Modeling Arbitrary Smooth Objects with Algebraic Surfaces

Baining Guo
Ph.D Thesis

91-1226
August 1991

Department of Computer Science
Cornell University
Ithaca, NY 14853-7501

MODELING ARBITRARY SMOOTH OBJECTS WITH ALGEBRAIC SURFACES

A Dissertation
Presented to the Faculty of the Graduate School
of Cornell University
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy

by
Baining Guo
August 1991

MODELING ARBITRARY SMOOTH OBJECTS WITH ALGEBRAIC SURFACES

Baining Guo, Ph.D.

Cornell University 1991

Representing, manipulating, and reasoning about physical objects is a central area of research in a wide range of applications, including solid modeling, computer-aided design, computer graphics, and robotics. In these applications, complex physical objects are modeled using *free-form surfaces*, surfaces that are smooth but otherwise arbitrary. Traditionally, free-form surfaces are constructed from parametric patches, which are very successful as far as design and rendering are concerned. However, manipulating and reasoning about physical objects with parametric patches poses fundamental difficulties. For example, the difficulty of evaluating and representing the intersection of parametric patches has hindered the development of solid modeling systems based on parametric patches alone.

This thesis addresses the problem of constructing free-form surfaces using low-degree implicit patches, which facilitate manipulating and reasoning about physical objects. We establish the fact that quadric patches and cubic patches are flexible enough for free-form surface constructions: (i) given an arbitrary polyhedron, we show how to fit a smooth piecewise quadric surface through the vertices of the

polyhedron without splitting its facets; (ii) given an arbitrary polyhedron with arbitrarily prescribed tangent planes at its vertices, we present an algorithm that fits a smooth piecewise cubic surface through the vertices of the polyhedron so that the surface is tangent to the prescribed tangent plane at each vertex. In constructing free-form surfaces, we also study three related issues. First, we examine the power and limitations of blending techniques. We show that on the one hand, blending techniques can be used to improve the efficiency of free-form surface constructions; on the other, blending techniques have their fundamental limitations. Second, we provide a complete understanding of the *weight functions* for quadric surfaces meeting with geometric continuity. Finally, we tackle some shape control issues of implicit patches. In particular, we demonstrate how to achieve the convexity of a quadric or cubic patch by manipulating its control points.

Biographical Sketch

Born March 1, 1962 in Suining, Sichuan Province, P.R.China, Baining Guo entered Beijing University in Beijing, China in 1978. In 1982, he graduated with bachelor's degree in mathematics. From 1982 to 1985, he worked for the Institute of Applied Physics and Computational Mathematics in Beijing on various scientific computation problems. In the fall of 1985, he became a graduate student in astrophysics at the graduate school of the Chinese Academy of Sciences (Academia Sinica). In the following year, he came to Cornell University as a Ph.D student in the Center for Applied Mathematics. He joined the Modeling and Simulation Project in the Department of Computer Science in 1988 and received his Master's degree in Computer Science in 1989. His research interests include solid and geometric modeling, computer graphics, and scientific computing.

To my parents and my sister

Acknowledgements

There are many people whom I would like to acknowledge for their support during my years at Cornell. My advisor, John Hopcroft, has provided much motivation, guidance, and encouragement for this work. Moreover, his high standard for technical writing has enhanced the presentation of the results in this thesis. I thank him for the time he spent with me on my dissertation research.

There are others that I would like to acknowledge as well. Louis Billera has been extremely helpful since the beginning of my thesis work. He has always been available to listen to a problem, or to suggest a solution, or just to talk. Lars Wahlbin and Keshav Pingali served on my committee, and Lars' insightful and humorous criticism has surely helped the accuracy of this work. Mike Stillman, Jorg Peters, Allen Back, Joe Warren, and Chanderjit Bajaj also helped me through many inspiring discussions.

In addition to those cited above, I would also like to thank the members of the Cornell Programmable Automation (CPA) Project in the mechanical engineering department for their interests. In particular, a senior member of CPA, Gene Hartquist, read this thesis and provided many helpful comments.

My parents and my sister have given me continuous love and support, without which this thesis would have not been possible. Gwen Raymond helped me survive the hard days of thesis writing, and she proofread part of this thesis.

Finally, I would like to express my gratitude to Keith Hjortshoj for his time and patience in helping me improve my writing skills.

Table of Contents

List of Figures

Chapter 1

Introduction

Representing, manipulating, and reasoning about physical objects is a central area of research in a wide range of applications, including solid and geometric modeling, computer-aided design and manufacturing, electronic prototyping, computer graphics, and robotics [WLPL+80,HK86,RVC88,Man88,Hof89,FD90]. As these applications extend their geometric coverage, they increasingly demand techniques for handling *free-form surfaces*, surfaces that are smooth but otherwise arbitrary. Free-form surfaces are essential for representing complex physical objects, such as mechanical parts, medical implants, and plastic flow in molds. In this thesis, we derive schemes for constructing free-form surfaces that facilitate manipulating and reasoning about physical objects.

1.1 Motivations

Techniques for constructing free-form surfaces are well-studied in computer-aided geometric design (CAGD), an increasingly prominent area in computer science and mechanical engineering [Pra79,BFK84,Mor87,BBB87,Far88]. Motivated by the automation the design of car bodies, ship hulls, and aircraft surfaces, CAGD has developed a large repertoire of algorithms for constructing free-form surfaces. As far as the design and display of physical objects are concerned, these algorithms have been very successful.

However, when it comes to manipulating and reasoning about physical objects, problems arise [HK86]. Traditional CAGD constructs free-form surfaces from parametric patches; yet manipulating and reasoning about physical objects with parametric patches poses fundamental difficulties. Some elementary operations in geometric modeling, such as sweeping and convolution, are not closed for parametric patches [BK87]. The results of these operations have to be approximated, making the operations very expensive. More serious problems are encountered when intersecting parametric patches. Because of their complicated topology, the intersections of parametric patches are extremely hard to represent and evaluate [PG86,BHLH88,Hof89].

The causes of these problems are two-fold: on the one hand, parametric surfaces form only a subset of the set of algebraic surfaces; on the other, the commonly used parametric patches have high algebraic degrees. To expand on the second point, we observe that bicubic patches [FD90] have degree 18, and the intersection of two degree 18 surfaces is a degree 324 space curve. It should not be surprising that such a space curve has a complicated topology [Sed83,HK86].

To avoid the above problems, we construct free-form surfaces from low-degree algebraic surface patches. Algebraic surfaces are closed under all common operations of geometric and solid modeling [Baj88], and low-degree algebraic surfaces greatly simplify intersections because the intersection of a degree m surface and a degree n surface is a degree mn curve [Har77]. Additionally, low-degree algebraic surfaces allow the use of algebraic methods as opposed to numerical methods in reasoning about physical objects, and free-form surfaces consisting of algebraic surface patches are compact to store and fast to ray trace. These features make low-degree algebraic surface patches a superior choice for constructing free-form surfaces in applications where manipulating and reasoning about physical objects are important.

1.2 The main results

In this thesis, we construct free-form surfaces from the input data of an arbitrary polyhedron with normals prescribed at its vertices. More precisely, we consider the following problem: given the input data of an arbitrary polyhedron with a normal $\mathbf{n}_i$ prescribed at each vertex $\mathbf{x}_i$ of the polyhedron, create a smooth piecewise algebraic surface that *smoothly contains* the vertices of the polyhedron. A surface defined by $f = 0$ *smoothly contains* the vertex $\mathbf{x}_i$ if

$$f(\mathbf{x}_i) = 0 \text{ and } \nabla f(\mathbf{x}_i) = \alpha \mathbf{n}_i$$

for some nonzero constant α.

Note that part of the input data is essential: the vertices of the polyhedron specify the points that are known to be on the free-form surface, and the topology of the polyhedron is the topology of the free-form surface. The rest of the input data, i.e. the collection of the prescribed normals at the vertices of the polyhedron, is optional but very important for the quality of the free-form surfaces. The prescribed normals control the shapes of the algebraic surface patches in the free-form surface so that a few algebraic surface patches can represent a curved object that would otherwise require thousands of polygons to approximate.

We are primarily interested in free-form surface constructions using low-degree algebraic surface patches. The algebraic surface patches of the lowest degree, quadric patches, are particularly inviting. Quadric patches are widely used as primitives in existing solid modeling and graphics systems [RV83,EKL+91,EKM+91], and techniques for manipulating and reasoning about physical objects bounded by quadric patches are well understood [Lev76,OSS83,EKL+91,EKM+91]. So, the first part of this thesis is devoted to the construction of free-form surfaces using quadric patches.

Given a polyhedron with normals prescribed at its vertices, we show that constructing a free-form surface by replacing each facet of the polyhedron with a quadric patch is possible only if the prescribed normals satisfy a condition. Under this condition, we give an algorithm for constructing the free-form surfaces. We also give a procedure for generating the normals at the vertices of the polyhedron according to the geometry of the polyhedron and the above condition. By combining the procedure for generating the normals and the algorithm for constructing the free-form surfaces, we can fit a smooth piecewise quadric surface through the vertices of an arbitrary polyhedron without splitting its facets.

In this thesis, we also apply blending techniques to free-form surface constructions. Blending is the process of providing smooth transitions between distinctive features. Through the work of many authors [MS85,Ros85,HH87,RO87,Woo87, War89], powerful blending techniques have been developed. The main difficulties in applying these techniques to free-form surface constructions are that blending surfaces have high degrees, and blending causes sharp corners at the vertices of the input polyhedron. We develop strategies for overcoming these difficulties. As an application of these strategies, we improve the performance of the free-form surface algorithm mentioned earlier using blending techniques.

Free-form surface constructions using quadric patches do have their limitations. Recall that a free-form surface obtained by replacing the facets of a polyhedron with quadric patches has a correlation between the normals of the free-form surface at adjacent vertices. This correlation prevents the control of the shape of the free-form surface through the normals at the vertices of the polyhedron.

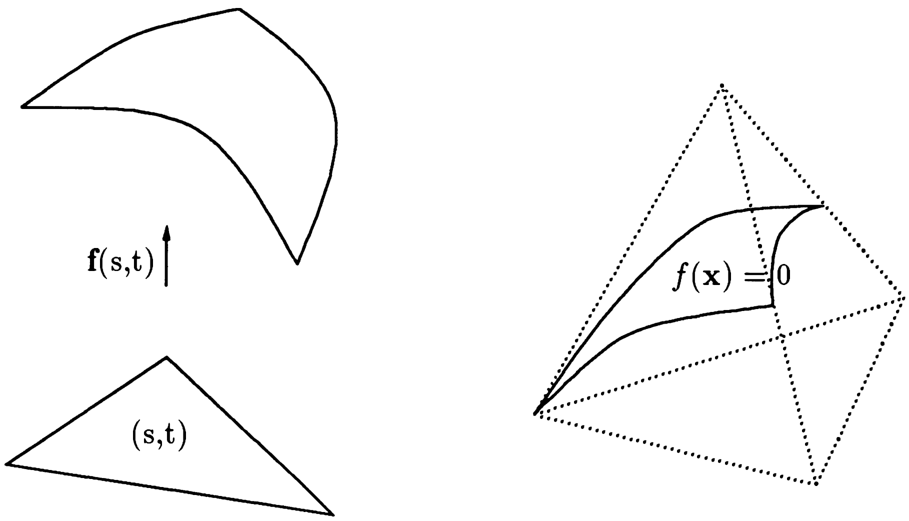
The above limitations can be overcome by constructing free-form surfaces using cubic patches. Given a polyhedron with arbitrarily prescribed normals at its vertices, we give an algorithm for constructing a smooth piecewise cubic surface that smoothly contains the vertices of the polyhedron. Using this algorithm, we can accurately model arbitrary smooth objects with cubic patches. This is encouraging because geometric operations on cubic surfaces, such as parametrizations and intersections, can be done efficiently [Sed87,AB87,PG86].

An important requirement for free-form surfaces is that they must easily lend themselves to engineering design. In engineering design, one often encounters various shape requirements, such as restrictions on the minimum radius of curvature and a nice pattern of reflection lines. Among all shape requirements, the most basic and frequent one is convexity. In this thesis, we show how to control the shape of a free-form surface to achieve convexity.

The free-form surface algorithms we derive have several attractive features that are appealing to a diverse range of applications. Free-form surfaces are needed for two purposes: representing existing objects and designing new objects [FD90]. For representing existing objects, the representation should be accurate, while for designing new objects, the shapes of the free-form surfaces should be easy to control. These requirements are met very well in the algorithms we derive. In terms of accuracy, our algorithms produce free-form surfaces with *quadratic precision*, meaning that if the input data to the algorithms comes from a quadric surface, the algorithms will reproduce the quadric surface. In terms of facilitating the design of new objects, our algorithms have two features. One, the free-form surfaces produced by the algorithms can be controlled through a set of intuitive *shape parameters* without knowing the details of the algorithms; two, the algorithms are *local*, which means modifying a surface patch affects only a small portion of the free-form surface. We will have more discussion on these features in the later chapters.

1.3 The basic ideas

This section describes the ideas that are important to all the algorithms to be presented in the later chapters. The main difference between our work and traditional CAGD is that we construct free-form surfaces using implicitly represented algebraic surface patches instead of parametric patches. So in this section, we will particularly emphasize the techniques for handling implicitly defined algebraic



A parametric patch

An implicit patch

Figure 1.1: A parametric patch and an implicit patch

surface patches, which will be referred to as *implicit patches* in the future.

1.3.1 The definition of implicit patches

When an algebraic surface can be represented parametrically, identifying a patch on the surface is straightforward. A parametric surface is defined as a mapping from the parameter space $\mathbf{R}^2$ to the three-dimensional x-y-z space $\mathbf{R}^3$, i.e.

$$\mathbf{x} = \mathbf{f}(s, t).$$

where $\mathbf{f}$ is a vector-valued rational function. A *parametric patch* may be obtained by restricting the parameter (s, t) to a domain in $\mathbf{R}^2$. The domain is usually triangular because triangular patches are flexible enough for modeling arbitrary shapes [MLL+90].

Identifying patches from an implicitly represented algebraic surface, $S(f)$, is not so easy. The surface $S(f)$ is defined as the set of points $\mathbf{x}$ such that

$$f(\mathbf{x}) = 0, \ \mathbf{x} \in \mathbf{R}^3.$$

With this definition, there is no obvious way to find two independent variables and to specify a range for them so that a patch on the surface is defined by the range.

We identify an implicit patch on the surface $S(f)$ by introducing a set of *clipping surfaces*, $S(g_1), \cdots, S(g_n)$. An implicit patch is defined as the set of points $\mathbf{x}$ on $S(f)$ such that

$$g_i(\mathbf{x}) \geq 0, \; i = 1, \cdots, n.$$

As opposed to clipping surfaces, the surface $S(f)$ is called the *primary surface* of the implicit patch.

In this thesis, we only consider planar clipping surfaces, and the clipping planes always form a tetrahedral bounding box. An implicit patch is the zero contour of a polynomial inside the bounding tetrahedron. Here tetrahedra are chosen because they facilitate the use of the Bezier-Bernstein surface technique, which is basic for our results.

A major difficulty in using implicit patches is handling topological anomalies [Sed85,PK89,BI89,MW90]. The zero contour of a polynomial inside a tetrahedron can exhibit very complicated behaviors, such as holes, extraneous sheets, self intersections, etc. In each free-form surface algorithm we derive, we incorporate a strategy for avoiding topological anomalies.

1.3.2 Free-form surface constructions using implicit patches

Recall that we described the problem of constructing free-form surfaces in Section 1.2. From that description, a free-form surface algorithm immediately suggests itself. Writing the implicit patches as algebraic equations with symbolic coefficients, we can express, in a system of algebraic equations with the symbolic coefficients as unknowns, the requirement that the free-form surface smoothly contains the vertices of the input polyhedron and the requirement that adjacent implicit patches meet smoothly. Solving this system of equations gives the free-form surface to be constructed.

Conceptually, this algorithm captures the mathematical nature of free-form surface constructions. However, it poses fundamental difficulties in practice: the computation needed for solving such a global, nonlinear system exceeds the power of modern computers, and the proposed algorithm is global.

To overcome the difficulties in solving the global, nonlinear system, we formulate the free-form surface construction as a series of local, linear subsystems (we "break" the global, nonlinear system, so to speak). In fact, how this is done is the gist of a free-form surface algorithm. The robustness, efficiency, and quality of the resulting free-form surfaces depend, to a large extent, on the way in which the global, nonlinear system is "broken".

Closely related to the issue of breaking the global, nonlinear system of equations is the partitioning of space into bounding tetrahedra for the implicit patches used in the free-form surface. Since the free-form surface is in a neighborhood of the input polyhedron, only this neighborhood needs to be partitioned into tetrahedra. In this thesis, the partition of space is done by constructing a neighborhood of the input polyhedron using tetrahedra.

After constructing the neighborhood with tetrahedra and forming a series of local, linear subsystems, a free-form surface is constructed by creating an implicit patch within each tetrahedron. One can create an implicit patch within a tetrahedron by simply defining a polynomial for the tetrahedron; the hard part is to achieve the smoothness between adjacent patches. The easiest way to ensure a smooth transition between a pair of adjacent implicit patches is to enforce C^1 conditions between the polynomials that define the implicit patches. A more sophisticated way to achieve smoothness involves the notion of geometric continuity, which is an important subject for its own sake.

1.3.3 Geometric continuity

So far we have been resorting to the reader's intuition whenever we mention "smooth". To precisely define smoothness, we need the concept of *geometric continuity*, which is created to express the exact conditions for visual smoothness between adjacent surface patches. In this thesis, we are mainly interested in first-order smoothness of implicit surfaces, in which case geometric continuity is the familiar tangent-plane continuity[1]. A sufficient condition for tangent-plane continuity is gradient continuity.

Geometric continuity first appeared in the research on parametric surfaces. The traditional measure of smoothness between two adjacent parametric patches, known as *parametric continuity*, requires that the two patches have equal derivatives along their common boundary. This requirement is overly restrictive because it depends on parameterizing details that are irrelevant to geometry [DeR85].

A major drawback of parametric continuity is that a closed free-form surface cannot be built from parametric patches under the notion of parametric continuity [Her85]. Roughly speaking, this is because there is no C^1 mapping that maps a domain in a plane onto a closed surface in $\mathbf{R}^3$. Since closed free-form surfaces are widely used to model physical objects, geometric continuity is essential for free-form surface constructions using parametric patches.

In the context of implicit patches, geometric continuity is also introduced as opposed to the gradient continuity [War86,GW89]. However, it is a common practice to derive a closed free-form surface from a gradient-continuous function [Dah89, MW90,PK89]. In this sense, geometric continuity is not crucial to free-form surface constructions using implicit patches. In fact, none of the research on free-form surfaces so far has been able to benefit from the work on the geometric continuity of implicit patches.

But geometric continuity does lead to more efficient algorithms for free-form

[1]Without cusps

surface constructions using implicit patches. This can be seen from the method for creating and assembling implicit patches in Section 1.3.2, where the gradients of two adjacent patches have to be equal over the entire plane separating the two tetrahedra where the patches lie. When geometric continuity is used, the gradients only need to be parallel along the intersection curve of the two implicit patches.

In this thesis, we will discuss the geometric continuity of implicit patches. In fact, without exploring the full power of geometric continuity, our algorithm for free-form surface constructions using quadric patches would not be possible.

1.4 Minimum input data

As was mentioned earlier, the free-form surface constructions in this thesis assume the input data of a polyhedron with normals prescribed at its vertices. But there are situations in which only a set of points is given as the input data. For example, in medical imaging, a CT scan gives the data points on the surface of a bone, and the shape of the bone needs to be reconstructed from these data points. We call input data consisting of only data points *minimum input data.*

This section shows how to construct free-form surfaces from minimum input data. For simplicity's sake, we assume that the free-form surface to be constructed is a *simple surface*, a surface without holes, without self intersections, and bounding a single connected region. Under this assumption, we can create the input data for free-form surface constructions as follows. First, we find the most "natural" simple polyhedron using the data points as vertices. Then, the normal at each vertex is chosen to be a weighted average of the normals of the incident facets.

Consider the problem of finding the most "natural" polyhedron having the data points as vertices. Conceptually, computing a polyhedron merely from a set of points is not straightforward. Because no information other than the points is known about the polyhedron, there are many choices. However, intuition tells us that, at least when the data points are dense enough, there is a polyhedron that is the most "natural" choice. So what we are seeking here are procedures that produce a reasonable polyhedron for sparse data points and "converge" for dense data points. Here by converge we mean that if the data points are from a simple surface, the Hausdorff distance between the surface and the polyhedron can be made arbitrarily small as the data points become increasingly dense.

The most "natural" polyhedron is computed using a heuristic search strategy. The basic idea of the heuristic search strategy is to incorporate both local nearness and global constraints (simplicity, covering all the data points, etc.).

To incorporate local nearness, we restrict the search for the most "natural" polyhedron to within the Delaunay triangulation of the data points. In other words, the candidates of the most "natural" polyhedron consist of only the polyhedra whose facets are the facets of the Delaunay triangulation. For each data point $\mathbf{p}$, there is a polyhedron defined by all the points in $\mathbf{R}^3$ which are closer to the point $\mathbf{p}$ than to any other data point. This division of space into polyhedra is called the Voronoi diagram of the data points. The geometric dual of the Voronoi diagram is the Delaunay triangulation, which triangulates the convex hull of the data points [PS85]. Since Delaunay neighbors are near in the sense that their Voronoi polyhedra share a common face, confining the search of the polyhedron to within Delaunay triangulation reflects local nearness.

