387–394. ACM SIGGRAPH, Addison Wesley, July 1998. ISBN 0-89791-999-8.

[92] R. Seidel. The nature and meaning of perturbations in geometric computing. *Discrete Comput. Geom.*, 19(1):1–17, 1998.

[93] A. Sheffer and E. de Sturler. Surface parameterization for meshing by triangulation flattening. In *Proc. 9th International Meshing Roundtable*, pages 161–172, 2000.

[94] J. R. Shewchuk. Adaptive precision floating-point arithmetic and fast robust geometric predicates. Technical report, Carnegie Mellon University, 1996.

[95] K. Singh and E. Fiume. Wires: A geometric deformation technique. *Proceedings of SIGGRAPH 98*, pages 405–414, July 1998.

[96] J. Stam. Exact evaluation of Catmull-Clark subdivision surfaces at arbitrary parameter values. In *Proceedings of SIGGRAPH 98*, Computer Graphics Proceedings, Annual Conference Series, pages 395–404, Orlando, Florida, July 1998. ACM SIGGRAPH / Addison Wesley. ISBN 0-89791-999-8.

[97] E. M. Stein. *Singular integrals and differentiability properties of functions*. Princeton University Press, Princeton, N.J., 1970. Princeton Mathematical Series, No. 30.

[98] A. J. Stewart. Local robustness and its application to polyhedral intersection. *Internat. J. Comput. Geom. Appl.*, 4(1):87–118, 1994.

[99] H. Suzuki, Y. Sakurai, T. Kanai, and F. Kimura. Interactive mesh dragging with an adaptive remeshing technique. *The Visual Computer*, 16(3-4):159–176, 2000.

[100] G. Taubin. A signal processing approach to fair surface design. In R. Cook, editor, *SIGGRAPH 95 Conference Proceedings*, Annual Conference Series, pages 351–358. ACM SIGGRAPH, Addison Wesley, August 1995.

[101] C. L. F. Tsang. Animated surface pasting. Master's thesis, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1, 1998. Available on WWW as ftp://cs-archive.uwaterloo.ca/cs-archive/CS-98-19/.

[102] F. W. Warner. *Foundations of differentiable manifolds and Lie groups*. Springer-Verlag, New York, 1983. Corrected reprint of the 1971 edition.

[103] J. Warren. Subdivision methods for geometric design. Unpublished manuscript, November 1995.

[104] D. Zorin. *Subdivision and Multiresolution Surface Representations*. PhD thesis, Caltech, Pasadena, 1997.

[105] D. Zorin. A method for analysis of C^1-continuity of subdivision surfaces. *SIAM Journal of Numerical Analysis*, 37(4), 2000.

[106] D. Zorin. Smoothness of subdivision on irregular meshes. *Constructive Approximation*, 16(3), 2000.

[107] D. Zorin, P. Schröder, and W. Sweldens. Interpolating subdivision for meshes with arbitrary topology. *Proceedings of SIGGRAPH 96*, pages 189–192, August 1996.

[108] D. Zorin, P. Schröder, and W. Sweldens. Interactive multiresolution mesh editing. *Proceedings of SIGGRAPH 97*, pages 259–268, August 1997. ISBN 0-89791-896-7. Held in Los Angeles, California.

[109] D. N. Zorin. *Subdivision and Multiresolution Surface Representations*. PhD thesis, Caltech, Pasadena, California, 1997.