© Copyright 1992
Stephen Mann

University of Washington

Abstract

Surface Approximation Using Geometric Hermite Patches

by Stephen Mann

Chair of Supervisory Committee: Professor Tony DeRose
Department of Computer Science and Engineering

Many surfaces have mathematically complex or computationally expensive representations. For some applications, an approximation to these surfaces is adequate. Numerous tangent plane smooth surface interpolants are reviewed, and inadequacies of these schemes for the approximation of surfaces are investigated. Two high-order-of-approximation surface patches are presented and used to construct continuous, approximating surfaces. The use of these patches is demonstrated for approximating offset surfaces, algebraic surfaces, and S-patches.

TABLE OF CONTENTS

List of Figures iv

List of Tables vi

Chapter 1: Introduction 1

1.1 Overview . . . 2

1.2 Notation and Mathematical Preliminaries. . . . 3

Chapter 2: G^1 Interpolants 7

2.1 Background. . . . 7

2.2 Tangent Plane Continuity. . . . 8

2.2.1 Farin . . . 8

2.2.2 Piper . . . 9

2.2.3 Chiyokura-Kimura, Herron, Jensen . . . 10

2.2.4 Peters . . . 13

2.3 Parametric Schemes. . . . 15

2.3.1 Split Domain Schemes . . . 15

2.3.2 Convex Combination Schemes . . . 20

2.3.3 Other Schemes . . . 28

2.4 Comparison. . . . 28

2.4.1 Data Sets . . . 29

2.4.2 Results of Comparison . . . 30

2.4.3 Boundary Curves . . . 32

2.4.4 Cubic Geometric Hermite Curves . . . 34

2.4.5 Boundary curve parameterization . . . 37

2.4.6 Degrees of Freedom . . . 38

Chapter 3: The Cubic Interpolant **48**
3.1 The Quadratic Interpolant 48
3.2 The Cubic Interpolant 49
3.3 Results 53
3.4 Order of Approximation of the Cubic Interpolant 54
3.4.1 Conjecture of Quintic Convergence 54
3.5 Proof of lemmas 57

Chapter 4: The Bicubic Interpolant **64**
4.1 Order of Approximation of the Bicubic Interpolant 69
4.2 Sixth Order of Convergence 70
4.3 Proof of Lemmas 72

Chapter 5: Applications **76**
5.1 Approximate Continuity 76
5.2 A Piecewise Cubic Surface Scheme 77
5.3 Bounding the Discontinuity 78
5.3.1 Analytic Bound 78
5.4 Approximation of Known Functions 79
5.4.1 Algebraic surfaces 80
5.4.2 Offset surfaces 82
5.4.3 S-Patches 83
5.5 Scattered Data 87
5.5.1 Partial Estimation 89
5.5.2 Local Estimation 89
5.5.3 Global Estimation 91
5.5.4 Results 92

Chapter 6: Conclusions and Future Work **102**

Bibliography **104**

Appendix A: Herron's Method **110**

Appendix B: S-patch Derivatives **111**
B.1 First Derivatives 111
B.2 Second Derivatives 112
B.3 Code 113

LIST OF FIGURES

1.1 Vertex ring. 6

2.1 Farin's G^1 conditions. 9
2.2 Piper's G^1 conditions. 10
2.3 Chiyokura and Kimura's G^1 construction. 11
2.4 Peters' G^1 conditions. 14
2.5 Split domain schemes. 16
2.6 Hansford. 19
2.7 Side-vertex method. 21
2.8 Triangular Gregory patch. 23
2.9 Herron. 24
2.10 Gregory-Charrot. 26
2.11 Data sets. 30
2.12 Tangent vector construction. 33
2.13 Bézier curves and their curvature plots. 34
2.14 Cubic Geometric Hermite Curves. 35
2.15 Curve parameterizations. 39
2.16 Curve networks. 39
2.17 Comparison of Bézier patches. 40
2.18 Shirman-Séquin sphere. 42
2.19 Shirman-Séquin torus. 44
2.20 Shirman-Séquin octahedron. 46

3.1 Control points of a quadratic patch. 49
3.2 Control points of a cubic patch. 50
3.3 Construction of a planar boundary curve. 52
3.4 Domain of S and P. 56
3.5 Cubic interpolant fit to torus. 60

3.6 Cubic interpolant fit to octahedron. 62

4.1 The bi-cubic interpolant's control points. 65
4.2 Quadrilateral torus mesh. 69
4.3 Tensor product domain. 70
4.4 Bicubic interpolant torus. 74

5.1 Refinement of the mesh. 77
5.2 Meshes for the function $4x^2 + 2y^2 + z^2 - 4 = 0$. 81
5.3 A five sided S-patch. 84
5.4 The branch mesh and its refinements. 86
5.5 Ring mesh. 88
5.6 Moreton's curvature estimation scheme. 91
5.7 Curvature plots of surfaces fit to $4x^2 + 2y^2 + z^2 - 4 = 0$. 94
5.8 An offset to a bicubic. 96
5.9 The S-patch "branch" surface. 98
5.10 The S-patch "ring" surface. 100

LIST OF TABLES

5.1 Refinement color map. 83
5.2 Approximation of S-patch branch surface. 87
5.3 Approximation of S-patch ring surface. 88

ACKNOWLEDGMENTS

I'd like to thank my reading committee, Tony DeRose, Steve Tanimoto, and David Salesin, for their helpful comments on the text of this dissertation. I'd like to thank my advisor Tony DeRose for his introducing me to the field and for countless discussions about the material. I'd also like to thank Ken Sloan for for his discussions of several ideas.

I'd like to thank my mother [Man68] and father [Man69] for encouraging me to continue my education.

I'd like to thank for graphics group (Michael Lounsbery, Jamie Painter, David Meyers, Charles Loop, Tony DeRose, and Kenneth Sloan) for their help with the initial surface survey.

Several people have allowed me to use their software. In particular, I would like to thank Hugues Hoppe for the use of his marching cubes code; Eng-wee Chionh for his quadric intersection code; and Henry Moreton, for his curve optimization code.

DEDICATION

In memory of my father

Kenneth Earl Mann

and of my grandfather

Jasper Williams

Chapter 1

INTRODUCTION

The problem of constructing a surface from a set of points arises in numerous applications such as automobile and ship hull design, medical imaging, scientific visualization, geological modeling, and geometric modeling. There are many variations of this problem, based on the form of the data, on extra information about the input, and on requirements for the resulting surface.

In this dissertation, I will focus on the approximation of known surfaces. A surface will be approximated by first sampling it, and then building an approximating surface that interpolates these samples. My primary interest is in finding an approximation method that is visually indistinguishable from the original surface.

One way to categorize surface fitting schemes is by the locality of data used in constructing a portion of the surface. A *global scheme* uses arbitrarily many of the data points in constructing each portion of the surface. A *local scheme* only considers those data points near the portion of the surface being created. Local schemes construct piecewise continuous surfaces, and ensure that adjacent surface patches meet "smoothly."

If the data lie above a plane, then they can be described by a function $\mathbf{S}(x, y) = (x, y, f(x, y))$. The data set is then called *scalar data*, as the surface can be thought of as a scalar-valued function over the plane. Such data can be interpolated with a differentiable surface, using, for instance, the methods surveyed by Barnhill [Bar83] and Franke and Nielson [Fra82, FN90].

A parametric scheme, on the other hand, constructs a vector-valued surface, $\mathbf{S}(u, v) = (x(u, v), y(u, v), z(u, v))$, and, unlike a scalar method, is capable of representing surfaces of arbitrary topological type. The parametric problem is generally considered to be more difficult than the scalar variant. It has been shown, for instance, that some data cannot be interpolated with a parametrically continuous surface (i.e., one with continuous parametric derivatives) [Her85]. Instead, the continuity condi-

tions have to be relaxed to G^1 (tangent plane) continuity. (Rather than continuity of derivatives, a G^1 surface only requires continuity of surface normals. While a G^1 surface may be locally reparameterized to be C^1, it is not always possible to find a global reparameterization having continuous derivatives everywhere.)

In addition to geometric data, most parametric interpolation schemes require information about the topological type of the desired surface (although some recent research investigates the reconstruction of surfaces where the topological type is unknown [HDD+92]). Topological information is usually specified as adjacency information relating the data points (vertices), edges, and faces. Most schemes assume that the faces are either triangular or quadrilateral.

Finally, surface fitting schemes may interpolate or approximate the given data. *Interpolating schemes* construct surfaces that pass through the given data points. *Approximating schemes* produce surfaces that retain the topology of the input data, but only pass near the data points. For some applications, an interpolating scheme is preferred, while for other applications, an approximating scheme may be a better choice.

1.1 Overview

In this dissertation, I concentrate on approximating known functions with local, triangular, parametric, interpolation schemes. In Chapter 2, I survey previous methods for constructing such surfaces. Most of these schemes fall into two broad categories: those that construct one patch per face and those that construct three patches per face. While the schemes all construct surfaces that meet the mathematical continuity conditions, I found that the constructed surfaces have "poor shape" (a typical surface appears on the left in Figure 2.19).

On further investigation, I found that after meeting the continuity conditions, these methods all have free parameters that are set using simple heuristics intended to have a neutral effect on surface quality. The resulting patches, however, are often constructed with poor distributions of curvature. Setting some of the free parameters to match higher order derivative constraints results in some improvement in the shape of the resulting surfaces (compare the image on the left to the one on the right in Figure 2.19).

In Chapter 3, I present the cubic interpolant, a triangular interpolant that in-

terpolates positions, normals, and surface curvature at the corners of a triangle. Piecewise continuous surfaces constructed using these patches, while only C^0, are visually smoother than the G^1 interpolants (Figure 3.5). In Chapter 4, I briefly look at quadrilateral patches, and present a quadrilateral surface patch that interpolates second order data at the patch corners.

In Chapter 5, I show how the cubic interpolant can be used to approximate known functions, including algebraic surfaces (Figure 5.7), offset surfaces (Figure 5.8), and S-patches (Figures 5.9 and 5.10). In each case, the main subproblem to be solved is the computation of the surface curvature. In Section 5.5, I investigate using the cubic interpolant for scattered data fitting.

1.2 Notation and Mathematical Preliminaries.

Throughout this dissertation, I denote scalars and scalar-valued functions by non-bold type italic letters and by Greek letters, such as r and α. Points and point-valued functions are denoted with boldface letters, such as $\mathbf{V}$. Vectors will be represented by boldface letters topped with an arrow, such as $\vec{\mathbf{T}}$. Unit vectors are denoted using a diacritical arrow, as in $\hat{\mathbf{C}}$.

In general, I denote a surface to be approximated by $\mathbf{S}$. However, functions above a plane are denoted by S. A directional derivative of $\mathbf{S}$ at a domain point $\mathbf{p}$ in direction $\vec{\mathbf{u}}$ is denoted $\mathbf{D}_{\vec{\mathbf{u}}}\mathbf{S}(\mathbf{p})$. As I will compare surfaces with different domains, it simplifies the notation to place $\vec{\mathbf{u}}$ in the tangent plane of $\mathbf{S}$, with the understanding that $\mathbf{D}_{\vec{\mathbf{u}}}\mathbf{S}$ denotes the derivative of $\mathbf{S}$ in the direction of the preimage of $\vec{\mathbf{u}}$ (a vector $\vec{\mathbf{v}}$ is said to be the preimage of a vector $\vec{\mathbf{u}}$ if $\vec{\mathbf{v}}$ maps to $\vec{\mathbf{u}}$ under the differential). Note that $\mathbf{D}_{\vec{\mathbf{u}}}\mathbf{S}(\mathbf{p}) = \vec{\mathbf{u}}$. However, for scalar functions, $D_{\vec{\mathbf{u}}}S$ is interpreted with $\vec{\mathbf{u}}$ in the domain of S.

$\mathbf{D}_{\vec{\mathbf{u}}\vec{\mathbf{v}}}\mathbf{S}(\mathbf{p})$ denotes the second derivative of $\mathbf{S}$ in directions $\vec{\mathbf{u}}$ and $\vec{\mathbf{v}}$. $< \vec{\mathbf{u}},\ \vec{\mathbf{v}} >$ is used to denote the Euclidean dot product of vectors $\vec{\mathbf{u}}$ and $\vec{\mathbf{v}}$. The unit normal to $\mathbf{S}$ at domain point $\mathbf{p}$ is denoted $\hat{\mathbf{N}}(\mathbf{p})$; $\mathbf{p}$ will be omitted when the point of evaluation is clear. The first derivative of a parametric curve $\mathbf{H}(t)$ is denoted $\mathbf{H}'(t)$, and the ith derivative of a curve is denoted $\mathbf{H}^{(i)}(t)$.

It is well known from differential geometry that the curvature properties of a surface $\mathbf{S}$ at a point $\mathbf{p}$ are characterized by a symmetric bilinear form, called the second fundamental form ([O'N66] pg 208). I denote this form as $\mathrm{II}^{\mathbf{S}}_{\mathbf{p}}(\vec{\mathbf{u}}, \vec{\mathbf{v}})$, where

$\vec{\mathbf{u}}$ and $\vec{\mathbf{v}}$ are vectors in the tangent plane of $\mathbf{S}$ at $\mathbf{p}$. The symbols $\mathbf{S}$ and $\mathbf{p}$ will be omitted when they are clear from the context. The second fundamental form is related to derivatives of the surface according to

$$\mathrm{II}(\vec{\mathbf{u}}, \vec{\mathbf{v}}) = \; - < \mathbf{D}_{\vec{\mathbf{u}}\vec{\mathbf{v}}}\mathbf{S}, \; \hat{\mathbf{N}} >,$$

implying that, up to sign, $\mathrm{II}(\vec{\mathbf{u}}, \vec{\mathbf{v}})$ is the length of the normal component of $\mathbf{D}_{\vec{\mathbf{u}}\vec{\mathbf{v}}}\mathbf{S}$. The normal component of the curvature $k_{\vec{\mathbf{u}}}$ corresponding to direction $\vec{\mathbf{u}}$ is proportional to $\mathrm{II}(\vec{\mathbf{u}}, \vec{\mathbf{u}})$:

$$k_{\vec{\mathbf{u}}} = \frac{\mathrm{II}(\vec{\mathbf{u}}, \vec{\mathbf{u}})}{< \vec{\mathbf{u}}, \; \vec{\mathbf{u}} >}.$$

In Chapter 3, I need the second fundamental form to be specified at each data point. One way to specify the second fundamental form is by fixing three values $\mathrm{II}(\vec{\mathbf{u}}, \vec{\mathbf{u}})$, $\mathrm{II}(\vec{\mathbf{u}}, \vec{\mathbf{v}})$, and $\mathrm{II}(\vec{\mathbf{v}}, \vec{\mathbf{v}})$, where $\vec{\mathbf{u}}$ and $\vec{\mathbf{v}}$ are any pair of independent tangent vectors [O'N66].

The second fundamental form can also be characterized by the principal directions. The principal directions are the directions of maximum and minimum normal section curvature. These two directions are perpendicular to one another. Further, if $\vec{\mathbf{u}}$ and $\vec{\mathbf{v}}$ are principal directions, then $\mathrm{II}(\vec{\mathbf{u}}, \vec{\mathbf{v}}) = 0$.

Bézier Patches

Let $B_i^n(t)$ denote the ith Bernstein polynomial of nth degree, i.e.,

$$B_i^n(t) = \left(\begin{array}{c} n \\ i \end{array} \right) (1-t)^{n-i} t^i.$$

A multivariate Bernstein polynomial of dimension k is a map from a k-simplex into the reals. For $\vec{i} = (i_0, i_1, \ldots, i_k)$ with $\sum i_j = n$, $B_{\vec{i}}^n$ will denote the degree n multivariate Bernstein polynomial given by

$$B_{\vec{i}}^n(\mathbf{t}) = \left(\begin{array}{c} n \\ \vec{i} \end{array} \right) t_0^{i_0} t_1^{i_1} \ldots t_k^{i_k},$$

where $(t_0, \ldots, t_k)$ are the barycentric coordinates of $\mathbf{t}$ relative to the domain k-simplex, and where

$$\left(\begin{array}{c} n \\ \vec{i} \end{array} \right) = \frac{n!}{i_0! i_1! \ldots i_k!}.$$

A triangular Bézier patch of degree n is given by

$$\mathbf{P}(\mathbf{t}) = \sum \mathbf{P}_{\vec{i}} B^n_{\vec{i}}(\mathbf{t}),$$

where the domain simplex is a triangle (i.e., $k = 2$). The $\mathbf{P}_{\vec{i}}$ are known as the Bézier control points of $\mathbf{P}$. A development of Bézier patches and their properties can be found in [Far90].

Surface patches will be denoted by the boldface letters $\mathbf{F}$, $\mathbf{G}$, and $\mathbf{P}$. Often $\mathbf{F}$ and $\mathbf{G}$ are adjacent patches. In this case, the Bézier control points associated only with patch $\mathbf{F}$ will be denoted by $\mathbf{F}_i$, the control points associated only with patch $\mathbf{G}$ will be denoted by $\mathbf{G}_i$, and the control points common to both patches will be denoted by $\mathbf{H}_i$.

The vertices of a triangle in the domain of a patch will be denoted by $\mathbf{p}$, $\mathbf{q}$, and $\mathbf{r}$. The corresponding vertices in the range will be described by $\mathbf{V}_p$, $\mathbf{V}_q$, and $\mathbf{V}_r$.

Blossoms

Ramshaw [Ram88] discusses the blossom, or polar form, of Bézier curves and surfaces. I will use the blossom of a Bézier simplex in Chapter 5. The blossoming principle states:

> For every polynomial $\mathbf{P}(t)$ of degree n, there is a unique, symmetric, n-affine map, $\mathbf{p}(t_1, \ldots, t_n)$, that agrees with $\mathbf{P}$ on the diagonal, i.e., $\mathbf{P}(t) = \mathbf{p}(t, \ldots, t)$.

Ramshaw defines $\mathbf{p}$ as the *blossom* of $\mathbf{P}$.

I will use a polynomial's blossom to determine its derivatives. If $\mathbf{b}$ is the blossom of $\mathbf{B}$, then

$$\mathbf{D}_{\vec{\mathbf{u}}}\mathbf{B}(\mathbf{p}) = n\mathbf{b}(\vec{\mathbf{u}}, \mathbf{p}, \ldots, \mathbf{p}),$$

where n is the degree of $\mathbf{B}$. The second derivatives are given by

$$\mathbf{D}_{\vec{\mathbf{u}}\vec{\mathbf{v}}}\mathbf{B}(\mathbf{p}) = n(n-1)\mathbf{b}(\vec{\mathbf{u}}, \vec{\mathbf{v}}, \mathbf{p}, \ldots, \mathbf{p}).$$

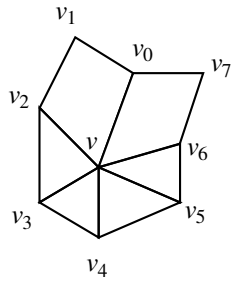


Figure 1.1: Vertex v and its vertex ring, $v_0, \ldots, v_7$.

Meshes

The input to a surface fitting scheme is a mesh. A *mesh* $M = (V, F)$ is a set of vertices V and a set of faces F. Each *face* $f \in F$ is an ordered set of $n > 2$ vertices, $f = (v_0, v_1, \ldots, v_{n-1})$, where $v_i \in V$, and $v_i \neq v_j$ for $i \neq j$. A directed edge is associated between each successive pair of face vertices v_i and $v_{(i+1) \bmod n}$. If a directed edge e appears in a mesh but the opposing directed edge does not appear, then the mesh is said to have a boundary at e.

A *ring* of neighbors of a vertex v is a list of n vertices v_i such that for $0 \leq i < n-1$, the vertices v_i, v_{i+1}, and v all appear in some face $f \in F$, and the directed edge v_i, v_{i+1} is in face f (Figure 1.1). Further, either the directed edge v_{n-1}, v_0 should be in a face containing v, or the directed edges v_{n-1}, v and v, v_0 should be boundary edges of the mesh.

A mesh is a topological data structure with data stored at the vertices. The data at each vertex varies, but a vertex will always contain a position in three space. As we are studying surfaces, I restrict the meshes to represent orientable 2-manifolds. This restriction is enforced by the requirement that each directed edge may appear no more than once in the set of faces F, and by the requirement that each vertex have exactly one ring of vertex neighbors.

Chapter 2

G^1 INTERPOLANTS

Previous work on surface construction has centered on G^1 surfaces. A G^1 surface is a C^0 surface with a continuous surface normal. In this chapter, I review several local G^1 interpolating schemes for triangular meshes, and investigate problems with these methods.

2.1 Background.

Local interpolation schemes generally construct a surface consisting of multiple surface patches. In order for the entire surface to be smooth, continuity conditions must be satisfied at patch boundaries. To avoid holes in the surface, each pair of neighboring patches must meet with C^0 continuity. To ensure that adjacent patches meet smoothly, one might also want them to meet with a continuous first derivative. However, this is not possible for surfaces of arbitrary topology [Her85]. An alternate approach is to construct the surface patches to meet with continuous tangent planes along the boundaries. The patches are then said to meet with G^1 continuity (cf. [Boe88, Her87]). Several methods of ensuring tangent plane continuity are presented in Section 2.2.

A second issue that must be addressed is what is sometimes called the *vertex consistency problem.* This problem occurs when trying to construct a single C^2 patch for each triangular face of data. The G^1 continuity conditions between patches set up a system of constraints around each data point. For a vertex with an even number of neighbors greater than four, it has been shown that this system cannot necessarily be satisfied [vW86].

There are primarily two approaches taken to avoid the vertex consistency problem. The first approach constructs multiple patches per face, thereby decoupling the cycle of constraints. A second approach is to construct patches that are not C^2 at the data points. The schemes surveyed in this chapter all use one of these two approaches.

A third approach solves the vertex consistency problem by constructing a "C^2 con-

sistent" curve network. Peters has shown that if the boundary curves adjacent to a data point all agree with a common second fundamental form, then the set of G^1 constraints can be satisfied [Pet91b]. Note that while this is a sufficient condition for satisfying the vertex consistency problem, it is not a necessary condition. For example, Loop gives an alternate approach for constructing a C^2 consistent curve network [Loo92].

2.2 Tangent Plane Continuity.

Joining surface patches with tangent plane continuity has been approached from several directions. In this section, I review tangent plane continuity schemes for polynomial patches. Most of these constructions are for quartic patches. While in general cubic patches can be used to achieve a G^1 join, Piper presented boundary data that could not be interpolated by two cubic polynomial patches meeting the G^1 conditions [Pip87]. Peters later gave a complete characterization of boundary data that cannot be interpolated by two cubic patches meeting with tangent plane continuity [Pet91c].

Farin and Piper gave sufficient conditions for two polynomial patches to meet with G^1 continuity [Far82, Pip87]. A second approach is to first create a cross-boundary tangent vector field for each boundary and then to construct patches that agree with these cross-boundary fields [CK83, Her85, Jen87, Nie87]. Peters [Pet91c] gives two approaches: the first is for cubic patches with quadratic boundaries and is similar to that of Chiyokura-Kimura. The second approach is for cubic patches with cubic boundaries; in this case, Peters finds the unique solution when it exists. While in general the solution for cubics exists, it does not exist in symmetric situations such as occur on the octahedron.

2.2.1 Farin

Farin [Far82] presented sufficient conditions for two degree n polynomial patches with a common degree $n-1$ boundary to meet with G^1 continuity. Labeling the Bézier control points as in Figure 2.1, Farin's conditions are as follows.

Given two degree n polynomial patches $\mathbf{F}$ and $\mathbf{G}$ with a common degree $n-1$

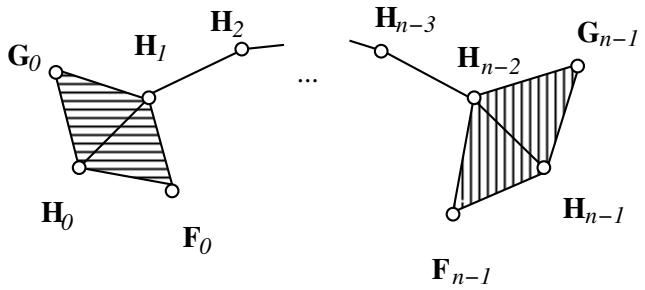


Figure 2.1: Bézier control points used in Farin's G^1 conditions. The **F**s and **G**s are from the degree n patch, while the **H**s are from the degree $n-1$ boundary.

boundary, where

$$\begin{array}{lcll} \mathbf{G}_0 & = & \alpha_1\mathbf{H}_0 + \alpha_2\mathbf{H}_1 + \alpha\mathbf{F}_0, & \alpha_1+\alpha_2+\alpha = 1, \\ \mathbf{G}_{n-1} & = & \alpha_3\mathbf{H}_{n-2} + \alpha_4\mathbf{H}_{n-1} + \alpha\mathbf{F}_{n-1}, & \alpha_3+\alpha_4+\alpha = 1, \end{array} \tag{2.1}$$

then the two patches meet with G^1 continuity if

$$\mathbf{G}_i = \frac{n-1-i}{n-1}(\alpha_1\mathbf{H}_i + \alpha_2\mathbf{H}_{i+1} + \alpha\mathbf{F}_i) + \frac{i}{n-1}(\alpha_3\mathbf{H}_{i-1} + \alpha_4\mathbf{H}_i + \alpha\mathbf{F}_i).$$

Here the **H**s are control points of the degree $n-1$ boundary, while the **F**s and **G**s are from the degree n patches. Equation 2.1 can also be formulated in terms of the areas of the shaded triangles of Figure 2.1:

$$\frac{\text{area}(\mathbf{G}_0, \mathbf{H}_0, \mathbf{H}_1)}{\text{area}(\mathbf{F}_0, \mathbf{H}_0, \mathbf{H}_1)} = \frac{\text{area}(\mathbf{G}_{n-1}, \mathbf{H}_{n-1}, \mathbf{H}_{n-2})}{\text{area}(\mathbf{F}_{n-1}, \mathbf{H}_{n-1}, \mathbf{H}_{n-2})}.$$

2.2.2 Piper

Piper developed sufficient conditions for two quartic patches with quartic boundaries to meet with G^1 continuity [Pip87]. He began by noting that the following equation must hold for patches **F** and **G** to meet G^1:

$$e(t)\mathbf{I}(t) + f(t)\mathbf{J}(t) + g(t)\mathbf{K}(t) + h(t)\mathbf{L}(t) = \vec{\mathbf{0}}, \tag{2.2}$$

where e, f, g, and h are scalar functions such that

$$e(t) + f(t) + g(t) + h(t) = 0$$

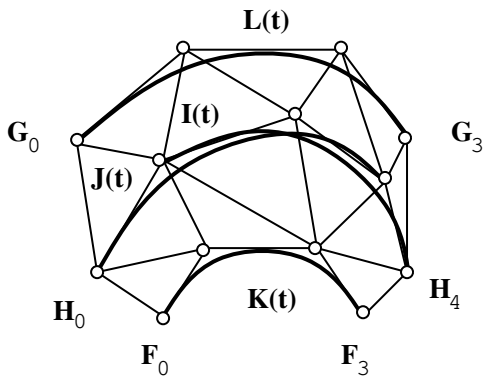


Figure 2.2: The curves used in Piper's G^1 conditions.

for $t \in [0,1]$, and where

$$\begin{aligned}
\mathbf{I}(t) &= \sum B_i^3(t)\mathbf{H}_{i+1}, \\
\mathbf{J}(t) &= \sum B_i^3(t)\mathbf{H}_i, \\
\mathbf{K}(t) &= \sum B_i^3(t)\mathbf{F}_i, \\
\mathbf{L}(t) &= \sum B_i^3(t)\mathbf{G}_i.
\end{aligned}$$

These curves are illustrated in Figure 2.2. The eight control points $\mathbf{F}_0$, $\mathbf{F}_3$, $\mathbf{G}_0$, $\mathbf{G}_3$, $\mathbf{H}_0$, $\mathbf{H}_1$, $\mathbf{H}_3$, and $\mathbf{H}_4$ are known; Piper sets the other five control points to achieve a G^1 join.

Piper restricts e, f, g, and h to be linear functions, reducing Equation 2.2 to a 3×5 system of linear equations, where the unknowns are the control points $\mathbf{F}_1$, $\mathbf{F}_2$, $\mathbf{G}_1$, $\mathbf{G}_2$, and $\mathbf{H}_2$. In certain situations, the functions e, f, g, and h are all constant functions. In this case, the 3×5 system of equations reduces to a 2×5 system. In summary, these constraints represent underdetermined conditions on the unknown control points to achieve a G^1 join.

2.2.3 Chiyokura-Kimura, Herron, Jensen

As mentioned earlier, one approach to creating surface patches that meet with G^1 continuity is to first construct a field of cross-boundary tangent vectors along the boundary using data common to both patches. This cross-boundary tangent field, together with the boundary curve's first derivative vector, defines a tangent plane field along the boundary. Next, two patches are constructed, one on either side of

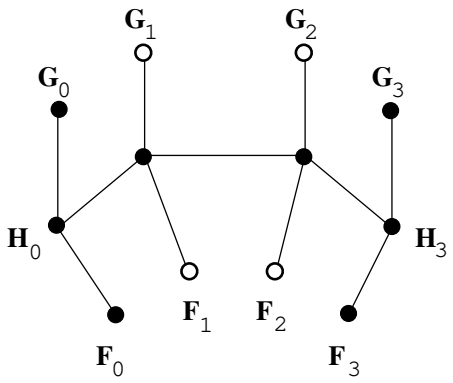


Figure 2.3: Boundary data used by Chiyokura and Kimura's construction. The solid points are the known control points; the hollow points are constructed so that the two patches meet G^1. Note that the **H**s are cubic control points, while the **F**s and **G**s are quartic control points.

the boundary, that match this tangent plane field along the boundary. The two patches will therefore meet with G^1 continuity. This is the approach taken by many authors [CK83, Her85, Jen87, Nie87]. Peters [Pet91c] also uses this method for joining cubic patches with quartic boundaries. I present now the method of Chiyokura and Kimura and show its relationship to other constructions.

Although Chiyokura and Kimura's cross-boundary construction was originally intended for rectangular patches, it readily extends to the construction of quartic triangular Bézier patches [SS87]. The method uses the boundary data for two adjacent patches to construct the Bézier control points that influence the tangent plane behavior along their common boundary. The boundary data shared by two patches **F** and **G** consist of a cubic polynomial boundary curve and a pair of tangent vectors at each end of the boundary curve (Figure 2.3 shows this data in Bézier form). Two quartic interior Bézier control points for each patch are set to match the tangent plane field. I give the construction for **F**; the construction for **G** is identical.