A related question is whether a simple polyhedron covering all data points (3-D generalization of a Hamiltonian cycle) always exists in the Delaunay triangulation of the data points. While a general answer to this question has not been found, a simple polyhedron covering all data points exists for cases that arise in practical applications.

A search of the Delaunay triangulation for a simple polyhedron using the data points as vertices can be done in two ways. One is to "shrink" the convex hull bounding the Delaunay triangulation until we get a polyhedron whose boundary has all the data points as vertices [Boi84,Boi85]. This polyhedron is then taken as the simple polyhedron to be computed. The "shrinking" is done by iteratively eliminating the following tetrahedra.

1. Tetrahedra with one face, three edges, and three points on the boundary.

2. Tetrahedra with two faces, five edges, and four points on the boundary.

The above rules enable us to obtain all simple polyhedra inside the Delaunay triangulation [Boi84]. To ensure that the final polyhedron looks "nice", we associate a value with each *boundary tetrahedron*, a tetrahedron having a facet on the boundary. The value is the maximum distance between the boundary facet and its corresponding portion of the sphere circumscribing the tetrahedron. The eliminating process stops only when all the data points are on the boundary and the values of boundary tetrahedra cannot be reduced any further. This process has complexity $O(n^2 logn)$, where n is the number of the data points.

Another way to obtain a simple polyhedron having the data points as vertices is to search the Voronoi diagram [OBW87]. Every simple polyhedron having all data points as vertices corresponds to a tree in the Voronoi diagram. The tree touches every Voronoi polyhedron, hence the name *spanning Voronoi tree.* Analysis of the relation between the *skeleton* [Kir79] of the polyhedron to be computed and the Voronoi diagram shows that the most concise, and hence the most "natural" polyhedron corresponds to a minimum spanning Voronoi tree [LBMSFB88, Lee82]. Such a tree can be computed using the usual greedy approach [AHU74] with simplicity checks [OBW87].

A problem with the second approach is that it can take up to $O(e^3)$ time, where e is the number of edges in the Voronoi diagram. For n data points, there can be as many as $O(n^2)$ edges for 3-D Voronoi diagram [Pre77], so $O(e^3) = O(n^6)$. Fortunately, if the data points are taken homogeneously[2] from a surface, the number of Voronoi edges can be bounded by $7n$ [LBMSFB88]. So in practice, $O(e^3) = O(n^3)$.

[2]Here homogeneous means that the data points are evenly distributed in space

It is worth noticing that while searching for the simple polyhedron, we also packed the interior of the polyhedron with tetrahedra. Such a packing is very useful for applying the powerful engineering tool of finite element analysis.

1.5 Overview

This thesis consists of seven chapters. Chapter 2 introduces the mathematical concepts needed for later chapters. Part of the chapter gives algebraic background: the projective space, the theory of ideals, and algebraic geometry. The rest of the chapter describes the Bernstein-Bezier representation, which is very useful for constructing free-form surfaces.

The purpose of Chapter 3 is to understand the role of quadric patches as modeling primitives. We would like to derive an algorithm which constructs a free-form surface by replacing each facet of the input polyhedron with a quadric patch, but analyzing the constraints placed on a typical quadric patch in a free-form surface shows that such an algorithm is possible only if the normals prescribed at the vertices of the input polyhedron satisfy a certain condition. Given an arbitrary polyhedron, we derive a procedure for generating normals at the vertices of the input polyhedron according to the geometry of the polyhedron and the above condition. Under the above condition, we give an algorithm for fitting a smooth piecewise quadric surface through the vertices of the polyhedron without splitting its facets. In Chapter 3, we also examine the meaning of geometric continuity in the case of quadric patches, and we show that the algorithm we derive explores the full power of geometric continuity.

Chapter 4 applies blending techniques to free-form surface constructions. On one hand, we show that blending techniques lead to very efficient constructions of free-form surfaces; on the other, we identify the limitations of blending techniques and show how to overcome them. In applying blending techniques, we develop strategies for avoiding blending corners and for reducing the degrees of the blending surfaces. We also show how our results can be derived using techniques from algebraic geometry.

Chapter 5 describes an algorithm for free-form surface construction using cubic patches. Using a Clough-Tocher like splitting scheme, the algorithm constructs a free-form surface that smoothly contains the vertices of the input polyhedron. In using cubic patches, avoiding topological anomalies is particularly important. Cubic patches can exhibit very complicated behaviors: holes, extraneous sheets, self intersections, etc. More dramatically, a handle may appear on an otherwise nice cubic patch. We present a general strategy for avoiding such undesirable behaviors.

Chapter 6 develops techniques for controlling the shapes of free-form surfaces. In particular, we show how to obtain a convex quadric patch or a convex cubic surface patch by manipulating the *control points*, i.e. the coefficients of the Bernstein-Bezier representation.

Finally, Chapter 7 summarize our results and points out directions for future research.

Chapter 2

Notations and background

The goal of this thesis is to design algorithms for generating free-form surfaces using low-degree algebraic surface patches. To achieve this goal, techniques from several fields are used. This chapter provides a brief introduction to these techniques.

We are interested in the algebraic surfaces in real three-dimensional x-y-z space $\mathbf{R}^3$. An algebraic surface $S(f)$ in $\mathbf{R}^3$ is defined as the zero contour of a polynomial f in $\mathbf{R}[\mathbf{x}]$, the ring of all the trivariate polynomials with real coefficients, i.e.

$$S(f) = \{\mathbf{x} | f(\mathbf{x}) = 0\}.$$

For example, $S(x^2 + y^2 + z^2 - 1)$ is the unit sphere in $\mathbf{R}^3$.

When the polynomial ring $\mathbf{R}[\mathbf{x}]$ is viewed as a linear space, two bases for the linear space are extremely useful. One is the frequently used power basis, $\{x_1^i x_2^j x_3^k\}$; the other consists of the Bernstein polynomials. The latter is well suited for constructing smooth piecewise polynomial surfaces [Far85,BBB87]. Both bases will be used in this thesis. The first three sections of this chapter are concerned with polynomials in the power basis, and the last section is about polynomials represented by Bernstein polynomials.

2.1 The theory of ideals

The theory of ideals plays an important role in the study of algebraic surfaces. We often express the geometric properties of algebraic surfaces in terms of ideals so that algebraic techniques can be applied. This section introduces some basic concepts of ideal theory that will be used later. Some knowledge of algebra including rings is assumed here [Her75]. Throughout this section, let R be an arbitrary ring.

Definition 2.1 *An ideal I is a nonempty subset of a ring R such that*

1. *I is closed under addition.*
2. *If $a \in I$ and $b \in R$, then ab and ba are both members of I.*

An important type of ideals is the *prime* ideal. An ideal is *prime* if $ab \in I$ implies that either a or b is in I.

From the definition, if $b_1, \cdots, b_n$ are members of ring R, then

$$(b_1, \cdots, b_n) = \{a | a = \sum a_i b_i \text{ for some } a_i \in R\}$$

is an ideal. This ideal is called the ideal generated by the *basis* $\{b_1, \cdots, b_n\}$. For example, the ideal (y, xy, y^2) is the set of polynomials of the form $p(x, y)y$, where $p(x, y)$ is an arbitrary polynomial.

Given two ideals I and J, several operations on them can be defined. The *sum* of the two ideals, $I + J$, is the ideal $\{a + b | a \in I \text{ and } b \in J\}$, which turns out to be the smallest ideal that contains $I \cup J$. The *product* of the two ideals, IJ, is the ideal $\{ab | a \in I \text{ and } b \in J\}$. Finally, the *intersection* of the two ideals, $I \cap J$, is also an ideal.

If the ideals I and J have bases $\{a_1, \cdots, a_m\}$ and $\{b_1, \cdots, b_n\}$, then the bases for the ideals $I + J$ and IJ can be found. A basis for $I + J$ is $\{a_1, \cdots, a_m, b_1, \cdots, b_n\}$, and a basis for IJ is $\{a_i b_j | 1 \leq i \leq m, 1 \leq j \leq n\}$. There is no direct relationship between the bases for I and J and the basis for $I \cap J$.

We are particularly interested in ideals of the polynomial ring $\mathbf{R}[\mathbf{x}]$. Given an ideal in $\mathbf{R}[\mathbf{x}]$ with a basis $\{g_1, \cdots, g_k\}$, deciding whether a member f of the ring $\mathbf{R}[\mathbf{x}]$ is a member of the ideal $(g_1, \cdots, g_k)$ is known as the ideal membership problem. An ideal can have many different bases. For solving the ideal membership problem, a particular basis of $\mathbf{R}[\mathbf{x}]$ called the Groebner basis is extremely useful. Roughly speaking, any member f of the ring $\mathbf{R}[\mathbf{x}]$ can be simplified into a unique normal form using the Groebner basis of an ideal, and f is in the ideal if and only if the normal form is zero [Buc85]. A Groebner basis can be constructed from any basis, so it provides a general technique for solving the ideal membership problem.

2.2 Algebraic geometry

The primary goal of algebraic geometry is to study the geometry of the solutions to a system of algebraic equations. Algebraic geometry provides both powerful tools and deep insights necessary for studying algebraic surfaces and their intersections [Har77].

One thing we have to be careful about when using algebraic geometry is the field of coefficients for the algebraic surfaces. The algebraic surfaces that can be used to construct free-form surfaces are defined by polynomials with real coefficients. However, the field of the real numbers, $\mathbf{R}$, is not an algebraically closed field. Since many results in algebraic geometry require that the coefficient field of the algebraic equations be closed, we work in the ring of complex polynomials $\mathbf{C}[\mathbf{x}]$ whenever algebraic geometry is used.

We use $S(F)$ to denote the following subset of $\mathbf{C}^3$,

$$\{\mathbf{x} | f(\mathbf{x}) = 0 \text{ for all } f \in F\}.$$

When $F = \{f_1, \cdots, f_n\}$, $S(F)$ is simplified to $S(f_1, \cdots, f_n)$. For example, $S(y, z)$ is the x-axis.

Definition 2.2 *A subset S of $\mathbf{C}^3$ is an algebraic set if there exists an ideal I in the ring $\mathbf{C}[\mathbf{x}]$ such that $S = S(I)$.*

When $I = (f_1, \cdots, f_n)$, it is easy to verify from the definitions that $S(I) = S(f_1, \cdots, f_n)$. For examples, the curve of intersection of two algebraic surfaces $S(f_1, f_2)$ is the algebraic set of the ideal generated by f_1 and f_2, and the algebraic surface $S(f)$ is the algebraic set of the ideal generated by f. By defining algebraic sets of ideals, we can treat curves and surfaces in a unified way.

An algebraic set is *reducible* if it is the union of two different algebraic sets. Otherwise, it is *irreducible.* For example, an algebraic surface $S(f)$ is irreducible if f does not factor nontrivially into two polynomials. In general, an algebraic set S is irreducible if and only if there exists a prime ideal I such that $S = S(I)$. Notice that two irreducible algebraic surfaces need not have a single irreducible intersection. An example is the intersection of two perpendicular cylinders $S(x^2 + y^2 - 1)$ and $S(z^2 + y^2 - 1)$.

We are particularly interested in the way in which two algebraic surfaces intersect. The intersection of two algebraic surfaces is *proper* if the intersection is of dimension one. Most of the time, two surfaces intersect properly, but nonproper intersections do occur. For example, two surfaces may touch each other only at two points, as is the case of a sphere and an ellipsoid that are tangent at two poles.

In general, if $S(f)$ is an algebraic surface that contains the curve of intersection $S(g, h)$ of two algebraic surfaces, then there is a connection between the ideal (g, h) and the polynomial f, given by:

Theorem 2.1 (Hilbert's Nullstellensatz) *The surface $S(f)$ contains $S(g, h)$ if and only if there is an integer k such that $f^k \in (g, h)$. Moreover, if $S(g, h)$ is irreducible, then $k = 1$.*

In applications, identifying when $k = 1$ is very useful. The condition given in the theorem, the intersection $S(g, h)$ being irreducible, is overly restrictive. For example, if we choose surfaces $S(g)$ and $S(h)$ to be the two perpendicular cylinders $S(x^2+y^2-1)$ and $S(y^2+z^2-1)$, then the intersection is reducible; but a surface $S(f)$ contains $S(g,h)$ if and only if $f \in (g,h)$, as we shall see later.

More careful analysis reveals that $k > 1$ occurs when the surfaces $S(g)$ and $S(h)$ are tangent along the intersection $S(g,h)$, and an example of such situation can be found in [War89]. So we introduce the concept that two algebraic surfaces $S(g)$ and $S(h)$ intersect *transversally*, which means there exists a point on each irreducible component of $S(g,h)$ such that the tangent planes to $S(g)$ and $S(h)$ exist and are distinct.

Theorem 2.2 *If the surfaces $S(g)$ and $S(h)$ intersect transversally, then the surface $S(f)$ contains $S(g,h)$ if and only if $f \in (g,h)$.*

Proof: See [Kun85]. ♣

Recall the example of the two perpendicular cylinders $S(x^2+y^2-1)$ and $S(y^2+z^2-1)$. Their intersection is transversal, so an algebraic surface $S(f)$ contains $S(x^2+y^2-1, y^2+z^2-1)$ if and only if $f \in (x^2+y^2-1, y^2+z^2-1)$.

2.3 Projective space

Some results from algebraic geometry depend on the concept of several algebraic surfaces having common components at infinity in complex projective space. We shall use these results in the later chapters. However, this needed generality does not distort the intuitive contents of our results. In fact, all the free-form surface constructions in this thesis are done in the real x-y-z space.

A familiar concept to the reader is an *affine space*, a linear space whose points are tuples $(x_1, \cdots, x_n)$, where n is the dimension of the affine space. An n-dimensional *projective space* is defined as follows. Consider the (n+1)-dimensional affine space with the origin excluded. We create equivalence classes according to the following relationship: $(y_0, \cdots, y_n)$ and $(z_0, \cdots, z_n)$ are equivalent if and only *ponents at infinity. If* $f \in (g_1, \cdots, g_m)$*, then there exists an* $a_i \in \mathbf{C}[\mathbf{x}]$ *such that* $f = \sum a_i g_i$ *and* $deg(a_i g_i) \leq deg(f)$ *for all* i*.*

Proof: See [War86]. ♣

An important advantage of introducing complex projective space is that many algebraic geometry statements are valid without exceptions. An important example is that if $S(f)$ is an algebraic surface of degree n, then a line intersects $S(f)$ in n points. This defines the degree of an algebraic surface in geometric terms: the degree of an algebraic surface is the number of points the surface intersects a line in general position. Similarly, the degree of a curve is the number of points the curve intersects a plane in general position.

Having defined the degrees of curves, we give a theorem that will be particularly useful in our work.

Theorem 2.4 (Bezout's Theorem) *A curve of degree* m *and a curve of degree* n *intersect in* mn *points or have a common component.*

This theorem is also true for surfaces. A surface $S(f)$ of degree m and a surface $S(g)$ of degree n intersect in a space curve of degree mn or have a common component. Notice that the common component may lie in the plane at infinity. Recall the example of two parallel cylinders $S(x^2 + (y+1)^2 - 4)$ and $S(x^2 + (y-1)^2 - 4)$ that intersect along two lines. By Bezout's theorem, they must have a common component at infinity.

2.4 The Bernstein-Bezier representation of algebraic surfaces

So far, we have been working with the power basis of the polynomial ring $\mathbf{R}[\mathbf{x}]$. Another equally important basis of $\mathbf{R}[\mathbf{x}]$ is the Bernstein polynomials.

Bernstein polynomials are best expressed in barycentric coordinates. Given a tetrahedron V having vertices $\mathbf{x}_1$, $\mathbf{x}_2$, $\mathbf{x}_3$, and $\mathbf{x}_4$ in general position, we can express any point $\mathbf{x}$ in the x-y-z space as:

$$\mathbf{x} = \sum_{i=1}^{4} \tau_i \mathbf{x}_i,$$

where

$$\sum_{i=1}^{4} \tau_i = 1.$$

The tuples $(\tau_1, \tau_2, \tau_3, \tau_4)$ is called the barycentric coordinate of $\mathbf{x}$.

Sometimes we shall refer to the points in x-y-z space by their barycentric coordinates $\tau = (\tau_1, \tau_2, \tau_3, \tau_4)$. A point τ is inside the tetrahedron V if all $\tau_i \geq 0$, otherwise the point is outside V. The barycentric coordinates are linearly related to Cartesian coordinates, so any polynomial $f(\mathbf{x})$ may be expressed as $f(\tau)$ via a linear change of variables.

Bernstein polynomials are defined for all nonnegative integer tuple $\lambda = (\lambda_1, \lambda_2, \lambda_3, \lambda_4)$ with $|\lambda| = \sum_{i=1}^{4} \lambda_i = n$:

$$B_\lambda^n(\tau) = \frac{n!}{\lambda_1!\lambda_2!\lambda_3!\lambda_4!} \tau_1^{\lambda_1} \tau_2^{\lambda_2} \tau_3^{\lambda_3} \tau_4^{\lambda_4}$$

This is a straightforward generalization of the univariate Bernstein polynomials,

$$B_k^n(t) = \frac{n!}{k!(n-k)!} t^k (1-t)^{n-k}$$

for $k = 0, \cdots, n$. Univariate Bernstein polynomials are important in elementary probability, statistics, and some approximation theory.

Bernstein polynomials form a basis for the vector space $\mathbf{R}[\mathbf{x}]$ of real polynomials. Any polynomial f of degree less than or equal to n can be uniquely expressed as a linear combination of Bernstein polynomials.

$$f(\tau) = \sum_{|\lambda|=n} b_\lambda B_\lambda^n(\tau).$$

This is called the Bernstein-Bezier representation of polynomials. The coefficient b_λ is also denoted as $b_{\lambda_1\lambda_2\lambda_3\lambda_4}$ or $b_{\lambda_1 e^1 + \lambda_2 e^2 + \lambda_3 e^3 + \lambda_4 e^4}$, where $e^1 = (1, 0, \cdots, 0)$, $e^2 =$

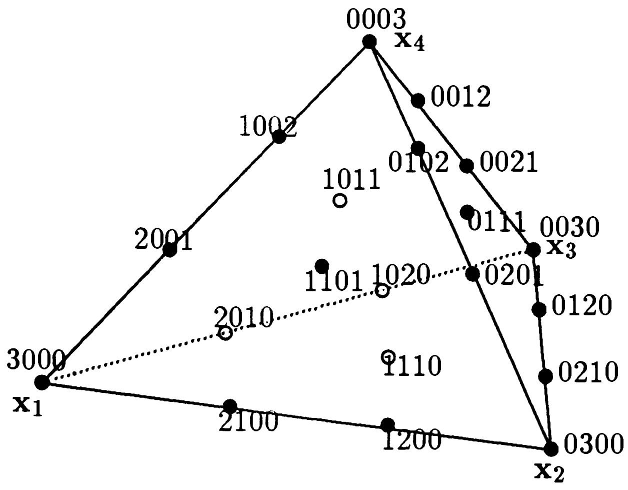


Figure 2.1: Cubic control points

$(0,1,0,\cdots,0)$, etc. These notations are convenient for arithmetics on the indexes. For examples, b_{3e^1} stands for b_{3000}, and $b_{\lambda+e^2}$ with $\lambda = (1,0,1,0)$ is b_{1110}.

The b_λ's are referred to as *control points* of the polynomial f and the algebraic surface $S(f)$, because the values of the control points can be used to control the geometry of the algebraic surface $S(f)$. If we associate b_λ with the point $\frac{1}{n}(\lambda_1, \lambda_2, \lambda_3, \lambda_4)$ on the tetrahedron V as shown in Figure 2.1, then the control point at a vertex of V assumes the function value of f at the vertex, and the control points next to the vertex completely determine the gradient at the vertex.

Lemma 2.1 *If*

$$f(\mathbf{x}) = \sum_{|\lambda|=k} c_\lambda B_\lambda^k(\tau),$$

then

$$c_{(k-1)e^i+e^j} = c_{ke^i} + \frac{1}{k}(\nabla f(\mathbf{x}_i), \mathbf{x}_j - \mathbf{x}_i), \; j = 1,2,3,4.$$

Here $(,)$ *denotes the scalar product of two vectors.*

Proof: From [Dah86],

$$(\mathbf{x}_i - \mathbf{x}_j, \nabla f(\mathbf{x})) = k \sum_{|\lambda|=k-1} (c_{\lambda+e^i} - c_{\lambda+e^j}) B_\lambda^{k-1}(\tau)$$

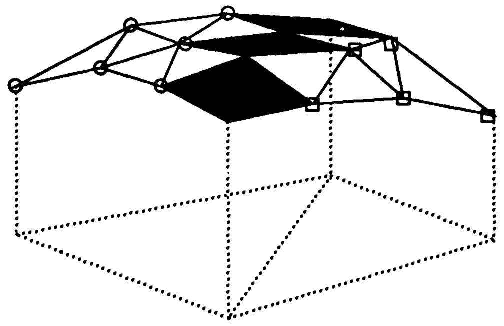


Figure 2.2: The C^1 conditions for two bivariate polynomials

Letting $\mathbf{x} = \mathbf{x}_i$ we prove the lemma. ♣

Since $\mathbf{x}_j - \mathbf{x}_i$ $(j \neq i)$ are three linearly independent vectors, the control points $c_{(k-1)e^i+e^j}$ $(j = 1,2,3,4)$ determine $\nabla f(\mathbf{x}_i)$.