The cross-boundary tangent vector field is defined by linearly blending two vectors, $\hat{\mathbf{C}}_0$ and $\hat{\mathbf{C}}_1$, one in each of the tangent planes at the endpoints. Chiyokura and Kimura choose these vectors to be unit vectors perpendicular to the tangents at the boundary curve's endpoints, i.e., $< \hat{\mathbf{C}}_0, \mathbf{H}_1 - \mathbf{H}_0 > = < \hat{\mathbf{C}}_1, \mathbf{H}_3 - \mathbf{H}_2 > = 0$. For patch **F**, the

cross-boundary field is given by

$$\vec{\mathbf{C}}(t) = (1-t)\hat{\mathbf{C}}_0 + t\hat{\mathbf{C}}_1.$$

The two perpendiculars $\hat{\mathbf{C}}_0$ and $\hat{\mathbf{C}}_1$ are unique, up to sign. The signs are chosen so that k_0 and k_1 below are positive. $\vec{\mathbf{C}}(t)$ together with $\mathbf{H}'(t)$ completely specifies the tangent plane field along the boundary.

For $\mathbf{F}$ to agree with the tangent plane field given by $\mathbf{H}'(t)$ and $\vec{\mathbf{C}}(t)$, there must exist functions $k(t)$ and $h(t)$ such that

$$\mathbf{D}_{\vec{\mathbf{r}}(t)}\mathbf{F}(0,t,1-t) = k(t)\cdot\vec{\mathbf{C}}(t) + h(t)\cdot\mathbf{H}'(t), \tag{2.3}$$

where $\vec{\mathbf{r}}(t)$ is the radial direction in the domain of $\mathbf{F}$ (i.e., if the domain of $\mathbf{F}$ is $\triangle\mathbf{pqs}$ where $\mathbf{F}(\mathbf{q}) = \mathbf{H}_0$ and $\mathbf{F}(\mathbf{s}) = \mathbf{H}_3$, then $\vec{\mathbf{r}}(t) = ((1-t)\mathbf{q} + t\mathbf{s}) - \mathbf{p}$).

Let $\vec{\mathbf{F}}_i = \mathbf{F}_i - \mathbf{H}_i$ and $\vec{\mathbf{H}}_i = \mathbf{H}_{i+1} - \mathbf{H}_i$. The values of $k(t)$ and $h(t)$ are determined at the endpoints by evaluating Equation 2.3 at $t = 0$ and $t = 1$:

$$\vec{\mathbf{F}}_0 = k_0\cdot\hat{\mathbf{C}}_0 + h_0\cdot\vec{\mathbf{H}}_0, \tag{2.4}$$

$$\vec{\mathbf{F}}_3 = k_1\cdot\hat{\mathbf{C}}_1 + h_1\cdot\vec{\mathbf{H}}_2, \tag{2.5}$$

where $k_0 = k(0)$, $k_1 = k(1)$, $h_0 = h(0)$, and $h_1 = h(1)$. For h and k to interpolate these endpoint conditions, they both must be at least linear functions. If we restrict them to be no more than linear, then each is uniquely determined:

$$\begin{aligned} k(t) &= k_0\cdot(1-t) + k_1\cdot t, \\ h(t) &= h_0\cdot(1-t) + h_1\cdot t. \end{aligned}$$

Rewriting Equation 2.3 in the cubic Bernstein basis, the coefficients of $B_1^3(t)$ and $B_2^3(t)$ determine the desired interior control points:

$$\mathbf{F}_1 = \frac{1}{3}\{(k_0+k_1)\hat{\mathbf{C}}_0 + k_0\hat{\mathbf{C}}_1 + 2h_0\vec{\mathbf{H}}_1 + h_1\vec{\mathbf{H}}_0\} + \mathbf{H}_1, \tag{2.6}$$

$$\mathbf{F}_2 = \frac{1}{3}\{k_1\hat{\mathbf{C}}_0 + (k_0+k_1)\hat{\mathbf{C}}_1 + h_0\vec{\mathbf{H}}_2 + 2h_1\vec{\mathbf{H}}_1\} + \mathbf{H}_2. \tag{2.7}$$

There is still some freedom left in Equation 2.3. If $h(t)$ is a linear function, then the product $k(t)\cdot\vec{\mathbf{C}}(t)$ must be a polynomial of no higher than third degree. In the

above formulation, this product is only a quadratic polynomial. Either $k(t)$ or $\vec{\mathbf{C}}(t)$ can be increased from a linear function to a quadratic function. Increasing $k(t)$ to a quadratic function gives a scalar degree of freedom, while increasing the degree of $\vec{\mathbf{C}}(t)$ yields a vector degree of freedom. Jensen [Jen87] used the former generalization. He used the same linear blend of unit vectors for $\vec{\mathbf{C}}(t)$, but used the following quadratic scale function:

$$k^*(t) = k_0 \cdot u_0(t) + C \cdot \frac{(k_0 + k_1)}{2} u_1(t) + k_1 \cdot u_2(t),$$

where

$$u_0(t) = 2t^2 - 3t + 1, \quad u_1(t) = 4t - 4t^2, \quad u_2(t) = 2t^2 - t,$$

and C is a scalar shape parameter.[1] For $C = 1$, $k^*(t) = k(t)$.

A second degree of freedom in Equation 2.3 is in the choice of $\hat{\mathbf{C}}_0$ and $\hat{\mathbf{C}}_1$. These two vectors may be chosen in any fashion that uses information available to both patches, where the construction from both sides gives the same vectors with opposite sign. For example, in a later paper [Chi86], Chiyokura defines $\hat{\mathbf{C}}_0$ and $\hat{\mathbf{C}}_1$ as

$$\hat{\mathbf{C}}_0 = \frac{\mathbf{G}_0 - \mathbf{F}_0}{|\mathbf{G}_0 - \mathbf{F}_0|},$$

$$\hat{\mathbf{C}}_1 = \frac{\mathbf{G}_3 - \mathbf{F}_3}{|\mathbf{G}_3 - \mathbf{F}_3|}.$$

Note that this definition of $\hat{\mathbf{C}}_0$ and $\hat{\mathbf{C}}_1$ is affine invariant and requires knowledge about both patches neighboring the boundary, whereas the earlier definition is not affine invariant but uses only information about the boundary curve.

Although Herron [Her85] has a different approach for constructing a G^1 join, his construction and the Chiyokura-Kimura construction result in the same field of cross-boundary tangent vectors along the boundary curves.

2.2.4 Peters

Peters [Pet91c] gives two constructions for tangent plane continuity. The first is for joining two cubics patches sharing a quadratic boundary, while the second is for joining two cubics with cubic boundaries.

---

[1] In Jensen's paper, u_1 is given as $u_1(t) = 4t - t^2$. However, without the factor of 4 scaling t^2, $k^*(1)$ does not interpolate k_1.

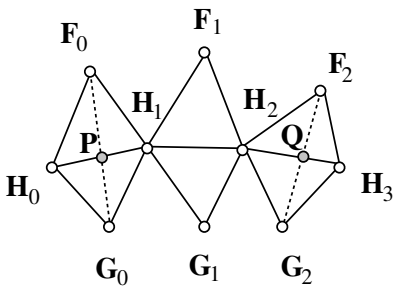


Figure 2.4: Control points along a cubic boundary.

The construction for cubics with quadratic boundaries is similar to that of Chiyokura-Kimura: each patch is constructed to agree with a cross-boundary function $\vec{\mathbf{C}}(t)$. Let $\mathbf{F}$ and $\mathbf{G}$ be two cubic patches with a shared degree raised [Far90] quadratic boundary $\mathbf{H}$ (Figure 2.4), with normals $\hat{\mathbf{N}}_0$ at $\mathbf{H}_0$ and $\hat{\mathbf{N}}_1$ at $\mathbf{H}_3$, and let

$$\vec{\mathbf{H}}_i = \mathbf{H}_{i+1} - \mathbf{H}_i, \quad \vec{\mathbf{F}}_i = \mathbf{F}_i - \mathbf{H}_i, \quad \vec{\mathbf{H}}_0^{\perp} = \hat{\mathbf{N}}_0 \times \vec{\mathbf{H}}_0, \quad \vec{\mathbf{H}}_2^{\perp} = \hat{\mathbf{N}}_1 \times \vec{\mathbf{H}}_2.$$

For $\mathbf{F}$ and $\mathbf{G}$ to meet with tangent plane continuity, the location of $\mathbf{F}_1$ is offset from $\mathbf{H}_1$ by a vector $\vec{\mathbf{F}}_1$ given by

$$\begin{aligned} 2\vec{\mathbf{F}}_1 \;=\; & \frac{<\vec{\mathbf{H}}_0^{\perp},\ \vec{\mathbf{F}}_0>}{<\vec{\mathbf{H}}_0^{\perp},\ \vec{\mathbf{H}}_0^{\perp}>}\vec{\mathbf{H}}_2^{\perp} + \frac{<\vec{\mathbf{H}}_2^{\perp},\ \vec{\mathbf{F}}_2>}{<\vec{\mathbf{H}}_2^{\perp},\ \vec{\mathbf{H}}_2^{\perp}>}\vec{\mathbf{H}}_0^{\perp} + \\ & \frac{<\vec{\mathbf{H}}_0,\ \vec{\mathbf{F}}_0>}{<\vec{\mathbf{H}}_0,\ \vec{\mathbf{H}}_0>}\vec{\mathbf{H}}_2 + \frac{<\vec{\mathbf{H}}_2,\ \vec{\mathbf{F}}_2>}{<\vec{\mathbf{H}}_2,\ \vec{\mathbf{H}}_2>}\vec{\mathbf{H}}_0. \end{aligned} \tag{2.8}$$

The location for $\mathbf{G}_1$ is determined in a similar manner.

If the boundary curve $\mathbf{H}$ is cubic, then a small system of linear equations must be solved to determine the points $\mathbf{F}_1$ and $\mathbf{G}_1$. In some case, such as Piper's counter example, this system fails to have a solution. Peters gives the following characterization. Referring to Figure 2.4, let $\mathbf{P}$ be the intersection of the segments $\mathbf{H}_0\mathbf{H}_1$ and $\mathbf{F}_0\mathbf{G}_0$, and let $\mathbf{Q}$ be the intersection of the segments $\mathbf{H}_2\mathbf{H}_3$ and $\mathbf{F}_2\mathbf{G}_2$. The system always has a solution unless

$$\frac{\mathbf{F}_0\mathbf{P}}{\mathbf{G}_0\mathbf{P}} = \frac{\mathbf{F}_2\mathbf{Q}}{\mathbf{G}_2\mathbf{Q}}.$$

In this case, there is a solution only if

$$\frac{\mathbf{H}_0\mathbf{P}}{\mathbf{H}_1\mathbf{P}} = \frac{\mathbf{H}_2\mathbf{Q}}{\mathbf{H}_3\mathbf{Q}}.$$

If the first equality holds, but the second one fails to hold, then no placement of $\mathbf{F}_1$ and $\mathbf{G}_1$ yields a G^1 join along the boundary between $\mathbf{H}_0$ and $\mathbf{H}_3$.

2.3 Parametric Schemes.

The schemes surveyed in this chapter fall into two categories: the split domain schemes [Jen87, Han91, Pet91c, Pip87, SS87] and the convex combination schemes [Gre86, HP89, Her85, Lon86, Nie87]. All methods build a surface for each triangular face by first computing boundary curves and cross-boundary derivatives around the triangle, and then constructing one or more patches that match this boundary data. The two categories of schemes differ in their solution to the vertex consistency problem.

A third class of interpolatory schemes are the subdivision schemes [DLG90, CM89]. These schemes construct a surface by successively adding vertices, faces, and edges to the mesh; the surface is defined to be the limit of the mesh. Currently, I have not investigated subdivision schemes.

2.3.1 Split Domain Schemes

There is an extensive body of literature that discusses the properties of polynomial Bézier patches (cf. Farin [Far90]). If the input data could be fit with Bézier patches, then we could draw on this body of knowledge to compute various surface properties. However, if the boundary curves are constructed independently of each other, then a single Bézier patch cannot in general be used to interpolate each face, since the vertex consistency problem cannot in general be solved. Split domain schemes avoid this problem by constructing three patches per face, splitting the domain triangle into three subtriangles, as was done by Clough and Tocher for scalar-valued data [CT65, Far85].

After splitting, each subpatch is used to interpolate the data along one boundary. Splitting allows the data along each boundary to be matched independently of the data on the other two boundaries. The remaining degrees of freedom are used to make the three subpatches meet with G^1 continuity along their internal boundaries.

Three of the split domain schemes presented here construct quartic polynomial patches. Figure 2.5 schematically shows a labeling of the Bézier control points of these

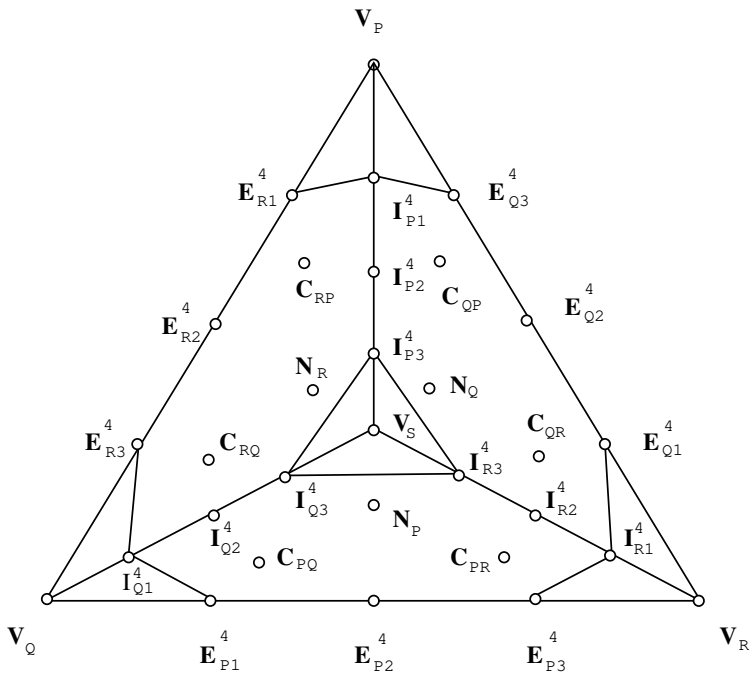


Figure 2.5: Bézier control points for split domain schemes.

patches. As the schemes compute cubic boundaries, we will need to refer to both the cubic and quartic control points of the boundaries. Symbols such as $\mathbf{I}^4$ and $\mathbf{E}^4$ refer to the quartic control points, whereas $\mathbf{I}^3$ and $\mathbf{E}^3$ refer to the corresponding cubic control points. The appearance of the resulting formulas is unfortunately visually complex, but it allows us to be specific about which quantities are to be used in the calculations.

Shirman-Séquin

The following split domain scheme was proposed by Shirman and Séquin [SS87] (a necessary correction to the method appears in [SS91]). For each face, three quartic triangular patches are constructed to interpolate the data points. The construction assumes that cubic boundary curves have been constructed and subsequently degree raised to quartics.

Farin's G^1 conditions are used on the internal boundaries to ensure that the three patches meet each other with G^1 continuity. For $ijk \in \{PQR, QRP, RPQ\}$, the

following relationships are thus imposed:

$$\mathbf{E}^4_{j3} = \alpha_{i2}\mathbf{I}^3_{i1} + \alpha_{i1}\mathbf{V}_i + \alpha_i\mathbf{E}^4_{k1}, \tag{2.9}$$

where $\alpha_{i1} + \alpha_{i2} + \alpha_i = 1$. Similarly, at the other end of the internal boundaries, the following relationships hold:

$$\mathbf{I}^4_{k3} = \alpha_{i3}\mathbf{I}^3_{i2} + \alpha_{i4}\mathbf{V}_S + \alpha_i\mathbf{I}^4_{j3}, \tag{2.10}$$

where $\alpha_{i3} + \alpha_{i4} + \alpha_i = 1$. For reasons of symmetry, we set $\alpha_i = -1$ for all i.

A setting of the αs determines the control points $\mathbf{I}^3_{i1}$ according to Equation 2.9. The method of Chiyokura and Kimura can now be used to establish tangent plane continuity across the external boundaries (Equations 2.6 and 2.7), thus determining six of the interior control points (the $\mathbf{C}_{ij}$s).

Using these αs, Farin's continuity conditions set up a system of equations involving the interior control points that has the following solution:

$$\mathbf{I}^3_{i2} = -\frac{\alpha_{i3}}{2\alpha_{i2}}\mathbf{V}_i - \left(\frac{\alpha_{i1}}{\alpha_{i2}} + \frac{\alpha_{i4}}{2\alpha_{i2}}\right)\mathbf{I}^3_{i1} + \frac{3}{2\alpha_{i2}}\left(\mathbf{C}_{ji} + \mathbf{C}_{ki}\right),$$

$$\begin{aligned}\mathbf{N}_i \;=\; & -\frac{\alpha_{i3}}{3}\mathbf{I}^3_{i1} + \frac{\alpha_{i3}}{3}(\mathbf{I}^3_{j1} + \mathbf{I}^3_{k1}) + \left(\frac{\alpha_{i2}}{18} - \frac{\alpha_{i1} + 2\alpha_{i4}}{6}\right)\mathbf{I}^3_{i2} + \\ & \left(\frac{\alpha_{i2}}{18} + \frac{\alpha_{i1} + 2\alpha_{i4}}{6}\right)\left(\mathbf{I}^3_{k2} + \mathbf{I}^3_{j2}\right).\end{aligned}$$

Setting $\mathbf{V}_S$ to be the centroid of the $\mathbf{I}^3_{i2}$s fixes the following αs:

$$\alpha_{i3} = -\frac{3}{4},\ \alpha_{i4} = \frac{11}{4}.$$

α_{i1} and α_{i2} are now related by $\alpha_{i1} + \alpha_{i2} = 2$. This leaves a scalar shape parameter to influence the shape of the patch interiors. By setting $\alpha_{i1} = -\frac{1}{4}$ and $\alpha_{i2} = \frac{9}{4}$ the $\mathbf{I}^3_{i1}$s are placed as in Shirman and Séquin's paper (i.e., $\mathbf{I}^3_{i1}$ will lie at the centroid of the triangle $\mathbf{V}_i\mathbf{E}^3_{j1}\mathbf{E}^3_{k3}$).

Jensen

Jensen's method [Jen87] is similar to the method of Shirman and Séquin, with two notable differences. The first difference is in the construction of the cross-boundary

tangent vector field along the boundaries. As mentioned earlier, both methods compute a linearly varying cross-boundary tangent vector field. However, Jensen then uses a quadratic scaling function instead of the linear one used by Shirman-Séquin. The second way in which Jensen's method differs from Shirman and Séquin's method is in the construction of the interior boundaries. While Shirman and Séquin construct their patches to meet with G^1 continuity, Jensen uses C^1 conditions for the construction of the interior points.

Piper

Piper's construction [Pip87] differs from the above two split domain schemes. First, a single cubic patch is constructed for each face. Next, this cubic patch is subdivided at the centroid into three cubic subpatches. These patches are then modified to produce "candidate" patches, which are degree raised to quartic patches. The control points of the quartic patches are adjusted so that they satisfy the tangent plane continuity conditions of Section 2.2.2 along the exterior boundaries. Because of rank deficiency of the linear system, there is more than one set of such control points that satisfy Piper's continuity conditions. The set chosen is the one closest to the candidate control points in a least squares sense. Finally, the control points along the interior boundaries are adjusted so that the three patches meet each other with C^1 continuity.

Hansford

I briefly sketch Hansford's scheme [Han91]. The method is a split domain scheme producing rational Bézier patches. The construction (shown diagrammatically in Figure 2.6) proceeds as follows: 1) Conic sections are fit to a triangle of data to form the boundaries of the patch. 2) These curves are extended to a rational quadratic patch filling the entire triangular face. 3) The patch is degree raised to a rational quartic, and then split to form three patches. 4) The rational patches are projected into a four dimensional space, where they become polynomial patches. 5) The control points of these polynomial patches are adjusted to achieve a G^1 join, and then 6) projected back into the three dimensional space yielding rational quartic patches that meet with tangent plane continuity.

Note the similarities between this scheme and Piper's scheme: we first construct a single patch, split it to form candidate patches, and then adjust the candidate patches

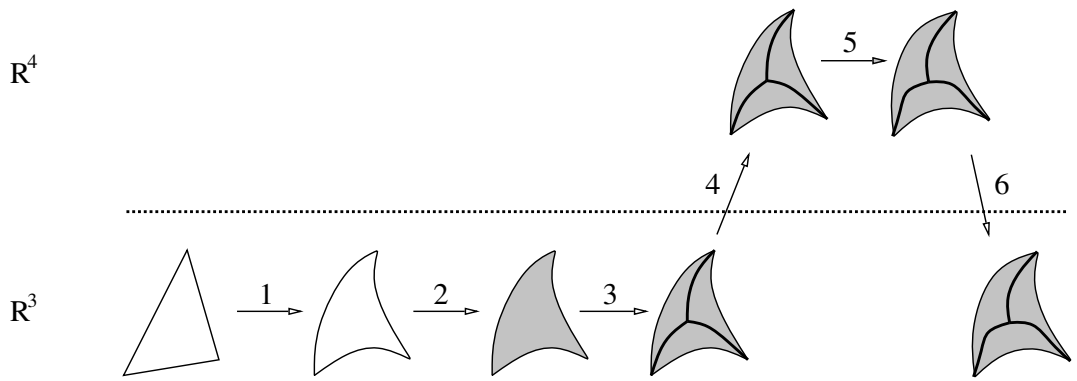


Figure 2.6: Hansford.

to achieve G^1 continuity. The differences between the two methods are in the type of surface patch (rational instead of polynomial) and in the manner the G^1 conditions are satisfied.

Peters

Peters [Pet91c] presents a split domain scheme for meshes with faces that have an arbitrary number of sides. Here, I present the restriction of his scheme to triangular faces. The patches constructed by Peters' scheme are generally cubic, although sometimes a quartic patch must be used. The labels of the cubic control points are similar to the labels used for Shirman-Séquin's scheme. Each interior control point $\mathbf{Q}_i$ replaces the triple of points $\mathbf{N}_i$, $\mathbf{C}_{ij}$, and $\mathbf{C}_{ik}$.

The scheme begins by first building boundary curves. The boundaries are constructed to be of minimum degree; they may be linear, quadratic, or cubic. For boundary endpoints $\mathbf{V}_r$, $\mathbf{V}_s$, the determination of the degree is made based on σ_0 and σ_1, where

$$\begin{aligned} \sigma_0 &= < \hat{\mathbf{N}}_r,\ (\mathbf{V}_s - \mathbf{V}_r) > \\ \sigma_1 &= < \hat{\mathbf{N}}_s,\ (\mathbf{V}_r - \mathbf{V}_s) > . \end{aligned} \tag{2.11}$$

If both σs are zero, then Peters constructs a linear boundary. If the product $\sigma_0\sigma_1$ is greater than or equal to zero, a quadratic curve can be used. If the product of the σs is negative, then the curve must have an inflection point, and a cubic curve is used.

(See Section 2.4.5 for a discussion on Peters' quadratic boundary curves.)

The interior control points are set in the following order: First, the $\mathbf{I}^3_{i1}$s are set, then the $\mathbf{Q}_i$s, then the $\mathbf{I}^3_{i2}$s, and finally $\mathbf{V}_S$ is set. The $\mathbf{I}^3_{i1}$s, the $\mathbf{I}^3_{i2}$s, and $\mathbf{V}_S$ are all set using averaging, similar to the method Shirman-Séquin use. More precisely:

$$\mathbf{I}^3_{i1} = \frac{\mathbf{V}_i + \mathbf{E}^3_{ij} + \mathbf{E}^3_{ik}}{3},$$

$$\mathbf{I}^3_{i2} = \frac{\mathbf{I}^3_{i1} + \mathbf{Q}_j + \mathbf{Q}_k}{3},$$

$$\mathbf{V}_S = \frac{\mathbf{I}^3_{i2} + \mathbf{I}^3_{j2} + \mathbf{I}^3_{k2}}{3}.$$

The construction of the $\mathbf{Q}_i$s is given in section 2.2.4.

2.3.2 Convex Combination Schemes

Convex combination schemes create a single patch for each face. The patches are C^2 everywhere except at the vertices. This successfully avoids the vertex consistency problem by having inconsistently defined mixed partial (i.e., *twist*) terms at the patch corners. Each patch is constructed by first building boundary curves and tangent plane fields along these boundary curves. Next, three patches are created, each of which interpolates part of the boundary data. Finally, a single patch is formed by taking a convex combination of the three patches in such a way that the resulting patch interpolates all the boundary data.

Nielson

Nielson [Nie87] presented two surface construction techniques. The first is a "transfinite method." The input to this scheme is a "triangle" of three boundary curves together with a tangent plane field along each of these curves. The only requirements on each input curve are that it is C^1, and that it meet the other curves with a consistent tangent plane at each vertex. The tangent plane fields are specified using a normal vector field. Each tangent plane field is the field of planes perpendicular to the normal at each point along a boundary curve. The normal fields are also required to be C^1, with the further restrictions that they must be non-zero everywhere and meet C^0 at the vertices. A surface patch is then constructed that matches this data, using a *side-vertex* method.

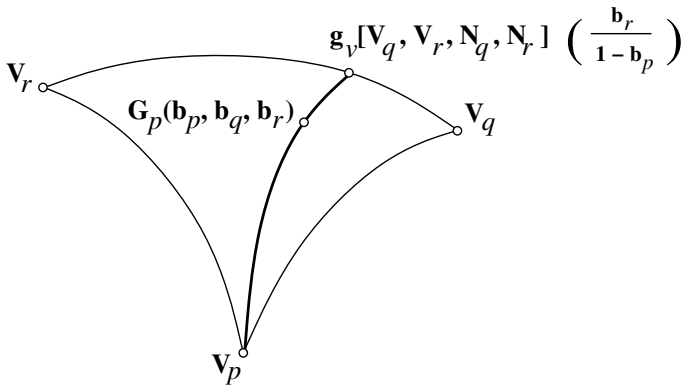


Figure 2.7: Side-vertex method.

The second scheme is a side-vertex method that fits into the problem domain of this work. The method proceeds by first constructing three boundary curves, one corresponding to each edge of the input triangle. Next, three patches are created, one for each boundary/opposite-vertex pair. The interior of each patch is constructed by passing curves from points along the boundary (or "side") to the opposite vertex. Hence the name "side-vertex", as shown in Figure 2.7. The three patches are then blended together to form the final patch.

All curves are constructed from two points and associated normals. We assume the existence of a curve construction operator $\mathbf{g}_v$ that takes two vertices with normals and constructs a curve:

$$\mathbf{g}_v[\mathbf{V}_0, \mathbf{V}_1, \hat{\mathbf{N}}_0, \hat{\mathbf{N}}_1](t),$$

such that $\mathbf{g}_v(0) = \mathbf{V}_0$, $\mathbf{g}_v(1) = \mathbf{V}_1$, $< \mathbf{g}_v'(0),\ \hat{\mathbf{N}}_0 > = 0$, and $< \mathbf{g}_v'(1)\, \hat{\mathbf{N}}_1 > = 0$. We also assume the existence of a normal field constructor $\mathbf{g}_n$ that constructs a continuous normal field along the curve $\mathbf{g}_v$, where $\mathbf{g}_n$ is required to interpolate $\hat{\mathbf{N}}_0$ and $\hat{\mathbf{N}}_1$ at the endpoints.

The construction proceeds by building three patches, $\mathbf{G}_i$, $i \in \{p, q, r\}$ defined as:

$$\begin{aligned}\mathbf{G}_i(b_p, b_q, b_r) \;=\; \mathbf{g}_v \Bigg[& \mathbf{V}_i,\ \mathbf{g}_v[\mathbf{V}_j, \mathbf{V}_k, \hat{\mathbf{N}}_j, \hat{\mathbf{N}}_k]\left(\frac{b_k}{1-b_i}\right), \\ & \hat{\mathbf{N}}_i,\ \mathbf{g}_n[\mathbf{V}_j, \mathbf{V}_k, \hat{\mathbf{N}}_j, \hat{\mathbf{N}}_k]\left(\frac{b_k}{1-b_i}\right)\Bigg](1-b_i),\end{aligned}$$

where b_i, b_j, b_k are the barycentric coordinates of the domain point. Nielson notes the following two properties of $\mathbf{G}_i$:

1. $\mathbf{G}_i$ interpolates all three of the boundaries.

2. $\mathbf{G}_i$ interpolates the tangent plane field of the boundary opposite vertex $\mathbf{V}_i$.

The final surface is defined to be

$$\mathbf{G}[\mathbf{V}_p, \mathbf{V}_q, \mathbf{V}_r, \hat{\mathbf{N}}_p, \hat{\mathbf{N}}_q, \hat{\mathbf{N}}_r] = \beta_p \mathbf{G}_p + \beta_q \mathbf{G}_q + \beta_r \mathbf{G}_r,$$

where

$$\beta_i = \frac{b_j b_k}{b_p b_q + b_q b_r + b_r b_p}. \tag{2.12}$$

Nielson shows that if three surfaces having properties 1 and 2 are blended with these β_i, then the resulting surface will interpolate all the boundary curves and tangent fields. The theorem is true for a large class of $\mathbf{g}_v$ and $\mathbf{g}_n$. The operator $\mathbf{g}_v$ that Nielson presents constructs tangent vectors from the two normals and interpolates the two points and these vectors with a cubic polynomial curve.

A variation of Nielson's scheme appears in [HFN91]. This scheme uses generalized conics [Bal74] for the operator $\mathbf{g}_v$. (A generalized conic is a rational cubic curve that, in the absence of inflection points, is a degree raised conic.) While this variation of Nielson's scheme produces good approximations to the sphere, it exhibits the other shape defects mentioned below.

Triangular Gregory Patches

Triangular Gregory patches are a variant of Gregory squares [Gre74]. The key idea is that the twist term at the vertices is a blend of two twists, one for each boundary curve incident to the vertex. The scheme I present here is due to Longhi [Lon86].

After constructing cubic boundary curves, this scheme uses the method of Chiyokura and Kimura to find a pair of cross-boundary control points for each edge (Figure 2.8). In this figure, points $\mathbf{I}_{ji}$ and $\mathbf{I}_{ki}$ are the points constructed for the boundary associated with $b_i = 0$. When evaluating the patch at the domain point (b_p, b_q, b_r), the two interior control points near each of the corner vertices are blended to form a single point, reducing the six interior points to three control points. Points $\mathbf{I}_{ij}$ and $\mathbf{I}_{ik}$ are blended to produce the control point $\mathbf{I}_i$ with the following blend:

$$\mathbf{I}_i = \frac{b_k(1-b_j)\mathbf{I}_{ik} + b_j(1-b_k)\mathbf{I}_{ij}}{b_k(1-b_j) + b_j(1-b_k)}.$$

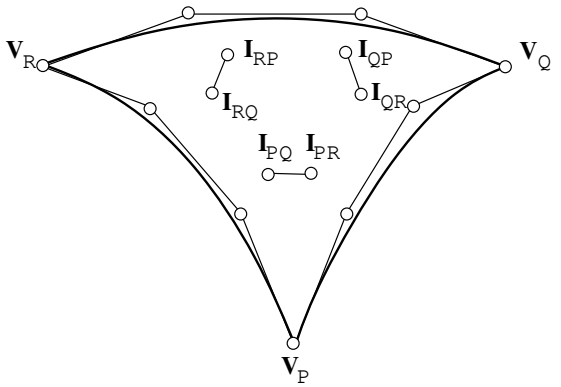


Figure 2.8: Control points of a triangular Gregory patch.

This yields a quartic Bézier patch that is evaluated at (b_p, b_q, b_r) to give a point on the surface.

Alternatively, we can think of triangular Gregory patches as a convex combination scheme. By putting the blending functions of the $\mathbf{I}_i$s over a common denominator, the scheme can be rewritten as a convex combination of seven quartic Bézier patches. In this form, the blending functions are sixth degree rational polynomials, four degrees higher than the ones used by Nielson.

Herron

In [Her85], Herron introduced a triangular surface fitting scheme in the following form:

$$\mathbf{F} = \mathcal{C} + b_p b_q b_r \mathcal{X},$$

where $\mathcal{C}$ interpolates the boundary curves and $\mathcal{X}$ is presented as a function that adjusts the cross-boundary tangents of $\mathcal{C}$ to meet specified tangent plane fields. Although it can be shown that $\mathbf{F}$ is a point-valued function, neither $\mathcal{C}$ nor $\mathcal{X}$ represent affine geometric entities (points, vectors, etc).

I present here an alternative description of Herron's method that is more geometric in nature. First observe that $\mathbf{F}$ can be rewritten as

$$\mathbf{F} = \sum_{i=p,q,r} \beta_i \mathbf{F}_i, \tag{2.13}$$

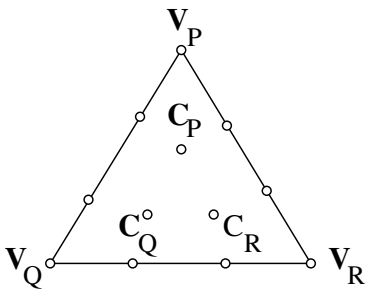


Figure 2.9: Bézier patch for Herron's scheme. The boundary points shown in the figure are cubic control points, while the $\mathbf{C}_i$s are quartic control points.

where

$$\beta_i = \frac{b_j b_k}{b_i b_j + b_j b_k + b_k b_i},$$

and where each $\mathbf{F}_i$ is a quartic Bézier patch whose construction is given below. Note that these β_i are identical to those used by Nielson (Equation 2.12).

The input required by Herron's scheme is a triangle of points and the six boundary curve tangents at those points. Cubic Hermite interpolation is used on the boundary data to construct the boundary curves of $\mathbf{F}_i$. This sets all the exterior control points for $\mathbf{F}_i$, leaving only the three interior control points, $\mathbf{C}_P$, $\mathbf{C}_Q$, and $\mathbf{C}_R$, undetermined (Figure 2.9).

For patch $\mathbf{F}_i$, points $\mathbf{C}_j$ and $\mathbf{C}_k$ are constructed using the triangular version of Chiyokura-Kimura. Equations 2.6 and 2.7 of Section 2.2.3 give formulas for these two points. The final control point $\mathbf{C}_i$ of $\mathbf{F}_i$ can be considered a free parameter. Herron's setting for this parameter is given in Appendix A.

Herron's scheme, then, can be thought of as a hybrid scheme using the cross-boundary tangent method of Chiyokura and Kimura for constructing triangular Bézier patches and the weighting functions of Nielson to produce the final patch. Note that we can use Nielson's proof above to show that $\mathbf{F}$ interpolates the tangent plane fields along all edges[2].

A third way to view Herron's scheme is as a "three point" triangular Gregory

[2] It is interesting to note that Herron's development of the method predated the publication of Chiyokura and Kimura [CK83] and Nielson [Nie87]

patch. By rewriting Equation 2.13 in Bernstein form, Herron's method can be expressed as a quartic Bézier patch, where each internal control point is a blend of three of the internal control points of the $\mathbf{F}_i$s. In Bézier form, then, the three point triangular Gregory patch blends together fewer patches than the two point Gregory patch (three patches instead of seven patches) and uses rational quadratic blending functions instead of the sextic rational polynomials of the two point scheme.

Gregory and Charrot

The scheme of Gregory and Charrot [GC80] is a convex combination scheme. The surface created is a blend of three patches, where each patch is a "two-sided interpolant" (i.e., a patch interpolating the cross-boundary information along two boundaries). These two-sided interpolants are constructed using what is known as a "boolean sum."

Again, as a first step, boundary curves are created that interpolate the corner data. The boundary curves are denoted by a single function $\mathbf{f}$, whose domain is the boundary of the domain triangle. Cross-boundary functions are then built along the edges; I denote the cross-boundary function along edge i as $\vec{\mathbf{f}}_i$.

If $\mathbf{w}$ is a point in the domain, the two-sided interpolant $\mathbf{P}_i$ (associated with corner i) is given by

$$\begin{aligned}\mathbf{P}_i(\mathbf{w}) \;=\; & \mathbf{f}(\mathbf{E}^i_{i+1}) + w_{i+1}\vec{\mathbf{f}}_{i+1}(\mathbf{E}^i_{i+1}) + \mathbf{f}(\mathbf{E}^i_{i-1}) + w_{i-1}\vec{\mathbf{f}}_{i-1}(\mathbf{E}^i_{i-1}) \\ & -\mathbf{f}(\mathbf{V}_i) - w_{i+1}\vec{\mathbf{f}}_{i+1}(\mathbf{V}_i) - w_{i-1}\vec{\mathbf{f}}_{i-1}(\mathbf{V}_i) \\ & -w_{i-1}w_{i+1}\left[\frac{w_{i-1}}{w_{i+1}+w_{i-1}}\vec{\mathbf{f}}_{i+1,i-1}(\mathbf{V}_i) + \frac{w_{i+1}}{w_{i+1}+w_{i-1}}\vec{\mathbf{f}}_{i-1,i+1}(\mathbf{V}_i)\right]\end{aligned}$$

where w_i are the barycentric coordinates of $\mathbf{w}$, and $\mathbf{E}^i_{i+1}$ is the intersection of the edge e_{i+1} with the line through $\mathbf{w}$ parallel to e_{i-1} (Figure 2.10). Note that the mixed partial term simplifies if the boundaries have consistent mixed partials.

The final patch is given by

$$\mathbf{P}(\mathbf{w}) = \sum_{i=0}^{2} \alpha_i \mathbf{P}_i(\mathbf{w}),$$

where

$$\alpha_i(\mathbf{w}) = \frac{w_j w_k}{w_i w_j + w_j w_k + w_k w_i}.$$

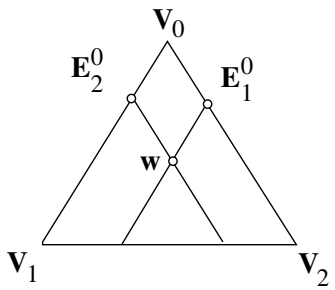


Figure 2.10: Construction of $\mathbf{E}_2^0$ and $\mathbf{E}_1^0$ used by Gregory and Charrot.

Hagen-Pottmann

Hagen and Pottmann [HP89] extended Nielson's scheme to construct a G^2 surface. As in Nielson's paper, two versions of the G^2 scheme are presented: a transfinite version and a scattered data version. Here I sketch the construction for the scattered data version to impart the ideas of the method and to show the increased complexity of creating a G^2 join between surfaces.

The construction proceeds in the same manner as Nielson's. Along each boundary, a tangent field and second fundamental form field are created. Three patches are created, each of which interpolates the all three boundary curves plus the first and second order data along one boundary. The three patches are then blended with the following weights:

$$\beta_i = \frac{b_j^2 b_k^2}{b_p^2 b_q^2 + b_q^2 b_r^2 + b_r^2 b_p^2}. \tag{2.14}$$

Hagen-Pottmann then show that the surface formed by blending these three surfaces with these β_i interpolates the normal and curvature data along the entire boundary.

Hagen-Pottmann use quintic Hermite curves to interpolate position, tangent, and curvature data at two points, defining a curve constructor $\mathbf{H}_5$:

$$\begin{aligned}\mathbf{H}_5[\mathbf{V}_0, \mathbf{V}_1, \mathbf{V}_0', \mathbf{V}_1', \mathbf{V}_0'', \mathbf{V}_1''](t) &= \mathbf{V}_0 H_0^5(t) + \mathbf{V}_0' H_1^5(t) + \mathbf{V}_0'' H_2^5(t) + \\ &\quad \mathbf{V}_1'' H_3^5(t) + \mathbf{V}_1' H_4^5(t) + \mathbf{V}_1 H_5^5(t),\end{aligned}$$

where $H_i^5(t)$ is the ith quintic Hermite polynomial. Typically, $\mathbf{V}'$ and $\mathbf{V}''$ (the first and second derivatives at the endpoints) are not known. Instead, we know unit

tangent vectors and curvature vectors:

$$\mathbf{V}_0' = \lambda_0 \vec{\mathbf{T}}_0, \qquad \mathbf{V}_1' = \lambda_1 \vec{\mathbf{T}}_1,$$

$$\vec{\mathbf{K}}_0 = \kappa_0 \hat{\mathbf{N}}_0, \qquad \vec{\mathbf{K}}_1 = \kappa_1 \hat{\mathbf{N}}_1.$$

$\mathbf{V}''$ can now be written as

$$\mathbf{V}_0'' = \lambda_0^2 \vec{\mathbf{K}}_0 + \mu_0 \vec{\mathbf{T}}_0, \qquad \mathbf{V}_1'' = \lambda_1^2 \vec{\mathbf{K}}_1 + \mu_1 \vec{\mathbf{T}}_1.$$

The λs and μs are shape parameters, with the restriction that $\lambda > 0$. A curve constructor operator can now be defined on two vertices:

$$\mathbf{g}_v[\mathbf{V}_0, \mathbf{V}_1, \hat{\mathbf{N}}_0, \hat{\mathbf{N}}_1, C_0, C_0] = \mathbf{H}_5[\begin{array}{ll} \mathbf{V}_0, & \mathbf{V}_1, \\ \lambda_0 \vec{\mathbf{T}}_0, & \lambda_1 \vec{\mathbf{T}}_1, \\ \lambda_0^2 \vec{\mathbf{K}}_0 + \mu_0 \vec{\mathbf{T}}_0, & \lambda_1^2 \vec{\mathbf{K}}_1 + \mu_1 \vec{\mathbf{T}}_1], \end{array}$$

where the Cs are curvature data at the $\mathbf{V}$s. $\vec{\mathbf{T}}_0$ and $\vec{\mathbf{T}}_1$ are tangent vectors built from the $\mathbf{V}$s and the $\hat{\mathbf{N}}$s.

Hagen and Pottmann formulate the second order data in terms of the "Dupin indicatrix." I give an equivalent formulation for the second fundamental form. The boundary curves are constructed using the operator $\mathbf{g}$; the boundary between vertices i and $i+1$ of the input triangle is

$$\mathbf{g}[\mathbf{V}_i, \mathbf{V}_{i+1}, \hat{\mathbf{N}}_i, \hat{\mathbf{N}}_{i+1}, C_i, C_{i+1}](t).$$

We now need to construct a "tangent plane field" and a "curvature field" along each boundary. The tangent plane field is built by first creating a cross-boundary tangent field, and then taking the cross product of the cross-boundary tangents with the boundary curve tangents to get the normal. Unlike the G^1 schemes, a cubic cross-boundary field is required, since the derivative of the tangent plane field must agree with the second order data at the vertices.

The second fundamental form at each point along the boundary is determined once we know its value in three directions. The curvature along the boundary curve gives us one of these directions, while the tangent plane field gives us a second value. Thus, we need only determine the third value to characterize the curvature field.

Hagen-Pottmann obtain this third value by linearly blending conjugate directions to determine the third direction, and a linear blend of the curvatures at the endpoints to assign the curvature in this direction.

Many details have been left out of the above description. In particular, there are several special cases that must be considered. These special cases, in turn, cause numerical problems for the surface construction, resulting in surfaces with poor shape.

2.3.3 Other Schemes

I know of two schemes (one proposed by Farin [Far83], the other by Peters [Pet91a]) that fit within the scope of this work but are not surveyed above. In this section, I briefly look at these schemes.

Farin's scheme is a split domain scheme noteworthy primarily because it was the first parametric triangular surface fitting scheme. The construction, however, has several problems. One problem is its asymmetric treatment of the ring of control points surrounding a vertex. This asymmetry is visible in the constructed surfaces, so I chose not to discuss this method in detail here.

Peters [Pet91a] has given an interpolation scheme that constructs a single cubic polynomial patch per triangular facet. The construction proceeds by first creating the boundary curves. Next, a linearly varying normal field is constructed along each boundary, and then the single interior control point is set. However, there are several constraints on the input data. One constraint is that along each boundary, it must be possible to construct a boundary curve without an inflection point. This restriction is needed in order to build the linearly varying normal field. A second restriction requires the data around a vertex to allow for the construction of a C^2 curve network. This restriction is required in order to solve the vertex consistency problem. The above restrictions are severe enough that this scheme cannot be used to solve our problem.

2.4 Comparison.

The primary concern in this work was with the visual appearance of the constructed surfaces. Other concerns, such as computational issues, were considered secondary, as I wanted to first find methods that produced adequate shapes. Numerical stability

issues are occasionally mentioned, as they can have a large impact on the shape of the resulting surface.

One problem with using visual appearance as a criterion is that it is a subjective measure of surface quality. In part, this stems from a lack of a general purpose "surface quality metric," that is, a commonly agreed upon definition of good shape. However, the problems with the shapes of surfaces I encountered were extreme, leaving little doubt about the poor quality of the surfaces.

In many applications, the data points themselves are not distinguished points on the surface. Therefore, these points should not be visually distinguishable in the constructed surface. All but one of the schemes surveyed in this chapter construct piecewise surfaces that have second order derivative discontinuities at the boundaries of the surfaces patches, implying that the boundaries of the patches (and, thus, the data points) will be distinguished to some extent. I feel that the visual impact of these discontinuities should be minimized.

In the past, many authors have used line drawing renditions to show the visual quality of their surfaces. I, as other researchers, found shaded images more useful in detecting various shape defects. To see more subtle defects I found that Gaussian curvature plots of the interpolants were often helpful. (The Gaussian curvature at a point on a surface is the product of the minimum and maximum normal curvature of the surface at the point.)

All implemented schemes produced surfaces with shape defects that were readily apparent in the shaded images or Gaussian curvature plots. However, a scheme should not be considered "good" just because it passes these two visual tests. Until a good quantitative measure of shape is devised, a surface fitting scheme should also be tested by a variety of other methods (such as reflection lines and isophotes [Poe84, HHS+92]) to determine surface quality.

2.4.1 Data Sets

A variety of data sets were used to test the surface fitting schemes. Three of these are shown in Figures 2.11a-c. In these figures, the lines represent the edges of the triangulated data. The data points are located at the intersection of the lines. The data sets of the sphere and the torus are samplings of those surfaces; the vertices and tangent planes of the octahedron data set were sampled from a sphere. (Both

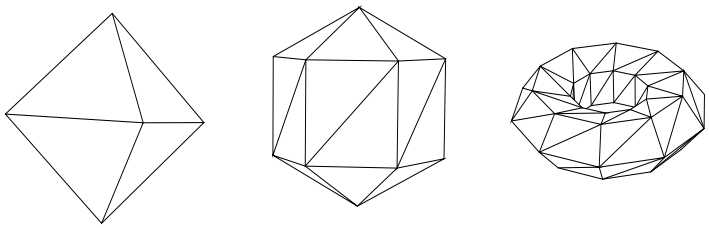


Figure 2.11: a) Octahedron b) Sphere c) Torus

the sphere and the octahedron data sets can be thought of as samplings of the unit sphere, which has constant Gaussian curvature of one.)

The sphere data sets tended to be particularly "mild", as the entire surface has positive constant Gaussian curvature, i.e., it has no flat spots or saddle points. The torus is a more interesting data set, as it has regions of positive, negative, and zero Gaussian curvature. One problem with the dense data sets is that they are somewhat complex, making the resulting surfaces difficult to analyze. The octahedron data set was chosen because it was simple enough to allow us to develop intuition governing the failure of the schemes.

Three materials (i.e., surface reflectance parameters) were used to construct the surfaces appearing in the color figures. The material used for the shaded surfaces has a red diffuse component with a white specular component. Some of the surfaces were false colored to show the Gaussian curvature of the surface. Areas of positive Gaussian curvature are colored red, areas of nearly zero Gaussian curvature are colored green, and areas of negative Gaussian curvature are colored blue. In all the Gaussian curvature images, the color scale was truncated to -4 and +4. A color bar at the bottom of these images indicates the relative values of the colors.

The third material is discussed in Chapter 5.

2.4.2 Results of Comparison

The goal of this work was to find which interpolation schemes produced "nice" surfaces and which schemes did not. All the schemes described in Section 2.3 (except for Hansford's scheme and the two schemes mentioned in Section 2.3.3) were imple-

mented and tested. Jensen's and Shirman-Séquin's schemes were similar enough that it seemed adequate to implement only one of them. I chose to implement Shirman and Séquin's scheme using Jensen's generalization of the cross-boundary tangents. The software developed to test these schemes is discussed elsewhere [LML+90]. To my surprise, all the schemes I tested created surfaces with severe shape defects. Moreover, they all suffered from shape defects that are qualitatively similar.

Parametric Data

On testing the surveyed schemes on the data sets in Figures 2.11a-c, I found a variety of flaws. As a representative scheme, I show pictures of the surfaces constructed by Shirman and Séquin's scheme. Most artifacts are visible in the shaded images, although Gaussian curvature plots tend to reveal the nature of the artifacts more clearly.

Inspection of Gaussian curvature plots for interpolants to the sphere data shows that the patches are mostly flat, with a few areas of high curvature (Figure 2.18, left). These effects occurred fairly uniformly for most schemes, with additional curvature discontinuities appearing along the interior boundaries produced by split domain schemes.

Shaded images of the interpolants to the torus data (Figure 2.19, left) showed problems more serious than the ones mentioned above. Images of Gaussian curvature indicated that there were large variations of curvature, even within a single patch. Discontinuous jumps from positive to negative curvature were also observed along patch boundaries.

Images of Gaussian curvature for interpolants to the simple data set of the octahedron clearly show that curvature is concentrated near the vertices and boundary curves (Figure 2.20, left). The center of the patch (or patches) created for a face tends to be flat by comparison. These problems were similar for all the schemes. This was somewhat unexpected, given the differences in the constructions of the split domain schemes and the convex combination schemes.

When Peters' quadratic boundary curves are used, then the patches show different curvature behavior: the patches are flat near the face vertices, having areas of high curvature near the middle of the face boundaries. Again, these affects occur uniformly for all schemes when quadratic curves are used for the boundaries.

Some of these problems can be alleviated by manually adjusting the shape parameters. For example, the curvature of the surfaces constructed for the octahedron can be spread more uniformly over the patches, although there are still flat spots on the surface. The improvements possible for the torus data set are less significant. It is also unclear how to set automatically the value of these parameters.

Three schemes had problems in addition to those mentioned above. A minor (and easily corrected) problem occurs with the boundary curve construction of Peters' scheme, as discussed in Section 2.4.5. Piper's scheme experienced numerical problems when "near" the special cases of the scheme. This occurs when the scalar functions of Equations 2.2 are nearly constant (i.e., when the 3×5 system of equations nearly reduces to a 2×5 system). Similar numerical problems occur with Hagen-Pottmann's scheme. I decided not to address these numerical stability issues, focusing instead on the other schemes.

2.4.3 Boundary Curves

The construction of boundary curves was a common element in all our implementations. Following constructions given by Piper and Shirman-Séquin, I originally constructed a cubic boundary curve for an edge of the data set as follows: First, the endpoints of the curve are set to interpolate the endpoints of the edge. Next, tangent directions are computed for each endpoint by perpendicularly projecting the vector along the edge into the tangent plane at each endpoint; these vectors are then scaled to be of length equal to the length of the edge (see Figure 2.12). The boundary curve is then set to be the cubic polynomial curve interpolating this data.

As published, the schemes surveyed in this chapter used a variety of methods for constructing boundary curves. However, the differences between these methods are minor. Shirman and Séquin use the method mentioned above. Nielson constructs cubic boundary curves with the same tangent directions, but leaves the length as a free parameter. Herron and Jensen essentially assume that the boundary curves have already been constructed. Piper's scheme constructs quartic boundary curves; the endpoint tangents have the same direction as those in the method presented above, but are shorter in length. Peters's scheme constructs linear, quadratic, and cubic boundary curves. The linear and cubic curves are constructed with the above the method; the quadratic curves are built to be of minimal length.

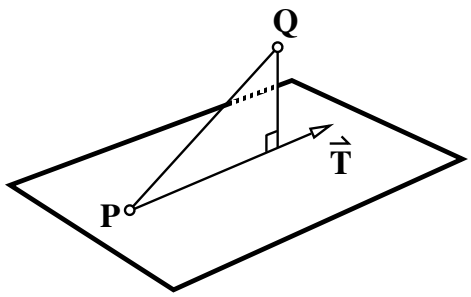


Figure 2.12: The construction of the tangent at $\mathbf{P}$. The tangent is used for the curve from $\mathbf{P}$ to $\mathbf{Q}$. $|\vec{\mathbf{T}}| = |\mathbf{P} - \mathbf{Q}|$.

A variation on the above boundary curve construction builds a planar cubic curve. For each boundary, a plane P is chosen that contains the endpoints of the curve. The tangent planes at the endpoints are intersected with P to find tangent directions at the ends; these vectors are then scaled to be of length equal to the distance between the endpoints. It was determined empirically that this planar method works better than the non-planar method given above (compare the surfaces on the left and center in Figure 2.19).

Curvature plots of boundary curves constructed using either the planar or non-planar method show that as the tangent vectors become more perpendicular to the edge, the curvature along the curve is concentrated near the endpoints (Figure 2.13a), leaving a relatively flat region in the middle of the curve. In all the surface schemes I implemented, this flatness was then propagated inward in the patch construction, resulting in relatively large flat areas in the middle of the patches.

Peters' quadratic boundary curves were the only curves exhibiting different curvature distribution. Often, these curves would be flat near the endpoints, with high curvature at the middle of the curve. Patches constructed with these curves exhibit the opposite curvature behavior of patches constructed with cubic curves: the patches for a face are flat near the vertices and have high curvature near the middle of the face boundaries. Near the face centers, patches constructed with quadratic boundaries are not as flat as those constructed with the standard cubic boundary curves.

The curvature behavior when using either quadratic or cubic curves suggested

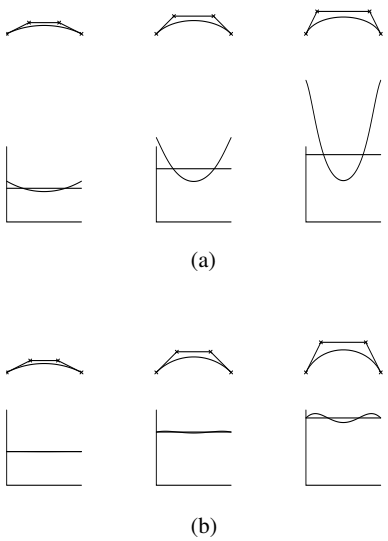


Figure 2.13: Bézier curves and their curvature plots. (a) Curves constructed using projection method. (b) Curve constructed by de Boor-Höllig-Sabin method. The horizontal lines in the curvature plots show the curvature of the circle interpolating the data.

that it might be possible to improve the curvature distribution of the patches by improving the curvature distribution of the boundary curves.

2.4.4 Cubic Geometric Hermite Curves

De Boor, Höllig, and Sabin [dBHS87] give a method for constructing a planar cubic curve that interpolates positions, tangent lines, and curvatures at two points. More precisely, they solve the following problem:

> Given: Two points $\mathbf{p}_0$ and $\mathbf{p}_1$, with unit tangents $\hat{\mathbf{t}}_0$ and $\hat{\mathbf{t}}_1$, and with curvatures k_0 and k_1, where the points and tangents lie in a common plane (Figure 2.14).
>
> Find: A cubic curve that interpolates this data at its endpoints. Since we know the positions and tangents, only the tangent lengths are unknown.

If the tangent vectors are $\delta_0\hat{\mathbf{t}}_0$ and $\delta_1\hat{\mathbf{t}}_1$, then solutions to

$$(\hat{\mathbf{t}}_0 \times \hat{\mathbf{t}}_1)\delta_0 \;=\; ((\mathbf{p}_1 - \mathbf{p}_0) \times \hat{\mathbf{t}}_1) - \frac{3}{2}k_1\delta_1^2,$$

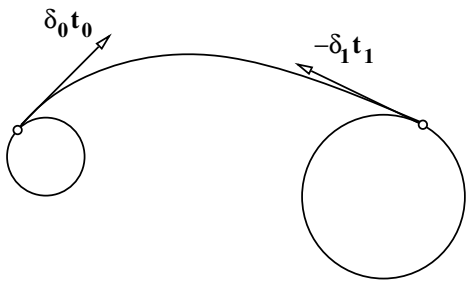


Figure 2.14: Cubic geometric Hermite curves. Find δs such that the tangents and curvature values are interpolated by a cubic Hermite curve.

$$(\hat{\mathbf{t}}_0 \times \hat{\mathbf{t}}_1)\delta_1 \quad = \quad ((\mathbf{p}_0 - \mathbf{p}_1) \times \hat{\mathbf{t}}_0) - \frac{3}{2} k_0 \delta_0^2$$

gives the values of the δs [dBHS87]. A Hermite curve interpolates position and derivatives. Since the de Boor-Höllig-Sabin curves interpolate geometric invariants rather than derivatives, I refer to them as *cubic geometric Hermite curves.* De Boor, Höllig, and Sabin show that these curves have sixth order of convergence.

More important for our purposes, these curves were observed to have relatively uniform curvature distributions. Figure 2.13b, for example, shows curves constructed by this method. The curvature values are taken from the circle passing through the data points with the given tangents. If the curvature values at the endpoints are not equal, the resulting curves are approximations to an ellipse.

At times, we will need to construct non-planar cubic geometric Hermite curves. A variation of the de Boor-Höllig-Sabin method is used to match normal section curvature at the endpoints (Peters has developed a similar method using quartic space curves [Pet89]). The variation of the problem we wish to solve is:

> Given: Two positions $\mathbf{p}_0$ and $\mathbf{p}_1$, with unit tangents $\hat{\mathbf{t}}_0$, $\hat{\mathbf{t}}_1$, normals $\hat{\mathbf{N}}_0$, $\hat{\mathbf{N}}_1$, and normal section curvature values k_0, k_1.
>
> Find: a cubic curve that interpolates this data at its endpoints. As normal section curvature is only defined for surface curves, we think of the data as being taken from a surface and the curve we build as being on a surface interpolating this data.

Let $\mathbf{C}(t)$ be a cubic curve with control points $\mathbf{C}_0$, $\mathbf{C}_1$, $\mathbf{C}_2$, $\mathbf{C}_3$. For $\mathbf{C}$ to interpolate the positions, we must set $\mathbf{C}_0 = \mathbf{p}_0$ and $\mathbf{C}_3 = \mathbf{p}_1$, leaving $\mathbf{C}_1$ and $\mathbf{C}_2$ as our unknowns. The second order constraints on $\mathbf{C}$ are:

$$\begin{aligned} k_0 &= \frac{< \hat{\mathbf{N}}_0,\ \mathbf{C}''(0) >}{< \mathbf{C}'(0),\ \mathbf{C}'(0) >} \\ k_1 &= \frac{< \hat{\mathbf{N}}_1,\ \mathbf{C}''(1) >}{< \mathbf{C}'(1),\ \mathbf{C}'(1) >} \end{aligned}$$

Rewritten in terms of our control points gives us

$$\begin{aligned} k_0 &= \frac{2 < \hat{\mathbf{N}}_0,\ (\mathbf{C}_2 - \mathbf{C}_1) >}{3 < \mathbf{C}_1 - \mathbf{C}_0, \mathbf{C}_1 - \mathbf{C}_0 >} \\ k_1 &= \frac{2 < \hat{\mathbf{N}}_1,\ (\mathbf{C}_1 - \mathbf{C}_2) >}{3 < \mathbf{C}_3 - \mathbf{C}_2, \mathbf{C}_3 - \mathbf{C}_2 >} \end{aligned}$$

Letting $\mathbf{C}_1 = \mathbf{C}_0 + \delta_0 \hat{\mathbf{t}}_0$ and $\mathbf{C}_2 = \mathbf{C}_3 - \delta_1 \hat{\mathbf{t}}_1$, we can rewrite these equations as

$$\begin{aligned} < \hat{\mathbf{N}}_1,\ \hat{\mathbf{t}}_0 > \delta_0 &= < \hat{\mathbf{N}}_1,\ (\mathbf{p}_0 - \mathbf{p}_1) > - \frac{3}{2}\delta_1^2 k_1 \\ < \hat{\mathbf{N}}_0,\ \hat{\mathbf{t}}_1 > \delta_1 &= < \hat{\mathbf{N}}_0,\ (\mathbf{p}_1 - \mathbf{p}_0) > - \frac{3}{2}\delta_0^2 k_0. \end{aligned}$$

Solutions to these equations gives the values of the δs that solve the stated problem.

To determine if using boundary curves with a more uniform distribution of curvature would yield surfaces with better shape, the de Boor-Höllig-Sabin curve construction technique was integrated into each of the surface fitting schemes. The resulting interpolants all show improvement in shape. Interpolants to the sphere data set, for example, exhibit more uniform distribution of Gaussian curvature (Figure 2.18, center).

The results for the octahedron data set are less encouraging. The curvature is spread along the patches somewhat more uniformly, but the patches are still flat in the interior (Figure 2.20, center). Additional improvements can be made by manually adjusting the shape parameters (Figure 2.20, right).

The improvement in the shape of the interpolants for the torus data set are minor. Using the de Boor-Höllig-Sabin method yields an interpolant that is a better approximation to the torus (using a radial distance metric); however, while there is some

improvement in shape, the surface still has many of the shape defects found in the surface produced using the standard boundary curve method (Figure 2.19, right). So, while the de Boor-Höllig-Sabin boundary curve method appears to improve the shape of the interpolants, it is only a partial solution. Further, there are several problems with this method.