The C^0 and C^1 conditions[1] of two polynomials defined on two tetrahedra sharing a common face can be expressed in terms of control points of the two polynomials. Let $V' = [\mathbf{x}_1\mathbf{x}_2\mathbf{x}_3\mathbf{x}'_4]$ be a tetrahedron sharing face $[\mathbf{x}_1\mathbf{x}_2\mathbf{x}_3]$ with tetrahedron V^2. If a polynomial f' is expressed in terms of the barycentric coordinate τ' of V' as

$$f'(\tau') = \sum_{|\lambda|=k} c'_\lambda B^k_\lambda(\tau'),$$

then the C^0 condition across the common face $[\mathbf{x}_1\mathbf{x}_2\mathbf{x}_3]$ is

$$c_\lambda = c'_\lambda, \text{ for all } \lambda \text{ with } \lambda_4 = 0.$$

Similarly, the C^1 condition across the common face $[\mathbf{x}_1\mathbf{x}_2\mathbf{x}_3]$ is

$$c'_{\lambda+e^4} = \mu_1 c_{\lambda+e^1} + \mu_2 c_{\lambda+e^2} + \mu_3 c_{\lambda+e^3} + \mu_4 c_{\lambda+e^4} \text{ for all } \lambda \text{ with } \lambda_4 = 0,$$

where the μ's are determined from the relation

$$\mathbf{x}'_4 = \mu_1\mathbf{x}_1 + \mu_2\mathbf{x}_2 + \mu_3\mathbf{x}_3 + \mu_4\mathbf{x}_4.$$

[1]We use C^k to denote that the k-th order derivatives of a function are continuous.
[2]We denote the convex hull of the set of points $\{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$ as $[\mathbf{x}_1 \cdots \mathbf{x}_n]$

Intuitively, the C^0 condition is that the control points for the two polynomials on the common face are equal, and the C^1 condition can be visualized as Figure 2.2 (the two-dimensional case), where three pairs of triangles are shaded. The C^1 condition is that each pair of shaded triangles is coplanar.

A useful technique when using polynomials in the Bernstein-Bezier representation is degree elevation. A degree n polynomial f can be represented as a degree $n+1$ polynomial as follows

$$\sum_{|\lambda|=n} c_\lambda B_\lambda^n(\tau) = \sum_{|\beta|=n+1} b_\beta B_\beta^{n+1}(\tau)$$

with

$$b_\beta = \frac{1}{n+1}\sum_{i=1}^{4} \beta_i c_{\beta-e^i}.$$

As an application of degree elevation, consider the C^1 conditions for polynomials over two tetrahedra sharing a common face. The conditions we gave earlier are for two polynomials of the same degree. To generalize the conditions to polynomials of different degrees, we simply elevate the degree of one of the polynomials.

One of the most important algorithms for polynomials in Bernstein-Bezier representation is de Casteljau algorithm, which is a recursive procedure for evaluating a polynomial. This algorithm reduces the task of evaluating a polynomial of arbitrary degree to the task of evaluating linear polynomials.

Theorem 2.5 *If*

$$f(\tau) = \sum_{|\lambda|=k} c_\lambda B_\lambda^k(\tau), \tag{2.1}$$

then

$$f(\tau) = \sum_{|\lambda|=k-r} c_\lambda^r(\tau) B_\lambda^{k-r}(\tau),\ 0 \le r \le k,$$

where

$$c_\lambda^0(\tau) = c_\lambda,$$

$$c_\lambda^r(\tau) = \sum_{i=1}^{4} \tau_i c_{\lambda+e^i}^{r-1}(\tau),\ |\lambda| = n - r.$$

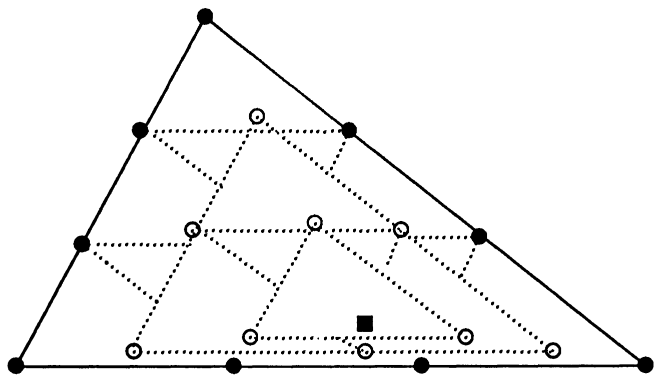


Figure 2.3: The de Casteljau algorithm for bivariate polynomials

Proof: see [Far85]. ♣

Figure 2.3 illustrates the idea of de Casteljau algorithm for a bivariate polynomial.

An important application of de Casteljau algorithm is domain transformation. Given f defined by equation (2.1) over the tetrahedron $V = [\mathbf{x}_1\mathbf{x}_2\mathbf{x}_3\mathbf{x}_4]$, the domain transformation problem is to compute the control points of f on the tetrahedron $V' = [\mathbf{x}'_1\mathbf{x}_2\mathbf{x}_3\mathbf{x}_4]$.

Theorem 2.6 *If*

$$f(\mathbf{x}) = \sum_{|\lambda|=k} c'_\lambda B^k_\lambda(\tau'),$$

where τ' is the barycentric coordinate with respect to tetrahedron V', then

$$c'_{\lambda^r} = c^r_{\lambda^0}(\sigma), \;\; 0 \le r \le n$$

where $\lambda^r = (r, \lambda_2, \lambda_3, \lambda_4)$, $|\lambda^r| = n$, and σ is the barycentric coordinate of $\mathbf{x}'_1$ with respect to V.

Proof: see [Far85]. ♣

Chapter 3

The power and limitations of quadric patches

Quadric patches are the simplest implicit patches that can be used for free-form surface constructions, so it is very natural to ask whether we can design an algorithm that constructs free-form surfaces using quadric patches. Such an algorithm is valuable for two reasons: one, quadric patches are widely used as primitives in solid modeling and other application systems [RV83]; two, techniques for manipulating and reasoning about physical objects bounded by quadric patches are well understood [Lev76,OSS83].

The goal of this chapter is to understand the role of quadric patches as building blocks for free-form surface constructions. On the one hand, we identify the limitations of quadric patches; on the other, we show that one can fit a smooth piecewise quadric surface through the vertices of an arbitrary polyhedron without splitting its facets.

Section 3.1 analyzes the constraints placed on a typical quadric patch in a free-form surface, which is obtained by replacing each facet of a polyhedron with a quadric patch. We show that there is a correlation between the normals of the free-form surface at adjacent vertices of the polyhedron.

In Section 3.2, we describe an algorithm which constructs a free-form surface by replacing each facet of the input polyhedron with a quadric patch. Because of the correlation mentioned above, such an algorithm is possible only if the normals prescribed at the vertices of the input polyhedron satisfy a condition, which we assume in deriving the free-form surface algorithm. Given an arbitrary polyhedron, we provide a procedure for generating normals at the vertices of the polyhedron according to the above condition and the geometry of the polyhedron. By combining the procedure with the free-form surface algorithm, one can fit a smooth piecewise quadric surface through the vertices of an arbitrary polyhedron without splitting its facets.

In designing the free-form surface algorithm, we also develop techniques for avoiding topological anomalies associated with quadric patches. Inside their bounding tetrahedra, quadric patches may exhibit various undesirable behaviors, such as self intersections[1] and multiple sheets. We discuss the causes of these problems and present solutions.

Section 3.3 addresses an important issue in the study of geometric continuity: the role of weight functions in surface construction. Through the analysis of the weight functions for quadric patches, we show that the algorithm we derive explores the full power of geometric continuity.

3.1 Hermite interpolation with quadric surfaces

In this section, we analyze the constraints placed on a typical quadric patch in a free-form surface obtained by replacing each facet of the input polyhedron with a quadric patch. Since the quadric patch smoothly contains the vertices of the facet, the primary surface of the quadric patch is a solution to the following special case of the Hermite interpolation problem.

[1]Self intersections occur where there is a singular point. An example may be found in a cone

The triangular Hermite problem

Given three noncollinear points, $\mathbf{x}_i$ $(i = 1,2,3)$, and a normal $\mathbf{n}_i$ for each point $\mathbf{x}_i$, find a quadric surface $S(f)$ that smoothly contains the given points.

3.1.1 A condition on the normals

Does the triangular Hermite problem always have a solution? To get some feel for the answer, we compare the degrees of freedom with the number of constraints involved in this problem. On the one hand, it is easy to see that a quadric surface $S(f)$ has nine degrees of freedom; on the other, a quadric surface $S(f)$ is a solution to the triangular Hermite problem if and only if $S(f)$ satisfies the constraints

$$f(\mathbf{x}_i) = 0 \quad (i = 1,2,3) \tag{3.1}$$

and

$$\nabla f(\mathbf{x}_i) \parallel \mathbf{n}_i \quad (i = 1,2,3). \tag{3.2}$$

Nine linear constraints result from (3.1) and (3.2) together because for each i, one of the following cases is selected for the constraint (3.2) [BI89].

1. If $n_{ix} \neq 0$, use $n_{ix}f_y(\mathbf{x}_i) - n_{iy}f_x(\mathbf{x}_i) = 0$ and $n_{ix}f_z(\mathbf{x}_i) - n_{iz}f_x(\mathbf{x}_i) = 0$.
2. If $n_{iy} \neq 0$, use $n_{iy}f_x(\mathbf{x}_i) - n_{ix}f_y(\mathbf{x}_i) = 0$ and $n_{iy}f_z(\mathbf{x}_i) - n_{iz}f_y(\mathbf{x}_i) = 0$.
3. If $n_{iz} \neq 0$, use $n_{iz}f_x(\mathbf{x}_i) - n_{ix}f_z(\mathbf{x}_i) = 0$ and $n_{iy}f_z(\mathbf{x}_i) - n_{iz}f_y(\mathbf{x}_i) = 0$.

Here n_{ix} is the x-component of $\mathbf{n}_i$, $f_x = \frac{\partial f}{\partial x}$, etc.

Comparing the degrees of freedom to the number of constraints, one might conclude that the triangular Hermite interpolation problem always has a solution. However, this is not true, as can be seen from the following theorem.

Theorem 3.1 *There exists a quadric surface solving the triangular Hermite problem if and only if the following conditions are satisfied.*

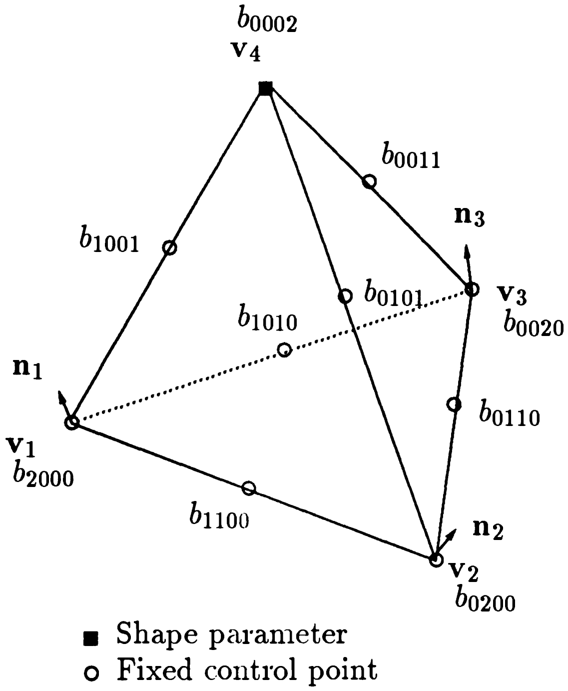


Figure 3.1: The triangular Hermite problem

1. *The two quantities* $(\mathbf{x}_i - \mathbf{x}_j, \mathbf{n}_j)$ *and* $(\mathbf{x}_j - \mathbf{x}_i, \mathbf{n}_i)$ *vanish at the same time.*

2. *There are* i *and* j *such that*

$$(\mathbf{x}_i - \mathbf{x}_j, \mathbf{n}_j) = (\mathbf{x}_j - \mathbf{x}_i, \mathbf{n}_i) = 0 \quad (i \neq j), \tag{3.3}$$

or

$$\frac{\frac{(\mathbf{x}_2 - \mathbf{x}_1, \mathbf{n}_1)}{(\mathbf{x}_1 - \mathbf{x}_2, \mathbf{n}_2)}}{\frac{(\mathbf{x}_3 - \mathbf{x}_1, \mathbf{n}_1)}{(\mathbf{x}_1 - \mathbf{x}_3, \mathbf{n}_3)}} = \frac{(\mathbf{x}_2 - \mathbf{x}_3, \mathbf{n}_3)}{(\mathbf{x}_3 - \mathbf{x}_2, \mathbf{n}_2)}. \tag{3.4}$$

Proof: We consider the necessity of the conditions (3.3) or (3.4) first. Suppose that a quadric surface $S(f)$ smoothly contains the given points. From the Bernstein-Bezier representation of polynomial surfaces [Dah86], we have

$$\begin{aligned}
\mu_1(\mathbf{x}_2 - \mathbf{x}_1, \mathbf{n}_1) &= \mu_2(\mathbf{x}_1 - \mathbf{x}_2, \mathbf{n}_2) \\
\mu_2(\mathbf{x}_3 - \mathbf{x}_2, \mathbf{n}_2) &= \mu_3(\mathbf{x}_2 - \mathbf{x}_3, \mathbf{n}_3) \\
\mu_1(\mathbf{x}_3 - \mathbf{x}_1, \mathbf{n}_1) &= \mu_3(\mathbf{x}_1 - \mathbf{x}_3, \mathbf{n}_3),
\end{aligned} \tag{3.5}$$

where μ_i is defined by $\nabla f(\mathbf{x}_i) = \mu_i \mathbf{n}_i$ $(i = 1, 2, 3)$. If $(\mathbf{x}_i - \mathbf{x}_j, \mathbf{n}_j) = 0$ for some $i \neq j$, then the condition (3.3) is satisfied. Otherwise, none of the quantities in (3.5) is zero, and we can eliminate the μ_i's to get the condition (3.4).

To prove the sufficiency, we construct a solution to the triangular Hermite problem. For this purpose, we form a tetrahedron $[\mathbf{x}_1\mathbf{x}_2\mathbf{x}_3\mathbf{x}_4]$ using a point $\mathbf{x}_4$ that is not coplanar with the triangle $[\mathbf{x}_1\mathbf{x}_2\mathbf{x}_3]$. A solution $S(f)$ to the triangular Hermite problem is represented in Bernstein-Bezier form as follows.

$$f(\tau) = \sum_{|\lambda|=2} b_\lambda B_\lambda^2(\tau).$$

Here τ is the barycentric coordinate defined by the tetrahedron $[\mathbf{x}_1\mathbf{x}_2\mathbf{x}_3\mathbf{x}_4]$.

The fact that the surface $S(f)$ passes through the vertex $\mathbf{x}_i$ implies

$$b_{2e^i} = 0, \;\; i = 1, 2, 3.$$

To decide the remaining control points so that $S(f)$ smoothly contains the vertices of the triangle $[\mathbf{x}_1\mathbf{x}_2\mathbf{x}_3]$, we make use of the conditions (3.3) and (3.4). When (3.3) is satisfied, say, for $i = 2$ and $j = 3$, we define

$$\nabla f(\mathbf{x}_1) = \mathbf{n}_1,$$

$$\nabla f(\mathbf{x}_2) = \begin{cases} \frac{(\mathbf{x}_2 - \mathbf{x}_1, \mathbf{n}_1)}{(\mathbf{x}_1 - \mathbf{x}_2, \mathbf{n}_2)} \mathbf{n}_2 & \text{when } (\mathbf{x}_1 - \mathbf{x}_2, \mathbf{n}_2) \neq 0 \\ \mathbf{n}_2 & \text{otherwise.} \end{cases}$$

$$\nabla f(\mathbf{x}_3) = \begin{cases} \frac{(\mathbf{x}_3 - \mathbf{x}_1, \mathbf{n}_1)}{(\mathbf{x}_1 - \mathbf{x}_3, \mathbf{n}_3)} \mathbf{n}_3 & \text{when } (\mathbf{x}_1 - \mathbf{x}_3, \mathbf{n}_3) \neq 0 \\ \mathbf{n}_3 & \text{otherwise.} \end{cases}$$

Otherwise, the condition (3.4) must be satisfied, and we define

$$\nabla f(\mathbf{x}_1) = \mathbf{n}_1,$$

$$\nabla f(\mathbf{x}_2) = \frac{(\mathbf{x}_2 - \mathbf{x}_1, \mathbf{n}_1)}{(\mathbf{x}_1 - \mathbf{x}_2, \mathbf{n}_2)} \mathbf{n}_2,$$

$$\nabla f(\mathbf{x}_3) = \frac{(\mathbf{x}_3 - \mathbf{x}_1, \mathbf{n}_1)}{(\mathbf{x}_1 - \mathbf{x}_3, \mathbf{n}_3)} \mathbf{n}_3,$$

The gradients of the polynomial f defined at $\mathbf{x}_i$ $(i = 1, 2, 3)$ give rise to the control points of f

$$b_{e^i+e^j} = \frac{1}{2}(\mathbf{x}_j - \mathbf{x}_i, \nabla f(\mathbf{x}_i)), \quad (i, j = 1, ...4;\ i \neq j). \tag{3.6}$$

It is straightforward to check that the quadric surface $S(f)$ defined this way solves the triangular Hermite problem. ♣

The above proof implies that if there is a quadric surface solving the triangular Hermite problem, then there is a family of quadric surfaces solving the triangular Hermite problem. This family of quadric surfaces is indexed by the free parameter $b_{0002} = f(\mathbf{x}_4)$, which we call a *shape parameter*. The existence of a family of quadric surfaces is good for free-form surface constructions. For example, if there is a quadric surface solving the triangular Hermite problem, then most of the time we can choose a value for the shape parameter so that the quadric surface does not degenerate. The only time such a value for the shape parameter does not exist is when all the $\mathbf{n}_i$'s are perpendicular to the facet $[\mathbf{x}_1\mathbf{x}_2\mathbf{x}_3]$, in which case we can use the shape parameter to eliminate the extra plane to obtain a single plane that interpolates the given points and normals.

Collecting the conditions in Theorem 3.1 for all facets of input polyhedron, we have a condition on the prescribed normals at the vertices of the input polyhedron. This condition will be referred to as *the normal condition*. The normal condition reflects the fact that if a free-form surface is constructed by replacing each facet of a polyhedron with a quadric patch, then there is a correlation between the normals of the free-form surface at the adjacent vertices of the polyhedron. The correlation is a fundamental limitation of quadric patches as building blocks for free-form surfaces.

3.1.2 The three-sided hole problem

As an application of Theorem 3.1, we consider the following problem.

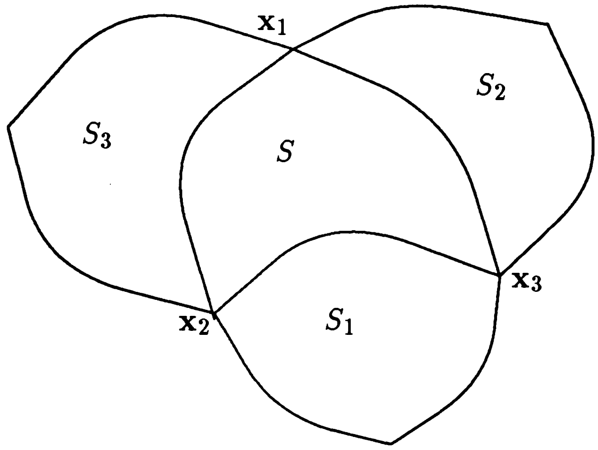


Figure 3.2: The three-sided hole problem

The three-sided hole problem

Given three surfaces S_1, S_2, and S_3 forming a three-sided hole as in Figure 3.2. At the point $\mathbf{x}_i$ ($i = 1, 2, 3$), two surfaces are tangent. Find a surface S that passes through $\mathbf{x}_1$, $\mathbf{x}_2$, and $\mathbf{x}_3$ and is tangent to S_i ($i = 1, 2, 3$) along the intersection of the surfaces S and S_i.

This problem arises if one proposes to build a free-form surface from a polyhedron by replacing the facets of the polyhedron by quadric patches, one at a time. When all three neighbors of a facet have been replaced by quadric patches, a three-sided hole is formed. The importance of this problem has also been recognized by other authors [Sed90].

Using Theorem 3.1, we can prove that the three-sided hole problem does not always have a solution if we restrict the surfaces S and S_i ($i = 1, 2, 3$) to quadric surfaces.

3.2 Free-form surface construction

This section gives an algorithm that constructs a free-form surface from the input data of an arbitrary polyhedron with a normal prescribed at each vertex of the polyhedron, and the algorithm does so by replacing each facet of the polyhedron with a quadric patch. From the discussion of Section 3.1, we know that such an algorithm is possible only if the input data satisfies the normal condition, so we assume the condition in this section. In the following, we denote the input data by $(\mathcal{P}, \mathcal{N})$, where $\mathcal{P}$ is the polyhedron and $\mathcal{N} = \{\mathbf{n}_1, \cdots, \mathbf{n}_k\}$ with $\mathbf{n}_i$ being the normal prescribed at the vertex $\mathbf{x}_i$ of $\mathcal{P}$. Without loss of generality, we assume the facets of the polyhedron $\mathcal{P}$ are triangular.

The algorithm consists of three steps: the packing step, the interpolating step, and the smoothing step. The packing step provides the bounding boxes for all the quadric patches to be used; the interpolating step replaces each facet of $\mathcal{P}$ by a quadric patch that smoothly contains the vertices of the facet; finally, the smoothing step creates smooth transitions between the adjacent quadric patches produced in the interpolating step.

3.2.1 The packing and interpolating step

The first step in the algorithm is the packing step, which provides the bounding boxes for the quadric patches to be used in the free-form surface. The union of these bounding boxes is a neighborhood Σ of the polyhedron $\mathcal{P}$, and the free-form surface to be constructed lies in Σ.