First, to have the curvature information needed by the de Boor-Höllig-Sabin scheme, second fundamental forms must be associated with the vertices of the data sets. If the data are sampled from a C^2 function, then second fundamental forms can be calculated directly. If only the positions are available at the vertices, then second fundamental forms must be estimated.

A more serious problem is that there may be from zero to three cubic curves that match the curvature data [dBHS87]. If no cubic curve interpolates the data, then the boundary curves must be computed by some other method. If there are multiple solutions, then a choice must be made between these solutions. In the cases I examined, all the solution curves given by the de Boor-Höllig-Sabin method have essentially the same shape. These solutions differ primarily in their parametrizations. I chose the curve with the "most uniform" parameterization, as discussed in the next section.

2.4.5 Boundary curve parameterization

In addition to shape, another important quality of a boundary curve is its parameterization. The standard boundary curve techniques construct curves with a near uniform speed parameterization. That is, the length of a section of the curve is nearly proportional to the corresponding length in the domain.

However, the method of de Boor-Höllig-Sabin sometimes creates curves with nonuniform speed parameterizations. For example, for the sphere data, there are up to three distinct cubic curves that match the endpoint data. The solution closest to a uniform speed parameterization gives the surface at the center of Figure 2.18. When a less uniform parameterization is used, large lumps appear in the surface (Figure 2.18, right). Note, however, that the surface is still G^1.

Sparse samplings of two curves constructed for one boundary of the sphere data are shown in Figure 2.15; the curve network for the sphere is given in Figure 2.16. In the faces of interest, only one of the three external boundary curves has a different

parameterization. While the shapes of the external patch boundary curves are similar in both figures, the internal boundary curves become twisted when the nonuniform speed parameterization is used. In the shaded image, this results in a pinching of the surface and in "lumps".

These effects are a result of the placement of the cross-boundary control points (for Shirman-Séquin, the $\mathbf{C}_{ij}$ of Figure 2.5), whose placement is strongly dependent on the parameterization of the boundary curves (Figure 2.17). From Equation 2.4, we see that h_0 is inversely proportional to the length of $\vec{\mathbf{H}}_0$. And as seen in Equations 2.6, h_0 weights $\vec{\mathbf{H}}_1$ in the computation of $\mathbf{F}_1$; a small $\vec{\mathbf{H}}_0$ results in an overly large weight of $\vec{\mathbf{H}}_1$.

Thus, in addition to good shape, it is important for the boundary curves to have a nearly uniform speed parameterization. Of the curve construction techniques I investigated, only the de Boor-Höllig-Sabin method and Peters' method would create curves with poor parameterizations. For the de Boor-Höllig-Sabin method, I selected the best parameterized solution.

Peters' method for building quadratic curves creates curves with poor parameterization when the σs of Equation 2.11 are of the same sign but have a small ratio (the ratio being formed by dividing the smaller σ by the larger). Again, the weights of some of the terms in the construction of the cross-boundary control point are inversely proportional to the length of $\vec{\mathbf{H}}_0$ (Equation 2.8). However, these poorly parameterized quadratic curves are easily detected and can be replaced with cubic curves having better parameterization.

2.4.6 Degrees of Freedom

The problem with the boundary curves is in a sense a problem of "too many degrees of freedom." To interpolate the position and tangent data, we must use cubic curves. After meeting these interpolation conditions, two degrees of freedom remain unset. Previous methods set these degrees of freedom with heuristics. Improvements to the surface shape occur when we use more principled settings, such as settings that match higher order data.

However, the G^1 schemes produce poorly shaped surfaces even when well shaped boundary curves with uniform parameterizations are used. On the interior of the patches, the above surface schemes have many degrees of freedom that are set using

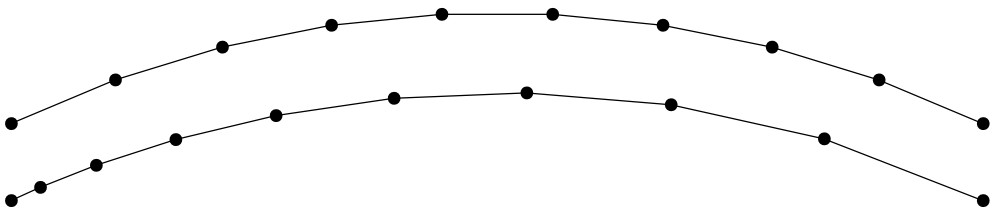


Figure 2.15: Comparison of parameterizations of two solutions given by de Boor-Höllig-Sabin. The dots show samples taken uniformly in the domain. The top curve yields the smooth sphere; the bottom one gives the lumpy sphere.

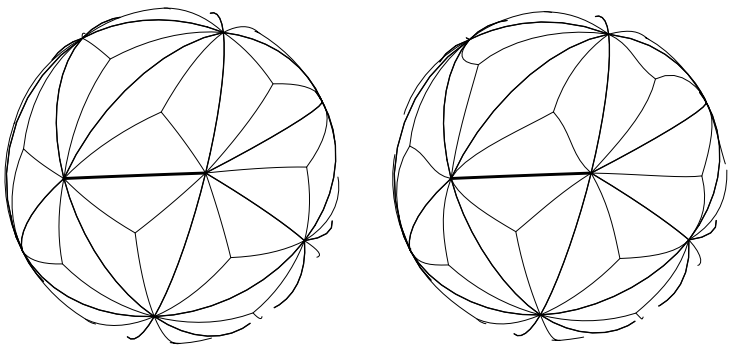


Figure 2.16: Comparison of curve networks for the sphere data. On the left is the curve network created when the nearly uniform speed parameterization is selected; on the right an arbitrary parameterization is chosen. The light lines are internal patch boundaries; the medium lines are external patch boundary; the dark line is the curve shown in Figure 2.15.

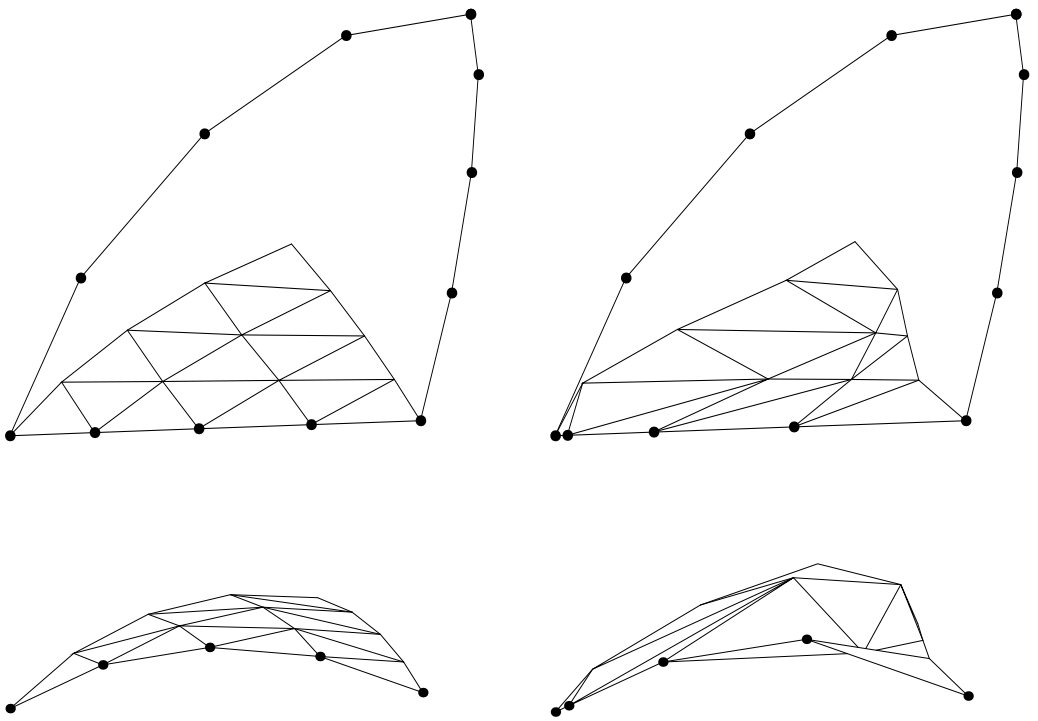


Figure 2.17: Comparison of Bézier patches. On the left are two views of the patch bordering the uniform speed parameterized curve of Figure 2.15; on the right, the patch bordering the other curve (the one that forms a "lump" in the surface). In addition to the surface patch, all three face boundaries have been drawn in the top view. The control points of the external boundary curves are shown as black dots.

heuristics. The quartic split domain schemes, for example, have 19 interior control points having 48 degrees of freedom. There are 31 constraints on these points; the remaining 17 degrees of freedom are set using heuristics, such as the averaging schemes of Shirman-Séquin and Peters (it can be argued, however, that some of these 17 degrees of freedom are constrained by symmetry).

The settings of these excess degrees of freedom are seen to strongly influence the shape of the surface. For example, with proper settings (determined empirically) of Jensen's and Shirman-Séquin's shape parameters, the Shirman-Séquin surface fit to the octahedron using de Boor-Höllig-Sabin boundary curves provides a good approximation of a sphere (Figure 2.20, right). Such free parameters can be useful in a design setting. However, default values are needed for these shape parameters. Currently, proper defaults are unknown.

Figure 2.18: Shirman-Séquin sphere. On the top row are shaded images. Below each shaded image is a Gaussian curvature plot of the surface.

Left: Shirman-Séquin. The maximum Gaussian curvature is 2.4; the minimum is 0.51.

Center: Gaussian Curvature of Shirman-Séquin surface fit to sphere data using de Boor-Höllig-Sabin boundary curves. The maximum Gaussian curvature is 1.3; the minimum is 0.76.

Right: Shirman-Séquin surface fit to sphere using poorly parameterized de Boor-Höllig-Sabin boundary curves. Maximum Gaussian curvature 235, minimum -298.

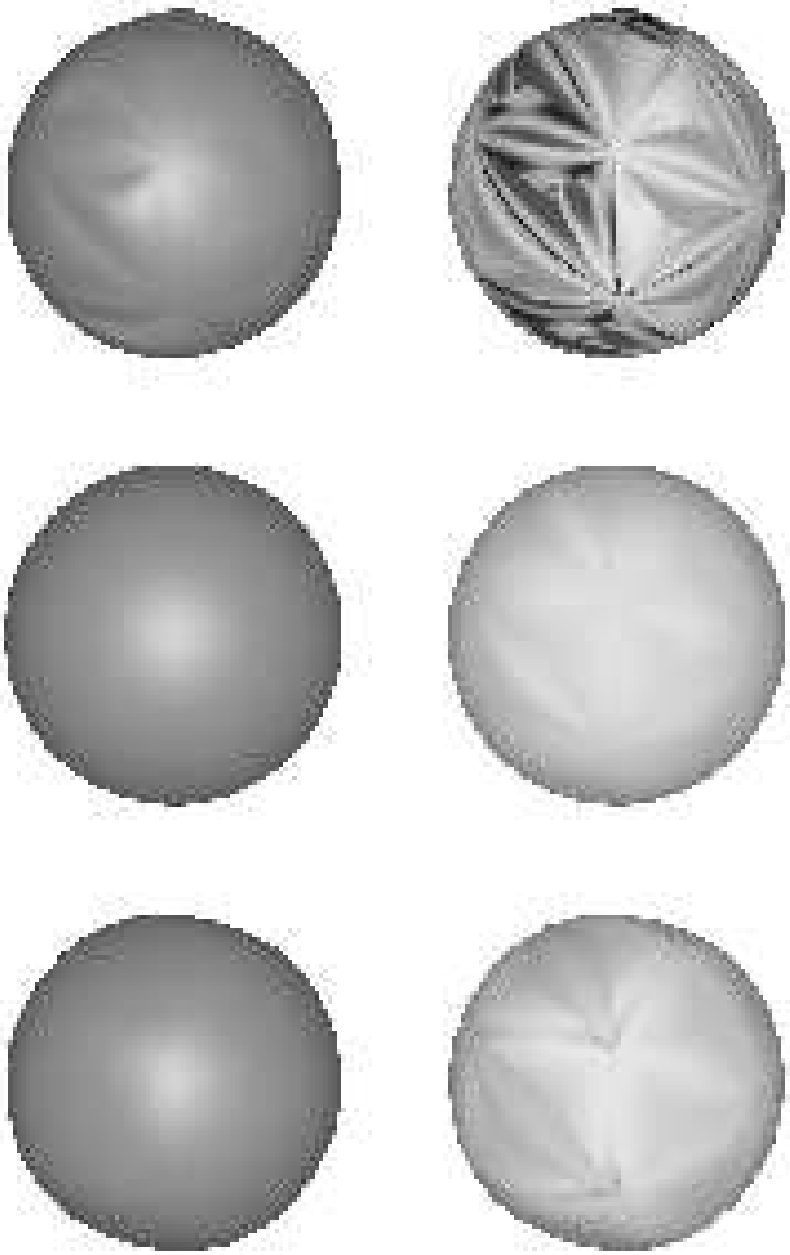


Figure 2.19: Shirman-Séquin torus. On the top row are shaded images. Below each shaded image is a Gaussian curvature plot of the surface.

Left: Shirman-Séquin surface fit to torus data using the non-planar boundary curve technique.

Center: Shirman-Séquin surface fit to torus data using the planar boundary curve technique.

Right: Shirman-Séquin surface fit to torus data using de Boor-Höllig-Sabin boundary curves.

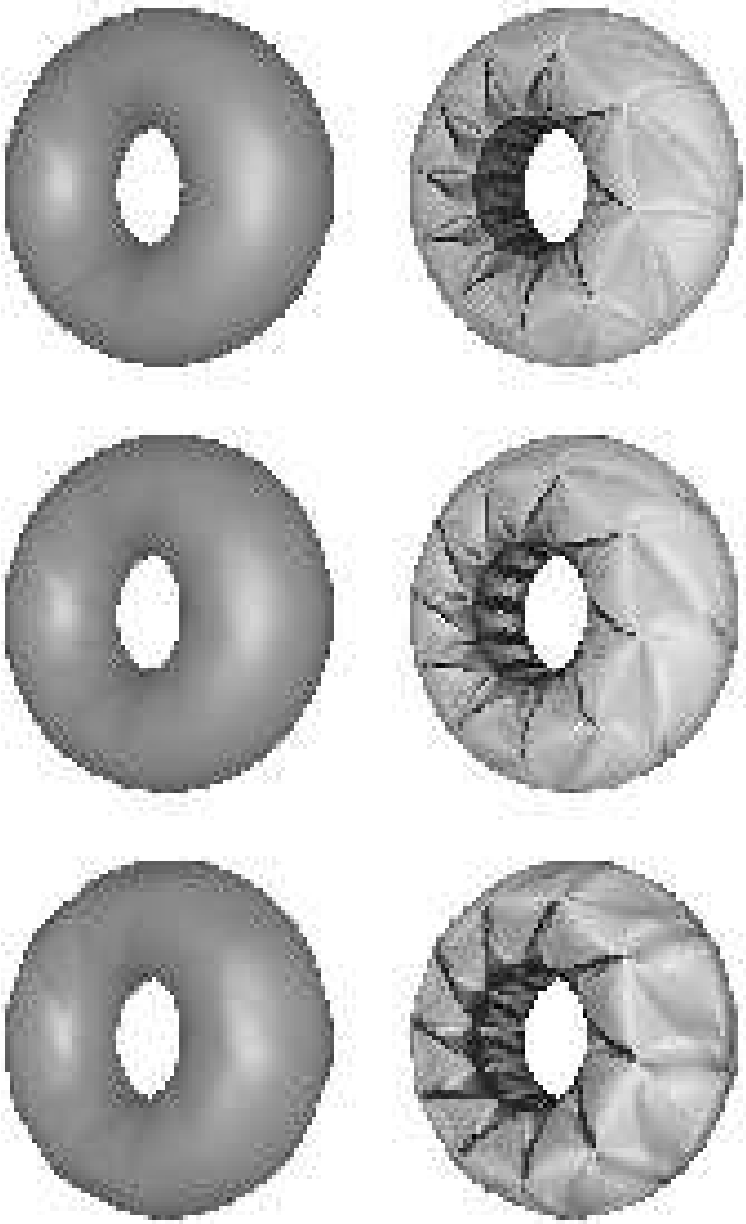
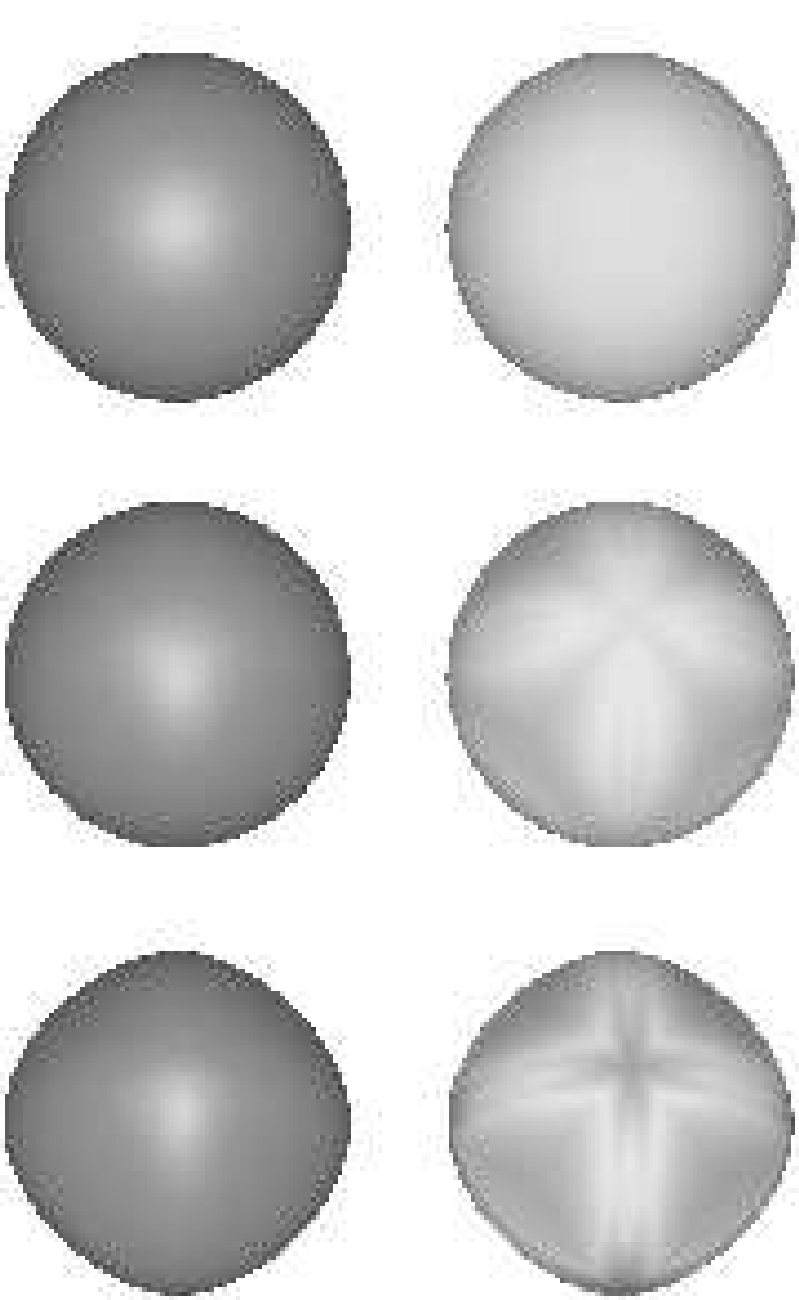


Figure 2.20: Shirman-Séquin octahedron. On the top row are shaded images. Below each shaded image is a Gaussian curvature plot of the surface.

Left: The surface constructed by Shirman-Séquin's scheme; the maximum Gaussian curvature is 2.5; the minimum is 0.28.

Center: The Shirman-Séquin's surface built using de Boor-Höllig-Sabin boundary curves; the maximum Gaussian curvature is 1.4; the minimum is 0.44.

Right: The Shirman-Séquin's scheme using de Boor-Höllig-Sabin boundary curves, with adjustments made to shape parameters (Jensen's $C = 1.055$, Shirman-Séquin's $\alpha_{i1} = 0.25$, $\alpha_{i2} = 1.75$); maximum Gaussian curvature is 1.03; minimum is 0.98.



Chapter 3

THE CUBIC INTERPOLANT

Despite the differences in the G^1 interpolation techniques, the constructed surfaces exhibit similar shape defects. Curvature plots show that the poor shape is due in part to poor curvature distribution over the patches. My experiences indicated that this non-uniform distribution was an indirect consequence of the G^1 requirement. Fairly high degree patches must be used to construct a G^1 join between patches. After satisfying the G^1 conditions, however, these patches have many degrees of freedom remaining that are typically set using heuristics. The poor distribution of curvature is partially a result of the arbitrary settings of these surplus degrees of freedom.

There are several approaches to improving the distribution of curvature over the surface patches. One approach is to improve the existing methods by using more principled settings of the remaining degrees of freedom, using for example, numerical optimization. However, for surface approximation, the excess degrees of freedom could be set using additional data sampled from the underlying surface. In this chapter, I look at constructing patches that set their degrees of freedom by matching higher order derivative information at the data points.

3.1 The Quadratic Interpolant

Given a mesh, our goal is to construct an interpolating surface while minimizing the number of degrees of freedom that remain unset after meeting the interpolation conditions. To simplify the the conditions on the surface patches, I will only require C^0 continuity between neighboring patches. If only positions are given at the vertices, then we can interpolate the data with a piecewise planar surface. If both position and normals are given, then the triangle of data can, in general, be fit with a quadratic patch (Figure 3.1).

The construction is simple. To interpolate the corners of a triangle $\triangle \mathbf{V}_r \mathbf{V}_s \mathbf{V}_t$,

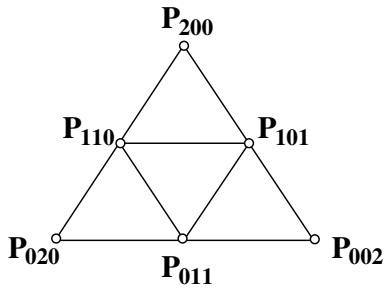


Figure 3.1: Control points of a quadratic patch.

the patch corners must interpolate the triangle vertices:

$$\mathbf{P}_{200} = \mathbf{V}_r, \qquad \mathbf{P}_{020} = \mathbf{V}_s, \qquad \mathbf{P}_{002} = \mathbf{V}_t.$$

To meet the given normals at the vertices $\mathbf{V}_r$ and $\mathbf{V}_s$, the point $\mathbf{P}_{110}$ must lie in the tangent planes of both $\mathbf{V}_r$ and $\mathbf{V}_s$. Therefore, the intersection of these two tangent planes restricts $\mathbf{P}_{110}$ to a line. Similarly, $\mathbf{P}_{101}$ and $\mathbf{P}_{011}$ can be also restricted to lines.

For any arbitrary selection of $\mathbf{P}_{110}$, $\mathbf{P}_{101}$, and $\mathbf{P}_{011}$ along the three lines, the resulting patch $\mathbf{P}$ will interpolate the corner data. To construct a C^0 surface, patches constructed for neighboring triangles must make this choice consistently. One possible choice is given in the next section.

Note that if two of the tangent planes are parallel, then either the intersection of the planes is empty (in which case there is no solution for the corresponding control point), or the intersection is a plane. If the quadratic interpolant is to be used to construct an ε-G^1 surface, then these two degeneracies must be addressed.

Another problem with quadratic boundaries is that the data may indicate that the boundary curve should have an inflection point, as discussed by Peters [Pet91c] (see also Section 2.3.1). While it may be possible to construct a quadratic patch to fit such data, the resulting surface might not have a well defined tangent plane at the mesh vertices.

3.2 The Cubic Interpolant

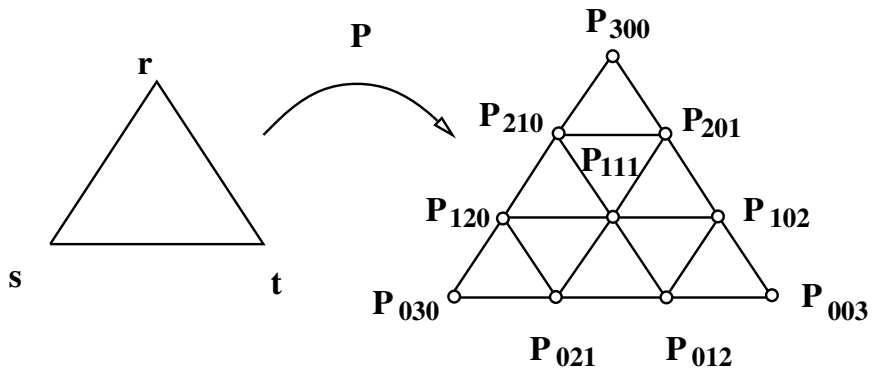


Figure 3.2: The domain triangle and control points of a cubic Bézier patch $\mathbf{P}$

If the data given at the vertices has position, normal, and curvature, then in general, we can interpolate it with triangular cubic Bézier patches. In the following, a single patch will be called $\mathbf{P}$. The domain of $\mathbf{P}$ is given by the triangle $\triangle\mathbf{rst}$. The control points of $\mathbf{P}$ are denoted with standard multi-index notation (Figure 3.2). The problem we wish to solve is:

> Given: Three points, with associated normals and second fundamental forms, $(\mathbf{V}_i, \hat{\mathbf{N}}_i, \mathrm{II}_i)$, $i \in \{\mathbf{r}, \mathbf{s}, \mathbf{t}\}$.
>
> Find: A cubic polynomial surface patch that in general interpolates this data.

I shall refer to this patch, when it exists, as the *cubic interpolant*. While the positions and normals are always interpolated, there are cases, shown below, when the second fundamental forms cannot be interpolated with a cubic patch.

The data at each input point imposes eight constraints on the patch: three for position, two for the normal, and three for the second fundamental form. For two linearly independent vectors $\vec{\mathbf{u}}$ and $\vec{\mathbf{v}}$ in the tangent plane of a patch $\mathbf{P}$ at $\mathbf{V}_r$, $\mathbf{P}$ must satisfy the following constraints to interpolate the data at $\mathbf{V}_r$:

$$\mathbf{P}(\mathbf{r}) = \mathbf{V}_r, \tag{3.1}$$

$$< \mathbf{D}_{\vec{\mathbf{u}}}\mathbf{P}(\mathbf{r}),\ \hat{\mathbf{N}}_r > = 0, \qquad < \mathbf{D}_{\vec{\mathbf{v}}}\mathbf{P}(\mathbf{r}),\ \hat{\mathbf{N}}_r > = 0, \tag{3.2}$$

$$< \mathbf{D}_{\vec{\mathbf{u}}\vec{\mathbf{u}}}\mathbf{P}(\mathbf{r}),\ \hat{\mathbf{N}}_r > = -\mathrm{II}_r(\vec{\mathbf{u}}, \vec{\mathbf{u}}), \qquad < \mathbf{D}_{\vec{\mathbf{v}}\vec{\mathbf{v}}}\mathbf{P}(\mathbf{r}),\ \hat{\mathbf{N}}_r > = -\mathrm{II}_r(\vec{\mathbf{v}}, \vec{\mathbf{v}}), \tag{3.3}$$

$$< \mathbf{D}_{\vec{\mathbf{u}}\vec{\mathbf{v}}}\mathbf{P}(\mathbf{r}),\ \hat{\mathbf{N}}_r > = -\mathrm{II}_r(\vec{\mathbf{u}}, \vec{\mathbf{v}}), \tag{3.4}$$

Similar constraints are imposed by the other two corners. The total number of constraints on the patch is therefore 24. A quadratic polynomial patch has only 18 degrees of freedom and cannot in general be used to solve this problem. A cubic patch is the minimum degree polynomial patch that can be used; it has 30 degrees of freedom in its ten control points.

A cubic polynomial patch generally has $30 - 24 = 6$ degrees of freedom unconstrained by the input data. These degrees of freedom manifest themselves in the tangents to the boundary curves. Each end tangent has an angular degree of freedom in the tangent plane. I set these degrees of freedom by restricting each boundary of the cubic patch to lie in a plane. The choice of planar boundaries uses three degrees of freedom, while the particular planes selected use the other three degrees of freedom.

There are ten control points in a cubic Bézier patch $\mathbf{P}$ (Figure 3.2). Since the corner control points of the patch are required to be the input data points, there are only seven unknown control points. Interaction of the second order constraints imposes a non-linear system of constraints on these seven points. Fortunately, the system decouples in such a way that each boundary of the patch can be constructed independently. The location of the control point $\mathbf{P}_{111}$ is then given by a 3×3 system of linear equations.

I now examine the system of constraints in more detail. Each boundary is constructed by first selecting a plane with normal $\hat{\mathbf{N}}$ containing the boundary's endpoints. Tangent directions for the boundary curve and curvature in these directions can then be determined, and a planar cubic geometric Hermite curve is fit to this data. More precisely, for the boundary between $\mathbf{V}_r$ and $\mathbf{V}_s$ the data to be interpolated is placed in the plane through $\mathbf{V}_r$ and $\mathbf{V}_s$ perpendicular to the unit vector $\hat{\mathbf{N}}$, where $\hat{\mathbf{N}}$ is parallel to $(\hat{\mathbf{N}}_r + \hat{\mathbf{N}}_s) \times (\mathbf{V}_s - \mathbf{V}_r)$ (Figure 3.3). The data to be interpolated is given by

$$\begin{aligned} \mathbf{P}_0 &= \mathbf{V}_r, & \mathbf{P}_1 &= \mathbf{V}_s. \\ \hat{\mathbf{t}}_0 &= \pm(\hat{\mathbf{N}} \times \hat{\mathbf{N}}_r)/|\hat{\mathbf{N}} \times \hat{\mathbf{N}}_r|, & \hat{\mathbf{t}}_1 &= \pm(\hat{\mathbf{N}} \times \hat{\mathbf{N}}_s)/|\hat{\mathbf{N}} \times \hat{\mathbf{N}}_s|, \\ k_0 &= \mathrm{II}_r(\hat{\mathbf{t}}_0, \hat{\mathbf{t}}_0)/|\hat{\mathbf{N}} \times \hat{\mathbf{N}}_r|, & k_1 &= \mathrm{II}_s(\hat{\mathbf{t}}_1, \hat{\mathbf{t}}_1)/|\hat{\mathbf{N}} \times \hat{\mathbf{N}}_s|. \end{aligned}$$

The signs of $\hat{\mathbf{t}}_0$ and $\hat{\mathbf{t}}_1$ are chosen so that $< \hat{\mathbf{t}}_0,\ \mathbf{P}_1 - \mathbf{P}_0 >$ and $< \hat{\mathbf{t}}_1,\ \mathbf{P}_1 - \mathbf{P}_0 >$ are positive. The denominator of the ks adjusts the curvature when the boundary curve

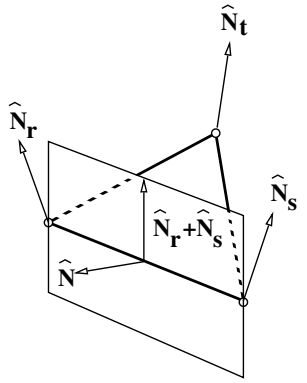


Figure 3.3: Construction of a planar boundary curve.

plane does not contain the normals at the endpoints.