The neighborhood Σ is constructed as follows. For each facet $F = [\mathbf{x}_1\mathbf{x}_2\mathbf{x}_3]$ of the polyhedron $\mathcal{P}$, two points $\mathbf{x}_4$ and $\mathbf{y}_4$ off each side of the plane containing F are chosen, and they determine two tetrahedra, $[\mathbf{x}_1\mathbf{x}_2\mathbf{x}_3\mathbf{x}_4]$ and $[\mathbf{x}_1\mathbf{x}_2\mathbf{x}_3\mathbf{y}_4]$. These two tetrahedra form the *double tetrahedron* of the facet F, denoted as $([\mathbf{x}_1\mathbf{x}_2\mathbf{x}_3\mathbf{x}_4], [\mathbf{x}_1\mathbf{x}_2\mathbf{x}_3\mathbf{y}_4])$. Consider an adjacent facet $F' = [\mathbf{x}_1'\mathbf{x}_2\mathbf{x}_3]$ and its double tetrahedron $([\mathbf{x}_1'\mathbf{x}_2\mathbf{x}_3\mathbf{x}_4'], [\mathbf{x}_1'\mathbf{x}_2\mathbf{x}_3\mathbf{y}_4'])$. Between the double tetrahedra of facets F and F', there are two gaps. One gap is between the tetrahedra $[\mathbf{x}_1'\mathbf{x}_2\mathbf{x}_3\mathbf{x}_4']$ and $[\mathbf{x}_1\mathbf{x}_2\mathbf{x}_3\mathbf{x}_4]$; the other is between $[\mathbf{x}_1\mathbf{x}_2\mathbf{x}_3\mathbf{y}_4]$ and $[\mathbf{x}_1'\mathbf{x}_2\mathbf{x}_3\mathbf{y}_4']$. The first gap is filled with a pair of tetrahedra $[\mathbf{x}_1''\mathbf{x}_2\mathbf{x}_3\mathbf{x}_4]$ and $[\mathbf{x}_1''\mathbf{x}_2\mathbf{x}_3\mathbf{x}_4']$, and the second gap is filled

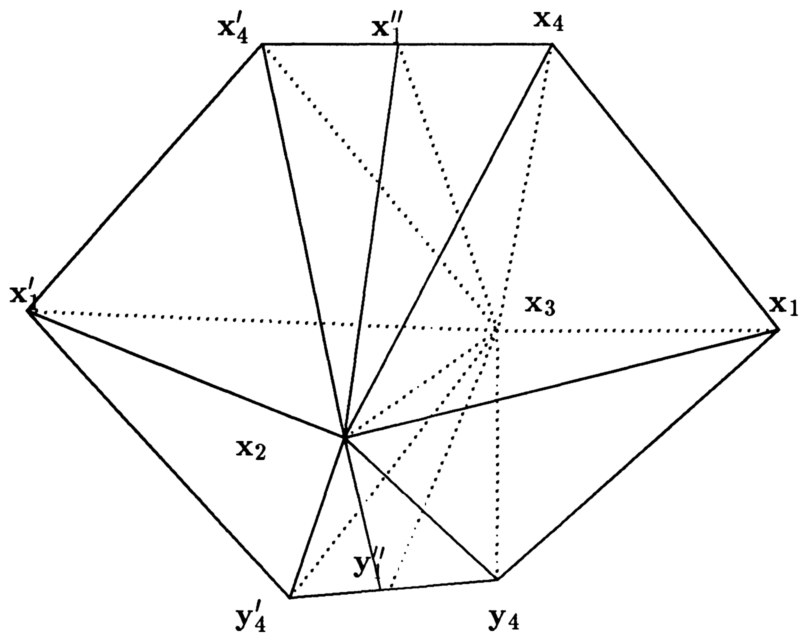


Figure 3.3: Filling gaps between two double tetrahedra

with another pair of tetrahedra $[\mathbf{y}_1''\mathbf{x}_2\mathbf{x}_3\mathbf{y}_4]$ and $[\mathbf{y}_1''\mathbf{x}_2\mathbf{x}_3\mathbf{y}_4']$. Here $\mathbf{x}_1''$ and $\mathbf{y}_1''$ are points on the line segments $[\mathbf{x}_4\mathbf{x}_4']$ and $[\mathbf{y}_4\mathbf{y}_4']$ respectively. All these are shown in Figure 3.3.

As an auxiliary geometric structure for the free-form surface construction, the neighborhood Σ must satisfy the following condition. At each vertex of the polyhedron $\mathcal{P}$, the neighborhood Σ should locally contain the tangent plane determined by the prescribed normal at the vertex. In other words, there is a disk D around the vertex $\mathbf{x}_i$ in the tangent plane at $\mathbf{x}_i$ such that

$$D \subset \Sigma.$$

The purpose of this condition is to ensure that at each vertex, the free-form surface lies inside the constructed neighborhood Σ.

The interpolating step produces a mesh of quadric patches by replacing each facet of the input polyhedron with a quadric patch that smoothly contains the vertices of the facet. For the double tetrahedron of each facet, a quadratic polynomial is created according to the constructive procedure in the proof of Theorem 3.1 and de Casteljau algorithm [Far85].

3.2.2 The smoothing step

The smoothing step provides smooth transitions between the quadric patches obtained in the interpolating step. Here smooth means tangent-plane continuity. The smoothing step is best illustrated using the two adjacent facets F and F' we mentioned in the packing step. After the interpolating step, two quadratic polynomials f and f' are defined over the double tetrahedron of F and F' respectively. The smoothing step produces quadratic polynomials over the four tetrahedra used to fill the two gaps between the two double tetrahedra so that the quadric patches defined by these polynomials and their bounding boxes provide a smooth transition between the quadric patches defined by f and f' and their bounding boxes.

Consider the problem of constructing the quadratic polynomials g and g' over tetrahedra $W = [\mathbf{x}_1''\mathbf{x}_2\mathbf{x}_3\mathbf{x}_4]$ and $W' = [\mathbf{x}_1''\mathbf{x}_2\mathbf{x}_3\mathbf{x}_4']$ which fill the gap between tetrahedra $V = [\mathbf{x}_1\mathbf{x}_2\mathbf{x}_3\mathbf{x}_4]$ and $V' = [\mathbf{x}_1'\mathbf{x}_2\mathbf{x}_3\mathbf{x}_4']$. The construction is based on the following lemma.

Lemma 3.1 *Suppose two quadric surfaces $S(f)$ and $S(f')$ are tangent at the points $\mathbf{x}_2$ and $\mathbf{x}_3$, and $\mathbf{t}$ is the unit vector along $[\mathbf{x}_2\mathbf{x}_3]$. If $(\mathbf{t}, \nabla f'(\mathbf{x}_3)) \neq 0$, then there is a constant c such that*

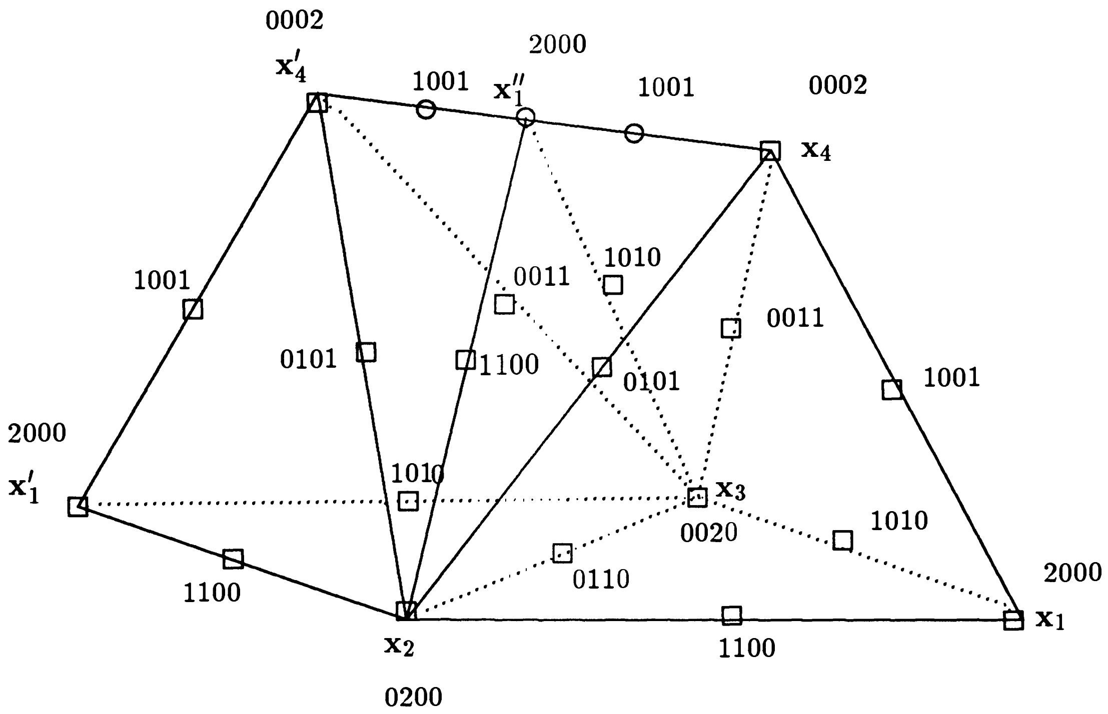
$$\nabla f(\mathbf{x}_2) = c\nabla f'(\mathbf{x}_2) \tag{3.7}$$

$$\nabla f(\mathbf{x}_3) = c\nabla f'(\mathbf{x}_3) \tag{3.8}$$

Proof: Because the surfaces $S(f)$ and $S(f')$ are tangent at the points $\mathbf{x}_2$ and $\mathbf{x}_3$, there are constants c and c' such that

$$\nabla f(\mathbf{x}_2) = c\nabla f'(\mathbf{x}_2) \tag{3.9}$$

$$\nabla f(\mathbf{x}_3) = c'\nabla f'(\mathbf{x}_3). \tag{3.10}$$



○ Control points decided by the smoothness conditions

□ Control points decided by the input data

Figure 3.4: The smoothing step

We show that $c = c'$.

Notice the restriction of $f - cf'$ to the line segment $[\mathbf{x}_2\mathbf{x}_3]$ is a univariate quadratic function satisfying

$$f(\mathbf{x}_2) - cf'(\mathbf{x}_2) = 0, \;\; f(\mathbf{x}_3) - cf'(\mathbf{x}_3) = 0. \tag{3.11}$$

From equations (3.9) and (3.11), we know the function $f - cf'$ is zero everywhere on the line segment $[\mathbf{x}_1\mathbf{x}_2]$, so

$$(\mathbf{t}, (\nabla f(\mathbf{x}_3) - c\nabla f'(\mathbf{x}_3))) = 0. \tag{3.12}$$

The equations (3.10) and (3.12) imply

$$\begin{aligned} c'(\mathbf{t}, \nabla f'(\mathbf{x}_3)) &= (\mathbf{t}, \nabla f(\mathbf{x}_3)) \\ &= c(\mathbf{t}, \nabla f'(\mathbf{x}_3)) \end{aligned}$$

Since $(\mathbf{t}, \nabla f'(\mathbf{x}_3)) \neq 0$, $c' = c$. ♣

Remark: From the proof, it is easy to see that if $(\mathbf{t}, \nabla f'(\mathbf{x}_3)) = 0$, then the surfaces $S(f)$ and $S(f')$ intersect along the line $[\mathbf{x}_2\mathbf{x}_3]$. If $\nabla f'(\mathbf{x}_2) \parallel \nabla f'(\mathbf{x}_3)$, then $S(f)$ and $S(f')$ are tangent along the line segment $[\mathbf{x}_2\mathbf{x}_3]$ as well. So, we require that $\mathbf{n}_2 \parallel \mathbf{n}_3$ when $(\mathbf{t}, \mathbf{n}_3) = 0$ in the input data.

Because of Lemma 3.1, we can assume $\nabla f(\mathbf{x}_2) = \nabla f'(\mathbf{x}_2) = \mathbf{n}_2$ and $\nabla f(\mathbf{x}_3) = \nabla f'(\mathbf{x}_3) = \mathbf{n}_3$, so the gradients of g and g' at the points $\mathbf{x}_2$ and $\mathbf{x}_3$ can be defined. This is the key to the smoothing step.

Now we construct g and g' using the C^1 conditions across the faces $[\mathbf{x}_2\mathbf{x}_3\mathbf{x}_4]$, $[\mathbf{x}_2\mathbf{x}_3\mathbf{x}_1'']$, and $[\mathbf{x}_2\mathbf{x}_3\mathbf{x}_4']$. Suppose f and f' are in Bernstein-Bezier representation as follows.

$$f(\mathbf{x}) = \sum_{|\lambda|=2} b_\lambda B_\lambda^2(\tau)$$

and

$$f'(\mathbf{x}) = \sum_{|\lambda|=2} b'_\lambda B_\lambda^2(\tau'),$$

where τ and τ' are the barycentric coordinates defined by the tetrahedra V and V'. We will construct g and g' in Bernstein-Bezier representation as well.

$$g(\mathbf{x}) = \sum_{|\lambda|=2} a_\lambda B^2_\lambda(\mu)$$

and

$$g'(\mathbf{x}) = \sum_{|\lambda|=2} a'_\lambda B^2_\lambda(\mu'),$$

where μ and μ' are the barycentric coordinates defined by the tetrahedra W and W'.

The control points of g and g' are defined as follows. The gradients at $\mathbf{x}_2$ and $\mathbf{x}_3$ determine the following control points.

$$a_{1100} = \frac{1}{2}(\mathbf{n}_2, \mathbf{x}''_1 - \mathbf{x}_2)$$

and

$$a_{1010} = \frac{1}{2}(\mathbf{n}_3, \mathbf{x}''_1 - \mathbf{x}_3).$$

If the barycentric coordinates of $\mathbf{x}''_1$ with respect to V and V' are (γ_i) and (γ'_i) respectively, then

$$a_{1001} = \gamma_1 b_{1001} + \gamma_2 b_{0101} + \gamma_3 b_{0011} + \gamma_4 b_{0002}$$

and

$$a'_{1001} = \gamma'_1 b'_{1001} + \gamma'_2 b'_{0101} + \gamma'_3 b'_{0011} + \gamma'_4 b'_{0002}.$$

Finally,

$$a_{2000} = a'_{2000} = \delta a_{1001} + \delta' a'_{1001},$$

where δ and δ' are defined by

$$\mathbf{x}''_1 = \delta \mathbf{x}_4 + \delta' \mathbf{x}'_4.$$

Similarly, we can construct the quadratic polynomials over the tetrahedra which fill the gap between the tetrahedra $[\mathbf{x}_1\mathbf{x}_2\mathbf{x}_3\mathbf{y}_4]$ and $[\mathbf{x}'_1\mathbf{x}_2\mathbf{x}_3\mathbf{y}'_4]$. The quadric patches defined by the quadratic polynomials constructed in the smoothing step provide smooth transitions between the adjacent patches produced by the interpolating step. The assembly of all the quadric patches created in the interpolating step and the smoothing step constitutes the free-form surface to be generated.

It is important to notice that the quadratic polynomials defined in the algorithm do not form a global C^1 function over Σ. In fact, they need not even be C^0. But the quadric patches defined by these quadratic polynomials form a visually smooth free-form surface because any two quadratic polynomials defined over two tetrahedra sharing a common face are C^1 across the common face after a scaling as in Lemma 3.1. This smoothness is the geometric continuity formalized in [War86]. The algorithm we derive needs the freedom given by geometric continuity. For example, when the smoothing step is performed between a quadric patch and its different neighboring patches, the quadratic polynomial defining the quadric patch may be scaled differently.

3.2.3 Avoiding topological anomalies

An important task in using implicit patches is to ensure taut and smooth patches, which are part of the zero contours of polynomials. The following situations may occur to the zero contour of a quadratic polynomial inside a tetrahedron.

1. There is no zero contour in the interior of the tetrahedron even when the zero contour is known to pass through several vertices of the tetrahedron.

2. There are self-intersection points, or singular points on the quadric patch.

3. There are multiple sheets of the zero contour inside the tetrahedron.

We use the quadratic polynomial f defined over tetrahedron $[\mathbf{x}_1\mathbf{x}_2\mathbf{x}_3\mathbf{x}_4]$ as an example to discuss the causes of these problem and to propose solutions.

Situation one can be avoided by properly choosing the function value at $\mathbf{x}_4$. Consider the line segment from the centroid of $[\mathbf{x}_1\mathbf{x}_2\mathbf{x}_3]$, $\mathbf{p}$, to $\mathbf{x}_4$. If the function value $f(\mathbf{x}_4)$ is chosen to be have a sign opposite to that of the function value at $\mathbf{p}$, then there must be a point on the line segment $[\mathbf{x}_4\mathbf{p}]$ where the quadratic polynomial is zero. In other words, the zero contour passes through a point inside the tetrahedron $[\mathbf{x}_1\mathbf{x}_2\mathbf{x}_3\mathbf{x}_4]$.

The other two situations can be avoided by enforcing monotonicity conditions on the quadratic polynomial along the direction from the centroid of $[\mathbf{x}_1\mathbf{x}_2\mathbf{x}_3]$ to $\mathbf{x}_4$. Function f is *monotone* in direction α if the directional derivative along α is positive. For the monotonicity of quadratic polynomials, we have the following theorem.

Theorem 3.2 *Let the quadratic polynomial in* $[\mathbf{x}_1\mathbf{x}_2\mathbf{x}_3\mathbf{x}_4]$ *be*

$$f(\tau) = \sum_{|\lambda|=2} b_\lambda B_\lambda^2(\tau).$$

A necessary and sufficient condition for f to be monotone along the direction α is

$$\sum_{i=1}^{4} \alpha_i b_{e^i+e^j} > 0, \quad (j = 1,2,3,4). \tag{3.13}$$

Proof: It is easy to verify that $\frac{\partial f}{\partial \alpha}$ can be written as

$$2\sum_{j=1}^{4}(\sum_{i=1}^{4} \alpha_i b_{e^i+e^j})\tau_j.$$

This is a convex combination of $\sum_{i=1}^{4} \alpha_i b_{e^i+e^j}$, hence the theorem. ♣

When α is the direction from the centroid of $[\mathbf{x}_1\mathbf{x}_2\mathbf{x}_3]$ to $\mathbf{x}_4$, the condition (13) is

$$b_{e^i+e^4} > \frac{1}{3}(b_{e^i+e^1} + b_{e^i+e^2} + b_{e^i+e^3}), \ (i = 1,2,3,4).$$

3.2.4 Some features of the algorithm

The above free-form algorithm has several attractive features. From the description of the algorithm, it is easy to see that the free-form construction in the algorithm is local. In the following, we discuss the quadratic precision of the algorithm and

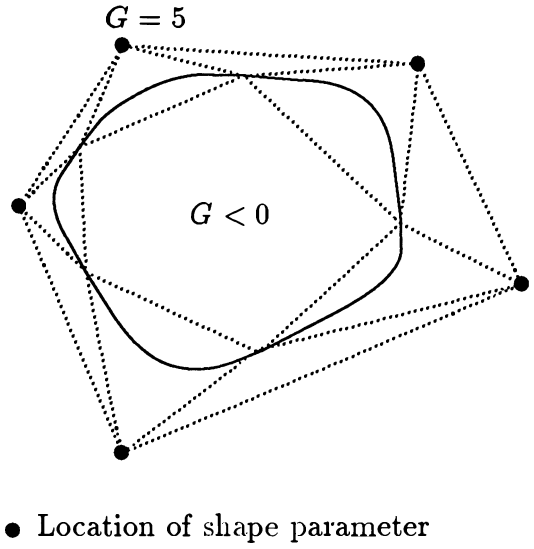


Figure 3.5: The shape control scheme

how to control the shape of the free-form surface without knowing the details of the algorithm.

Quadratic precision is a measure of accuracy in terms of how well an algorithm can reproduce a known surface if the input data is taken from the surface. The algorithm we derive reproduces all quadrics. To see this, we need to analyze the algorithm. When the input data of a polyhedron with normals prescribed at its vertices is taken from a quadric surface, the algorithm will accept the input since it automatically satisfy the normal condition. If the shape parameters are also taken from the quadric surface[2], then the interpolating step will produce the same quadric surface for all facets. Because the quadric patches produced in the smoothing step are completely determined by the quadrics produced in the interpolating step, the algorithm reproduces the quadric surface.

An important feature of the algorithm we derive is that it allows a user to control the shape of the free-form surface produced by the algorithm without knowing

[2]This can be done in many different ways, and an easy way is as follows: randomly choose enough vertices of $\mathcal{P}$ so that these vertices determine a quadratic polynomial q whose zero contour passes though the chosen vertices, then compute the the shape parameters by evaluating q.

the details of the algorithm. This feature is very important for applications like CAD/CAM, where the shapes of the free-form surfaces are manipulated to achieve functional or aesthetic objectives.

Recall that for each facet $F = [\mathbf{x}_1\mathbf{x}_2\mathbf{x}_3]$ of the polyhedron $\mathcal{P}$, the quadratic polynomial over the double tetrahedron of the facet F is not completely fixed. A double tetrahedron has a vertex outside $\mathcal{P}$, and we call the vertex *the apex* of the double tetrahedron. At the apex of the double tetrahedron of the facet F, the value of the quadratic polynomial is left as a shape parameter, as is shown in Theorem 3.1.

If we think of the algorithm as producing a global function over the constructed neighborhood Σ of $\mathcal{P}$, then the value of the global function G at the apex of each double tetrahedron within Σ is a shape parameter. Since the interior of the free-form surface is exactly the region where $G < 0$, decreasing a shape parameter at an apex pulls the free-form surface towards the apex. Moreover, only nearby quadric patches are affected by this shape parameter because the free-form surface construction in the algorithm is local. So, the apexes form a net which controls the shape of the free-form surface through the shape parameters at the node of the net.

Figure 3.5 and Figure 3.6 illustrate a two dimensional analogy of our shape control scheme. The situation in the three dimensional space is the same but harder to draw.

3.2.5 Representation of arbitrary shapes

The end goal of this work is to derive a method for representing arbitrary objects using quadric patches. A natural way of accomplishing this goal is to generate a polyhedron that describes the general shape of an object and to fit a smooth piecewise quadric surface through the vertices of the polyhedron. This section addresses the problem of creating a free-form surface from an arbitrary polyhedron

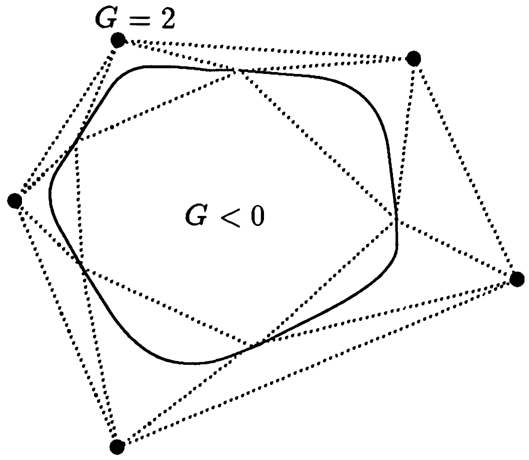


Figure 3.6: Figure 3.5 with a shape parameter decreased

$\mathcal{P}$ using the algorithm described in the last section.

A major task in using the algorithm described in the last section is to generate the normals at the vertices of the input polyhedron $\mathcal{P}$ so that free-form surfaces of pleasing shape can be created. From the discussion of Section 3.2, we know that the normals must satisfy the normal condition. But randomly chosen normals satisfying the normal condition often fail to produce free-form surfaces of pleasing shape. To obtain free-form surfaces of pleasing shape, we generate the normals according to the geometry of the input polyhedron $\mathcal{P}$ as follows. At each vertex $\mathbf{x}_i$, a normal $\mathbf{n}'_i$ is obtained by averaging the normals of the facets incident to $\mathbf{x}_i$. In general, the $\mathbf{n}'_i$'s do not satisfy the normal condition, but they can be perturbed using a least square technique [LH74] so that the perturbed normals satisfy the normal condition, and the geometric information in $\mathbf{n}'_i$ $(i = 1, ...k)$ is preserved to maximum extent.

Consider the problem of preserving the geometric information in $\mathbf{n}'_i$ $(i = 1, ...k)$. From the proof of Theorem 3.1, we know that the control point $a_{e^i+e^j}$ $(1 \leq i < j \leq 3)$ can be defined as either $\frac{1}{2}(\mathbf{n}'_i, \mathbf{x}_i - \mathbf{x}_j)$ or $\frac{1}{2}(\mathbf{n}'_j, \mathbf{x}_j - \mathbf{x}_i)$, and requiring the two quantities to be equal gives rise to the constraints in the theorem. But whether the two quantities are equal or not, the geometric information from the normals $\mathbf{n}_i'$ $(i = 1, 2, 3)$ is contained in the average of the two quantities:

$$a'_{e^i+e^j} = \frac{1}{4}((\mathbf{n}_i', \mathbf{x}_j - \mathbf{x}_i) + (\mathbf{n}_j', \mathbf{x}_i - \mathbf{x}_j)).$$

So, we can preserve the geometric information in the original input data by requiring that the newly generated normals $\mathbf{n}_i$ define almost the same control points.

It is important to notice that for two facets incident on a vertex, the two quadric patches used to replace the two facets may not have the same gradient at the vertex, but the two patches are tangent at the vertex. Again, we use geometric continuity rather than derivative continuity, and by doing so we can gain the freedom needed for creating a mesh of quadric patches without splitting the facet of $\mathcal{P}$.

3.3 Exploring the full power of geometric continuity

An important feature of the algorithm we derive is the use of geometric continuity. In this section, we study the meaning of geometric continuity for quadric patches, and we show that the algorithm we derive explores the full power of geometric continuity.

Two surfaces $S(f)$ and $S(g)$ meet with geometric continuity if there are polynomials a and b not identically zero on $S(f, g)$ such that $a\nabla f = b\nabla g$ on $S(f, g)$. In other words, two surfaces meet with geometric continuity if they are tangent along their curve of intersection. A basic result about geometric continuity is the following theorem.

Theorem 3.3 *Let $S(g)$ and $S(h)$ be distinct and irreducible polynomials. If $S(g)$ and $S(h)$ intersect transversally, then a surface $S(f)$ that meets the surface $S(g)$ with geometric continuity along the curve $S(g, h)$ must be in the ideal (g, h^2).*

Proof: See [War89]. ♣

The polynomials a and b in the definition of geometric continuity are known as *weight functions*, and they are not constant in general [War86]. In constructing a free-form surface from parametric patches, nonconstant weight functions are widely used to improve efficiency [MLL+90], so it is very natural to ask if we can also use weight functions to improve efficiency in constructing implicit free-form surfaces. We show that for quadric surfaces meeting with geometric continuity, the weight functions are nonconstant only when the intersection curve degenerates to lines. For this purpose, we need the following lemma.

Lemma 3.2 *If two quadric surfaces $S(f)$ and $S(g)$ are tangent along their curve of intersection, then $S(f,g)$ is a conic.*

Intuitively, this is because Bezout theorem says that a surface of degree m and a surface of degree n intersect in a space curve of degree mn. In the lemma, the curve is counted twice because it is a curve of tangency.

Proof: Recall that the degree of an algebraic space curve is defined as the number of intersections with a plane in generic position. Consider the intersection points of the curve $S(f,g)$ with a plane $S(P)$ in generic position. At these points, the conic $S(P,f)$ and $S(P,g)$ must be tangent because S(f) and S(g) are tangent. So these points count twice. From Bezout's theorem, the conic $S(P,f)$ and $S(P,g)$ can intersect at most four time, so $S(f,g)$ and $S(P)$ can intersect at most twice. Hence $S(f,g)$ is of degree two. ♣

Theorem 3.4 *If two irreducible quadric surfaces $S(f)$ and $S(g)$ meet with geometric continuity and $S(f,g)$ does not degenerate to lines, then the weight functions are constants.*

Proof: By Lemma 3.2, we can find a plane $S(P)$ containing the intersection $S(f,g)$. Theorem 3 implies $f = ag + bP^2$ for polynomials a and b. Since $S(P)$ and $S(g)$ intersect along the irreducible conic $S(f,g)$, we can use Bezout's Theorem to show that $S(P)$ and $S(g)$ have no common component at infinity. Then Theorem 2.3 implies that a is a constant. ♣

The algorithm described in Section 3.2 rescales the polynomials of adjacent quadric patches with constants, so the algorithm explores the full power of geometric continuity.

Chapter 4

Blending techniques in free-form surface constructions

Blending is the process of creating a smooth transition between distinct features of a physical object. Many techniques for constructing blending surfaces have been developed using implicit surfaces [Ros85,RO87,MS85,HH86,HH87,HH88]. In fact, the successful use of implicit surfaces in blending is, to a large extent, responsible for the recent emerging role of algebraic techniques in geometric modeling.

It is very natural to apply blending techniques to free-form surface constructions. A basic problem in constructing a free-form surface is the smooth transitions between adjacent patches in the free-form surface, and such smooth transitions can be achieved by blending.

This chapter applies blending techniques to free-form surface constructions. We discuss the difficulties of doing so and general strategies to overcome these difficulties. Using these strategies, we improve the performance of the free-form surface algorithm in Chapter 3. We also point out the limitations of blending techniques.

A major problem in applying blending techniques to free-form surface constructions is that the degrees of the blending surfaces are too high. For example, a surface that blends two quadric surfaces has degree four in general. In Section 4.1, we identify the causes of this problem and propose a general strategy for solving it.

As an application of the general strategy proposed in Section 4.1, Section 4.2 applies a blending technique to improve the performance of the free-form surface algorithm described in Chapter 3. In the algorithm, two quadric patches are used in the smoothing step to smooth two adjacent quadric patches produced in the interpolating step. Using blending techniques, we can replace these two quadric patches by a single quadric patch. We also show how to derive our results using techniques from algebraic geometry in Section 4.3.

Finally, Section 4.4 points out the limitations of blending techniques.

4.1 Low-degree blending surfaces

A blending surface smoothly joins two existing primary surfaces that otherwise would intersect in an edge. For two primary surfaces $S(f_1)$ and $S(f_2)$ with a curve $S(f_i, g_i)$ prescribed on surface $S(f_i)$, a blending surface is the surface $S(f)$ that is tangent to the surface $S(f_i)$ along the curve $S(f_i, g_i)$ for $i = 1, 2$.

Mathematically, finding the blending surface $S(f)$ is equivalent to computing a basis for the intersection of two ideals. From Theorem 3.3, if f_i and g_i are distinct, irreducible polynomials, the fact that $S(f)$ is tangent to the surface $S(f_i)$ along the curve $S(f_i, g_i)$ implies that the polynomial f is a member of the ideal (f_i, g_i^2). It is easy to see that if f is member of the ideal $(f_1, g_1^2) \cap (f_2, g_2^2)$ and f is not singular along $S(f_i, g_i)$, then $S(f)$ is a blending surface.

Unfortunately, finding a basis for the intersection of two ideals is not trivial. For two ideals I_1 and I_2 with bases $\{a_1, \cdots, a_m\}$ and $\{b_1, \cdots b_n\}$ respectively, most of the operations on the two ideals result in ideals whose bases are easy to compute. For examples, the basis for the ideal product $I_1 I_2$ is $\{a_i b_j : 1 \leq i \leq m,\ 1 \leq j \leq m\}$, and the basis for the ideal sum $I_1 + I_2$ is $\{a_1, \cdots, a_m, b_1, \cdots, b_n\}$. But in general, there is no direct relationship between the bases for I_1 and I_2 and the basis for $I_1 \cap I_2$. Usually, the basis of the intersection of two ideals is computed using the following lemma.

Lemma 4.1 *Let $R[t]$ be the polynomial ring in the indeterminate t over a ring R. If I_1 and I_2 are ideals in R, then*

$$(I_1(t) + I_2(1-t)) \cap R = I_1 \cap I_2 \tag{4.1}$$

Proof: See [GTZ88]. ♣

To compute a basis for $I_1 \cap I_2$, one first computes a basis for $I_1(t) + I_2(1-t)$. The fact that $I_1 = (a_1, \cdots, a_m)$ and $I_2 = (b_1, \cdots, b_n)$ implies that $\{a_1 t, \cdots, a_m t, b_1(1-t), \cdots, b_n(1-t)\}$ is a basis for $I_1(t) + I_2(1-t)$. From the above lemma, restricting a basis for $I_1(t) + I_2(1-t)$ to the subring R gives a basis for $I_1 \cap I_2$. Finding the basis for an ideal restricted to a subring can be easily solved using Groebner basis technique [Buc85]. Such computations can be done interactively with most symbolic computing systems.

Although a basis of $I_1 \cap I_2$ can be easily computed, there is no formula relating the basis of $I_1 \cap I_2$ to the bases of I_1 and I_2. So most of the blending techniques approximate the ideal intersection $(f_1, g_1^2) \cap (f_2, g_2^2)$ by the ideal product $(f_1, g_1^2)(f_2, g_2^2) = (f_1 f_2, f_1 g_2^2, f_2 g_1^2, g_1^2 g_2^2)$, a subset of $(f_1, g_1^2) \cap (f_2, g_2^2)$. In general blending problems, $S(f_1, g_1)$ and $S(f_2, g_2)$ are often disjoint, in which case the above approximation turns out to be the same as $(f_1, g_1^2) \cap (f_2, g_2^2)$ [Zar71]. But in such cases, the blending surfaces have high degrees.

The above analysis suggests the following strategy for reducing the degree of the blending surfaces. If $S(f_1, g_1)$ and $S(f_2, g_2)$ intersect, then the ideal intersection $(f_1, g_1^2) \cap (f_2, g_2^2)$ will be larger than the ideal product $(f_1, g_1^2)(f_2, g_2^2)$. In general blending problems, $S(f_i)$ and $S(g_i)$ are given, so there is nothing we can do if $S(f_1, g_1)$ and $S(f_2, g_2)$ do not intersect. But in free-form surface constructions, $S(f_1)$, $S(f_2)$, $S(g_1)$, and $S(g_2)$ are part of the constructions, and we can always make $S(f_1, g_1)$ and $S(f_2, g_2)$ intersect. This strategy will be used in the next section.

4.2 Performance improvements using blending techniques

As an application for the strategy described in the last section, we apply blending techniques to improve the performance of the algorithm described in Chapter 3.

Consider the smoothing step in the algorithm. This step provides a smooth transition between adjacent quadric patches obtained in the interpolating step in the algorithm. For each pair of adjacent quadric patches produced in the interpolating step, the smoothing step uses two quadric patches to create a smooth transition. In this section, we show that this smooth transition can be achieved with one quadric patch via blending.

Given two quadric surfaces $S(f_1)$ and $S(f_2)$ which are tangent at two vertices $\mathbf{x}_1$ and $\mathbf{x}_2$, we need to find clipping planes $S(M_i)$ $(i = 1, 2)$ and a blending surface $S(f)$ such that $S(f)$ is tangent to $S(f_i)$ along $S(f_i, M_i)$.

We require the clipping planes to pass through the edge $[\mathbf{x}_1\mathbf{x}_2]$. There are two reasons for this requirement. First, it forces the two curves $S(f_1, M_1)$ and $S(f_2, M_2)$ to intersect at the vertices $\mathbf{x}_1$ and $\mathbf{x}_2$, so the ideal intersection $(f_1, M_1^2) \cap (f_2, M_2^2)$ is larger than the ideal product $(f_1, M_1^2)(f_2, M_2^2)$. As the earlier discussions indicate, such an ideal intersection allows low-degree blending surfaces.

Another reason for requiring the clipping planes to pass through the edge $[\mathbf{x}_1\mathbf{x}_2]$ is to avoid blending corners, which is a major difficulty in applying blending techniques to free-form surface constructions. A blending surface patch is defined by a blending surface and two clipping surfaces, which are planes when the blending surface is quadric. Consider a quadric patch blending two quadric patches which are tangent at the vertices $\mathbf{x}_1$ and $\mathbf{x}_2$. If we choose the clipping planes to pass through edge $[\mathbf{x}_1\mathbf{x}_2]$, then the blending quadric patch narrows down to a point at the vertices $\mathbf{x}_1$ and $\mathbf{x}_2$. Such blending quadric patches ensure the smoothness at each vertex, so no corner blending is needed.

Now we construct the blending surface $S(f)$. The construction is based on the special structure of the intersection $S(f_1, f_2)$ resulting from the fact that $S(f_1)$ and $S(f_2)$ are tangent at $\mathbf{x}_1$ and $\mathbf{x}_2$. This special structure allows quadric blending surfaces.

Consider the intersection of $S(f_1)$ and $S(f_2)$. From Lemma 3.1, there is a constant c such that

$$\nabla f_1(\mathbf{x}_i) - c\nabla f_2(\mathbf{x}_i) = 0, \ i = 1, 2. \tag{4.2}$$

Since $\nabla(f_1 - cf_2)$ is a linear vector-valued function, the equation (4.2) implies that $\nabla(f_1 - cf_2) = 0$ over the entire line $[\mathbf{x}_1\mathbf{x}_2]$. As for the function $f_1 - cf_2$, we know that

$$f_1(\mathbf{x}_i) - cf_2(\mathbf{x}_i) = 0, \ i = 1, 2 \tag{4.3}$$

because $S(f_1)$ and $S(f_2)$ pass through $\mathbf{x}_1$ and $\mathbf{x}_2$. The equations (4.2) and (4.3) imply that for any two different planes $S(N_1)$ and $S(N_2)$ passing through $[\mathbf{x}_1\mathbf{x}_2]$, there are constants c_1, c_2, and c_3, such that

$$f_1 - cf_2 = c_1N_1^2 + c_2N_1N_2 + c_3N_3^2. \tag{4.4}$$

Depending on the values of c_1, c_2, and c_3, there are three cases.

1. There exist a constant c_4 and a plane $S(P)$ passing through $[\mathbf{x}_1\mathbf{x}_2]$ such that

$$f_1 - cf_2 = c_4P^2. \tag{4.5}$$

2. There exist two different planes $S(P)$ and $S(Q)$ passing through $[\mathbf{x}_1\mathbf{x}_2]$ such that

$$f_1 - cf_2 = \pm(P^2 + Q^2). \tag{4.6}$$

3. There exist two different planes $S(P)$ and $S(Q)$ passing through $[\mathbf{x}_1\mathbf{x}_2]$ such that

$$f_1 - cf_2 = PQ \tag{4.7}$$

passing through $[\mathbf{x}_1\mathbf{x}_2]$ is crucial to the construction of the clipping planes $S(M_1)$ and $S(M_2)$.

To derive the blending surface, we notice from equation (4.7) that M_1 and M_2 satisfy

$$M_1^2 - M_2^2 = 4aPQ = 4c(f_1 - cf_2).$$

Or, equivalently,

$$4af_1 - M_1^2 = -4acf_2 - M_2^2 \tag{4.9}$$

This equation is the key to the construction of the blending surface. If a quadric surface $S(f)$ is chosen according to

$$f = 4af_1 - M_1^2 = -4acf_2 - M_2^2, \tag{4.10}$$

then for all $\mathbf{x} \in S(f_1, M_1)$

$$f(\mathbf{x}) = 4af_1(\mathbf{x}) - M_1^2(\mathbf{x}) = 0$$

and

$$\nabla f(\mathbf{x}) = 4a\nabla f_1(\mathbf{x}) - M_1(\mathbf{x})\nabla M_1(\mathbf{x}) = 4a\nabla f_1(\mathbf{x}).$$

In other words, the surface $S(f)$ is tangent to $S(f_1)$ along $S(f_1, M_1)$. Similarly, the surface $S(f)$ is tangent to $S(f_2)$ along $S(f_2, M_2)$, so $S(f)$ is the required quadric blending surface.

Therefore, we have produced the blending surface $S(f)$ and the clipping planes $S(M_1)$ and $S(M_2)$, which trim $S(f)$ into a two-sided quadric patch smoothing $S(f_1)$ and $S(f_2)$. The shape of the quadric patch can be controlled through the parameter a in the expression of M_i. The blending quadric patch shrinks as a decreases and expands as a increases.

The general theory for blending two quadric surfaces with a quadric surface was developed in [War87], but the theory has two drawbacks. First, it generates blending surfaces in the complex space as opposed to real space. Second, there is no known procedure for computing the quadric blending surface and its clipping surfaces. The quadric blending surface we derive is obviously in real space. We can also give a procedure for computing the quadric blending surface and its clipping surfaces. In previous work, the main difficult in computing the blending surface $S(f)$ is the computation of the constant c in the equation (4.7). Using Lemma 3.1, we can easily compute the constant c from $\nabla f_1(\mathbf{x}_1) = c\nabla f_2(\mathbf{x}_1)$.

4.3 An algebraic approach to blending

In this section, we show how to derive our results using techniques from algebraic geometry. By doing so we hope to demonstrate the power of algebraic techniques and to reveal the mathematical nature of the problem under consideration.

To derive the blending surface and its clipping surfaces, we analyze the intersection $S(f_1, f_2)$[1]. We show that each irreducible component C of the intersection is a conic. Choose a point $\mathbf{p} \neq \mathbf{x}_i$ $(i = 1, 2)$ on the curve C, and denote by $S(P)$ the plane passing through points $\mathbf{x}_1$, $\mathbf{x}_2$, and $\mathbf{p}$. The conics $S(P, f_1)$ and $S(P, f_2)$ are tangent at $\mathbf{x}_1$ and $\mathbf{x}_2$ since $S(f_1)$ and $S(f_2)$ are. This makes $\mathbf{x}_1$ and $\mathbf{x}_2$ count twice as intersection points, so $S(P, f_1)$ and $S(P, f_2)$ intersect at five points. By Bezout's theorem, both conics $S(P, f_1)$ and $S(P, f_2)$ must be identical to C.

If the surfaces $S(f_1)$ and $S(f_2)$ are not tangent along C, then one of $S(f_i)$, say, $S(f_2)$ intersects with $S(P)$ transversally. From Theorem 2.2, we know that

$$f_1 = cf_2 + QP \tag{4.11}$$

with $c, Q \in \mathbf{C}[\mathbf{x}]$. Here $\mathbf{C}$ is the complex field.

The polynomial c is actually a constant. To see this, we use Bezout's Theorem to show that the intersection $S(f_2, P)$ has no common component at infinity; then by Theorem 2.3, c must be a constant and $deg(Q) \leq 1$. Bezout's Theorem states that the intersection of surfaces $S(g_i)$ in projective space must satisfy the following

[1]We assume the intersection is proper, and it does not degenerate into lines, even through these exceptional cases can be treated

property. If $\{S(g_i)\}$ intersect in components $C_1, \cdots, C_k$ whose multiplicities are $d_1, \cdots, d_k$ respectively, then

$$\sum d_j deg(C_j) = \prod deg(g_i).$$

According to this relation, the fact that $S(f_2)$ and $S(P)$ intersect in a conic implies that they cannot have common components at infinity.