After constructing the boundaries, the center control point is the only remaining unknown. Constraints 3.1 – 3.3 have been satisfied, leaving only constraint 3.4 unsatisfied. The center control point influences the mixed partial at the three corners of the patch. Letting $\vec{\mathbf{u}} = 3(\mathbf{P}_{210} - \mathbf{P}_{300})$ and $\vec{\mathbf{v}} = 3(\mathbf{P}_{201} - \mathbf{P}_{300})$, the mixed partial's normal component at the corner $\mathbf{V}_r$ is

$$\begin{aligned} \mathrm{II}_r(\vec{\mathbf{u}}, \vec{\mathbf{v}}) &= -< \mathbf{D}_{\vec{\mathbf{u}}\vec{\mathbf{v}}}\mathbf{P}(r),\ \hat{\mathbf{N}}_r > \\ &= -< 6(\mathbf{P}_{111} + \mathbf{P}_{300} - \mathbf{P}_{210} - \mathbf{P}_{201}),\ \hat{\mathbf{N}}_r > \\ &= -6 < \mathbf{P}_{111} - \mathbf{P}_{300},\ \hat{\mathbf{N}}_r > . \end{aligned}$$

Thus, $\mathbf{P}$ agrees with the mixed partial information at $\mathbf{V}_r$ if $\mathbf{P}_{111}$ is placed anywhere in the plane passing though the point $\mathbf{P}_{300} - \frac{\mathrm{II}_r(\vec{\mathbf{u}},\vec{\mathbf{v}})}{6}\hat{\mathbf{N}}_r$ and perpendicular to $\hat{\mathbf{N}}_r$. Each corner restricts $\mathbf{P}_{111}$ to such a plane. In general, these three planes can be intersected to find a unique position for $\mathbf{P}_{111}$ that satisfies all the input constraints.

This construction can fail in two ways. First, the construction of one or more of the boundary curves may fail because no solution exists. In this case, I currently set the end-tangent lengths to be the distance between the curve's two endpoints. (It is also possible for there to be multiple boundary curve solutions, in which case I choose the solution closest to a uniform speed parameterization.)

The second way the construction may fail occurs when intersecting the three planes to determine the center point $\mathbf{P}_{111}$. The intersection may not exist, or if it does exist, it may not be unique. A related difficulty is that if the planes become nearly parallel, the intersection calculation becomes numerically unstable. To address these problems, I use singular value decomposition to solve the 3×3 linear system [PFTV88]. The principal advantage of singular value decomposition for our purposes is that it automatically handles degeneracies. As the system of equations degenerates, the solution is placed near a user specified point. I choose this point to achieve quadratic precision [Far83].

3.3 Results

In this section, the cubic interpolant is compared to the G^1 schemes, using the method of Shirman-Séquin [SS87] as the representative G^1 scheme. Since the cubic interpolant uses second order information, I used the modification of Shirman-Séquin's scheme that builds cubic Hermite boundary curves (as described in Section 2.4.3). Note that this means that the surfaces being compared have the same boundary curves; only the patch interiors are different.

In the Figure 3.5, interpolants to a regular sampling of fifty points on the torus are shown. Shaded images of the surfaces constructed appear on the top row. Below each shaded image is a Gaussian curvature plot of the surface.

The surface constructed by the modified Shirman-Séquin scheme appears in the leftcolumn. This G^1 surface consists of 300 quartic Bézier patches. The right column shows the cubic interpolant surface for the same data set. This C^0 the cubic interpolant surface is comprised of only 100 cubic Bézier patches.

The cubic interpolant surface fit to the octahedron data set appears in the center of Figure 3.6. In this picture, the discontinuities are clearly visible. On the right is the cubic interpolant fit to a uniform refinement of the octahedron: each edge of the octahedron was split at the center, and the data retriangulated. The resulting mesh has twenty vertices and thirty-two faces.

Further examples appear in Chapter 5.

3.4 Order of Approximation of the Cubic Interpolant

In Chapter 5, I will use the cubic interpolant to approximate known functions. As we increase the number of samples on the surface, we would like to know how quickly the interpolant converges to the surface. The quadratic interpolant has cubic convergence [Pre75]. A cubic function that interpolates position and tangents at three vertices and also interpolates the triangle's center of gravity has fourth order convergence [Zlá68]. In this section, I conjecture that the cubic interpolant has fifth order of convergence.

The order of approximation can be conjectured by comparing the number of restrictions imposed by the data to the number of dimensions in a polynomial space. If an interpolant has order n approximation, then the number of constraints is greater than the dimension of the space of bivariate polynomials of degree n.

The space of bivariate polynomials of degree less than or equal to n has dimension $\binom{n+2}{2} = \sum_{i=0}^{n} i$. For example, the basis function for degree 0 polynomials is $B_0 = \{1\}$; for degree 1, $B_1 = \{x, y\} \cup B_0$; for degree 2, $B_2 = \{x^2, xy, y^2\} \cup B_1$. The data at each of the cubic interpolant's corners imposes three constraints for position, two for tangent plane, and three for the curvature, giving a total of 24 constraints. The space of bivariate polynomials of degree less than or equal to five has 21 dimensions; therefore, I conjecture that the cubic interpolant has fifth order convergence. In the next section, I show that this conjecture holds if it can be shown that certain derivatives of the cubic interpolant are bounded.

3.4.1 Conjecture of Quintic Convergence

In this section, I show that if the fifth derivatives of a reparameterization of the cubic interpolant are bounded uniformly in h (the maximum distance between the corners of the triangle) then the cubic interpolant has fifth order convergence. Intuitively, the requirement on the derivatives is that they must not increase without bound as h decreases.

I will begin by proving a theorem about functions over the plane. The proof of this theorem is based on the error term for Hermite interpolation. The following theorem, a variant of Hermite error taken from [Pre75], suffices for our purposes:

Theorem: Given a sequence of real numbers $x_1 < \ldots < x_k$, and positive

integers $m_1, \ldots, m_k$, there exists a unique univariate polynomial $p(x)$ of degree $\sum m_i = N + 1$ or less that solves the interpolation problem

$$p^{(j)}(x_i) = f^{(j)}(x_i)$$

at each x_i for all $i = 1, 2, \ldots, k$ where $j = 0, \ldots, m_i - 1$ and where f is a function having $m_i - 1$ consecutive derivatives at x_i. Further, if f is C^N then for some $\zeta \in [x_1, x_k]$

$$|f(x) - p(x)| \leq |f^{(N+1)(\zeta)}| h^{N+1},$$

for $x_1 \leq x \leq x_k$, where $h = x_k - x_1$.

Intuitively, the theorem states that we can interpolate $n + 1$ pieces of data uniquely with a degree n polynomial having an error term proportional to h^{n+1}.

The theorem I prove for functions above the plane is:

Theorem: Let $T = \triangle \mathbf{p}_0 \mathbf{p}_1 \mathbf{p}_2$ be the domain of a triangular interpolant $P(x, y)$ that interpolates a C^5 surface $S(x, y)$ to second order at the corners of T. If the fifth derivatives of P are uniformly bounded in h (the maximum distance between any two of the data points) then

$$|P - S| \leq O(h^5)$$

for all points in T.

Only certain fifth derivatives need to be uniformly bounded in h. However, to simplify the proof statement, I assume that all fifth derivatives are uniformly bounded in h.

Proof: Let $\phi(\mathbf{p}) = P(\mathbf{p}) - S(\mathbf{p})$ denote the error function over T. The following two lemmas are needed; proofs of these lemmas are given in the next section.

Lemma 1: Let $f(0) = f'(0) = f(l) = f'(l) = 0$ and $|f^{(4)}(s)| \leq K_4$ in $(0, l)$. Then for $s \in (0, l)$, $|f(s)| \leq K_4 l^4$.

Lemma 2: Let $f(0) = \nu_0$, $f'(0) = \tau_0$, $f''(0) = \kappa_0$, $f(l) = \nu_1$, and $f'(l) = \tau_1$, and $|f^{(5)}(s)| \leq K_5$ in $(0, l)$. Then

$$|f(s)| \leq \max(|\nu_i|) + l \max(|\tau_i|) + l^2 |\kappa_0| + K_5 l^5.$$

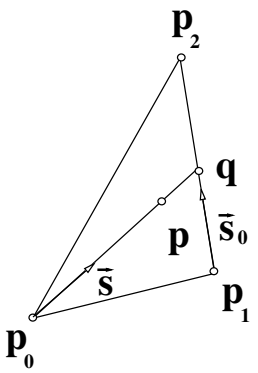


Figure 3.4: Domain of S and P.

Let $\vec{\mathbf{s}}$ be a direction in the domain of ϕ, and let $\mathbf{t}$ be on the boundary of T. Consider $D_{\vec{\mathbf{s}}}\phi(\mathbf{t})$. Since $D_{\vec{\mathbf{s}}}\phi(\mathbf{p}_i) = D^2_{\vec{\mathbf{s}}\vec{\mathbf{s}}_i}\phi(\mathbf{p}_i) = D_{\vec{\mathbf{s}}}\phi(\mathbf{p}_{i+1}) = D^2_{\vec{\mathbf{s}}\vec{\mathbf{s}}_i}\phi(\mathbf{p}_{i+1}) = 0$ where $\vec{\mathbf{s}}_i$ is the unit direction parallel to the edge from $\mathbf{p}_{i+1}$ to $\mathbf{p}_{i+2}$, we know by Lemma 1 that $|D_{\vec{\mathbf{s}}}\phi(\mathbf{t})| \leq O(h^4)$ (the fourth derivative of $D_{\vec{\mathbf{s}}}\phi$ is bounded since the fifth derivatives of ϕ are bounded).

Next, let $\mathbf{p}$ be an arbitrary point in the interior of T. Pass a line from $\mathbf{p}_0$ through $\mathbf{p}$, and find the intersection of this line and the segment $\mathbf{p}_1, \mathbf{p}_2$. Call this intersection point $\mathbf{q}$. Let $\vec{\mathbf{s}}$ denote the unit vector in direction $\mathbf{q} - \mathbf{p}_0$ (see Figure 3.4).

Consider the five values $\phi(\mathbf{p}_0)$, $D_{\vec{\mathbf{s}}}\phi(\mathbf{p}_0)$, $D^2_{\vec{\mathbf{s}}\vec{\mathbf{s}}}\phi(\mathbf{p}_0)$, $\phi(\mathbf{q})$, and $D_{\vec{\mathbf{s}}}\phi(\mathbf{q})$. The first three values are zero, by definition of ϕ. Since the fifth derivatives along the edges are bounded, we know from the error term of Hermite interpolation that $|\phi(\mathbf{q})| \leq O(h^5)$ (the error term is only fifth order since we only have a bound on fifth derivatives; if sixth derivatives are bounded, then $|\phi(\mathbf{q})| \leq O(h^6)$). And from Lemma 1, we know that $|D_{\vec{\mathbf{s}}}\phi(\mathbf{q})| \leq O(h^4)$. Applying Lemma 2, we see that

$$|\phi(\mathbf{p})| \leq \max\left(|\phi(\mathbf{p}_0)|,\ |\phi(\mathbf{q})|\right) + h\max\left(|D_{\vec{\mathbf{s}}}\phi(\mathbf{p}_0)|,\ |D_{\vec{\mathbf{s}}}\phi(\mathbf{q})|\right) + K_5 h^2 |D_{\vec{\mathbf{s}}\vec{\mathbf{s}}}\phi(\mathbf{p}_0)|,$$

where K_5 is the bound in the fifth derivatives of P. Since $\phi(p_0) = D_{\vec{\mathbf{s}}}\phi(\mathbf{p}_0) = D^2_{\vec{\mathbf{s}}\vec{\mathbf{s}}}\phi(\mathbf{p}_0) = 0$, and $\phi(\mathbf{q}) = O(h^5)$, and $|D_{\vec{\mathbf{s}}}\phi(\mathbf{q})| = O(h^4)$, we know that $|\phi(\mathbf{p})|$ is $O(h^5)$ on the interior of T. As $|\phi(\mathbf{p})| \leq O(h^5)$ along the boundaries, P has $O(h^5)$ convergence. □

Conjecture: The cubic interpolant has fifth order convergence.

The surface **S** and the cubic interpolant patch **P** are reparameterized to lie over a common plane, yielding the corresponding functions S and P over triangle $T = \triangle \mathbf{p}_0\mathbf{p}_1\mathbf{p}_2$. The above theorem can be applied (and thus, the conjecture will hold) if the necessary derivative bounds are found to hold for the cubic interpolant.

One approach to showing the derivative bounds on the reparameterized cubic interpolant patch is to follow the method of de Boor-Höllig-Sabin. If the first derivative (of the parametric patch) is proportional to h, and the second derivative is $O(h^2)$, and the third derivative is $O(h^3)$, then the fifth derivatives (of the reparameterized patch) will be bounded uniformly in h.

3.5 Proof of lemmas

Lemma 1: Let $f(0) = f'(0) = f(l) = f'(l) = 0$ and $|f^{(4)}(s)| \leq K_4$ in $(0, l)$. Then $|f(s)| \leq K_4 l^4$ for $s \in (0, l)$.

Proof: The cubic Hermite polynomial interpolating the above values is identically zero. From the theorem on Hermite interpolation, the error term is $O(l^4)$, with a constant factor less than the bound on the fourth derivatives of f. □

Lemma 2: Let $f(0) = \nu_0$, $f'(0) = \tau_0$, $f''(0) = \kappa_0$, $f(l) = \nu_1$, and $f'(l) = \tau_1$, and $|f^{(5)}(s)| \leq K_5$ in $(0, l)$. Then

$$|f(s)| \leq \max(|\nu_i|) + l \max(|\tau_i|) + l^2 |\kappa_i| + K_5 l^5. \quad \square$$

Proof: The quartic Hermite polynomials defined on the interval $(0, 1)$ that interpolate the above data are

$$\begin{aligned} H_0(t) &= 1 - 4\,t^3 + 3\,t^4, & H_1(t) &= t - 3\,t^3 + 2\,t^4, \\ H_2(t) &= \tfrac{t^2}{2} - t^3 + \tfrac{t^4}{2}, & H_3(t) &= -t^3 + t^4, \\ H_4(t) &= 4\,t^3 - 3\,t^4. \end{aligned}$$

Adjusting the parameterization from the unit interval to the interval $(0, l)$ gives

$$h(t/l) = \nu_0 H_0(t/l) + \tau_0 l H_1(t/l) + \kappa_0 l^2 H_2(t/l) + \tau_1 l H_3(t/l) + \nu_1 H_4(t/l)$$

f can now be written as

$$f(t) = h(t/l) + \psi(t),$$

for some function ψ. The error estimate for Hermite polynomials tells us that

$$|\psi(t)| \leq K_5 l^5$$

Since from Hermite interpolation theory [Pre75] we know

$$|h(t/l)| \leq \max(|\nu_i|) + l \max(|\tau_i|) + l^2 |\kappa_i|,$$

the lemma follows. $\square$

Figure 3.5: Cubic interpolant fit to torus.

Left: Shirman-Séquin with de Boor-Höllig-Sabin boundary curve.

Right: Cubic interpolant surface. The surface is ε-G^1 for $\varepsilon = 6.2$ degrees.

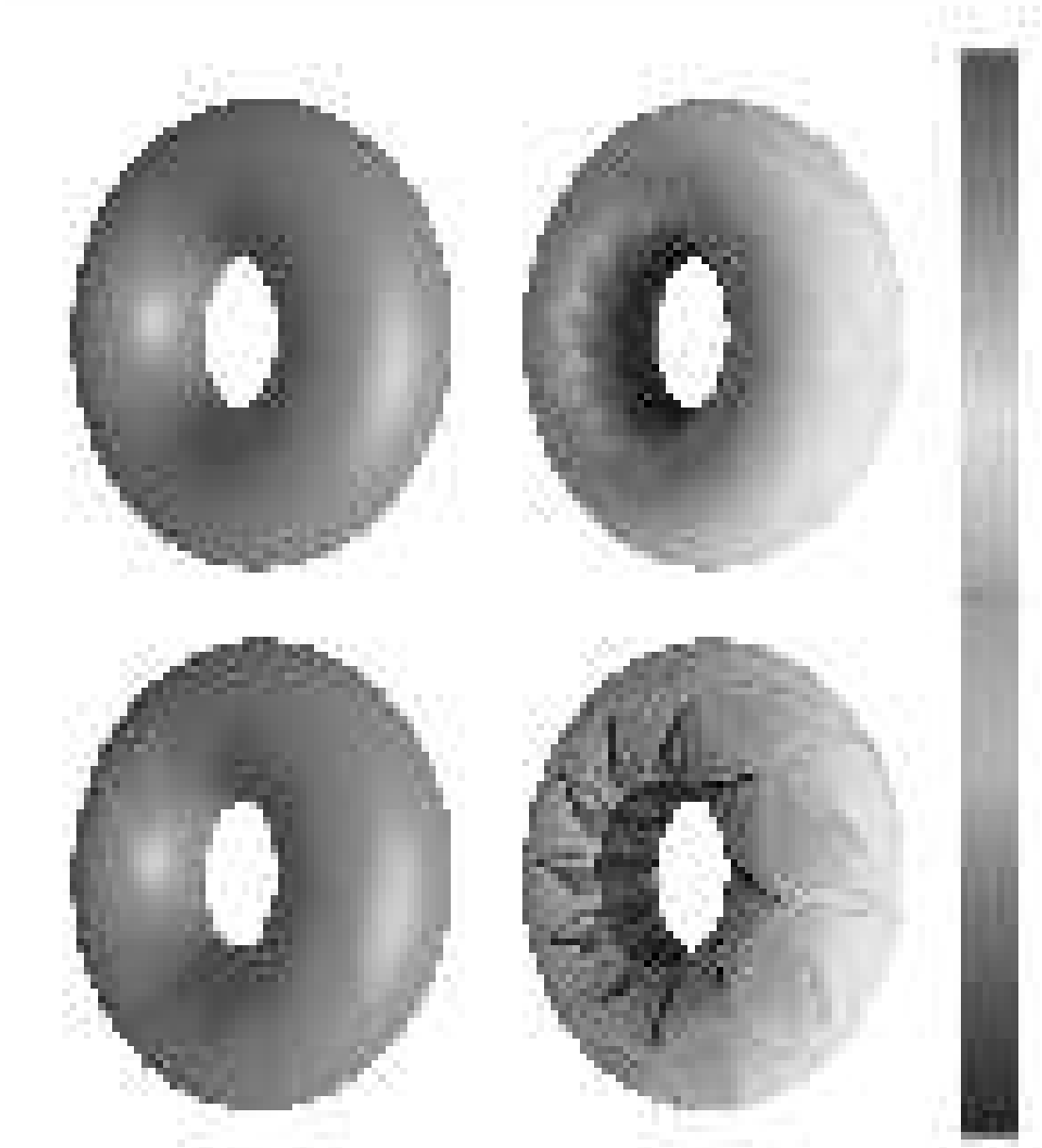
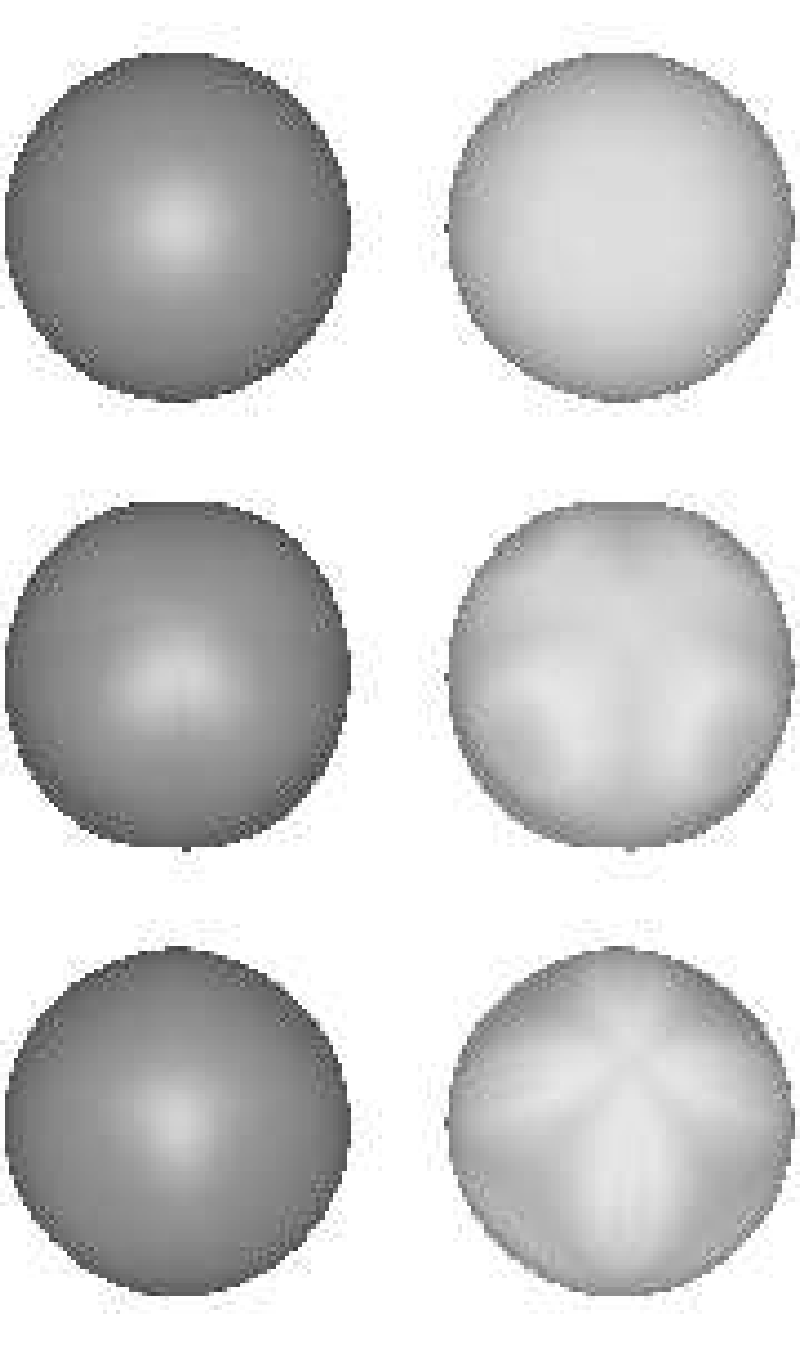


Figure 3.6: Cubic interpolant fit to octahedron.

Left: Shirman-Séquin with de Boor-Höllig-Sabin boundary curves; the maximum Gaussian curvature 1.4, the minimum is 0.44.

Center: The cubic interpolant fit to the octahedron. The maximum Gaussian curvature is 1.44; the minimum is 0.75.

Right: The cubic interpolant after one uniform refinement. The maximum Gaussian curvature is 1.08; the minimum is 0.98.



Chapter 4

THE BICUBIC INTERPOLANT

In the previous chapter, I introduced a high-order-of-approximation surface patch with a triangular domain. In this chapter, I present a tensor product surface patch having a quadrilateral domain.

A tensor product Bézier patch is a map from a rectangular domain $u \times v$ to $\mathcal{R}^k$ given by

$$\mathbf{C}(u,v) = \sum\sum \mathbf{P}_{i,j} B_i^n(u) B_j^m(v).$$

We will work with a bicubic patch in $\mathcal{R}^3$, i.e., where $k = 3$ and $m = n = 3$.

A bi-cubic patch has sixteen control points. If we are given second-order data at four vertices, and we wish to interpolate this data with a bicubic patch, then the method of de Boor-Höllig-Sabin can be used to set the boundary control points of the patch, leaving four interior points unset. These four points are known as *twist points*. The second-order data at a corner imposes a linear constraint on the closest interior control point. This leaves eight degrees of freedom to be set.

The linear constraint on each twist point may be thought of as a plane in which the twist point must lie. One approach for positioning a twist point is to set it to the projection of the zero twist point into the plane given by the second order data. (The zero twist point $\mathbf{C}_{1,1}$ is given by $\mathbf{C}_{1,0} + \mathbf{C}_{0,1} - \mathbf{C}_{0,0}$.) I will refer to this method as the *zero-twist bicubic*. One advantage of the zero-twist bicubic is that the setting for the twist points is simple. However, while this approach creates a patch that interpolates the data at four points to second order, the setting of the extra degrees of freedom is somewhat arbitrary.

An alternate approach for setting these degrees of freedom is to sample an additional point (in the patch interior) for position, tangent plane, and second fundamental form. This imposes eight more constraints, exactly matching the degrees of freedom. For reasons of symmetry, I take this sample at the patch center, i.e., $\mathbf{C}(\frac{1}{2}, \frac{1}{2})$.

Therefore, the problem we are solving is

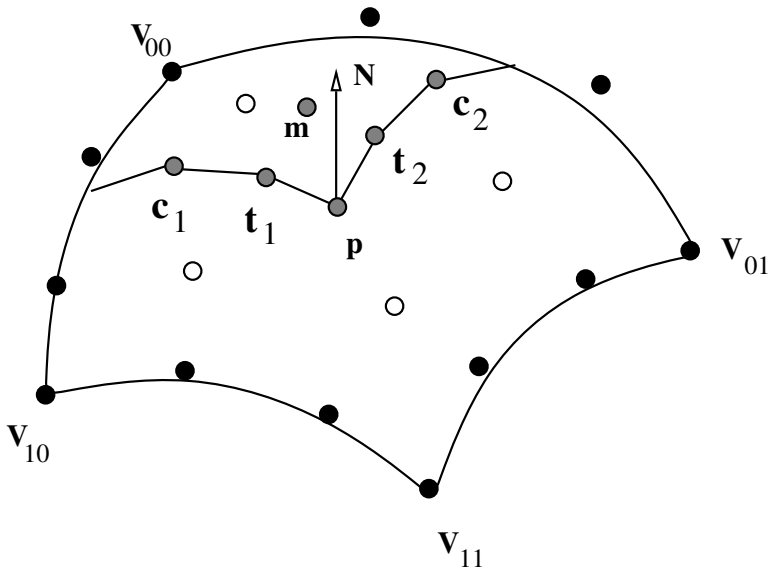


Figure 4.1: The bi-cubic interpolant's control points.

Given: Five points, $\mathbf{p}, \mathbf{V}_{0,0}, \mathbf{V}_{1,0}, \mathbf{V}_{0,1}$, and $\mathbf{V}_{1,1}$ with associated normals $\mathbf{N}, \hat{\mathbf{N}}_{0,0}, \hat{\mathbf{N}}_{1,0}, \hat{\mathbf{N}}_{0,1}$, and $\hat{\mathbf{N}}_{1,1}$ and second fundamental forms $\mathrm{II}, \mathrm{II}_{0,0}, \mathrm{II}_{1,0}, \mathrm{II}_{0,1}$, and $\mathrm{II}_{1,1}$.

Find: A quadrilateral parametric polynomial surface patch that interpolates the $\mathbf{V}_{i,j}$ to second order at the patch corners, and interpolates $\mathbf{p}$ to second order at the center of the patch.

I call my solution $\mathbf{C}$ the *bi-cubic interpolant*. Let $\mathbf{C}_{i,j}$, $i,j \in \{0,1,2,3\}$ be its control points. We subdivide $\mathbf{C}$ at (0.5,0.5) [Far90] and consider the control points given in Figure 4.1. In this figure, the black and white points are the control points of $\mathbf{C}$; the black points are the boundary points, determined by the method of de Boor-Höllig-Sabin, and the white points are the unknown interior points. The gray points are control points on one subdivision patch.

Let

$$\mathcal{C} = \begin{vmatrix} \mathbf{C}_{0,0} & \mathbf{C}_{0,1} & \mathbf{C}_{0,2} & \mathbf{C}_{0,3} \\ \mathbf{C}_{1,0} & \mathbf{C}_{1,1} & \mathbf{C}_{1,2} & \mathbf{C}_{1,3} \\ \mathbf{C}_{2,0} & \mathbf{C}_{2,1} & \mathbf{C}_{2,2} & \mathbf{C}_{2,3} \\ \mathbf{C}_{3,0} & \mathbf{C}_{3,1} & \mathbf{C}_{3,2} & \mathbf{C}_{3,3} \end{vmatrix}.$$

We can now express the subdivision points in terms of $\mathbf{C}$'s control points:

$$\mathbf{p} = \frac{1}{64}\mathcal{C}\begin{vmatrix} 1 & 3 & 3 & 1 \\ 3 & 9 & 9 & 3 \\ 3 & 9 & 9 & 3 \\ 1 & 3 & 3 & 1 \end{vmatrix} \tag{4.1}$$

$$\mathbf{t}_1 = \frac{1}{32}\mathcal{C}\begin{vmatrix} 1 & 2 & 1 & 0 \\ 3 & 6 & 3 & 0 \\ 3 & 6 & 3 & 0 \\ 1 & 2 & 1 & 0 \end{vmatrix} \qquad \mathbf{t}_2 = \frac{1}{32}\mathcal{C}\begin{vmatrix} 1 & 3 & 3 & 1 \\ 2 & 6 & 6 & 2 \\ 1 & 3 & 3 & 1 \\ 0 & 0 & 0 & 0 \end{vmatrix}$$

$$\mathbf{c}_1 = \frac{1}{16}\mathcal{C}\begin{vmatrix} 1 & 1 & 0 & 0 \\ 3 & 3 & 0 & 0 \\ 3 & 3 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{vmatrix} \qquad \mathbf{c}_2 = \frac{1}{16}\mathcal{C}\begin{vmatrix} 1 & 3 & 3 & 1 \\ 1 & 3 & 3 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{vmatrix}$$

$$\mathbf{m} = \frac{1}{16}\mathcal{C}\begin{vmatrix} 1 & 2 & 1 & 0 \\ 2 & 4 & 2 & 0 \\ 1 & 2 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{vmatrix}$$

where the "product" of two matrices should be considered a "dot product," i.e., $CD = \sum c_{ij}d_{ij}$.

After setting the boundary control points with the method of de Boor-Höllig-Sabin, the constraints on the interior control points of $\mathbf{C}$ can be written as follows:

$$\mathbf{C}(0.5, 0.5) = \mathbf{p} \tag{4.2}$$

$$< (\mathbf{t}_1 - \mathbf{p}),\ \hat{\mathbf{N}} > = 0 \qquad < (\mathbf{t}_2 - \mathbf{p}),\ \hat{\mathbf{N}} > = 0 \tag{4.3}$$

$$< (\mathbf{C}_{1,1} - \mathbf{C}_{0,0}),\ \hat{\mathbf{N}}_{00} > \ = \ \mathrm{II}_{00}(\mathbf{C}_{1,0} - \mathbf{C}_{0,0},\ \mathbf{C}_{0,1} - \mathbf{C}_{0,0}) \tag{4.4}$$

$$< (\mathbf{C}_{2,1} - \mathbf{C}_{3,0}),\ \hat{\mathbf{N}}_{10} > \ = \ \mathrm{II}_{10}(\mathbf{C}_{2,0} - \mathbf{C}_{3,0},\ \mathbf{C}_{3,1} - \mathbf{C}_{3,0}) \tag{4.5}$$

$$< (\mathbf{C}_{1,2} - \mathbf{C}_{0,3}),\hat{\mathbf{N}}_{01} > \ = \ \mathrm{II}_{01}(\mathbf{C}_{1,3} - \mathbf{C}_{0,3},\ \mathbf{C}_{0,2} - \mathbf{C}_{0,3}) \tag{4.6}$$

$$< (\mathbf{C}_{2,2} - \mathbf{C}_{3,3}),\hat{\mathbf{N}}_{11} > \ = \ \mathrm{II}_{11}(\mathbf{C}_{3,2} - \mathbf{C}_{3,3},\ \mathbf{C}_{2,3} - \mathbf{C}_{3,3}) \tag{4.7}$$

$$< (\mathbf{m} - \mathbf{t}_1),\ \hat{\mathbf{N}} > = \mathrm{II}(\mathbf{t}_1 - \mathbf{p},\ \mathbf{t}_2 - \mathbf{p}), \tag{4.8}$$

$$2 < [(\mathbf{c}_1 - \mathbf{t}_1) - (\mathbf{t}_1 - \mathbf{p})],\ \hat{\mathbf{N}} > \quad = \quad 3\ \mathrm{II}(\mathbf{t}_1 - \mathbf{p},\ \mathbf{t}_1 - \mathbf{p}), \tag{4.9}$$

$$2 < [(\mathbf{c}_2 - \mathbf{t}_2) - (\mathbf{t}_2 - \mathbf{p})],\ \hat{\mathbf{N}} > \quad = \quad 3\ \mathrm{II}(\mathbf{t}_2 - \mathbf{p},\ \mathbf{t}_2 - \mathbf{p}). \tag{4.10}$$

To simplify the constraints, I express the data in an orthonormal coordinate frame $\mathcal{F}$ whose origin is $\mathbf{p}$ and whose basis vector $(0,0,1)$ is $\hat{\mathbf{N}}$. The selection of the other two basis vectors is arbitrary. In this frame, we then express II in the basis $(1,0,0)$, $(0,1,0)$, and denote the normal section curvature in direction $(1,0,0)$ as L, in direction $(0,1,0)$ as N, and the normal section of the mixed partial as M.

The unknowns are the coordinates of the control points $\mathbf{C}_{i,j}$, where $i,j \in \{1,2\}$. Let $x(\mathbf{p})$, $y(\mathbf{p})$, and $z(\mathbf{p})$ denote the coordinate functions with respect to a given frame. Relative to $\mathcal{F}$, we can now write Equation 4.2 as three independent linear equations:

$$9(x(\mathbf{C}_{1,1}) + x(\mathbf{C}_{1,2}) + x(\mathbf{C}_{2,1}) + x(\mathbf{C}_{2,2})) = -x \left(\mathcal{C} \begin{vmatrix} 1 & 3 & 3 & 1 \\ 3 & 0 & 0 & 3 \\ 3 & 0 & 0 & 3 \\ 1 & 3 & 3 & 1 \end{vmatrix} \right).$$

$$9(y(\mathbf{C}_{1,1}) + y(\mathbf{C}_{1,2}) + y(\mathbf{C}_{2,1}) + y(\mathbf{C}_{2,2})) = -y \left(\mathcal{C} \begin{vmatrix} 1 & 3 & 3 & 1 \\ 3 & 0 & 0 & 3 \\ 3 & 0 & 0 & 3 \\ 1 & 3 & 3 & 1 \end{vmatrix} \right).$$

$$9(z(\mathbf{C}_{1,1}) + z(\mathbf{C}_{1,2}) + z(\mathbf{C}_{2,1}) + z(\mathbf{C}_{2,2})) = -z \left(\mathcal{C} \begin{vmatrix} 1 & 3 & 3 & 1 \\ 3 & 0 & 0 & 3 \\ 3 & 0 & 0 & 3 \\ 1 & 3 & 3 & 1 \end{vmatrix} \right).$$

Note that the right hand side of each equation is known.

Equations 4.3 simplify to the following:

$$< (\mathbf{t}_1 - \mathbf{p}),\ \hat{\mathbf{N}} > = z(\mathbf{t}_1) = 0,$$

$$< (\mathbf{t}_2 - \mathbf{p}),\ \hat{\mathbf{N}} > = z(\mathbf{t}_2) = 0.$$

Rewritten in terms of the unknown control points, we find

$$6z(\mathbf{C}_{1,1}) + 3z(\mathbf{C}_{1,2}) + 6z(\mathbf{C}_{2,1}) + 3z(\mathbf{C}_{2,2}) = -z \left(\mathcal{C} \begin{vmatrix} 1 & 2 & 1 & 0 \\ 3 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{vmatrix} \right),$$

$$6z(\mathbf{C}_{1,1}) + 6z(\mathbf{C}_{1,2}) + 3z(\mathbf{C}_{2,1}) + 3z(\mathbf{C}_{2,2}) = -z \left(\mathcal{C} \begin{vmatrix} 1 & 3 & 3 & 0 \\ 2 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{vmatrix} \right).$$

Equations 4.4-4.7 gives four linear equations, each involving three unknowns. Equation 4.4 can be rewritten as

$$\begin{aligned} x(\mathbf{C}_{1,1})x(\mathbf{N}_{0,0}) + y(\mathbf{C}_{1,1})y(\mathbf{N}_{0,0}) + z(\mathbf{C}_{1,1})z(\mathbf{N}_{0,0}) \quad = \quad & (2x(\mathbf{C}_{1,0}) - x(\mathbf{C}_{0,0}))x(\mathbf{N}_{0,0}) + \\ & (2y(\mathbf{C}_{1,0}) - y(\mathbf{C}_{0,0}))y(\mathbf{N}_{0,0}) + \\ & (2z(\mathbf{C}_{1,0}) - z(\mathbf{C}_{0,0}))z(\mathbf{N}_{0,0}) + \\ & \mathrm{II}(\mathbf{C}_{1,0} - \mathbf{C}_{0,0}, \mathbf{C}_{0,1} - \mathbf{C}_{0,0}). \end{aligned}$$

The other three equations have a similar form.

Equations 4.8, 4.9, and 4.10, are non-linear in the unknowns. Writing Equation 4.8 relative to $\mathcal{F}$ and our representation of II yields

$$64z(\mathbf{m}) = x(\mathbf{t}_1)x(\mathbf{t}_2)L + x(\mathbf{t}_1)y(\mathbf{t}_2)M + y(\mathbf{t}_1)x(\mathbf{t}_2)M + y(\mathbf{t}_1)y(\mathbf{t}_2)N.$$

Similarly, we rewrite Equations 4.9 and 4.10 as

$$\frac{128}{3}z(\mathbf{c}_1) = x(\mathbf{t}_1)^2L + 2x(\mathbf{t}_1)y(\mathbf{t}_1)M + y(\mathbf{t}_1)^2N,$$

$$\frac{128}{3}z(\mathbf{c}_2) = x(\mathbf{t}_2)^2L + 2x(\mathbf{t}_2)y(\mathbf{t}_2)M + y(\mathbf{t}_2)^2N.$$

When expanded in terms of $\mathbf{C}_{i,j}$'s coefficients, each of these last three equations is quadratic in twelve unknowns. The other nine equations are linear and can be used to reduce the number of unknowns in the quadratic equations to three. I use the method of Chionh et. al. to solve this quadratic system of equations [CGM91].

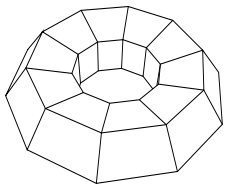


Figure 4.2: Quadrilateral torus mesh.

As an example, I fit the bicubic interpolant to the the mesh shown in Figure 4.2. The resulting surface and its curvature plot are shown in Figure 4.4. Solutions to the quadratic systems were found for the exterior patches, but none were found for the interior patches. Thus, in the regions along the exterior, the patches interpolate five data points to second order, while the patches on the interior only interpolate four points to second order. When the quadratic system fails to have a solution, I use the zero-twist bicubic scheme.

4.1 Order of Approximation of the Bicubic Interpolant

For scalar functions, the data at the corners of a rectangle can be interpolated with a bicubic patch that interpolates the position, the first partials and the mixed partial derivative with an order of approximation of $O(h^4)$ [Pre75] (note that second derivatives are not interpolated; only the mixed partial). This gives fourth order convergence for the bicubic interpolant. A degrees of freedom analysis indicates that the bicubic interpolant might have seventh order of convergence. However, the order of convergence of the interpolant can be no higher than that of its boundary curves, which only have sixth order convergence.

The theorem of Section 3.5 can be applied to the bicubic interpolant to give fifth order convergence, provided we can obtain the necessary derivative bounds. As the data at the center would not be used in this proof, we would also have fifth order convergence for the zero-twist bicubic (again, we will need to show that the derivatives are bounded).

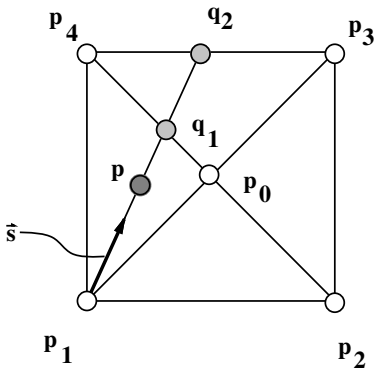


Figure 4.3: The domain of the reparameterized bicubic interpolant.

4.2 Sixth Order of Convergence

In this section, I show that if the sixth derivatives of a reparameterization of the bicubic interpolant are bounded uniformly in h, then the bicubic interpolant has sixth order convergence. The proof is similar to the proof of fifth order convergence for the cubic interpolant. Again, I start with a theorem about functions over the plane:

> **Theorem:** Let $Q = \mathbf{p}_1\mathbf{p}_2\mathbf{p}_3\mathbf{p}_4$ be the rectangular domain with center $\mathbf{p}_0$ (Figure 4.3) of an interpolant $P(x, y)$ that interpolates a C^6 surface $S(x, y)$ to second order at the five $\mathbf{p}_i$. If the sixth derivatives of P are uniformly bounded in h (the maximum distance between any two of the data points) then
>
> $$|P - S| \leq O(h^6)$$
>
> for all points in Q.

To prove this theorem, I will need the following two lemmas:

Lemma 3: If $f(0) = 0$, $f'(0) = 0$, $f(l/2) = 0$, $f(l) = 0$, $f'(l) = 0$, $|f^{(5)}(t)| \leq K_5$ for $t \in (0, l)$, then $|f(t)| \leq K_5 l^5$,

Lemma 4: If $f(0) = \nu_0$, $f'(0) = \tau_0$, $f''(0) = \kappa_0$, $f(\tilde{t}) = \nu_1$, $f'(\tilde{t}) = \tau_1$, for some $\tilde{t} \in (0, l)$, $f(l) = \nu_2$, and $|f^{(6)}(t)| \leq K_6$ for $t \in (0, l)$, then $|f(t)| \leq \max|\nu_i| + l\max(|\tau_i|) + l^2|\kappa_0| + K_6 l^6$.

Proof: Let ϕ denote the error function: $\phi = P - S$. Without loss of generality, let $\mathbf{p}$ be a point on the interior of the domain as shown in Figure 4.3. Let $\vec{\mathbf{s}}$ be the unit direction from $\mathbf{p}_1$ to $\mathbf{p}$. Let ℓ be the line given by $\mathbf{p}$ and $\vec{\mathbf{s}}$. Let $\mathbf{q}_1$ be the intersection of line ℓ with the segment $\mathbf{p}_4\mathbf{p}_0$, and let $\mathbf{q}_2$ be the intersection of ℓ with the segment $\mathbf{p}_3\mathbf{p}_4$.

Since P interpolates S to second order at both $\mathbf{p}_2$ and $\mathbf{p}_4$, we know from Hermite interpolation that $|\phi(\mathbf{q}_1)| \leq O(h^6)$. And since

$$D_{\vec{\mathbf{s}}}\phi(\mathbf{p}_4) = D^2_{\vec{\mathbf{s}},\vec{\mathbf{s}}_{4,0}}\phi(\mathbf{p}_4) = D_{\vec{\mathbf{s}}}\phi(\mathbf{p}_0) = D^2_{\vec{\mathbf{s}},\vec{\mathbf{s}}_{4,0}}\phi(\mathbf{p}_2) = D^2_{\vec{\mathbf{s}},\vec{\mathbf{s}}_{4,0}}\phi(\mathbf{p}_2) = 0,$$

where $\vec{\mathbf{s}}_{4,0}$ is the unit direction from $\mathbf{p}_4$ to $\mathbf{p}_0$, we know by Lemma 3 that $|D_{\vec{\mathbf{s}}}\phi(\mathbf{q}_1)| \leq O(h^5)$.

Therefore, we have

$$\phi(\mathbf{p}_1) = D_{\vec{\mathbf{s}}}\phi(\mathbf{p}_1) = D^2_{\vec{\mathbf{s}},\vec{\mathbf{s}}}\phi(\mathbf{p}_1) = 0,$$

$$|\phi(\mathbf{q}_1)| \leq O(h^6), \quad |D_{\vec{\mathbf{s}}}\phi(\mathbf{q}_1)| \leq O(h^5),$$

and

$$|\phi(\mathbf{q}_2)| \leq O(h^6).$$

By Lemma 4 we get the result $|\phi(\mathbf{p})| \leq O(h^6)$. □

Conjecture: The bicubic interpolant has sixth order of convergence.

The surface $\mathbf{S}$ and the interpolant $\mathbf{C}$ are reparameterized over the plane to give functions S and C. If the sixth derivatives of C are uniformly bounded in h, then the theorem can be applied and the conjecture holds.

One thing to note is that the above proof does not make use of the second order data at $\mathbf{p}_0$. Thus, if the system of second degree equations fails to have a solution, the interpolant still has sixth order convergence. The second fundamental form at $\mathbf{p}_0$ is used only for setting degrees of freedom in a geometric fashion.

To summarize, if the construction succeeds everywhere, the bicubic interpolant will have sixth order convergence. If the system of equations used to determine the interior control points can be reduced to a system of three quadratic equations that cannot be solved, then the interpolant still has sixth order convergence (assuming a method is devised for setting the three remaining degrees of freedom). If instead of determining settings for these degrees of freedom we use the zero-twist bicubic

interpolant, then the interpolant has fifth order convergence. And if de Boor-Höllig-Sabin fails on any of the boundary curves, then the order of approximation drops to fourth order.

4.3 Proof of Lemmas

Lemma 3: If $f(0) = 0$, $f'(0) = 0$, $f(l/2) = 0$, $f(l) = 0$, $f'(l) = 0$, $|f^{(5)}(t)| \leq K_5$ for $t \in (0, l)$. then $|f(t)| \leq K_5 l^5$,

A quartic Hermite polynomial interpolating the above five values is identically zero with an error term of $O(h^5)$. The constant factor is less than bound on the fifth derivatives of f. □

Lemma 4: If $f(0) = \nu_0$, $f'(0) = \tau_0$, $f''(0) = \kappa_0$, $f(s) = \nu_1$, $f'(s) = \tau_1$, $f(l) = \nu_2$, $|f^{(6)}(t)| \leq K_6$ for $t \in (0, l)$, for some $s \in (0, l)$, then $|f(t)| \leq \max|\nu_i| + l\max(|\tau_i|) + l^2|\kappa_0| + K_6 l^6$.

The proof of this lemma is similar to the proof of Lemma 2. Write f as a Hermite polynomial with an error term:

$$f(t) = H^6(t) + \psi(t),$$

where

$$H^6(t) = \nu_0 H_0^6(t/l) + \tau_0 l H_1^6(t/l) + \kappa l^2 H_2^6(t/l) + \nu_1 H_3^6(t/l) + \tau_1 l H_4^6(t/l) + \nu_2 H_5^6(t/l).$$

The error estimate for Hermite polynomials gives us

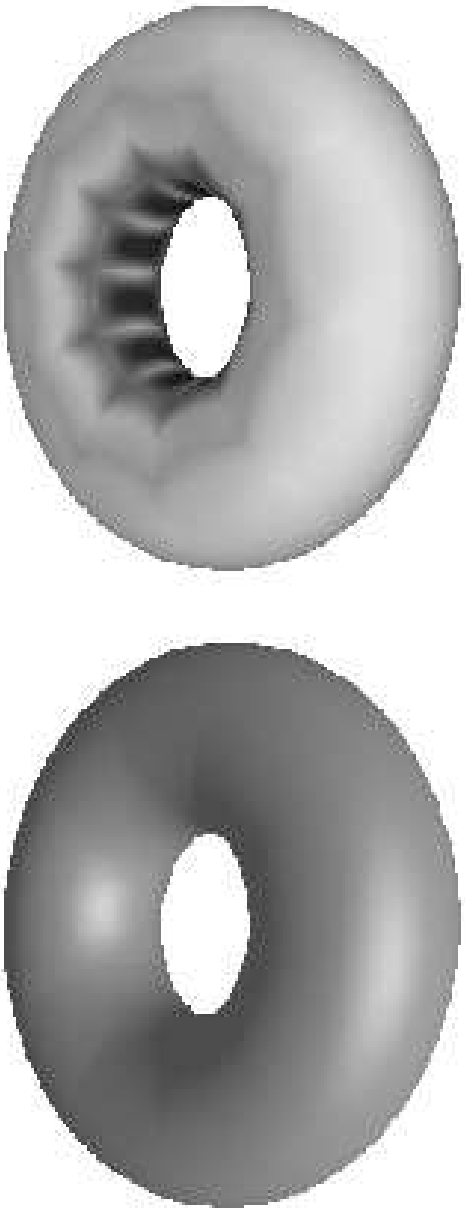
$$|\psi(t)| \leq \frac{K_6}{6!} l^6.$$

Since

$$|H(t)| \leq \max(|\nu_i|) + l\max(|\tau_i|) + l^2|\kappa_0|.$$

the lemma follows. □

Figure 4.4: Bicubic interpolant torus and its curvature plot



Chapter 5

APPLICATIONS

The surface interpolation methods survey in Chapter 2 construct a piecewise continuous surface with patches that meet with tangent plane continuity. Most applications, however, approximate the surface with an approximation that is not tangent plane continuous. For example, in a manufacturing process, machining tools create surfaces that are only approximately continuous. Thus, it may be advantageous to relax the continuity conditions earlier in the design process if it simplies the construction, simplifies the representation, improves efficiency, or improves the surface shape. In this chapter, I show how to use the cubic interpolant to construct approximately continuous surfaces, and use this interpolation scheme for the approximation of known surfaces.

5.1 Approximate Continuity

In the previous two chapters, I fit surfaces to meshes without considering the discontinuity in surface normals between neighboring patches. As all the degrees of freedom in each patch are used to match derivative information, we are unable to form a G^1 join in general. To ensure some degree of smoothness, I use the following relaxation of tangent plane continuity:

> **Definition:** Let $\mathbf{S}$ be a piecewise C^1, globally C^0 surface. I define $\mathbf{S}$ to be ε-G^1 if the the maximum angle between two surface normals at any point $\mathbf{p}$ on $\mathbf{S}$ is bounded by ε.

As stated, the definition of ε-G^1 allow for a surface to have a "razor edge". However, I consider such surfaces to not be ε-G^1 for $\varepsilon \leq 90$ degrees.

Note that a G^1 surface is ε-G^1 for every $\varepsilon \geq 0$. It has been informally recognized for some time that this weakened form of continuity is adequate for many applications. For example, a milling machine constructs a surface that is only ε-G^1, where ε is governed by the machine's precision.

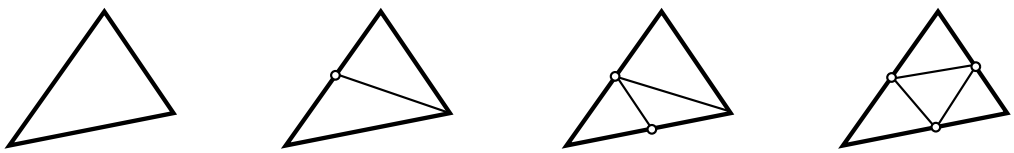


Figure 5.1: Mesh refinement. On left is the original triangle. The other three triangles show the retriangulation when one, two, or three sides are subdivided.

In this dissertation, surface quality has been measured by visually examining the surface. Similar criteria will be used to evaluate the approximately continuous surfaces constructed in this chapter. For the red, diffuse material described in Section 2.4, I empirically determined that angles less than 2 degrees are not visible in shaded images. Note that the visibility of the discontinuity is also a function of the viewing angle, and of the direction to the light source. The surfaces shown in this dissertation were evaluated by rotating the objects and looking at them from various angles.

5.2 A Piecewise Cubic Surface Scheme

The previous chapters shows how to fit a single geometric Hermite patch to one face of data. In this section, I will show how to create an ε-G^1 surface with the cubic interpolant.

For a triangular net of data, we can construct a cubic interpolant patch for each face. A surface constructed in this way is globally C^0, at least G^1 at all vertices, and G^2 at vertices where the construction succeeded on all the surrounding patch boundaries and patch interiors.

Neighboring patches of this surface, however, may join with large G^1 discontinuities along boundary curves. In Section 5.3, methods of determining the magnitude of the discontinuity are discussed. If, by any method, the discontinuity along an edge is determined to be small enough, the edge is left unchanged. If the discontinuity is large, then the mesh is refined adaptively in that region as indicated in Figure 5.1.

The refinement process generates new data points. At each new point, the position, normal and surface curvature must be determined. When approximating a

known surface, this data is sampled from the surface. The cubic interpolant is then used to construct patches for the new faces, and the process is repeated.

One slight difficulty occurs when the cubic interpolant is being used to approximate a surface with a smooth boundary. The above method restricts the boundaries of the cubic patches to be planar. Since the particular plane used for each patch boundary is chosen independent of the other patches, the resulting boundary of the surface will not in general be G^1. To achieve a G^1 surface boundary, I sample the tangents of the surface along the boundary and fit a non-planar geometric Hermite space curve to the data (Section 2.4.4).

5.3 Bounding the Discontinuity

To ensure that the piecewise cubic surface is ε-G^1 for a given ε, we need to bound the discontinuity in surface normals between two cubic patches. Currently, I sample the normals of both patches at ten locations on their common boundary and use the maximum discontinuity between these pairs of normals as an estimate of the maximum discontinuity along the entire boundary.

Ideally, we could use a more precise method. The purely analytic method given below requires finding the roots of a high degree polynomial. Therefore, it may be computationally less expensive to bound the discontinuity rather than compute it exactly. This is a topic for future research.

5.3.1 Analytic Bound

Let $\mathbf{F}$ and $\mathbf{G}$ be two cubic polynomial patches sharing a boundary

$$\mathbf{H}(t) = \sum_{i=0}^{3} \mathbf{H}_i B_i^3(t),$$

and let the second layer of control points of $\mathbf{F}$ and $\mathbf{G}$ be $\mathbf{F}_i$ and $\mathbf{G}_i$, $i = 0, 1, 2$. The tangent vector along the boundary is $\mathbf{H}'(t) = \sum_{i=0}^{2} \vec{\mathbf{H}}_i B_i^2(t)$, where $\vec{\mathbf{H}}_i = \mathbf{H}_{i+1} - \mathbf{H}_i$. Cross-boundary tangents are given by $\vec{\mathbf{F}}(t) = \sum_{i=0}^{2} \vec{\mathbf{F}}_i B_i^2(t)$ and $\vec{\mathbf{G}}(t) = \sum_{i=0}^{2} \vec{\mathbf{G}}_i B_i^2(t)$ where $\vec{\mathbf{F}}_i = \mathbf{F}_i - \mathbf{H}_i$ and $\vec{\mathbf{G}}_i = \mathbf{G}_i - \mathbf{H}_i$. The normal to patch $\mathbf{F}$ along the common boundary is then

$$\hat{\mathbf{N}}_a(t) = \frac{\mathbf{H}'(t) \times \vec{\mathbf{F}}(t)}{|\mathbf{H}'(t) \times \vec{\mathbf{F}}(t)|}$$

The normal to patch $\mathbf{G}$ along the boundary is

$$\hat{\mathbf{N}}_c(t) = \frac{\vec{\mathbf{G}}(t) \times \mathbf{H}'(t)}{|\vec{\mathbf{G}}(t) \times \mathbf{H}'(t)|}$$

The cosine of the angle between the two normals is given by $d(t) = < \mathbf{N}_a(t), \mathbf{N}_c(t) >$ Note that $-1 \leq d(t) \leq 1$ for all t. If $d(t)$ is identically 1 for all t, then the patches meet with a common tangent plane. The maximum discontinuity in the normals between two patches occurs at the minimum value of $d(t)$ for $0 \leq t \leq 1$.

Let $N = < (\mathbf{H}' \times \vec{\mathbf{F}}), (\vec{\mathbf{G}} \times \mathbf{H}') >$ and $D = |\mathbf{H}' \times \vec{\mathbf{F}}||\vec{\mathbf{G}} \times \mathbf{H}'|$. Then $d(t) = N/D$. To find the extrema of d, I work with d^2, as d^2 is a rational function. The zeros of the first derivative give us the extrema:

$$\begin{aligned} d^2(t) &= \frac{N^2(t)}{D^2(t)} \\ \frac{d}{dt}d^2(t) &= 2d(t)d'(t) \\ &= \frac{2NN'D^2 - N^2(D^2)'}{D^4}. \end{aligned}$$

Setting $d'(t)$ to zero and multiplying both sides by D^4 gives

$$0 = 2N(2N'D^2 - N(D^2)').$$

Thus, the extrema of d^2 can be determine by finding the roots of N (a degree 8 polynomial) and the roots of the degree 19 polynomial, $2N'D^2 - N(d^2)'$.

5.4 Approximation of Known Functions

One application of the cubic interpolant and the bicubic interpolant occurs in the approximation to known functions. In the previous chapter, these interpolants have been used to approximate a toroidal data set. In this section, I show how to use the cubic interpolant to approximate algebraic surfaces, offset surfaces, and S-patches. For all three applications, the primary problem is to compute the second fundamental forms.

5.4.1 Algebraic surfaces

An algebraic surface is an implicit polynomial surface. That is, it is the set of all points such that the equation

$$P(x, y, z) = 0$$

is satisfied, where P is a polynomial in x, y, and z. Algebraic surfaces and parametric polynomial surfaces both have uses in CAGD. Sometimes it is useful to convert between the two representations. While a parametric surface can always be converted to implicit form, it is sometimes impossible to convert an algebraic surface to a parametric polynomial surface. If an approximation to the algebraic surface is acceptable, then the cubic interpolant can be used to obtain a piecewise polynomial approximation.

Since the evaluation of an algebraic function yields a scalar that is zero if the point is on the surface, as a first step we must locate points on an implicit surface. This is typically done by searching a portion of space, leading to the related difficulty of determining which region of space to search. As this problem is beyond the scope of this dissertation, I assume that the region of interest is given. A variation of marching cubes is then used to find an initial sampling of the surface [HDD+92].

The data points in the marching cubes mesh are only near the surface. Thus, the points are projected back onto the implicit surface, and normals and curvature data are added using the method of Schweitzer and DeRose [SD]. Schweitzer and DeRose show that if f is the blossom of an algebraic function F, and $\mathbf{p}$ is a point on F, then the normal of F at $\mathbf{p}$ is given by

$$\hat{\mathbf{N}}(\mathbf{p}) = \frac{\sum_{i=1}^{3} f(\hat{\mathbf{e}}_i, \mathbf{p}, \ldots \mathbf{p})\hat{\mathbf{e}}_i}{(\sum_{i=1}^{3} f(\hat{\mathbf{e}}_i, \mathbf{p}, \ldots \mathbf{p})^2)^{1/2}},$$

where $\{\hat{\mathbf{e}}_1, \hat{\mathbf{e}}_2, \hat{\mathbf{e}}_3\}$ is a basis for the space. The second fundamental form of $\mathbf{F}(\mathbf{p}) = 0$ is

$$\mathrm{II}_{\mathbf{p}}(\vec{\mathbf{u}}, \vec{\mathbf{v}}) = \frac{(n-1)f(\vec{\mathbf{u}}, \vec{\mathbf{v}}, \mathbf{p}, \ldots, \mathbf{p})}{(\sum_{i=1}^{3} f(\hat{\mathbf{e}}_i, \mathbf{p}, \ldots \mathbf{p})^2)^{1/2}}.$$

Once I have an initial mesh, the cubic interpolant is fit to the data, and the mesh is refined where needed.

Meshes for the function $4x^2 + 2y^2 + z^2 - 4 = 0$ are given in Figure 5.2 . The number of faces in each mesh appears below below the mesh. The left most mesh is

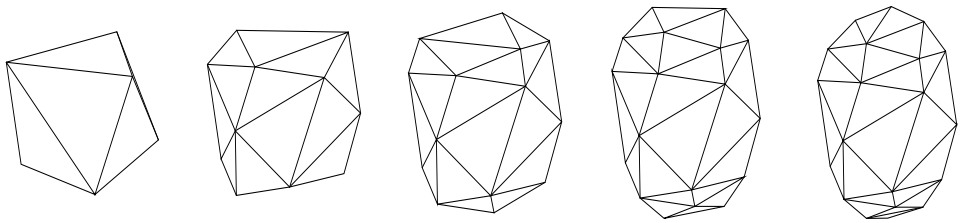


Figure 5.2: Meshes for the function $4x^2 + 2y^2 + z^2 - 4 = 0$.

the initial mesh, and the remaining four show the mesh after 1, 2, 3, and 4 levels of refinement. In Figure 5.7, the Gaussian curvature plots of the surface corresponding to these meshes are shown in two views: one matching the view of the meshes, and one from the top. The image in the bottom right is a shaded image of the final surface. The final surface is ε-G^1 for $\varepsilon = 0.4$ degrees.

Analysis

The example in the previous section is fairly simple. In particular, there are no degeneracies in the surface (i.e., singular points, singular lines, etc). In the presence of degeneracies, a different algorithm would have to be devised to compute the initial mesh, as the marching cubes algorithm fails to account for the degeneracies.