We prove that $S(Q)$ is in fact a plane passing through $[\mathbf{x}_1\mathbf{x}_2]$. From Lemma 3.1, there is a constant c' such that

$$\nabla f_1(\mathbf{x}_1) = c' \nabla f_2(\mathbf{x}_1)$$

and

$$\nabla f_1(\mathbf{x}_2) = c' \nabla f_2(\mathbf{x}_2).$$

From the equation (4.11) and the fact that $P(\mathbf{x}_i) = 0$ $(i = 1, 2)$,

$$\nabla f_1(\mathbf{x}_i) - c\nabla f_2(\mathbf{x}_2) = Q(\mathbf{x}_i)\nabla P(\mathbf{x}_i) \ (i = 1, 2).$$

So we have

$$(c' - c)\nabla f_2(\mathbf{x}_i) = Q(\mathbf{x}_i)\nabla P(\mathbf{x}_i) \ (i = 1, 2).$$

Since $\nabla P(\mathbf{x}_i)$ is the normal of the plane $S(P)$ passing through $[\mathbf{x}_1\mathbf{x}_2]$,

$$(c' - c)\mathbf{t} \cdot \nabla f_2(\mathbf{x}_i) = Q(\mathbf{x}_i)\mathbf{t} \cdot \nabla P(\mathbf{x}_i) = 0 \ (i = 1, 2),$$

where $\mathbf{t}$ is the unit vector along $[\mathbf{x}_1\mathbf{x}_2]$. Since $\mathbf{t} \cdot \nabla f_2(\mathbf{x}_i) \neq 0$ $(i = 1, 2)$, we conclude that $c' = c$ and $Q(\mathbf{x}_i) = 0$ $(i = 1, 2)$, which imply that $S(Q)$ must pass through $[\mathbf{x}_1\mathbf{x}_2]$.

Using equation (4.11) and the fact that the plane $S(Q)$ passes through $[\mathbf{x}_1\mathbf{x}_2]$, we can derive the clipping surfaces and the blending surfaces as in the previous section. The only problem is that we need to show that both c and Q are real. Techniques for doing so are of general importance because most of the algebraic techniques are developed for complex space while the final results only make sense in real space.

Theorem 4.1 *Let f, $g_1, \cdots, g_k$ be real polynomials. If $f \in (g_1, \cdots, g_k)$ over* $\mathbf{C}[\mathbf{x}]$, *then $f \in (g_1, \cdots, g_k)$ over* $\mathbf{R}[\mathbf{x}]$.

Proof: See [War86]. ♣

4.4 A limitation of the blending techniques

Blending techniques are very useful in free-form surface constructions, but they do have limitations. A blending surface provides a smooth transition between two primary surfaces that would otherwise join in an edge. However, because of the highly complex input data for free-form surface constructions, the two primary surfaces may not join in an edge in the first place, or they join in an edge outside the primary region of interest. In such cases, the blending techniques are bound to fail.

The above arguments can be best illustrated by an example. Consider two adjacent facets $[\mathbf{x}_1\mathbf{x}_2\mathbf{x}_3]$ and $[\mathbf{x}_1\mathbf{x}_2\mathbf{x}'_3]$ with two quadrics $S(f_1)$ and $S(f_2)$ produced for each of the facets by the interpolating step. Here

$$\mathbf{x}_1 = (1,0,0),\ \mathbf{x}_2 = (-1,0,0),\ \mathbf{x}_3 = (-1,0,1),\ \mathbf{x}'_3 = (-1,0,-1),$$

$$f_1 = x^2 + y^2 - 1, \text{ and } f_2 = x^2 + 4y^2 - 1.$$

Surfaces $S(f_1)$ and $S(f_2)$ are tangent to each other, but there is no smooth transition between the two quadrics in the primary area of interest , i.e. somewhere around the line $[\mathbf{x}_1\mathbf{x}_2]$. In this case, all blending techniques fail because they predict that no blending is necessary.

Chapter 5

Smooth piecewise cubic surfaces

Given a polyhedron with arbitrarily prescribed normals at the vertices of the polyhedron, we would like to design an algorithm which constructs a free-form surface that smoothly contains the vertices of the polyhedron. As was mentioned in Section 1.2, allowing arbitrarily prescribed normals is very important. It enables us to control the implicit patches in the free-form surface so that only a few implicit patches are needed for representing an object that would otherwise requires a polygon with thousands of facets to approximate [FD90]. In addition, allowing arbitrarily prescribed normals is necessary for applications where the prescribed normals determine physical quantities being modeled.

The algorithm we derived in Chapter 3 does not allow arbitrarily prescribed normals because of the fundamental limitations of quadric patches: when a free-form surface is constructed by replacing each facet of the input polyhedron with a quadric patch, a correlation is introduced between the normals at the adjacent vertices of the input polyhedron.

One way to overcome the limitations of quadric patches is to split the edges of the input polyhedron, and an algorithm for doing so was given in [Dah89]. However, the splitting approach has many drawbacks. Most importantly, it causes undesirable oscillations which result in free-form surfaces having nonpleasing shapes.

In this chapter, we take a different approach to overcome the limitations of quadric patches: we construct free-form surfaces using cubic patches. Given a polyhedron with arbitrarily prescribed normals at its vertices, we derive an algorithm for constructing a free-form surface that smoothly contains the vertices of the polyhedron. The algorithm is local, and it has quadratic precision. The shapes of the free-form surfaces produced by the algorithm can be controlled through a set of intuitive shape parameters without knowing the details of the algorithm.

5.1 Conceptual Overview

The algorithm described in this chapter builds free-form surfaces from the input data of a polyhedron with a normal vector prescribed at each vertex of the polyhedron. The input data is denoted by $(\mathcal{P}, \mathcal{N})$, where $\mathcal{P}$ is a arbitrary polyhedron with vertex set $\{\mathbf{x}_1, \cdots, \mathbf{x}_k\}$, and $\mathcal{N}$ is a set of normals $\{\mathbf{n}_1, \cdots, \mathbf{n}_k\}$ with $\mathbf{n}_i$ being the normal prescribed at $\mathbf{x}_i$.

The basic idea of the algorithm is very simple. The free-form surface to be built must be in the neighborhood of the input polyhedron $\mathcal{P}$, so we construct a neighborhood Σ of $\mathcal{P}$ using tetrahedra and create a cubic polynomial for each tetrahedron used. By ensuring C^1 conditions between adjacent tetrahedra, we construct a global C^1 function whose zero contour within the neighborhood Σ is the free-form surface to be generated.

The following three aspects are crucial to the success of the algorithm.

1. The construction of a neighborhood Σ of the input polyhedron using tetrahedra. The neighborhood must locally contain the tangent plane determined by the prescribed normal at each vertex of the input polyhedron $\mathcal{P}$, and the neighborhood must have the same topology as the polyhedron $\mathcal{P}$.

2. A scheme for defining a globally C^1 function which is a cubic polynomial over each tetrahedron within the neighborhood Σ. The scheme must leave

free control points in the definition of each cubic polynomial so that the zero contour of the cubic polynomial can be controlled by them.

3. A mechanism to control the cubic polynomial defined for each tetrahedron so that the zero contour of the cubic polynomial inside the tetrahedron is a single-sheeted cubic patch without holes, extraneous sheets, self intersections, or other topological anomalies.

The first aspect was covered in Chapter 3, where we described how to construct a neighborhood Σ of the input polyhedron $\mathcal{P}$. In particular, we placed a requirement on Σ: at each vertex of the polyhedron $\mathcal{P}$, the neighborhood should locally contain the tangent plane defined by the prescribed normal at the vertex.

In Section 5.2 and Section 5.3, we cover the second and the third aspects, and in Section 5.4, we explain some features of the algorithm.

5.2 A scheme for enforcing C^1 conditions over the neighborhood Σ

Having built a neighborhood Σ of the polyhedron $\mathcal{P}$, we construct a C^1 function f over the neighborhood Σ so that

$$f(\mathbf{x}_i) = 0, \ \ \nabla f(\mathbf{x}_i) = \mathbf{n}_i, \ \ i = 1, \cdots, k.$$

The zero contour of f within Σ is the free-form surface to be generated.

The construction can be outlined as follows. We split each facet of $\mathcal{P}$ into three subfacets, and we split the double tetrahedron of the facet accordingly. The neighborhood Σ is kept the same except some of its tetrahedra are split into three tetrahedra. After splitting, the function f is defined by constructing a cubic polynomial for each tetrahedron within the neighborhood Σ.

To show the splitting scheme, we take a facet $[\mathbf{x}_1\mathbf{x}_2\mathbf{x}_3]$ and its double tetrahedron $([\mathbf{x}_1\mathbf{x}_2\mathbf{x}_3\mathbf{x}_4], [\mathbf{x}_1\mathbf{x}_2\mathbf{x}_3\mathbf{y}_4])$ as an example. Let $\mathbf{w}$ be the intersection point of

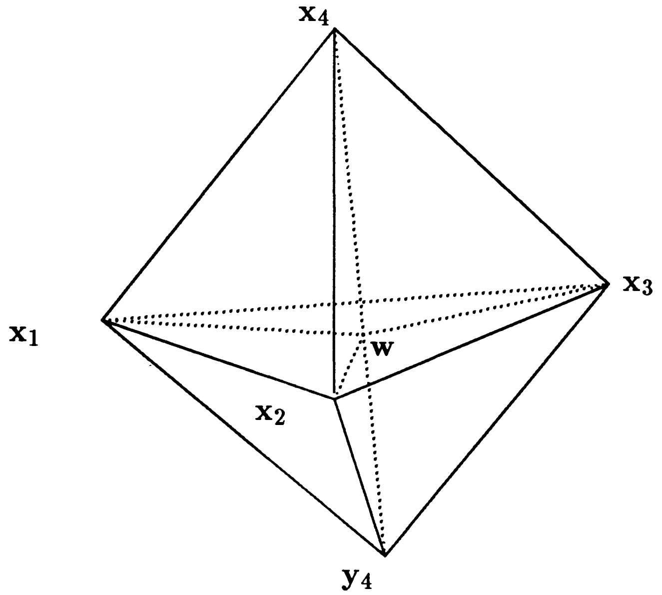


Figure 5.1: Splitting a double tetrahedron

line segment $[\mathbf{y}_4\mathbf{x}_4]$ and facet $[\mathbf{x}_1\mathbf{x}_2\mathbf{x}_3]$. We split the double tetrahedron into six tetrahedra: $[\mathbf{x}_i\mathbf{x}_j\mathbf{w}\mathbf{x}_4]$ and $[\mathbf{x}_i\mathbf{x}_j\mathbf{w}\mathbf{y}_4]$ for $1 \leq i < j \leq 3$. For symmetry and robustness reasons, $\mathbf{w}$ is often chosen to be the centroid of the triangle $[\mathbf{x}_1\mathbf{x}_2\mathbf{x}_3]$, and $\mathbf{y}_4$ and $\mathbf{x}_4$ are chosen to be on the line that passes through $\mathbf{w}$ and is perpendicular to $[\mathbf{x}_1\mathbf{x}_2\mathbf{x}_3]$.

The construction of cubic polynomials over the tetrahedra within Σ takes two steps. Consider a facet $F' = [\mathbf{x}_1'\mathbf{x}_2\mathbf{x}_3]$ adjacent to facet $F = [\mathbf{x}_1\mathbf{x}_2\mathbf{x}_3]$ and the double tetrahedron of F', $([\mathbf{x}_1'\mathbf{x}_2\mathbf{x}_3\mathbf{x}_4'], [\mathbf{x}_1'\mathbf{x}_2\mathbf{x}_3\mathbf{y}_4'])$. Facet F' and its double tetrahedron are split at the centroid $\mathbf{w}'$ of F' in the same way $([\mathbf{x}_1\mathbf{x}_2\mathbf{x}_3\mathbf{x}_4], [\mathbf{x}_1\mathbf{x}_2\mathbf{x}_3\mathbf{y}_4])$ is split at $\mathbf{w}$. For facet F and F', the first step takes place over the tetrahedra $V_1 = [\mathbf{x}_2\mathbf{x}_3\mathbf{x}_4\mathbf{w}]$, $V_2 = [\mathbf{x}_2\mathbf{x}_3\mathbf{x}_4'\mathbf{w}']$, $W_1 = [\mathbf{x}_2\mathbf{x}_3\mathbf{x}_1''\mathbf{x}_4]$, $W_2 = [\mathbf{x}_2\mathbf{x}_3\mathbf{x}_1''\mathbf{x}_4']$, $V_1' = [\mathbf{x}_2\mathbf{x}_3\mathbf{y}_4\mathbf{w}]$, $V_2' = [\mathbf{x}_2\mathbf{x}_3\mathbf{y}_4'\mathbf{w}']$, $W_1' = [\mathbf{x}_2\mathbf{x}_3\mathbf{y}_1''\mathbf{y}_4]$, and $W_2' = [\mathbf{x}_2\mathbf{x}_3\mathbf{y}_1''\mathbf{y}_4']$ as in Figure 5.2. We construct the cubic polynomials over the tetrahedra W_1, W_2, W_1', and W_2'. At the same time, the cubic polynomials over the tetrahedra V_1, V_2, V_1', and V_2' are partially determined through C^1 conditions. The same process is carried out between every pair of adjacent facets in $\mathcal{P}$, so at the end of the first step, the cubic polynomials over tetrahedra $[\mathbf{x}_i\mathbf{x}_j\mathbf{x}_4\mathbf{w}]$ and $[\mathbf{x}_i\mathbf{x}_j\mathbf{y}_4\mathbf{w}]$ are partially constructed for all $i < j \leq 3$. Then, the second step completes the construction of these cubic polynomials according to C^1 conditions.

Now we describe the first step in detail. Throughout this description, we assume $i = 1, 2$ whenever i appears. By doing so we are taking advantage of the symmetry of the problem in consideration.

Let the cubic polynomials f_i over V_i, f_i' over V_i', g_i over W_i, and g_i' over W_i' be expressed in Bernstein-Bezier forms as follows.

$$f_i(x) = \sum_{|\lambda|=3} a^i_\lambda B^3_\lambda(\tau_i), \tag{5.1}$$

$$g_i(x) = \sum_{|\lambda|=3} b^i_\lambda B^3_\lambda(\rho_i), \tag{5.2}$$

$$f_i'(x) = \sum_{|\lambda|=3} c^i_\lambda B^3_\lambda(\tau_i'), \tag{5.3}$$

and

$$g_i'(x) = \sum_{|\lambda|=3} d^i_\lambda B^3_\lambda(\rho_i'), \tag{5.4}$$

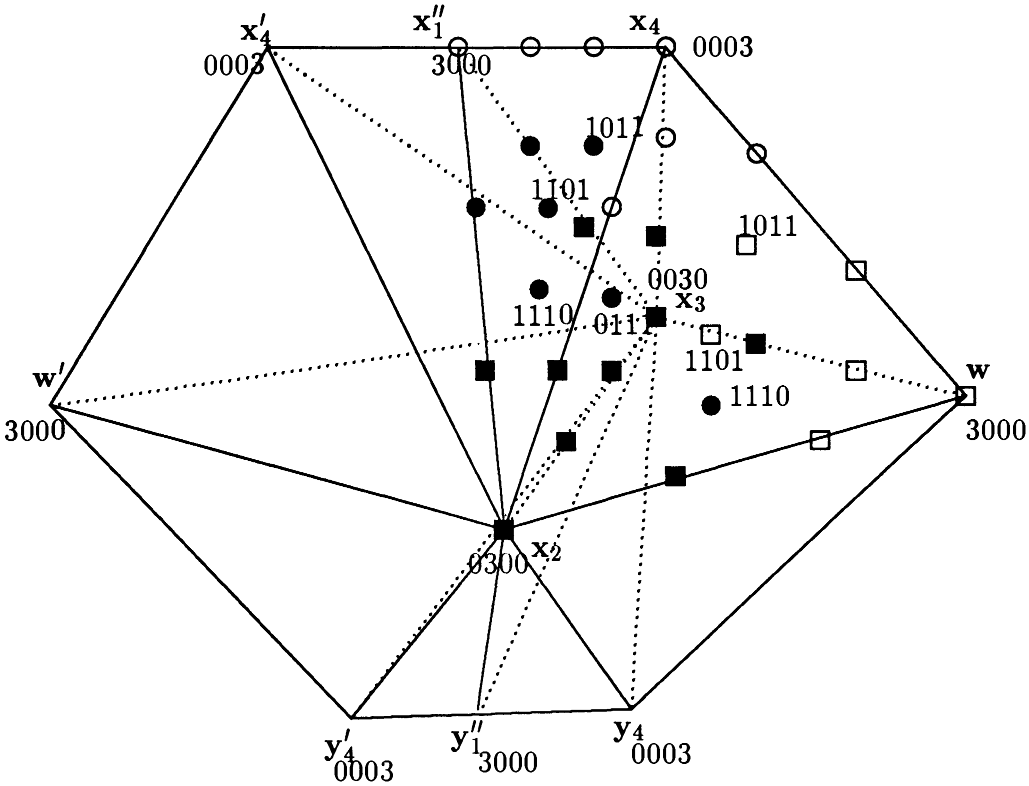
where τ_i, τ_i', ρ_i' and ρ_i are the barycentric coordinates on V_i, V_i', W_i', and W_i respectively. Our task is to determine the control points a_λ's, b_λ's, c_λ's, and d_λ's.

For notational convenience, if two tetrahedra share a common face, we equate the control points of the associated cubic polynomials on the common face to ensure C^0 continuity. Hence such control points will be defined only once.

All the control points over tetrahedra V_i, V_i', W_i, and W_i' that can be determined from the input data are as follows. The fact that the zero contours of f_i, f_i', g_i, and g_i' pass through $\mathbf{x}_2$ and $\mathbf{x}_3$ implies

$$a^i_{0300} = a^i_{0030} = 0,$$

$$c^i_{0300} = c^i_{0030} = 0,$$



□ control points to be decided

○ control points that are free or are determined by free control points

■ control points from the input data

● control points from C^1 conditions

Figure 5.2: The C^1 conditions between two adjacent double tetrahedra

$$b^i_{0300} = b^i_{0030} = 0,$$

and

$$d^i_{0300} = d^i_{0030} = 0.$$

More control points are determined by the normals at the vertices $\mathbf{x}_2$ and $\mathbf{x}_3$. For example,

$$a^1_{2e^j+e^1} = \frac{1}{3}(\mathbf{n}_j, \mathbf{w} - \mathbf{x}_j),\ j = 2, 3.$$

Similar expressions are used to determine the control points $a^i_{2e^j+e^k}$ for $j = 2, 3$ and $k = 1, 4$, $c^i_{e^j+e^4}$ for $j = 2, 3$, $b^i_{2e^j+e^1}$ for $j = 2, 3$, and $d^i_{2e^j+e^1}$ for $j = 2, 3$.

Before determining the rest of the control points according to C^1 conditions, we have to choose some control points to be *free control points* whose value will be left unspecified at this point. This is because creating a piecewise cubic C^1 function f over the neighborhood Σ is only an intermediate step in the free-form surface construction. The zero contour of a cubic polynomial inside a tetrahedron is not necessarily a "nice" cubic patch. There may not be a cubic patch inside the tetrahedron at all, or even if there is, the cubic patch can have self intersections, holes, and extra sheets. More dramatically, a handle can appear on an otherwise nice cubic patch. If such a cubic patch is used in a free-form surface constructed from the input polyhedron $\mathcal{P}$, the topology of the free-form surface is bound to be different from that of $\mathcal{P}$.

We choose the following free control points $a^i_{2e^4+e^j}$ $(j = 1, 2, 3, 4)$, $c^i_{2e^4+e^j}$ $(j = 1, 2, 3, 4)$, b^i_{1002}, and d^i_{1002} for f_i, f'_i, g_i, and g'_i respectively. The intuitive meaning of these control points are as follows. The control points $a^i_{2e^4+e^j}$ $(j = 1, 2, 3, 4)$ are equivalent to the function values and gradients at $\mathbf{x}_4$ and $\mathbf{x}'_4$, and the control points b^i_{1002} enable us to have complete control of the function values of g_i along line segment $[\mathbf{x}_4\mathbf{x}'_4]$. The same statement can be made about the control points $c^i_{2e^4+e^j}$ $(j = 1, 2, 3, 4)$ and d^i_{1002}. In 5.3, we will explain how these free control points affect the associated cubic patches.

Now we determine the rest of the control points to ensure C^1 conditions. Consider the C^1 conditions across faces $[\mathbf{x}_2\mathbf{x}_3\mathbf{x}_4]$ and $[\mathbf{x}_2\mathbf{x}_3\mathbf{x}_4']$. Suppose

$$\mathbf{x}_1'' = \beta_1^1\mathbf{x}_1 + \beta_2^1\mathbf{x}_2 + \beta_3^1\mathbf{x}_3 + \beta_4^1\mathbf{x}_4$$

and

$$\mathbf{x}_1'' = \beta_1^2\mathbf{x}_1' + \beta_2^2\mathbf{x}_2 + \beta_3^2\mathbf{x}_3 + \beta_4^2\mathbf{x}_4'.$$

Then, the C^1 conditions are the following.

$$b_{1002}^i = \beta_1^i a_{1002}^i + \beta_2^i a_{0102}^i + \beta_3^i a_{0012}^i + \beta_2^i a_{0003}^i,$$

$$b_{1101}^i = \beta_1^i a_{1101}^i + \beta_2^i a_{0201}^i + \beta_3^i a_{0111}^i + \beta_2^i a_{0102}^i, \tag{5.5}$$

$$b_{1011}^i = \beta_1^i a_{1011}^i + \beta_2^i a_{0111}^i + \beta_3^i a_{0021}^i + \beta_2^i a_{0012}^i, \tag{5.6}$$

and

$$b_{1110}^i = \beta_1^i a_{1110}^i + \beta_2^i a_{0210}^i + \beta_3^i a_{0120}^i + \beta_2^i a_{0111}^i. \tag{5.7}$$

The first three equations can be viewed as definitions for control points b_{1101}^i, b_{1011}^i, and b_{1110}^i, leaving a_{1011}^i and a_{1101}^i to be determined. Equation (5.7) will be treated later.

Moving on to the C^1 conditions across $[\mathbf{x}_2\mathbf{x}_3\mathbf{x}_1'']$, we see that if

$$\mathbf{x}_1'' = \mu_1\mathbf{x}_4 + \mu_2\mathbf{x}_4',$$

then the C^1 conditions are the following.

$$b_{3000}^i = \mu_1 b_{2001}^1 + \mu_2 b_{2001}^2, \tag{5.8}$$

$$b_{1020}^i = \mu_1 b_{1101}^1 + \mu_2 b_{1101}^2, \tag{5.9}$$

$$b_{1200}^i = \mu_1 b_{1011}^1 + \mu_2 b_{1011}^2, \tag{5.10}$$

and

$$b_{1110}^i = \mu_1 a_{0111}^1 + \mu_2 a_{0111}^2 \tag{5.11}$$

Again, the first three equations can be viewed as definitions for control points b^i_{3000}, b^i_{1020}, and b^i_{1200}; and the last equation will be treated later. Notice that b^i_{1011} and b^i_{1101} in the above equations are defined earlier by the equations (5.5) and (5.6).

Finally, we consider the C^1 conditions across faces $[\mathbf{x}_2\mathbf{x}_3\mathbf{y}_4]$, $[\mathbf{x}_2\mathbf{x}_3\mathbf{y}''_1]$, and $[\mathbf{x}_2\mathbf{x}_3\mathbf{y}'_4]$. All the control points of g'_i and some of the control points of f'_i can be fixed as f_i and g_i. In doing so, we also have two equations left untreated,

$$d^i_{1110} = \gamma^i_1 a^i_{1110} + \gamma^i_2 a^i_{0210} + \gamma^i_3 a^i_{0120} + \gamma^i_2 c^i_{0111} \tag{5.12}$$

and

$$d^i_{1110} = \eta_1 c^1_{0111} + \eta_2 c^2_{0111}, \tag{5.13}$$

where the coefficients η's and γ's come from the following relations:

$$\mathbf{y}''_1 = \gamma^2_1 \mathbf{x}'_1 + \gamma^2_2 \mathbf{x}_2 + \gamma^2_3 \mathbf{x}_3 + \gamma^2_4 \mathbf{y}'_4$$

and

$$\mathbf{y}''_1 = \eta_1 \mathbf{y}_4 + \eta_2 \mathbf{y}'_4.$$

Now we collectively treat the equations (5.7), (5.11), (5.12), and (5.13) as promised. These equations can be rewritten as

$$\eta_1 c^1_{0111} + \eta_2 c^2_{0111} = \gamma^i_1 a^i_{1110} + \gamma^i_2 a^i_{0210} + \gamma^i_3 a^i_{0120} + \gamma^i_2 c^i_{0111} \tag{5.14}$$

and

$$\mu_1 a^1_{0111} + \mu_2 a^2_{0111} = \beta^i_1 a^i_{1110} + \beta^i_2 a^i_{0210} + \beta^i_3 a^i_{0120} + \beta^i_2 a^i_{0111} \tag{5.15}$$

Here c^i_{0111} can be determined from a^i's through the C^1 conditions across $[\mathbf{x}_1\mathbf{x}_2\mathbf{x}_3]$ and $[\mathbf{x}'_1\mathbf{x}_2\mathbf{x}_3]$,

$$c^i_{0111} = \alpha^i_1 a^i_{1110} + \alpha^i_2 a^i_{0210} + \alpha^i_3 a^i_{0120} + \alpha^i_1 a^i_{0111}, \tag{5.16}$$

where the α's come from

$$\mathbf{y}_4 = \alpha^1_1 \mathbf{x}_1 + \alpha^1_2 \mathbf{x}_2 + \alpha^1_3 \mathbf{x}_3 + \alpha^1_4 \mathbf{x}_4$$

and

$$\mathbf{y}'_4 = \alpha_1^2\mathbf{x}'_1 + \alpha_2^2\mathbf{x}_2 + \alpha_3^2\mathbf{x}_3 + \alpha_4^2\mathbf{x}'_4.$$

The equations (5.14), (5.15), and (5.16) form a system of six linear equations with six unknowns, a^i_{0111}, a^i_{1110}, and c^i_{0111}. When $\mathbf{x}_4$, $\mathbf{x}'_4$, $\mathbf{y}_4$, and $\mathbf{y}'_4$ are in general position, this system always has a solution, from which d^i_{1110} and b^i_{1110} can be defined using equations (5.11) and (5.13) respectively.

This finishes the first step in the construction of the cubic polynomials over the tetrahedra within the neighborhood Σ. Now g_i and g'_i are completely constructed, while f_i and f'_i are partially constructed.

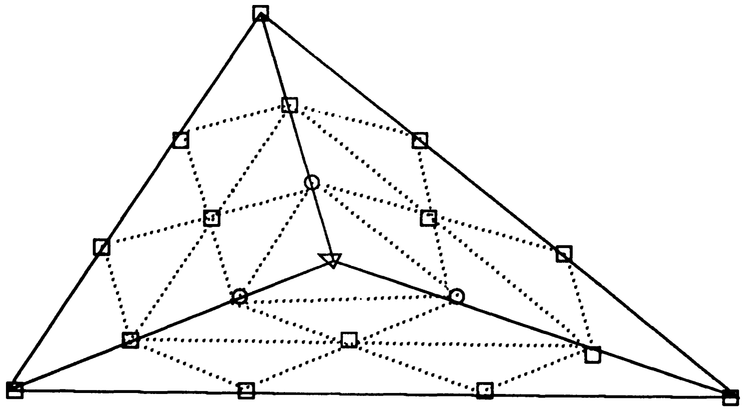
The second step completes the construction of f_i and f'_i. For this purpose, we shift our focus to the double tetrahedron $([\mathbf{x}_1\mathbf{x}_2\mathbf{x}_3\mathbf{x}_4], [\mathbf{x}_1\mathbf{x}_2\mathbf{x}_3\mathbf{y}_4])$, which has been split into tetrahedra $[\mathbf{x}_1\mathbf{x}_2\mathbf{x}_4\mathbf{w}]$, $[\mathbf{x}_1\mathbf{x}_3\mathbf{x}_4\mathbf{w}]$, $[\mathbf{x}_3\mathbf{x}_2\mathbf{x}_4\mathbf{w}]$, $[\mathbf{x}_3\mathbf{x}_2\mathbf{y}_4\mathbf{w}]$, $[\mathbf{x}_3\mathbf{x}_2\mathbf{y}_4\mathbf{w}]$, and $[\mathbf{x}_3\mathbf{x}_2\mathbf{y}_4\mathbf{w}]$.

Consider the problem of completing the construction of the partially constructed cubic polynomials for tetrahedra $U_1 = [\mathbf{x}_2\mathbf{x}_3\mathbf{x}_4\mathbf{w}]$, $U_2 = [\mathbf{x}_1\mathbf{x}_3\mathbf{x}_4\mathbf{w}]$, and $U_3 = [\mathbf{x}_1\mathbf{x}_2\mathbf{x}_4\mathbf{w}]$. Denote the cubic polynomials for U_i by

$$h_i(\nu_i) = \sum_{|\lambda|=3} a^i_\lambda B^3_\lambda(\nu_i),$$

where ν_i is the barycentric coordinates of U_i. It is easy to recognize that the polynomial h_1 is the same as f_1 in the first step. More generally, the partially constructed function h_i is the result of carrying out the first step for $[\mathbf{x}_1\mathbf{x}_2\mathbf{x}_3]$ and the facet sharing edge $[\mathbf{x}_m\mathbf{x}_n]$ $(m \neq i, n \neq i, 1 \leq m, n \leq 3)$ with $[\mathbf{x}_1\mathbf{x}_2\mathbf{x}_3]$. We denote f_1, V_1, and τ_1 by h_1, U_1, and ν_1 in the second step to reflect the new symmetry.

The task of ensuring C^1 conditions between U_i's is greatly simplified by taking advantage of the fact that $\mathbf{w} \in [\mathbf{x}_1\mathbf{x}_2\mathbf{x}_3]$. The control points over U_i can be divided into four groups. The i-th group, called i-th *layer*, is the set of $a^i_{\lambda_1\lambda_2\lambda_3\lambda_4}$ such that $\lambda_4 = i$. Because $\mathbf{w} \in [\mathbf{x}_1\mathbf{x}_2\mathbf{x}_3]$, the C^1 conditions between U_i and U_j only



o centrod of surrounding boxes
□ constructed from previous step
▽ centroid of surrounding circles

Figure 5.3: The Clough-Tocher bivariate splines

involve control points from the same layer. So we can satisfy the C^1 conditions by examining each layer as if we were working on a bivariate polynomial.

For the 0-th layer, the control points $a^i_{\lambda_1\lambda_2\lambda_3 0}$ are defined previously for all $\lambda_1 \leq 1$. Determining the rest of the control points in this layer is exactly the famous Clough-Tocher interpolation in finite element analysis. Figure 5.3 illustrates a standard solution [Far85].

For the 1st layer, the control points $a^i_{\lambda_1\lambda_2\lambda_3 1}$ are defined earlier for all $\lambda_1 = 0$. Since this layer can be viewed as a bivariate quadratic function, the known control points uniquely determine the rest of the control points within the layer through the C^1 conditions [Far85].

The control points in the 2nd and 3rd layers are trivially determined by the the function value and gradient at $\mathbf{x}_4$.

To complete the second step, we carry out the same argument for tetrahedra $[\mathbf{x}_1\mathbf{x}_2\mathbf{y}_4\mathbf{w}]$, $[\mathbf{x}_2\mathbf{x}_3\mathbf{y}_4\mathbf{w}]$, and $[\mathbf{x}_1\mathbf{x}_3\mathbf{y}_4\mathbf{w}]$. As for the C^1 conditions across $[\mathbf{x}_1\mathbf{x}_2\mathbf{x}_3]$, notice that these conditions only involve control points from the 0-th layer and the

1st layers. From equation (5.16) and the way the control points in the 1st layer are determined, it is easy to see that the C^1 conditions are indeed satisfied.

Therefore, we have constructed a C^1 piecewise cubic function f satisfying

$$f(\mathbf{x}_i) = 0, \ \nabla f(\mathbf{x}_i) = \mathbf{n}_i, \ i = 1, \cdots, k. \tag{5.17}$$

If the free control points are chosen so that a "nice" cubic patch is obtained inside each tetrahedron within the neighborhood Σ, then the zero contour of f inside the neighborhood Σ is the free-form surface to be constructed.

5.3 Obtaining and controlling the cubic patches

As we mentioned earlier, creating a C^1 function f over the neighborhood Σ satisfying condition (5.17) is only an intermediate step. In general, such a function rarely yields the free-form surface we expect. The problem is that some of the control points of a cubic polynomial strongly affect the zero contour of the cubic polynomial inside the associated tetrahedron. If we let these control points be decided by the C^1 condition, then the zero contour of the cubic polynomial inside the tetrahedron exhibits various behaviors undesirable for free-form surface constructions.

The following situations may occur for the zero contour of a cubic polynomial inside a tetrahedron.

1. There is no zero contour in the interior of the tetrahedron even when the zero contour is known to pass through several vertices of the tetrahedron.

2. There are self-intersection points, or singular points on a cubic patch.

3. There are holes on a cubic patch caused by the zero contour of the cubic polynomial leaving and coming back to the tetrahedron.

4. There are multiple sheets of the zero contour inside the tetrahedron.

5. More dramatically, there may be even handles on a cubic patch.

Notice that we listed singular points together with self intersection because for implicit patches, singular points appear where the self intersections occur.

We use tetrahedra $[\mathbf{x}_1\mathbf{x}_2\mathbf{w}\mathbf{x}_4]$ in the above construction as an example to explain how the situations list above can be avoided by controlling the free control points we have chosen. In this example, the free control point are the function value $f(\mathbf{x}_4)$ and gradient $\nabla f(\mathbf{x}_4)$. The same argument with minor modifications applies to the cubic polynomials defined for other tetrahedra.

Situation one can be avoided by properly choosing the function value at $\mathbf{x}_4$. Consider the line segment from the centroid of $[\mathbf{x}_1\mathbf{x}_2\mathbf{w}]$, $\mathbf{p}$, to $\mathbf{x}_4$. If the function value $f(\mathbf{x}_4)$ is chosen to be have a sign opposite to that of the function value at $\mathbf{p}$, then there must be a point on the line segment $[\mathbf{x}_4\mathbf{p}]$ where the cubic polynomial is zero. In other words, the zero contour passes through the interior of the tetrahedron $[\mathbf{x}_1\mathbf{x}_2\mathbf{w}\mathbf{x}_4]$.

Situations two through five can be avoided by enforcing monotonicity conditions on the cubic polynomial along the direction from $\mathbf{w}$ to $\mathbf{x}_4$. Function f is monotone in direction α if the directional derivative along α is positive. Let the cubic polynomial in $[\mathbf{x}_1\mathbf{x}_2\mathbf{x}_4\mathbf{w}]$ be

$$h_3(\nu_3) = \sum_{|\lambda|=3} a_\lambda B^3_\lambda(\nu_3).$$

A sufficient condition for the cubic polynomial h_3 to be monotone along the direction form $\mathbf{w}$ to $\mathbf{x}_4$ within the tetrahedron is that

$$a_{\lambda-e^1+e^4} - a_\lambda \geq 0, \text{ for all } \lambda \text{ with } \lambda_1 \geq 1. \tag{5.18}$$

When $\lambda_1 > 1$, condition (5.18) can be enforced by the function value and gradient at $\mathbf{x}_4$. As for $\lambda_1 = 1$, the control points involved in condition (5.18) are completely determined from the prescribed normals in the input data, so the monotonicity conditions may not be satisfied for certain input data no matter how the free control points $h_3(\mathbf{x}_4)$ and $\nabla h_3(\mathbf{x}_4)$ are chosen. But remember that the purpose of the prescribed normals in the input data is to control the behavior of each cubic patch. If we choose these normals within proper ranges, condition (5.18) may be enforced.

In practice, the free control points are computed using the degree-elevation property of Bernstein-Bezier representation. The idea is to extend the effects of prescribed normals to the free control points. Even this method for choosing free control points comes with no guarantee, but it gives a reasonable choice for the free control points and it is easy to implement. Most importantly, it works very well in practice. From our experience, the ranges for free control points where the cubic patches behave well are fairly large. As long as the free control points are not in the relative small "bad" ranges, the cubic patches are in good shape.

More specifically, the free control points are generated as follows. First, we define a quadric polynomial q over the tetrahedron $[\mathbf{x}_1\mathbf{x}_2\mathbf{x}_3\mathbf{x}_4]$.

$$q(\tau) = \sum_{|\lambda|=2} b_\lambda B_\lambda^2(\tau)$$

where τ is the barycentric coordinate defined by the tetrahedron $[\mathbf{x}_1\mathbf{x}_2\mathbf{x}_3\mathbf{x}_4]$, and

$$b_{2e^i} = 0, \ i = 1, 2, 3;$$

$$b_{e^i+e^j} = \frac{1}{4}((\mathbf{n}_i, \mathbf{x}_j - \mathbf{x}_i) + (\mathbf{n}_j, \mathbf{x}_i - \mathbf{x}_j)) \ (1 \le i, j \le 3);$$

$$b_{e^i+e^j} = \frac{1}{2}(\mathbf{x}_j - \mathbf{x}_i, \mathbf{n}_i), \quad (i, j = 1, ...4; \ i \ne j).$$

If this is done for all facets of the input polyhedron $\mathcal{P}$, then quadric polynomials over tetrahedra such as $[\mathbf{x}_2\mathbf{x}_3\mathbf{x}_1''\mathbf{x}_4]$ can be determined also. These quadric polynomials are then degree elevated to cubic polynomials, whose control points corresponding to the free control points are used as the default values for the free control points.

5.4 Features of the algorithm

The above free-form algorithm has many attractive features. From the description of the algorithm, it is easy to see that the free-form construction in the algorithm is local. In the following, we discuss the quadratic precision of the algorithm and how to control the shape of the free-form surface without knowing the details of the algorithm.

Consider the quadratic precision first. Notice that the input polyhedron $\mathcal{P}$, prescribed normals at the vertices of $\mathcal{P}$, and the shape parameter completely determine the free-form surface. If the input data is taken from a quadric surface and the shape parameters are from the quadric surface, then the algorithm with the above default values for the free control points will produce the original quadric surface.

The shape control scheme with this algorithm is very similar to the algorithm described in Chapter 3. Recall that for each facet, the cubic polynomial over the double tetrahedron of the facet is not completely fixed. At the apex of the double tetrahedron of each facet, the value of the cubic polynomial is left as a shape parameter, as shown in 5.3. With these facts in mind, we can use the same shape control scheme as described in 3.4.2 here.

Chapter 6

Controlling the shapes of implicit patches

In computer-aided design and computer graphics, the free-form surfaces used to model complex smooth objects often have to satisfy various shape requirements for practical and aesthetic reasons. Examples of such shape requirements are convexity, restrictions on the minimum radius of curvatures, and a nice pattern of reflection lines. In this chapter, we demonstrate how to control the shapes of free-form surfaces by manipulating the control points of the implicit patches in the free-form surfaces.

The Bernstein-Bezier representation of implicit patches is particularly suitable for shape control. In Section 6.1, we develop techniques for controlling the shape of an implicit patch represented by Bernstein polynomials. As an application of these techniques, we show how to achieve the convexity of an implicit patch in Section 6.2. Among all the shape requirements, convexity is the most basic and frequent one. We give a condition for an implicit patch to be convex. This condition allows the descriptions of the convexity of quadric patches and cubic patches in terms of their control points. Using these descriptions as guidelines, we can manipulate the control points of quadric patches and cubic patches to achieve convexity.

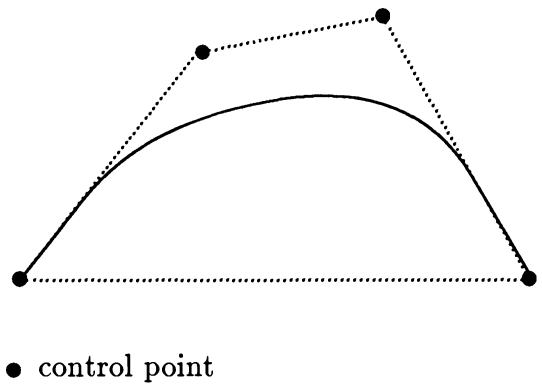


Figure 6.1: Function values of a univariate cubic function

6.1 Shape control properties of Bernstein-Bezier polynomials

A major advantage of the Bernstein-Bezier representation of a surface is the direct connection between the control points of the Bernstein-Bezier representation and the geometry of the surface. In this section, we show how to manipulate the shape of an implicit surface $S(f)$ by changing the control points. Here

$$f(\tau) = \sum_{|\lambda|=k} b_\lambda B_\lambda^k(\tau)$$

where τ is the barycentric coordinate defined by the tetrahedron $[\mathbf{x}_1\mathbf{x}_2\mathbf{x}_3\mathbf{x}_4]$.

To gain some intuition on the effects of control points, we consider the case where f is a cubic univariate polynomial over the 1-simplex $[0,1]$. In this case,

$$f(u) = b_{30}B_{30}^3 + b_{21}B_{21}^3 + b_{12}B_{12}^3 + b_{03}B_{03}^3.$$

The polynomial f and the convex hull of points $\mathbf{p}_1 = (0, b_{30})$, $\mathbf{p}_2 = (1/3, b_{21})$, $\mathbf{p}_3 = (2/3, b_{12})$, and $\mathbf{p}_4 = (1, b_{03})$ are shown in Figure 6.1, and the Bernstein polynomials B_{30}^3, B_{21}^3, B_{12}^3, and B_{03}^3 are shown in Figure 6.2.

From these figures, we can see the following.

1. At the vertices of the simplex $[0,1]$, the control points b_{30} and b_{03} are the function values of f.

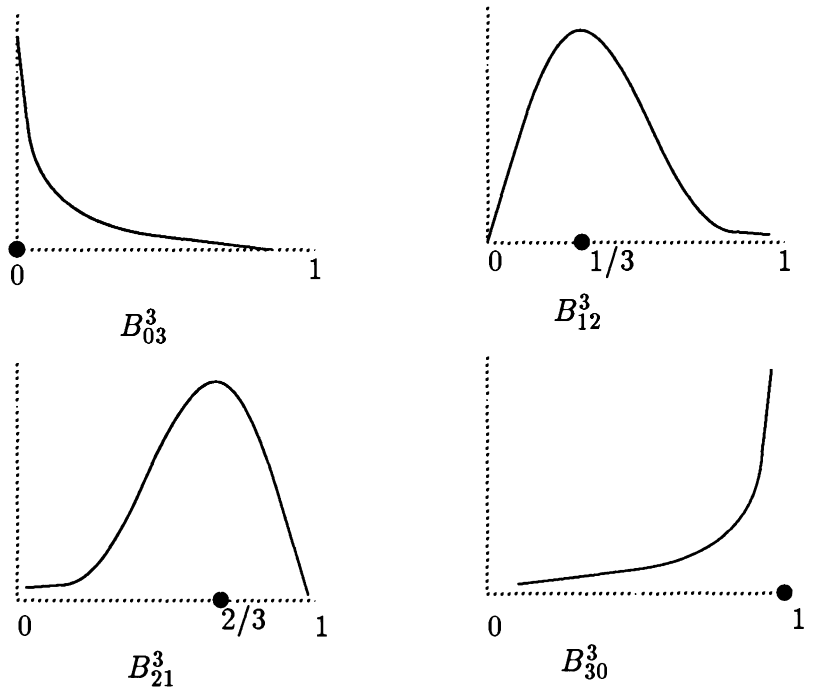


Figure 6.2: Weight functions for a univariate cubic function

2. The gradients at the vertices are determined by b_{21} and b_{30}, or b_{12} and b_{03}.

3. The control point b_{30} has effect on the value of $f(u)$ for all u except $u = 0$. Similar things can be said about the other control points.