One appealing property illustrated in Figure 5.2 is that the cubic interpolant refines the approximation "where needed." Typically, such refinement is needed in regions of high curvature. While the refinement in the example exhibits these properties, in general the current refinement strategy (of refining along boundaries with a high normal discontinuity) is inadequate to catch these regions. For example, given a function that is a bump on a plane, the cubic interpolant will reconstruct a plane unless one of the initial samples lies on the bump.

5.4.2 Offset surfaces

Given a surface $\mathbf{S}$, the offset surface $\mathbf{S}^*$ is computed by moving a fixed distance along the normals to $\mathbf{S}$. More precisely, if $\hat{\mathbf{N}}_\mathbf{S}$ is the normal to $\mathbf{S}$, then

$$\mathbf{S}^*(\mathbf{t}) = \mathbf{S}(\mathbf{t}) + d\hat{\mathbf{N}}_\mathbf{S}(\mathbf{t}),$$

where d is a fixed constant.

An offset surface is typically more complex mathematically than the base surface. For example, if $\mathbf{S}$ is a parametric polynomial surface, then the offset to $\mathbf{S}$ is not in general a parametric polynomial surface. This causes difficulties for many modeling systems, as such systems are equipped to deal with parametric polynomial surfaces but cannot explicitly represent offsets to polynomial surfaces. Often, however, users are satisfied with an approximation to the offset surface.

To use the cubic interpolant to approximate offsets to surfaces, the offset surface must be sampled at a variety of locations for position, tangent plane, and second fundamental form. The position is given by the definition of the offset surface. The mapping of a vector $\vec{\mathbf{v}}$ in the tangent plane of $\mathbf{S}$ to $\vec{\mathbf{v}}^*$ in the tangent plane of $\mathbf{S}^*$ is given by the following:

$$\begin{aligned} \mathbf{D}_{\vec{\mathbf{v}}^*}\mathbf{S}^* &= \mathbf{D}_{\vec{\mathbf{v}}}\mathbf{S} + d\mathbf{D}_{\vec{\mathbf{v}}}\hat{\mathbf{N}}_\mathbf{S} \\ &= \vec{\mathbf{v}} - d(\mathrm{II}^\mathbf{S}(\vec{\mathbf{v}}, \hat{\mathbf{p}}_1)\hat{\mathbf{p}}_1 + \mathrm{II}^\mathbf{S}(\vec{\mathbf{v}}, \hat{\mathbf{p}}_2))\hat{\mathbf{p}}_2, \end{aligned} \tag{5.1}$$

where $\hat{\mathbf{p}}_1$ and $\hat{\mathbf{p}}_2$ are the principal directions of $\mathbf{S}$. Note that $\vec{\mathbf{v}}^*$ is a linear combination of vectors in the tangent plane of $\mathbf{S}$. Thus, we also have

$$\hat{\mathbf{N}}_{\mathbf{S}^*}(\mathbf{p}) = \hat{\mathbf{N}}_\mathbf{S}(\mathbf{p}).$$

Further note that $\vec{\mathbf{v}}*$ is parallel to $\vec{\mathbf{v}}$ if and only if $\vec{\mathbf{v}}$ is a principal direction. This fact may be used to show that principle directions on the base surface map to principal directions on the offset surface.

The mapping of curvatures in a principal direction $\vec{\mathbf{p}}$ having curvature k quickly follows:

$$\begin{aligned} \vec{\mathbf{p}}* &= \vec{\mathbf{p}} - d\mathrm{II}(\vec{\mathbf{p}}, \vec{\mathbf{p}})\vec{\mathbf{p}} \\ &= (1 - dk)\vec{\mathbf{p}}, \end{aligned}$$

Table 5.1: Refinement color map. A color map also appears to the right of the refinement images.

Refinement Level	Color
6+	white
5	light green
4	lavender
3	red
2	blue
1	yellow-brown
0	dark green

implying that the curvature of the offset surface in direction $\vec{\mathbf{p}}*$ is $k/(1-dk)$. The mapping of the principal directions gives us a complete characterization of the second fundamental form of $\mathbf{S}^*$.

On the left in Figure 5.8 is an offset to a bicubic tensor product B-spline surface consisting of nine bicubic patches. The cubic interpolant approximation to this surface appears in the center of this figure. The cubic interpolant surface was refined seven times; the surface consists of 148 patches and has is ε-G^1 for $\varepsilon = 0.72$ degrees. A map showing the refinement level appears on the right in the figure. Table 5.1 gives the mapping of colors to levels of refinement. The small region that was refined six and seven times occurs along the surface's right edge.

Note that in this example, the boundary of the offset surface is composed of four G^1 curves. To construct G^1 boundaries for the approximation, I sampled the tangents of the boundary of the offset surface (Equation 5.1) and constructed non-planar geometric Hermite curves.

5.4.3 S-Patches

In the previous chapters, I have looked at three and four sided surface patches. Sometimes n-sided patches are needed, where n is greater than four. An S-patch is one type of patch that can be used to interpolate a face with an arbitrary number of vertices. One difficultly with S-patches is they have high rational degree. In this

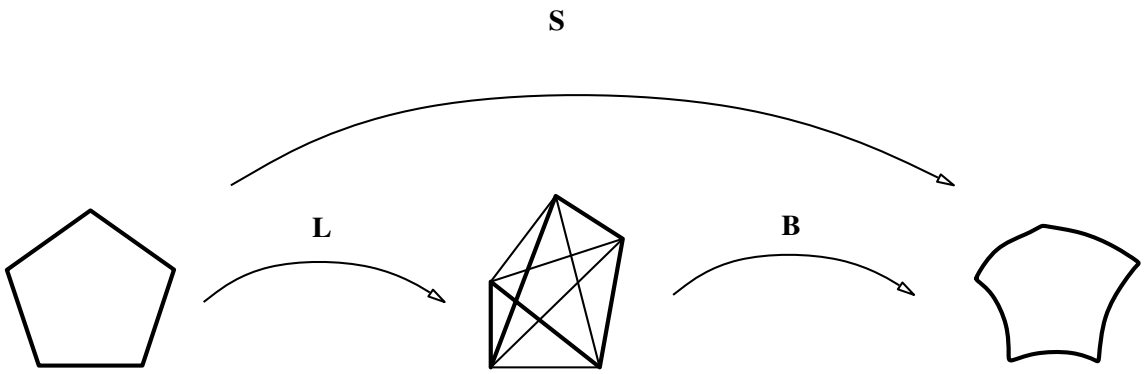


Figure 5.3: A five sided S-patch.

section I show how to approximate a G^1 S-patch network with the cubic interpolant.

An S-patch is the composition of two maps, $\mathbf{S} = \mathbf{B} \circ \mathbf{L}$. The map $\mathbf{L}$ is a rational map from a regular $n$gon to an n-simplex; the map $\mathbf{B}$ is a Bézier simplex. Thus, the composition $\mathbf{S}$ is a map from an $n$gon to an n-sided surface patch (Figure 5.3). The *depth* of an S-patch is defined to be the degree of $\mathbf{B}$. Loop and DeRose give a detailed description of S-patches [LD89].

I begin by noting a few properties of S-patches. First, S-patches are a generalization of both triangular Bézier patches and of tensor product Bézier patches: three sided S-patches are Bézier patches, and tensor product Bézier patches are a special case of four sided S-patches. A second property is that while in general S-patches are rational polynomial surfaces, their boundaries are polynomial curves.

The first subproblem I solve is: Given an S-patch, find a set of cubic interpolant patches that approximate it. The easiest approach is to subdivide the S-patch at the center of its domain and split the n-sided patch into n triangular regions. Each one of these triangular regions can now be approximated with a cubic interpolant patch.

To make such an approximation, we must compute normals and second fundamental forms at the sample points. Since we know the rational polynomial equation for the S-patch, we can compute these values directly. Essentially, $\mathbf{L}$ is a function of two variables, say x and y, and $\mathbf{B}$ is a Bézier simplex. Blossoming $\mathbf{B}$ gives the

following form for $\mathbf{S}$:

$$\begin{aligned}\mathbf{S}(x,y) &= \mathbf{B}(\mathbf{L}(x,y)) \\ &= \mathbf{b}(\mathbf{L}(x,y),\ldots,\mathbf{L}(x,y)).\end{aligned}$$

To compute the first and second fundamental form, we need to take partial derivatives of $\mathbf{S}$. As is shown in Appendix B, the first and second partial derivatives of $\mathbf{S}$ with respect to x are:

$$\begin{aligned}\mathbf{D}_x\mathbf{S}(x,y) &= d\mathbf{b}\left(\mathbf{D}_x\mathbf{L}(x,y),\mathbf{L}(x,y),\ldots,\mathbf{L}(x,y)\right), \\ \mathbf{D}_{x,x}\mathbf{S}(x,y) &= d(d-1)\mathbf{b}\left(\mathbf{D}_x\mathbf{L}(x,y),\mathbf{D}_x\mathbf{L}(x,y),\mathbf{L}(x,y),\ldots,\mathbf{L}(x,y)\right) + \\ &\quad n\mathbf{b}\left(\mathbf{D}_{x,x}\mathbf{L}(x,y),\mathbf{L}(x,y),\ldots,\mathbf{L}(x,y)\right),\end{aligned}$$

where d is the degree of $\mathbf{B}$. Similar formulas give the derivatives with respect to y and the mixed partials. With this information, we can compute the normal and second fundamental form at a point on $\mathbf{S}$.

The boundaries between S-patches pose a slight problem when approximating a network of S-patches. Since the S-patch surface may be only G^1, the second fundamental forms of two S-patches meeting along a boundary do not in general agree. If the curvatures used to construct the boundary curves fail to match, then the approximation will not be C^0. To avoid this problem, I sample the tangents and curvatures directly from the boundary curves of the S-patches. A non-planar geometric Hermite curve is then fit to this data. Note that if the S-patches are of depth no greater than three, then we can use the boundary of the S-patches as the boundary curves for the cubic interpolant patches.

Results

I give two examples of approximating an S-patch surface with the cubic interpolant. Both S-patch surfaces are from [LD90, Loo92]. The initial mesh for the first example appears on the left in Figure 5.4. Shaded images of the S-patch surface, of the cubic interpolant approximation to this surface, and of the refinement map appear in Figure 5.9.

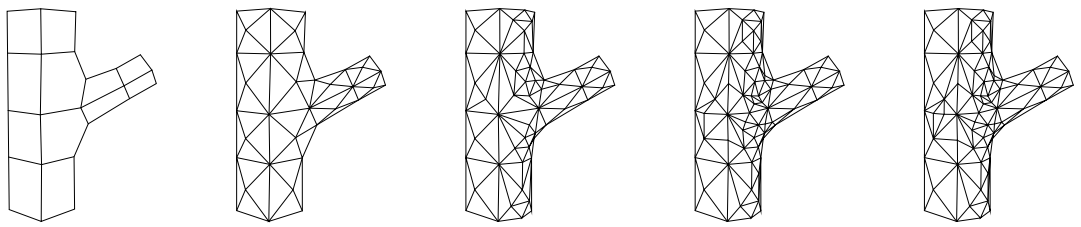


Figure 5.4: The branch mesh and its refinements.

The S-patch surface is comprised of twenty-four S-patches. Twenty of these are four sided patches of depth three, while four are five sided patches of depth six. The cubic interpolant surface was refined three times; the final approximation is ε-G^1 for $\varepsilon = 0.04$ degrees and has 238 patches.

A comparison of the storage requirements appears in Table 5.2. With no sharing of vertices, the S-patch surface requires 1240 vertices. The cubic interpolant requires 2380 vertices for the approximation. If sharing is used, the cubic interpolant storage cost decreases to roughly half (1100 points), while the S-patch scheme decreases to about 1000 points. Also note that in this example the cubic interpolant is being used to approximate polynomial bicubic tensor product patches. As these bicubics are already simple patches, I could use the cubic interpolant to approximate only the five sided patches. In this case, the storage requirements for the cubic interpolant would be less.

The primary advantage to using the cubic interpolant to approximate S-patches is in time required to evaluate a point on the surface. The cost to evaluate S-patches of high degree is relatively expensive compared to the cost of evaluating the cubic interpolant. For example, it is about 42 times as expensive to evaluate a depth six, five sided S-patch than it is to evaluate the cubic interpolant. The cost increases dramatically as the number of sides increases; as seen in the next example, it is 92 times as expensive to evaluate a depth six, six sided S-patch than it is to evaluate the cubic interpolant.

The total cost of evaluating a tessellation of the example S-patch surface is about 4.2 times as expensive as evaluating its cubic interpolant approximation. Note that these costs are conservative: the tessellation of the cubic interpolant approximation

Table 5.2: Approximation of S-patch branch surface.

	S-Patch Branch		Cubic Interpolant
Sides	4	5	3
Depth	3	6	3
Number in surface	20	4	238
Control points per patch	20	210	10
Total control points	1240		2380
Relative Evaluation Cost Per Point	2	42	1
Total Evaluation Cost For Surface	1000		238

has 2.4 times the number of triangles that are in the S-patch tessellation, and the costs are only for evaluating the Bézier simplex; they do not include the cost of evaluating the **L** function.

The second example is of the ring mesh. The initial S-patch mesh appears in Figure 5.5. Shaded images appear in Figure 5.10. A comparison of the costs and storage is given in Table 5.3. Here the evaluation cost for the S-patch surface is about 4.2 times that of the cubic interpolant approximation, while the number of triangles in the cubic interpolant tessellation is about 3 times that of the S-patch surface. The cubic interpolant surface is ε-G^1 for $\varepsilon = 0.052$ degrees. (The number of control points in a depth six, six sided S-patch is unrelated to the number of cubic interpolant patches in the approximation; the fact that these two quantities are equal in this example is a coincidence.)

5.5 Scattered Data

I would like to use the cubic interpolant to construct a surface from scattered data. If the data stored in the mesh has only positional information, then tangent planes and second fundamental forms must be estimated. I tried three estimation techniques: partial estimation, local estimation, and global estimation. The results for all three methods have been disappointing.

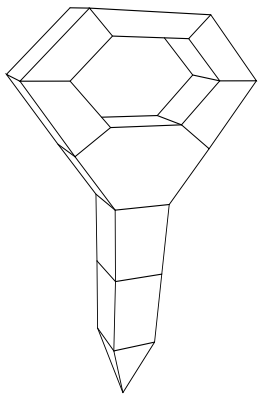


Figure 5.5: Ring mesh.

Table 5.3: Approximation of S-patch ring surface.

	S-patch Ring			Cubic Interpolant
Sides	3	4	6	3
Depth	6	3	6	3
Number in surface	4	31	2	462
Control points per patch	56	20	462	10
Total control points	1768			4620
Relative Evaluation Cost Per Point	5.6	2	92	1
Total Evaluation Cost For Surface	1379			462

5.5.1 Partial Estimation

As an initial attempt at estimating second order data, I assumed that the initial mesh had second order information, and that estimation was needed only at the refinement points. In this setting, the initial cubic interpolant surface is constructed without any estimation. This surface is then refined along boundaries with large tangent plane discontinuities, and estimation is needed at the refinement points.

Three things need to be estimated: the refinement point, the tangent plane, and the second fundamental form. If the de Boor-Höllig-Sabin curve construction succeeded on a boundary, then an accurate estimate for position can be made by sampling the curve. The normals of both patches at this point are then averaged to give the tangent plane. To determine the second fundamental form, the curvature in three directions must be estimated. The boundary curve curvature is sampled for one of these values. Each of the last two normal section curvatures is estimated by sampling the curvature of both patches along the boundary in an arbitrary direction and averaged.

5.5.2 Local Estimation

The local methods of estimation I experimented with proceed in two steps. In the first step, I estimate the normal to the surface at a vertex. In the second step, I estimate the second fundamental form. After refining, the position of each refined vertex is taken from the boundary curve, and the normals and second fundamental forms are re-estimated for the entire surface.

Normal Estimation

Several methods have been proposed for estimating normals. Most involve a weighted sum of the normals to the faces surrounding a vertex. More precisely, if a vertex $\mathbf{V}$ has n neighbors $\mathbf{V}_i$, then the estimated normal is given by

$$\mathbf{N} = \sum_{i=0}^{n-1} \alpha_i \frac{(\mathbf{V}_i - \mathbf{V}) \times (\mathbf{V}_{i+1} - \mathbf{V})}{|(\mathbf{V}_i - \mathbf{V}) \times (\mathbf{V}_{i+1} - \mathbf{V})|},$$

where the α_is are the weights associated with each face, and where $\mathbf{V}_n = \mathbf{V}_0$. Several weighting functions have been suggested, including

1. Unit weights (i.e, $\alpha_i = 1$).

2. Weight by face area.

3. Weight by the reciprocal of the face area.

4. Weight by the angle $\angle \mathbf{V}_i \mathbf{V} \mathbf{V}_{i+1}$.

Another method uses the principal component as the tangent plane. Despite the differences in these normal estimations, none of these methods seems particularly good to use for a surface interpolation scheme.[1]

Todd and McLeod

Todd and McLeod [TM86] estimate second order data as follows: at each point $\mathbf{V}$, select two neighbors of $\mathbf{V}$ and pass a circle through the three points. Circles are constructed for various pairs of $\mathbf{V}$'s neighbors. The curvature of each circle is adjusted, yielding an overconstrained system of normal section curvatures. A least squares fit is used to determine the Dupin indicatrix (an equivalent formulation of the second fundamental form).

Moreton

Moreton [MS92] uses a global optimization method to compute second order information at the vertices of a control net. However, the initial value of the second order information is computed using a local method. At each vertex $\mathbf{V}$ with normal $\hat{\mathbf{N}}$ and neighbors $\mathbf{V}_i$, the initial second fundamental form is estimated as a least squares fit to curvature values estimated in several directions.

The construction is illustrated in Figure 5.6. For each neighbor $\mathbf{V}_l$ of $\mathbf{V}$, the tangent is estimated by projecting $\mathbf{V}_l$ into the tangent plane at $\mathbf{V}$ (call this point $\mathbf{V}_{lp}$); the tangent direction $\hat{\mathbf{t}}_l$ is then given by the unit vector parallel to $\mathbf{V}_{lp} - \mathbf{V}$.

The curvature in direction $\hat{\mathbf{t}}_l$ is estimated by computing the circle that passes through the points $\mathbf{V}$ and $\mathbf{V}_l$, and is perpendicular to $\hat{\mathbf{N}}$ at $\mathbf{V}$. The unsigned curvature

[1] Note that weighting the face normals by the face area yields a normal that depends only on the ring of vertices $\mathbf{V}_i$ and not on $\mathbf{V}$ itself. This method is probably a poor choice for surface reconstruction.

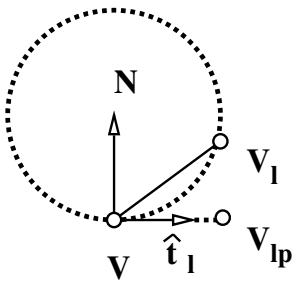


Figure 5.6: Moreton's curvature estimation scheme.

is then estimated to be the reciprocal of this circle's radius. The curvature's sign is given by the sign of $< \hat{\mathbf{N}},\ \mathbf{V}_l - \mathbf{V} >$.

If $\mathbf{V}$ has more than three neighbors, then II will be over constrained. A least squares fit to the curvature values is used to determine II.

Quadric surface

Another approach I tried was to estimate the normal at each vertex $\mathbf{V}$ by finding the principal component of the ring of neighboring vertices. The second fundamental form is then estimated by constructing a quadric surface over the plane defined by the normal. This quadric surface is constrained to pass through $\mathbf{V}$, and a least squares fit to the neighbors of $\mathbf{V}$ is used to set the quadric's remaining coefficients. I use the second fundamental form of the quadric at $\mathbf{V}$ as the estimate for the second order data of $\mathbf{V}$ in the mesh.

5.5.3 Global Estimation

As mentioned above, another approach to estimating the second order data at the vertices of a mesh is to use a global optimization technique. The idea is to start with an initial guess for the second order data. Next, I perform global optimization over the mesh, minimizing a functional. The choices for the variables of the optimization are extensive; however, I am primarily interested in optimizing the normals and second fundamental forms at the vertices of the mesh.

Moreton

Moreton [MS92] uses global optimization to fit biquintic patches to a rectilinear data net. As a first step in this process, a curve network is constructed for the boundaries of the patches. This network minimizes the variation in curvature [MS91], i.e., the functional

$$\int \left(\frac{d\kappa}{ds}\right)^2 ds$$

is minimized over a network of quintic polynomial curves. The curves meeting at a vertex are constrained to agree with a common second fundamental form, resulting in a G^2 consistent curve network.

As a starting point for the optimization routine, Moreton uses the method described in Section 5.5.2. As the optimization process proceeds, the second order data is adjusted until a minimal variation curve network is obtained.

Moreton's method together with the cubic interpolant can be combined to fit a piecewise (bi)cubic to scattered data. I use Moreton's method to construct a curve network interpolating the mesh vertices. The second order data at the vertices of the mesh is taken from the curve network, and a cubic interpolant surface is constructed. The boundaries between two patches with a high tangent plane discontinuity are refined, and the process is repeated (i.e., I run the global optimization method on the new network to determine new values for the second order data and then construct a new cubic interpolant surface). The subdivision point on the boundary is obtained from Moreton's G^2 curve network, avoiding problems caused by the failure of de Boor-Höllig-Sabin.

5.5.4 Results

I implemented all the above methods except for Todd and McLeod and tested them on the torus data set. The quadric surface method gave very poor results. The remaining methods all yielded similar results: The initial surface had a greater discontinuity in the surface normals than the surface constructed for the true data (except for the partial estimation method, which starts with the true data), although the curvature distribution was reasonable.

Boundaries with high normal discontinuities were refined. After refinement, the normal discontinuity decreases, but the curvature distribution across the patch con-

centrates near the boundaries, giving surfaces similar to those constructed by the G^1 schemes. The global scheme gave additional artifacts after refinement: the shape of the surface became worse, with bulges appearing along some of the patch boundaries.

All the local and global schemes I tried were inadequate for fitting surfaces to scattered data. Moreton's local scheme seems to work best, and as expected, all the methods perform better on dense data sets. It should be noted that the variant of Moreton's global scheme I used was designed for building planar curves. The data sets I tested the scheme on caused Moreton's method to build non-planar curves that fail to minimize the functional. A variant that is designed to optimize non-planar curves is expected to preform better.

Figure 5.7: Curvature plots of surfaces fit to $4x^2 + 2y^2 + z^2 - 4 = 0$. The bottom row shows the curvature plot and the surface after four refinements. (The final surface is ε-G^1 for $\varepsilon = 0.4$ degrees.)

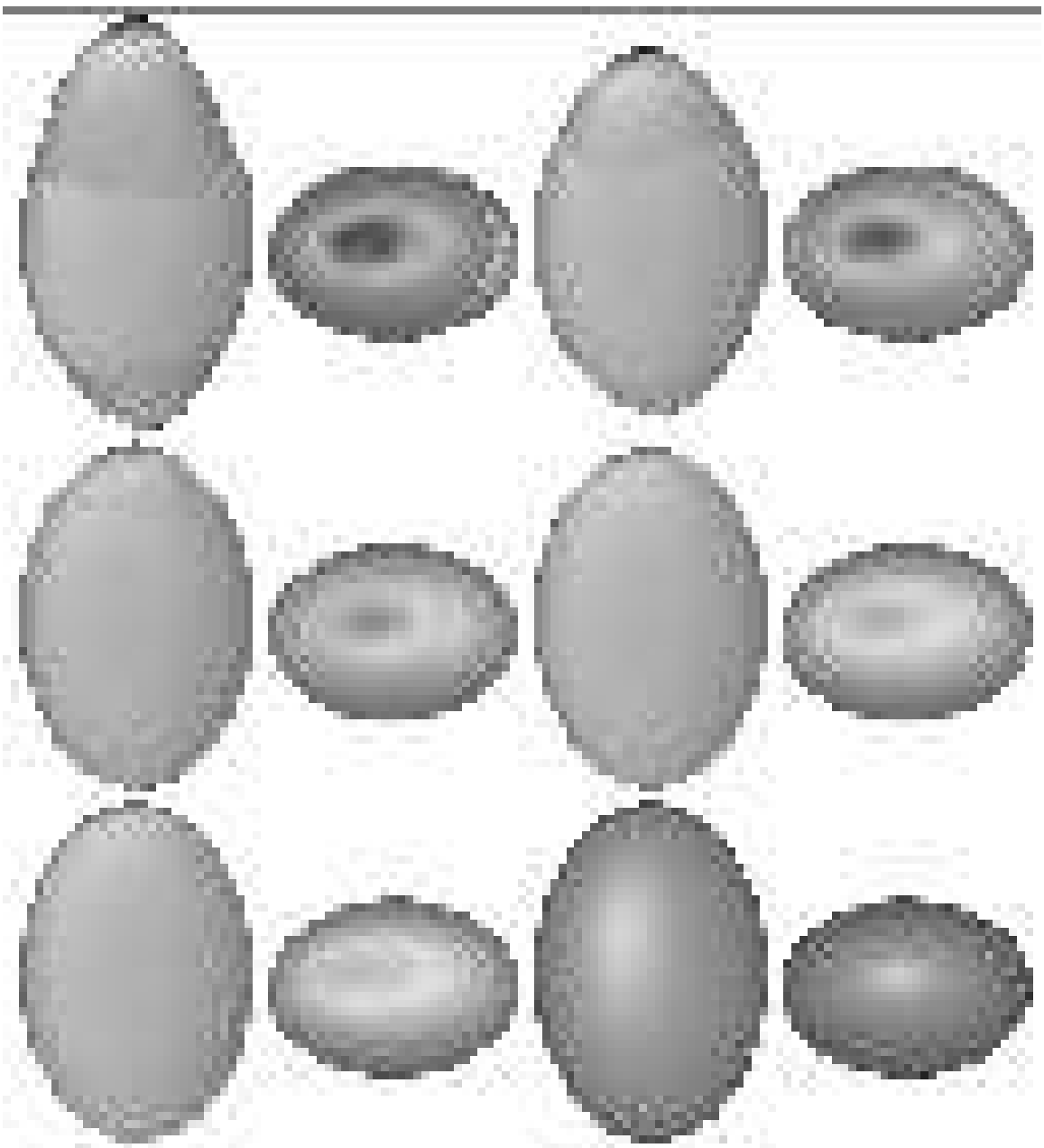


Figure 5.8: An offset to a bicubic.

Left: An offset surface.

Center: The cubic interpolant approximation to this surface. The surface is ε-G^1 for $\varepsilon = 0.72$ degrees.

Right: is a refinement map of the approximate. Table 5.1 gives the mapping of colors to refinement level.

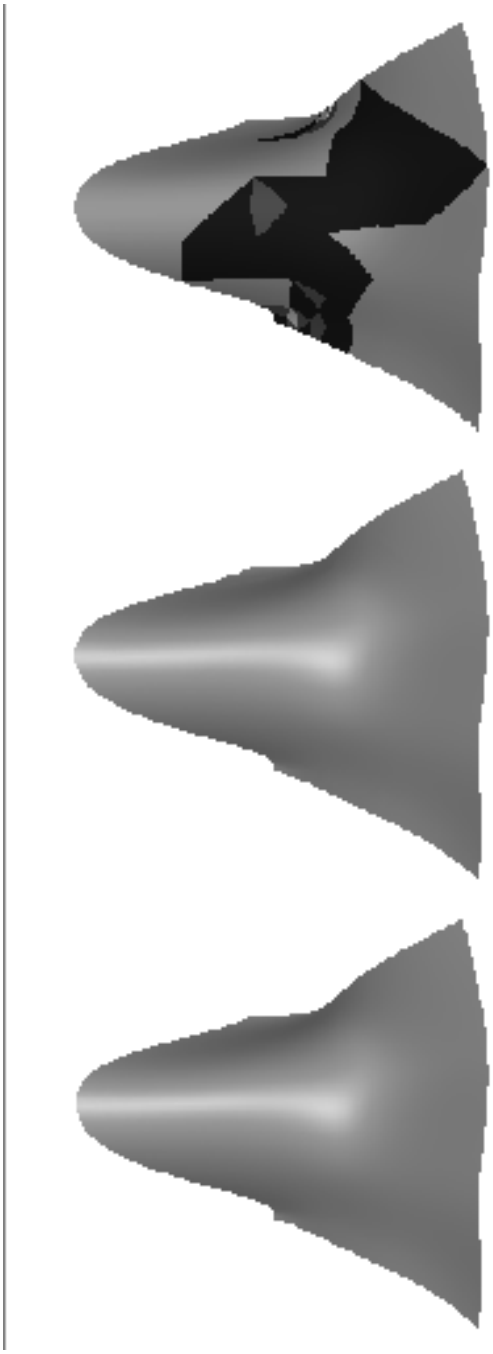


Figure 5.9: The S-patch "branch" surface.

Left: The S-patch surface.

Center: The cubic interpolant approximation to this surface. The surface is ε-G^1 for $\varepsilon = 0.04$ degrees.

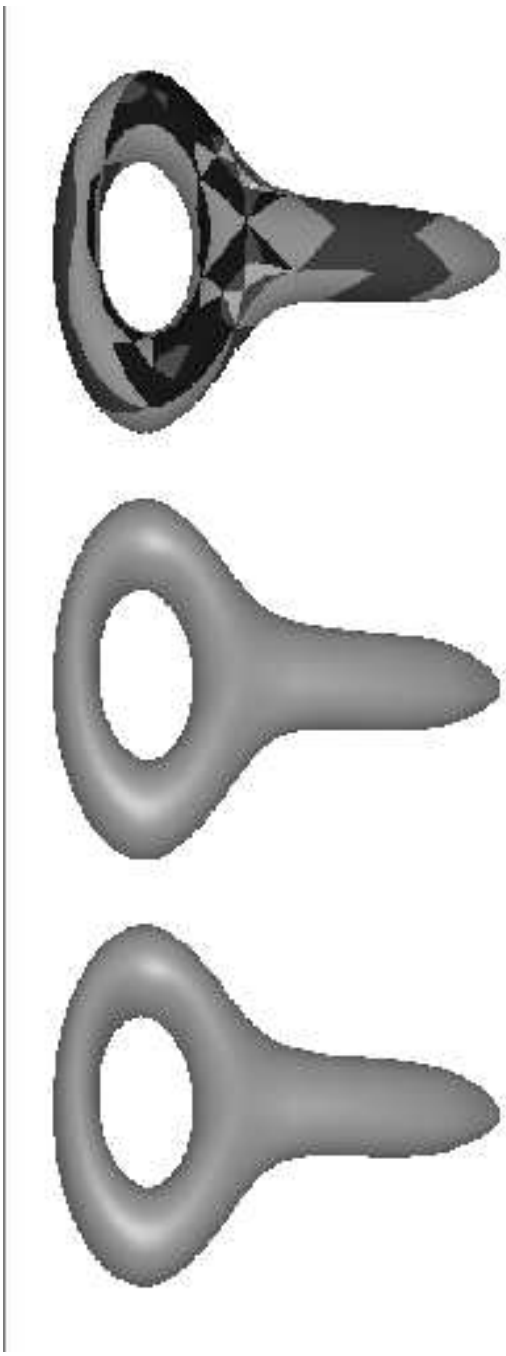
Right: A refinement map of the approximate. Table 5.1 gives the mapping of colors to refinement level.

Figure 5.10: The S-patch "ring" surface.

Left: The S-patch surface.

Center: The cubic interpolant approximation to this surface. The surface is ε-G^1 for $\varepsilon = 0.052$ degrees.

Right: A refinement map of the approximate. Table 5.1 gives the mapping of colors to refinement level.



Chapter 6

CONCLUSIONS AND FUTURE WORK

I surveyed G^1 surface interpolation schemes and found that tangent plane continuity is not sufficient for constructing visually smooth surfaces. Constructing a G^1 surface necessitates the use of a high-degree surface patch. Once the G^1 constraints have been met, a large number of degrees of freedom remain unset. Poor settings of these degrees of freedom result in the poor surface shape.

I then introduced two high-order-of-approximation surfaces patches, the cubic interpolant and the bicubic interpolant. These patches interpolate the input mesh to second order at the vertices and can be used to construct ε-G^1 surfaces.

We have seen cubic interpolant approximations to several types of surfaces, including offset surfaces, algebraic surfaces, and S-patch surfaces. Although the approximations are only ε-G^1, it is visually difficult to see differences between these approximations and the original surfaces.

Both theoretical and empirical aspects of approximate continuity deserve further investigation. In particular, more accurate methods of determining the discontinuity between two surface patches should be devised. To improve practical applications of approximate continuity, experiments should be made to determine the magnitude of smallest discontinuity perceivable to a viewer, recognizing that this angle may also be a function of the material property of the surface.

The current technique to construct an ε-G^1 surface is to build the surface, determine if it meets the ε-G^1 requirement, and refine the surface in regions where it does not. An alternative approach to try is to build a surface that meets the continuity bound by construction. It should be cautioned that such a construction will need to satisfy a cycle of constraints around each vertex, perhaps creating a global problem or a problem similar to the vertex consistency problem.

The use of the cubic interpolant for surface construction can be improved, however. When used to approximate surfaces, better refinement criteria should be developed. In particular, we want to know "how good the approximation is", in terms

of the maximum distance from the approximate and the true surface. Not only is this a difficult theoretical problem, I have encountered surfaces where refining along boundaries with high normal discontinuity yields an inadequate approximation.

The results for scattered data interpolation are inadequate. A better means of estimating the second fundamental form needs to be determined. Global techniques should be investigated further. A first step is to improve Moreton's optimization technique to optimize non-planar boundaries. Another approach is to use a global surface interpolation scheme to construct a surface (for example, Moreton's scheme), and then approximate this surface with the cubic interpolant. It is hoped, however, that a computationally less expensive method can be devised.

The current cubic interpolant scheme can be improved by finding a better refinement technique. The current patterns for subdividing patches (Figure 5.1) lead to the creation of long, skinny triangles. Triangles with better aspect ratios should allow for an approximation with fewer surface elements. A refinement technique such as the one in [BMSW91] should be investigated.

Further studies of the G^1 schemes should also be made. In particular, one deficiency of tangent plane continuity is that it is a differential property and only guarantees smoothness in a differential neighborhood near the boundary. Additional requirements are needed to ensure that the entire patch appears smooth. Moreover, it is still unclear why all the G^1 schemes exhibit similar shape defects, and in particular, why they usually construct surface patches with curvature concentration near the patch boundaries.

BIBLIOGRAPHY

[Bal74] A. A. Ball. CONSURF I, introduction of the conic lofting. *cad*, 6:243–249, 1974.

[Bar83] R. Barnhill. Computer aided surface representation and design. In R. Barnhill and W. Boehm, editors, *Surfaces in computer aided geometric design*, pages 1–24. North-Holland, 1983.

[BMSW91] D. Baum, S. Mann, K. Smith, and J. Winget. Making radiosity usable. In *SIGGRAPH 1991*, pages 51–60, July 1991.

[Boe88] W. Boehm. Visual continuity. *Computer Aided Design*, 20(6):307–311, 1988.

[CGM91] E.-W. Chionh, R. Goldman, and J. Miller. Using multivariate resultants to find the intersection of three quadric surfaces. *TOG*, 10(4):378–400, October 1991.

[Chi86] H. Chiyokura. Localized surface interpolation method for irregular meshes. In Tosiyasu L. Kunii, editor, *Advanced Computer Graphics*, pages 3–19. Springer-Verlag, 1986.

[CK83] H. Chiyokura and F. Kimura. Design of solids with free-form surfaces. *Computer Graphics*, 17(3):289–298, 1983.

[CM89] A. Cavaretta and C. Micchelli. The design of curves and surfaces by subdivision algorithms. In T. Lyche and L. Schumaker, editors, *Mathematical Methods in Computer Aided Geometric Design*, pages 115–153. Academic Press, 1989.

[CT65] R. Clough and J. Tocher. Finite element siffness matrices for analysis of plates in bending. In *Proceedings of Conference on Matrix Methods in Structural Analysis*, 1965.

[dBHS87] C. de Boor, K. Höllig, and M. Sabin. High accuracy geometric Hermite interpolation. *CAGD*, 4(4):269–278, December 1987.

[DLG90] N. Dyn, D. Levin, and J. Gregory. A butterfly subdivision scheme for surface interpolation with tension control. *ACM Transactions on Graphics*, 9(2):160–169, April 1990.

[Far82] G. Farin. A construction for visual C^1 continuity of polynomial surface patches. *Computer Graphics and Image Processing*, 20:272–282, 1982.

[Far83] G. Farin. Smooth interpolation to scattered 3D data. In R. Barnhill and W. Boehm, editors, *Surfaces in Computer Aided Geometric Design*, pages 43–63. North-Holland, 1983.

[Far85] G. Farin. A modified Clough-Tocher interpolant. *CAGD*, 2(1-3):19–27, September 1985.

[Far90] G. Farin. *Curves and Surfaces for Computer Aided Geometric Design*. Academic Press, San Diego, second edition, 1990.

[FN90] R. Franke and G. Nielson. Scattered data interpolation: A tutorial and survey. In Hans Hagen, editor, *Geometric Modeling: Methods and Applications*. Springer-Verlag, to appear 1990.

[Fra82] R. Franke. Scattered data interpolation: tests of some methods. *Mathematics of Computation*, 38(157):181–200, January 1982.

[GC80] J. Gregory and P. Charrot. A C^1 triangular interpolation patch for computer-aided geometric design. *Computer Graphics and Image Processing*, 13:80–87, 1980.

[Gre74] J. Gregory. Smooth interpolation without twist constraints. In R. Barnhill and R. Riesenfeld, editors, *Computer Aided Geometric Design*, pages 71–87. Academic Press, 1974.

[Gre86] J. Gregory. N-sided surface patches. In J. Gregory, editor, *The Mathematics of Surfaces*, pages 217–232. Clarendon Press, 1986.

[Han91] D. Hansford. *Boundary Curves with Quadric Precision for a Tangent Continuous Scattered Data Interpolant.* PhD thesis, Arizona State University, 1991.

[HDD+92] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Surface reconstruction from unorganized points. In *SIGGRAPH 1992*, pages 71–79, July 1992.

[Her85] G. Herron. Smooth closed surfaces with discrete triangular interpolants. *CAGD*, 2(4):297–306, December 1985.

[Her87] G. Herron. Techniques for visual continuity. In G. Farin, editor, *Geometric Modeling: Algorithms and New Trends*, pages 163–174. SIAM, 1987.

[HFN91] B. Hamann, G. Farin, and G. M. Nielson. A parametric triangular patch based on generalized conics. In G. Farin, editor, *NURBS for Curve and Surface Design*, chapter 6, pages 75–85. SIAM, 1991.

[HHS+92] H. Hagen, S. Hahmann, T. Schreiber, Y. Nakajima, B. Wördenweber, and P. Hollemann-Grundstedt. Surface interrogation algorithms. *IEEE*, 12(5):53–60, September 1992.

[HP89] H. Hagen and H. Pottmann. Curvature continouse triangular interpolants. In T. Lyche and L. L. Schumaker, editors, *Mathematical Methods in Computer Aided Gometric Design*, pages 373–384. Academic Press, 1989.

[Jen87] T. Jensen. Assembling triangular and rectangular patches and multivariate splines. In G. Farin, editor, *Geometric Modeling: Algorithms and New Trends*, pages 203–220. SIAM, 1987.

[LD89] C. Loop and T. DeRose. A multisided generalization of Bézier surfaces. *TOG*, 8(3):204–234, July 1989.

[LD90] C. Loop and T. DeRose. Generalized B-splines of arbitrary topology. In *SIGGRAPH 1990*, pages 347–356, 1990.

[LML+90] M. Lounsbery, S. Mann, C. Loop, D. Meyers, J. Painter, T. DeRose, and K. Sloan. A testbed for the comparison of parametric surface methods. In *SPIE/SPSE Symposium on Electronic Imaging Science and Technology, Santa Clara, CA.*, February 1990.

[Lon86] L. Longhi. Interpolating patches between cubic boundaries. Technical Report T.R. UCB/CSD 87/313, University of California, Berkeley, Berkeley, CA 94720, October 1986.

[Loo92] C. Loop. *Generalized B-spline Surfaces of Arbitrary Topological Type.* PhD thesis, Univerity of Washington, Seattle, WA, 1992.

[Man68] J. Mann. *Nathanial West.* PhD thesis, University of Missouri, Columbia, Columbia, MO., 1968.

[Man69] K. Mann. *EPR Studies of Impurity Ions and F^+-centers in Single Crystals of Barium Oxide.* PhD thesis, University of Missouri, Columbia, Columbia, MO., August 1969.

[MS91] H. Moreton and C. Séquin. Surface design with minimum energy networks. In *ACM/SIGGRAPH Symposium on Solid Modeling Foundations and CAD/CAM Applications*, Austin, TX, June 1991.

[MS92] H. Moreton and C. Séquin. Functional optimization for fair surface design. In *SIGGRAPH 1992*, July 1992.

[Nie87] G. M. Nielson. A transfinite, visually continuous, triangular interpolant. In G. Farin, editor, *Geometric Modeling: Algorithms and New Trends*, pages 235–246. SIAM, 1987.

[O'N66] B. O'Neill. *Elementary Differential Geometry.* Academic Press, 1966.

[Pet89] J. Peters. Local generalized Hermite interpolation by quartic C^2 space curves. *TOG*, 8:235–242, 1989.

[Pet91a] J. Peters. Local cubic and bicubic C^1 surface interpolation with linearly varying boundary normal. *CAGD*, 7(6):499–516, 1991.

[Pet91b] J. Peters. Smooth interpolation of a mesh of curves. *Constructive Approximation*, 7:221–247, 1991.

[Pet91c] J. Peters. Smooth mesh interpolation with cubic patches. *CAD*, 22(2):109–120, 1991.

[PFTV88] W. Press, B. Flannery, S. Teukolsky, and W. Vetterling. *Numerical Recipes in C.* Cambridge University Press, 1988.

[Pip87] B. Piper. Visually smooth interpolation with triangular Bézier patches. In G. Farin, editor, *Geometric Modeling: Algorithms and New Trends*, pages 221–233. SIAM, 1987.

[Poe84] T. Poeschl. Detecting surface irregularities using isophotes. *Computer Aided Geometric Design*, 1(2):163–168, 1984.

[Pre75] P. Prenter. *Splines and Variational Methods.* John Wiley & Sons, 1975.

[Ram88] L. Ramshaw. Béziers and B-splines as multiaffine maps. In R. Earnshaw, editor, *Theoretical Foundations of Computer Graphics and CAD*, pages 757–776. Springer Verlag, 1988.

[SD] J. Schweitzer and T. DeRose. Applying blossoming to algebraic curves and surfaces. In preparation.

[SS87] L. A. Shirman and C. H. Séquin. Local surface interpolation with Bézier patches. *CAGD*, 4(4):279–295, December 1987.

[SS91] L. A. Shirman and C. H. Séquin. Local surface interpolation with Bézier patches: errata and improvements. *CAGD*, 8(3):217–221, August 1991.

[TM86] P. H. Todd and R. J. Y. McLeod. Numerical estimation of the curvature of surfaces. *CAD*, 18(1):33–37, January/February 1986.

[vW86] J. J. van Wijk. Bicubic patches for approximating non-rectangular control-point meshes. *CAGD*, 3(1):1–13, May 1986.

[Zlá68] M. Zlámal. On the finite element method. *Numerische Mathematik*, 12:394–409, 1968.

Appendix A

HERRON'S METHOD

This appendix gives Herron's setting of the control point $\mathbf{C}_i$ (Figure 2.9). We denote the tangent from point $\mathbf{V}_i$ to $\mathbf{V}_j$ as $\vec{\mathbf{T}}_{ij}$. In the construction of patch $\mathbf{F}_i$, Herron implicitly sets $\mathbf{C}_i$ to be:

$$\begin{aligned}\mathbf{C}_i &= \tfrac{1}{12}\Big\{-6\mathbf{V}_j - 2\vec{\mathbf{T}}_{ji} + \vec{\mathbf{T}}_{ik} - \vec{\mathbf{T}}_{ij} + \tfrac{9}{4}\Big[\vec{\mathbf{R}}(\tfrac{1}{3}) - \mathcal{Q}(\tfrac{1}{3})\Big] \\ &\quad + \tfrac{9}{4}\Big[\vec{\mathbf{R}}(\tfrac{2}{3}) - \mathcal{Q}(\tfrac{2}{3})\Big]\Big\} + \mathbf{E}^4_{i2},\end{aligned}$$

where

$$\mathbf{E}^4_{i2} = \frac{1}{2}\mathbf{V}_j + \frac{1}{6}\vec{\mathbf{T}}_{ji} + \frac{1}{2}\mathbf{V}_i + \frac{1}{6}\vec{\mathbf{T}}_{ij},$$

and $\vec{\mathbf{R}}(t) = k(t) \cdot \vec{\mathbf{C}}(t)$ (from Section 2.2.3) and

$$\begin{aligned}\mathcal{Q}(t) &= 6t(1-t)p_j(t) \cdot \mathbf{V}_j + 6t(1-t)p_k(t) \cdot \mathbf{V}_k \\ &\quad + [(1-t)^2 p_k(t) + 2t(1-t)p_j(t)] \cdot \vec{\mathbf{T}}_{jk} \\ &\quad + [t^2 p_j(t) + 2t(1-t)p_k(t)] \cdot \vec{\mathbf{T}}_{kj} \\ &\quad + (1-t)^2 \vec{\mathbf{T}}_{ji} + t^2 \vec{\mathbf{T}}_{ki}.\end{aligned}$$

Here $p_j(t)$ and $p_k(t)$ are the scalar functions

$$p_j(t) = t(r_{ji} + r_{ki} + 1) - r_{ji} - 1$$

and

$$p_k(t) = (1-t)(r_{ji} + r_{ki} + 1) - r_{ki} - 1,$$

where r_{ji} and r_{ki} are:

$$r_{ji} = \frac{< \vec{\mathbf{T}}_{jk},\ \vec{\mathbf{T}}_{ji} >}{< \vec{\mathbf{T}}_{jk},\ \vec{\mathbf{T}}_{jk} >} \quad \text{and} \quad r_{ki} = \frac{< \vec{\mathbf{T}}_{kj},\ \vec{\mathbf{T}}_{ki} >}{< \vec{\mathbf{T}}_{kj},\ \vec{\mathbf{T}}_{kj} >}.$$

Appendix B

S-PATCH DERIVATIVES

For $n \geq 3$, let $\mathbf{S} = \mathbf{B} \circ \mathbf{L}$ be defined as in [LD89]; i.e., $\mathbf{B}$ is a degree d Bézier $(n-1)$-simplex, and $\mathbf{L}$ is a map from a regular $n$gon with domain $(\mathbf{p}_1, \ldots, \mathbf{p}_n)$ defined as:

$$\mathbf{L} = (l_1, \ldots, l_n)$$

where

$$l_i(\mathbf{p}) = \frac{\pi_i(\mathbf{p})}{\pi_1(\mathbf{p}) + \ldots + \pi_n(\mathbf{p})}$$

and where

$$\pi_i(\mathbf{p}) = \alpha_1(\mathbf{p}) \ldots \alpha_{i-2}(\mathbf{p}) \alpha_{i+1}(\mathbf{p}) \ldots \alpha_n(\mathbf{p}).$$

Here, α_is is the ratio of the signed area of the triangle $\mathbf{p}\mathbf{p}_i\mathbf{p}_{i+1}$ to the area of the triangle $\mathbf{p}_i\mathbf{p}_{i+1}\mathbf{p}_{i+2}$. The αs can be written as

$$\alpha_i(\mathbf{p}) = x(\mathbf{p})(y(\mathbf{p}_i) - y(\mathbf{p}_{i+1})) + y(\mathbf{p})(x(\mathbf{p}_{i+1}) - x(\mathbf{p}_i)) + c.$$

where x and y are the coordinate functions, and c is a constant not needed for the derivative calculation.

B.1 First Derivatives

To compute the normal, we need to calculate the partials of l with respect to x and y. I give the derivation for x. By the quotient rule,

$$D_x l_i(\mathbf{p}) = \frac{D_x \pi_i(\mathbf{p}) \Pi(\mathbf{p}) - \pi_i(\mathbf{p}) D_x \Pi(\mathbf{p})}{\Pi(\mathbf{p})^2},$$

where $\Pi = \sum \pi_i$.

From the definition of π we see

$$D_x \pi_i(\mathbf{p}) = \sum_{j \notin \{i-1, i\}} \left((y(\mathbf{p}_j) - y(\mathbf{p}_{j+1})) \prod_{k \notin \{i-1, i, j\}} \pi_k(\mathbf{p}) \right)$$

and

$$D_x\Pi(\mathbf{p}) = \sum D_x\pi_i(\mathbf{p}).$$

I give the partial of $\mathbf{S}$ with respect to x in terms of the blossom of $\mathbf{S}$ by applying the chain rule to $\mathbf{B}(\mathbf{L}(\mathbf{p}))$:

$$\begin{aligned}\mathbf{D}_x\mathbf{S}(\mathbf{p}) &= \mathbf{D}_x\mathbf{B}(\mathbf{L}(\mathbf{p})) \\ &= d\mathbf{b}(\mathbf{D}_x\mathbf{L}(\mathbf{p}), \mathbf{L}(\mathbf{p}), \mathbf{L}(\mathbf{p}), \ldots, \mathbf{L}(\mathbf{p})).\end{aligned}$$

B.2 Second Derivatives

The second derivative of l with respect to x is

$$D^2_{x,x}l_i(\mathbf{p}) = \frac{2\pi_i(\mathbf{p})D_x\Pi(\mathbf{p})^2}{\Pi(\mathbf{p})^3} - \frac{2D_x\pi_i(\mathbf{p})D_x\Pi(\mathbf{p})}{\Pi(\mathbf{p})^2} - \frac{\pi_i(\mathbf{p})D^2_{x,x}\Pi(\mathbf{p})}{\Pi(\mathbf{p})^2} + \frac{D^2_{x,x}\pi_i(\mathbf{p})}{\Pi(\mathbf{p})}.$$

The second derivatives of π and Π are similar to the first derivatives:

$$D^2_{x,x}\pi_i(\mathbf{p}) = \sum_{j\notin\{i-1,i\}} \sum_{k\notin\{i-1,i,j\}} \left((y(\mathbf{p}_j) - y(\mathbf{p}_{j+1}))(y(\mathbf{p}_k) - y(\mathbf{p}_{k+1})) \prod_{\ell\notin\{i-1,i,j,k\}} \pi_\ell(\mathbf{p}) \right)$$

and

$$D^2_{x,x}\Pi(\mathbf{p}) = \sum D^2_{x,x}\pi_i(\mathbf{p}).$$

The mixed partial is similar, though more complicated:

$$\begin{aligned}D^2_{x,y}l_i(\mathbf{p}) &= \frac{2\pi_i(\mathbf{p})D_y\Pi(\mathbf{p})D_x\Pi(\mathbf{p})}{\Pi(\mathbf{p})^3} - \frac{D_y\Pi(\mathbf{p})D_x\pi_i(\mathbf{p})}{\Pi(\mathbf{p})^2} - \frac{D_y\pi_i(\mathbf{p})D_x\Pi(\mathbf{p})}{\Pi(\mathbf{p})^2} - \\ &\quad \frac{\pi_i(\mathbf{p})D^2\Pi_{x,y}(\mathbf{p})}{\Pi(\mathbf{p})^2} + \frac{D^2_{x,y}\pi_i(\mathbf{p})}{\Pi(\mathbf{p})}\end{aligned}$$

with

$$D^2_{x,y}\pi_i(\mathbf{p}) = \sum_{j\notin\{i-1,i\}} \sum_{k\notin\{i-1,i,j\}} \left((y(\mathbf{p}_j) - y(\mathbf{p}_{j+1}))(x(\mathbf{p}_{k+1}) - x(\mathbf{p}_k)) \prod_{\ell\notin\{i-1,i,j,k\}} \pi_\ell(\mathbf{p}) \right)$$

We can now define the second derivatives of $\mathbf{S}$. Differentiating the first derivative and blossoming gives us

$$\begin{aligned}\mathbf{D}_{x,z}\mathbf{S}(\mathbf{L}(\mathbf{p})) &= d(d-1)\mathbf{b}(\mathbf{D}_x\mathbf{L}(\mathbf{p}), \mathbf{D}_z\mathbf{L}(\mathbf{p}), \mathbf{L}(\mathbf{p}), \ldots, \mathbf{L}(\mathbf{p})) + \\ &\quad d\mathbf{b}(\mathbf{D}^2_{x,z}\mathbf{L}(\mathbf{p}), \mathbf{L}(\mathbf{p}), \ldots, \mathbf{L}(\mathbf{p})),\end{aligned}$$

where "z" may be either x or y, and d is degree of $\mathbf{B}$. Note that for a three sided S-patch $\mathbf{D}^2_{x,z}\mathbf{L}(\mathbf{p})$ is the zero vector. As a blossom evaluated on the zero vector gives the zero vector, the expression for the second derivative of a three sided S-patch simplifies to

$$\mathbf{D}_{x,z}\mathbf{S}(\mathbf{L}(\mathbf{p})) = d(d-1)\mathbf{b}(\mathbf{D}_x\mathbf{L}(\mathbf{p}), \mathbf{D}_z\mathbf{L}(\mathbf{p}), \mathbf{L}(\mathbf{p}), \ldots, \mathbf{L}(\mathbf{p})),$$

as we expect from the theory of Bézier surfaces.

B.3 Code

This section gives the code for computing $\mathbf{L}$, $\mathbf{DL}$, and $\mathbf{D}^2\mathbf{L}$. The routine for computing $\mathbf{DL}$ is based on a routine written by Charles Loop. The routines assume that a domain polygon has been computed and stored in `dp[][2]`. The point of evaluation is passed to the routines in `bb[2]`.

```
/*
 *----------------------------------------------------------------------
 *  File:  lmap.c
 *        Compute the 'L' function of Loop-DeRose, TOG 1989.
 *----------------------------------------------------------------------
 */

#include <stdio.h>

#define SQ(A) ((A)*(A))
#define CUBE(A) ((A)*(A)*(A))
#define MAX_SIDES 100

static double a[MAX_SIDES];
static double pi[MAX_SIDES];
static double sum_pi;
static double sum_px;
static double sum_py;
static double px[MAX_SIDES];
static double py[MAX_SIDES];

static double determ(double m[3][3])
{
    return m[0][0]*(m[1][1]*m[2][2] - m[2][1]*m[1][2]) -
           m[0][1]*(m[1][0]*m[2][2] - m[2][0]*m[1][2]) +
           m[0][2]*(m[1][0]*m[2][1] - m[2][0]*m[1][1]);
}

static double alpha(double dp[][2], double p[2], int i, int s)
{
    int j;
    double m[3][3];
    double d;

    m[0][0] = p[0];  m[0][1] = dp[i][0];  m[0][2] = dp[(i+1)%s][0];
    m[1][0] = p[1];  m[1][1] = dp[i][1];  m[1][2] = dp[(i+1)%s][1];
    m[2][0] = 1.;    m[2][1] = 1.;        m[2][2] = 1.;

    d = determ(m);
    return d;
}
```

```
void computeL(double dp[][2], int sides, double b[], double l[])
{
    int i;

    if ( sides > MAX_SIDES ) {
        fprintf(stderr, "ComputeL: sides %d is too many.\n");
        exit(1);
    }

    for (i=0; i<sides; i++) {
        a[i] = alpha(dp, b, i, sides);
    }

    sum_pi = 0.;
    for (i=0; i<sides; i++) {
        int j;

        pi[i] = 1.;
        for (j=0; j<sides; j++) {
            if ( j != i  &&  (j+1)%sides != i ) {
                pi[i] *= a[j];
            }
        }
        sum_pi += pi[i];
    }

    for (i=0; i<sides; i++) {
        l[i] = pi[i]/sum_pi;
    }
}
```

```
void computeLDL(double dp[][2], int sides, double b[],
                double l[], double Dxl[], double Dyl[])
{
    int i;

    computeL(dp, sides, b, l);

    sum_px = 0;  sum_py = 0;
    for (i=0; i<sides; i++) {
        int j;

        px[i] = 0;
        py[i] = 0;
        for (j=1; j<sides-1; j++) {
            int k;
            double tmp1=dp[(i+j)%sides][1]-dp[(i+j+1)%sides][1];
            double tmp2=dp[(i+j+1)%sides][0]-dp[(i+j)%sides][0];

            for (k=1; k<sides-1; k++) {
                if ( j==k ) continue;
                tmp1 *= a[(i+k)%sides];
                tmp2 *= a[(i+k)%sides];
            }
            px[i] += tmp1;
            py[i] += tmp2;
        }
        sum_px += px[i];
        sum_py += py[i];
    }
    for (i=0; i<sides; i++) {
        Dxl[i] = px[i]/sum_pi - (pi[i]*sum_px)/(sum_pi*sum_pi);
        Dyl[i] = py[i]/sum_pi - (pi[i]*sum_py)/(sum_pi*sum_pi);
    }
}
```

```
void computeLDLDDL(double dp[][2], int sides, double b[],
                   double l[],
                   double Dxl[], double Dyl[],
                   double Dxxl[], double Dxyl[], double Dyyl[])
{
    double pxx[MAX_SIDES];
    double pxy[MAX_SIDES];
    double pyy[MAX_SIDES];
    double sum_pxx;
    double sum_pyy;
    double sum_pxy;
    int i;

    computeLDL(dp, sides, b, l, Dxl, Dyl);

    sum_pxx = sum_pyy = sum_pxy = 0;
    for (i=0; i<sides; i++) {
        int j;

        pxx[i] = 0;  pxy[i] = 0;  pyy[i] = 0;
        for (j=1; j<sides-1; j++) {
            int k;

            for (k=1; k<sides-1; k++) {
                int m;
                double tmpxx;
                double tmpyy;
                double tmpxy;

                if ( j==k ) continue;

                tmpxx  = dp[(i+k)%sides][1] - dp[(i+k+1)%sides][1];
                tmpxx *= dp[(i+j)%sides][1] - dp[(i+j+1)%sides][1];

                tmpyy  = dp[(i+k+1)%sides][0] - dp[(i+k)%sides][0];
                tmpyy *= dp[(i+j+1)%sides][0] - dp[(i+j)%sides][0];

                tmpxy  = dp[(i+k)%sides][1] - dp[(i+k+1)%sides][1];
                tmpxy *= dp[(i+j+1)%sides][0] - dp[(i+j)%sides][0];

                for (m=1; m<sides-1; m++) {
                    if ( m == k || m == j ) {
                        continue;
                    } else {
```

```
                    tmpxx *= a[(i+m)%sides];
                    tmpyy *= a[(i+m)%sides];
                    tmpxy *= a[(i+m)%sides];
                }
            }
            pxx[i] += tmpxx;
            pyy[i] += tmpyy;
            pxy[i] += tmpxy;
        }
    }
    sum_pxx += pxx[i];
    sum_pyy += pyy[i];
    sum_pxy += pxy[i];
}
for (i=0; i<sides; i++) {
    Dxxl[i] = 2*pi[i]*SQ(sum_px)/CUBE(sum_pi) -
      2*sum_px*px[i]/SQ(sum_pi) -
        pi[i]*sum_pxx/SQ(sum_pi) +
          pxx[i]/sum_pi;
    Dyyl[i] = 2*pi[i]*SQ(sum_py)/CUBE(sum_pi) -
      2*sum_py*py[i]/SQ(sum_pi) -
        pi[i]*sum_pyy/SQ(sum_pi) +
          pyy[i]/sum_pi;
    Dxyl[i] = 2*pi[i]*sum_py*sum_px/CUBE(sum_pi) -
      sum_py*px[i]/SQ(sum_pi) -
        py[i]*sum_px/SQ(sum_pi) -
          pi[i]*sum_pxy/SQ(sum_pi) +
        pxy[i]/sum_pi;
}
}
```

VITA

Stephen Mann was born on July 24th, 1963 in Columbia, Missouri. In 1986 he received a B.A. degree from the University of California at Berkeley majoring in Computer Science and Mathematics. In 1988 he received a M.S. degree in computer science at the University of Washington, and completed his Ph.D. degree at the University of Washington in 1992.

List of publications:

- "Functional Composition via Blossoming," to appear in ACM Transactions on Graphics (with T. DeRose, R. Goldman, and H. Hagen).

- "An Approximately G^1 Cubic Surface Interpolant," in *Mathematical Methods in Computer Aided Geometric Design II*, T. Lyche and L. Schumaker (eds), Academic Press, 1992, (with T. DeRose).

- "A Survey of Triangular Scattered Data Fitting," in *Curve and Surface Design*, H. Hagen (ed), SIAM, 1992 (with M. Lounsbery, C. Loop, D. Meyers, J. Painter, T. DeRose and K. Sloan).

- "An Overview of Parametric Scattered Data Fitting," in CG&A, September 1992 (with M. Lounsbery, T. DeRose).

- "Making Radiosity Usable", SIGGRAPH 1991 (with D. Baum, K. Smith, and J. Winget).

- "A Testbed for the Comparison of Parametric Surface Methods," in the proceedings of *SPIE/SPSE Symposium on Electronic Imaging Science and Technology,* Santa Clara, CA, February, 1990 (with M. Lounsbery, C. Loop, D. Meyers, J. Painter, T. DeRose and K. Sloan).