4. A control point has the strongest effect on the function values of nearby space points.

All these relations between the control points and the properties of the polynomial f generalize to trivariate polynomials. Using these relations, we can control the geometry of the surface $S(f)$.

Consider the problem of interpolating points and lines in space by the surface $S(f)$. As mentioned in Section 1.4, the function values of f at the vertices of the tetrahedron $[\mathbf{x}_1\mathbf{x}_2\mathbf{x}_3\mathbf{x}_4]$ are exactly the values of the control points. So in order to force $S(f)$ to pass through the vertex $\mathbf{x}_i$, we set the control point at the vertex, b_{ke^i}, to zero. This method of interpolating points can be easily generalized to interpolate the edges of the tetrahedron $[\mathbf{x}_1\mathbf{x}_2\mathbf{x}_3\mathbf{x}_4]$. For example, to interpolate the edge $[\mathbf{x}_1\mathbf{x}_2]$, we set all the control points along the edge, $b_{\lambda_1\lambda_2 00}$, to zero.

Moving on to the problem of controlling the tangent planes of the surface $S(f)$, we consider the tangent plane of the surface $S(f)$ at the vertex $\mathbf{x}_1$. The tangent plane at $\mathbf{x}_1$ is defined by the gradient $\nabla f(\mathbf{x}_1)$, which is related to the control points next to the vertex $\mathbf{x}_1$ as follows,

$$b_{(k-1)e^1+e^j} = \frac{1}{k}(\nabla f(\mathbf{x}_1), \mathbf{x}_j - \mathbf{x}_1), \ j = 2, 3, 4.$$

Since the vectors $\mathbf{x}_j - \mathbf{x}_1$ for $j = 2, 3, 4$ are three linearly independent vectors, equation (1.1) implies that the control points next to the vertex $\mathbf{x}_1$ completely determine the gradient $\nabla f(\mathbf{x}_1)$, hence the tangent plane at $\mathbf{x}_1$.

More sophisticated examples of shape control are easy to come by. The restrictions of f to the faces and edges of the tetrahedron $[\mathbf{x}_1\mathbf{x}_2\mathbf{x}_3\mathbf{x}_4]$ can be viewed as bivariate and univariate Bernstein-Bezier polynomials. This fact can be used to control the intersections of the surface $S(f)$ with the edges and faces. If the control points on an edge are all positive or all negative, then the surface $S(f)$ does not intersect the edge. Otherwise, the surface $S(f)$ intersects an edge exact once if there is exactly one sign change in the list of control points along the edge. Similarly, the intersections of $S(f)$ and the the faces of $[\mathbf{x}_1\mathbf{x}_2\mathbf{x}_3\mathbf{x}_4]$ can be controlled.

6.2 Achieving convexity through shape control

As an application of the techniques for shape control, we derive the convexity condition of an implicit patch in terms of its control points. Throughout this section, we concentrate on the implicit patch defined as the portion of a surface $S(f)$ inside a tetrahedron V.

The reader is familiar with convex objects as a set of points in three dimensional space such that the line segment connecting two points in the set is contained in the set. Convex surfaces are often defined as surfaces whose Gaussian curvatures are positive over the entire surface. Convex objects and convex surfaces are related in that if a convex surface is closed and it bounds a point set with finite volume, then the point set is a convex object.

Defining a convex surface in terms of Gaussian curvature is not convenient when dealing with implicit surfaces. So we use the definition of convex surfaces in terms of the tangent planes. Let an implicit surface $S(f)$ have a tangent plane $S(P_{\mathbf{x}})$ at point $\mathbf{x} \in S(f)$. The surface $S(f)$ is convex at the point $\mathbf{x}$ if the surface $S(f)$ is in the half space bounded by $S(P_{\mathbf{x}})$ and pointed to by $-\nabla f(\mathbf{x})$. An implicit patch is convex if its primary surface is convex at every point on the implicit patch.

Notice the relationship between the convexity of the surface $S(f)$ and the convexity of the polynomial f. A polynomial f is convex over the tetrahedron V if for any two points $\mathbf{x}$ and $\mathbf{y}$ in the tetrahedron,

$$f(\frac{\mathbf{x}+\mathbf{y}}{2}) \leq \frac{1}{2}(f(\mathbf{x}) + f(\mathbf{y})).$$

It is easy to show that if the polynomial f is convex over the tetrahedron V, then the implicit patch defined as the portion of $S(f)$ inside V is convex. However, the converse is not true.

Motivated by the design of parametric convex surfaces, researchers in CAGD have obtained many results on the convexity conditions of polynomials over triangles [CD84,Gra89,Goo89]. It is possible to generalize these results to polynomials over tetrahedra [DM85,DM88], thus obtaining sufficient conditions for implicit patches to be convex. However, the convexity conditions obtained this way are often overly restrictive. So in the following, we derive the convexity conditions of an implicit patch directly from the definition of a convex implicit patch.

Let $\mathbf{p}' = (\tau_1', \tau_2', \tau_3', \tau_4')$ be a point close to a point $\mathbf{p} = (\tau_1, \tau_2, \tau_3, \tau_4)$ on the surface $S(f)$. The Taylor expansion of f, with higher order terms omitted, is

$$f(\mathbf{p}') = f(\mathbf{p}) + \sum_{i=1}^{4} \frac{\partial f}{\partial \tau_i}(\tau_i' - \tau_i) + \frac{1}{2} \sum_{i,j=1}^{4} \frac{\partial^2 f}{\partial \tau_i \partial \tau_j}(\tau_i' - \tau_i)(\tau_j' - \tau_j) \tag{6.1}$$

The first term on the right-hand side of the above equation, $f(\mathbf{p})$, vanishes because $\mathbf{p}$ is on $S(f)$. Moving on to the second term, we notice that this term is the same as the left-hand side of the tangent plane equation at $\mathbf{p}$,

$$\sum_{i=1}^{4} \frac{\partial f}{\partial \tau_i}(\tau_i' - \tau_i) = 0. \tag{6.2}$$

So the definition of a convex implicit surface implies that $S(f)$ is convex at $\mathbf{p}$ if

$$\sum_{i,j=1}^{4} \frac{\partial^2 f}{\partial \tau_i \tau_j}(\tau_i' - \tau_i)(\tau_i' - \tau_j) \geq 0 \tag{6.3}$$

for all $\mathbf{p}'$. Introducing a new variable $\sigma_i = \tau_i' - \tau_i$, we can write (6.3) as

$$\sum_{i,j=1}^{4} \frac{\partial^2 f}{\partial \tau_i \partial \tau_j}\sigma_i \sigma_j \geq 0. \tag{6.4}$$

The σ's satisfy the constraint

$$\sum_{i=1}^{4} \sigma_i = 0 \tag{6.5}$$

since the barycentric coordinates (τ_i) and (τ_i') satisfy the constraints

$$\sum_{i=1}^{4} \tau_i = 1 \text{ and } \sum_{i=1}^{4} \tau_i' = 1.$$

To eliminate the constraint (6.5), we substitute $-\sum_{i=1}^{3} \sigma_i$ for σ_4 in (6.4). The result is

$$\sum_{i,j=1}^{3} a_{ij}(\mathbf{p})\sigma_i \sigma_j \geq 0 \tag{6.6}$$

for arbitrary $(\sigma_1, \sigma_2, \sigma_3)$ with

$$a_{ij}(\mathbf{p}) = \frac{\partial^2 f}{\partial \tau_i \partial \tau_j} + \frac{\partial^2 f}{\partial \tau_4 \partial \tau_4} - \frac{\partial^2 f}{\partial \tau_i \partial \tau_4} - \frac{\partial^2 f}{\partial \tau_j \partial \tau_4}. \tag{6.7}$$

The condition (6.6) is the condition for the surface $S(f)$ to be convex at the point $\mathbf{p}$.

Applying the condition (6.6) to every point on an implicit patch, we have the following theorem.

Theorem 6.1 *An implicit patch is convex if the* 3×3 *matrix* $A = (a_{ij})$ *is positive definite for all points* $\mathbf{p}$ *on the implicit patch.*

Proof: Obvious from the above arguments. ♣

Generalizing the convexity conditions for bivariate polynomials would give a sufficient condition requiring the matrix A to be positive definite over the entire tetrahedron as opposed to the implicit patch. The condition in Theorem 6.1 is much less restrictive.

Although Theorem 6.1 gives a condition for the convexity of an implicit patch, the condition is hard to use because checking the condition for the infinitely many points on the implicit patch is impossible. So in the rest of the section, we use Theorem 6.1 to derive the convexity condition of an implicit patch in terms of its control points.

If f is a degree k polynomial given by

$$f = \sum_{|\lambda|=k} b_\lambda B_\lambda^k(\tau),$$

then

$$\frac{\partial^2 f}{\partial \tau_i \partial \tau_j} = k(k-1) \sum_{|\mu|=k-2} b_{\mu+e^i+e^j} B_\mu^{k-2}(\tau),$$

and $A = (a_{ij}(\mathbf{p}))$ is a 3×3 symmetric matrix whose entries are homogeneous degree $k-2$ polynomials in (τ_i). From linear algebra, A is positive definite if and only if

$$a_{11} \geq 0, \quad \begin{vmatrix} a_{11} a_{12} \\ a_{12} a_{22} \end{vmatrix} \geq 0, \text{ and } |A| \geq 0. \tag{6.8}$$

In order to decide whether A is positive definite for all points $\mathbf{p}$ on an implicit patch, we have to determine the signs of the minimum values of the quantities listed in (6.8) under the following the constraints,

$$f(\mathbf{p}) = 0, \tag{6.9}$$

$$\tau_1 + \tau_2 + \tau_3 + \tau_4 = 1, \quad \text{and} \quad \tau_i \geq 0 \ \ (i = 1, 2, 3, 4). \tag{6.10}$$

Here the constraints characterize the points on the implicit patch. The inequality constraints and nonlinear constraints in (6.9) and (6.10) make the problem of deciding the convexity of a general implicit patch very hard.

Fortunately, practical criterions for the convexity of quadric patches and cubic patches can be derived. For quadric patches, notice that

$$\frac{\partial^2 f}{\partial\tau_i \partial\tau_j} = b_{e^i+e^j}$$

is independent of $\mathbf{p}$, so the convexity of a quadric patch can be decided by evaluating (6.8) with the constant $a_{ij} = b_{e^i+e^j} + b_{0002} - b_{e^i+e^4} - b_{e^j+e^4}$.

Deciding the convexity of a cubic patch is a little bit harder. Using the formula for Bernstein-Bezier polynomials, it is easy to verify that

$$\frac{\partial^2 f}{\partial\tau_i \partial\tau_j} = 6\sum_{m=1}^{4} b_{e^m+e^i+e^j}\tau_m$$

Using this relation, we can rewrite (6.6) as

$$\sum_{m=1}^{4} \tau_m Q_m(\sigma) \geq 0. \tag{6.11}$$

where

$$Q_m(\sigma) = \sum_{i,j=1}^{4} (b_{e^i+e^j+e^m} + b_{2e^4+e^m} - b_{e^4+e^i+e^m} - b_{e^4+e^j+e^m})\sigma_i\sigma_j.$$

Since the left-hand side of inequality (6.11) is a convex combination of $Q_m(\sigma)$, the inequality (6.11) is valid over the entire tetrahedron enclosing the cubic patch if and only if the inequality is valid at the vertices of the tetrahedron, i.e.

$$Q_m(\sigma) \geq 0, \text{ for } m = 1, 2, 3, 4.$$

So the cubic patch is convex if the inequalities in (6.8) hold for $m = 1, 2, 3, 4$ with constant

$$a_{ij} = b_{e^i+e^j+e^m} + b_{2e^4+e^m} - b_{e^4+e^i+e^m} - b_{e^4+e^j+e^m}.$$

An important observation is that for each m, the above condition is exactly the same as the convexity condition for a quadric patch. Using the terminology of CAGD, we can say that a cubic patch inside a tetrahedron is convex if the subpolynomials at the vertices of the tetrahedron are convex.

Chapter 7

Conclusions

This thesis addresses the problem of constructing free-form surfaces using low-degree implicit patches: given an arbitrary polyhedron with normals prescribed at its vertices, construct a smooth, implicitly defined, and piecewise low-degree algebraic surface that smoothly contains the vertices of the polyhedron.

We established the fact that quadric patches and cubic patches are flexible enough for free-form surface constructions. For free-form surface constructions using quadric patches, we showed how to fit a smooth piecewise quadric surface through the vertices of an arbitrary polyhedron without splitting its facets. For free-form surface construction using cubic patches, we gave an algorithm which takes the input of an arbitrary polyhedron with arbitrarily prescribed normals at its vertices, and constructs a smooth piecewise cubic surface that smoothly contains the vertices of the input polyhedron.

In deriving the free-form surface algorithms, we also explored the power and limitations of blending techniques. We showed that on the one hand, blending techniques can be used to improve the efficiency of free-form surface constructions; on the other, blending techniques have their fundamental limitations.

Good shape control is one of the main reasons that parametric patches have become so popular in computer graphics. This thesis tackled some shape control issues of implicit patches. In particular, we showed how to achieve the convexity of a quadric or cubic patch through manipulating its control points.

There are still a number of interesting problems in implicit free-form surface constructions that remain unsolved.

1. Improve the robustness of the free-form surface constructions. A major difficulty in using implicit patches is handling topological anomalies. We presented partial solutions to the problem of topological anomalies, but these solutions are sensitive to the input data. A robust solution would be a major contribution to the constructions of implicit free-form surfaces.

2. Design algorithms for constructing implicit free-form surfaces of prescribed shapes. In computer-aided design and other applications, free-form surfaces used to model physical objects often have to satisfy various shape requirements for practical and aesthetic reasons. Examples of such shape requirements are convexity and restrictions on the minimum radius of curvatures. Algorithms for constructing free-form surfaces of prescribed shapes are very valuable for industrial applications such as CAD/CAM.

3. Construct free-form surfaces of higher order smoothness. In this thesis, we only considered the first-order smoothness. Free-form surfaces of higher order smoothness are also needed in applications. For example, designing aerodynamic surfaces for car bodies needs second-order smooth surfaces. Designing ship hulls often requires surfaces with third-order smoothness.

4. Find a way for fast rendering of piecewise algebraic surfaces. Currently, piecewise algebraic surfaces are rendered using either ray tracing or polygonizing. Ray tracing generates very good images, but it is slow and view dependent, so it is impossible to create dynamic sequences with ray tracing. Polygonizing is view independent, and it is easy to generate dynamic sequences. However,

polygonizing algebraic surfaces takes a certain amount of time, which hinders the interactive design of free-form surfaces.

To sum up, we have demonstrated that arbitrary physical objects can be represented using low-degree algebraic surfaces. Such representations greatly facilitate manipulating and reasoning about physical objects.

Bibliography

[AB87] S. Abhyankar and C. Bajaj. Automatic rational parameterization of curves and surfaces ii: cubics and cubicoids. *Computer Aided Design: CAD*, pages 499–502, 1987.

[AHU74] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *The Design and Analysis of Computer Algorithms.* Addison-Wesley, Reading, Mass., 1974.

[Baj88] C. Bajaj. Geometric modeling with algebraic surfaces. Technical Report CSD-TR-825, Computer Science, Purdue University, 1988.

[BBB87] R. Bartels, J. Beatty, and B. Barsky. *An Introduction to Splines for Use in Computer Graphics and Geometric Modeling.* Morgan Kaufmann Publishers, Los Altos, California, 1987.

[BFK84] W. Boehm, G. Farin, and J. Kahmann. A survey of curve and surface methods in computer-aided geometric design. *Computer Aided Geometric Design*, 1:1–60, 1984.

[BHLH88] C. Bajaj, C. Hoffmann, R. Lynch, and J. Hopcroft. Tracing surface intersection. *Computer Aided Geometric Design*, 5:285–307, 1988.

[BI89] C. Bajaj and I. Ihm. Hermite interpolation using real algebraic surfaces. In *Proceedings of the ACM Symposium on Computational Geometry*, pages 94–103, 1989.

[BK87] C. Bajaj and M. Kim. Compliant motion planning with geometric models. In *Proceedings of the 3rd ACM Symposium on Computational Geometry*, pages 171–180, 1987.

[Boi84] J. Boissonnat. Geometric structure for three-dimensional shape representation. *ACM Transactions on Graphics*, 3(4):266–286, 1984.

[Boi85] J. Boissonnat. Reconstruction of solids. In *Proceedings of the 2nd ACM Symposium on Computational Geometry*, pages 46–54, 1985.

[Buc85] B. Buchberger. Groebner bases: An algorithmic method in polynomial ideal theory. In N. K. Bose, editor, *Recent Trends in Multidimensional System Theory*, 1985.

[CD84] G. Chang and P. Davis. The convexity of bernstein polynomials over triangles. *Journal of Approximation Theory*, 40:11–28, 1984.

[Dah86] W. Dahmen. Bernstein-bezier representation of polynomial surfaces. In *Extension of B-spline Curve Algorithms to Surfaces.* SIGGRAPH'86 Course Notes, 1986.

[Dah89] W. Dahmen. Smooth piecewise quadric surfaces. In T. Lyche and L. Schumaker, editors, *Mathematical Methods in CAGD*, pages 181–193, 1989.

[DeR85] A. DeRose. *Geometric continuity: a parameterization independent measure of continuity for computer aided geometric design.* Ph.D. dissertation, Computer Science, University of California at Berkeley, 1985.

[DM85] W. Dahmen and C. Micchelli. Convexity of multivariate bernstein polynomials and box spline surfaces. Technical Report RC11176, T. J. Watson Research Center, IBM, 1985.

[DM88] W. Dahmen and C. Micchelli. Convexity and bernstein polynomials k-simploids. Technical Report RC13969, T. J. Watson Research Center, IBM, 1988.

[EKL+91] J. Ellis, G. Kedem, T. Lyerly, R. Marisa, J. Menon, D. Thielman, and H. Voelcker. Ray representations and the raycasting engine. In *Proceedings of ACM Symposium on Solid Modeling Foundations and CAD/CAM Applications*, 1991.

[EKM+91] J. Ellis, G. Kedem, R. Marisa, J. Menon, and H. Voelcker. Breaking barriers in solid modeling. *ASME Mechanical Engineering*, pages 28–34, February 1991.

[Far85] G. Farin. Triangular bernstein-bezier patches. *Computer Aided Geometric Design*, 3:83–127, 1985.

[Far88] G. Farin. *Curves and Surfaces for Computer Aided Geometric Design.* Academic Press, Boston, 1988.

[FD90] J Foley and A. Van Dam. *Fundamentals of interactive computer graphics.* Addison-Wesley, 1990.

[Goo89] T. Goodman. Shape preserving representations. In *Mathematical Methods in Computer Aided Geometric Design*, pages 145–152. Academic Press, 1989.

[Gra89] T. Gradine. On the convexity of piecewise polynomial functions on triangulations. *Computer Aided Geometric Design*, 6:181–187, 1989.

[GTZ88] P. Gianni, B. Trager, and G. Zacharias. Groebner bases and primary decomposition of polynomial ideals. *Journal of Symbolic Computation*, 6(2-3):149–167, 1988.

[GW89] T. Garrity and J. Warren. Geometric continuity. Technical Report Rice COMP TR89-89, Department of Computer Science, Rice University, 1989.

[Har77] R. Hartshorne. *Algebraic Geometry.* Springer-Verlag, New York, 1977.

[Her75] I. Herstein. *Topics in Algebra.* John Wiley & Sons, New York, 1975.

[Her85] G. Herron. Smooth closed surfaces with discrete triangular interpolants. *Computer Aided Geometric Design*, 2:297–306, 1985.

[HH86] C. Hoffmann and J. Hopcroft. Quadratic blending surfaces. *Computer Aided Design: CAD*, 18:301–306, 1986.

[HH87] C. Hoffmann and J. Hopcroft. The potential method for blending surfaces and corners. In *Geometric Modeling: Algorithms and New Trends*, pages 347–366, 1987.

[HH88] C. Hoffmann and J. Hopcroft. The geometry of projective blending surfaces. *Artificial Intelligence*, 37:357–376, 1988.

[HK86] J Hopcroft and D. Krafft. The challenge of robotics for computer science. In C. Yap and J. Schwartz, editors, *Advances in Robotics, Vol. 1: Algorithmic and Geometric Aspects of Robotics*, 1986.

[Hof89] C. Hoffmann. *Solid and Geometric Modeling.* Morgan Kaufmann Publishers, 1989.

[Kir79] D. Kirkpatrik. Efficient computations of continuous skeletons. In *Proceedings of 20th Symposium on Foundations of Computer Science*, pages 18–27, 1979.

[Kun85] E. Kunz. *Introduction to Commutative Algebra and Algebraic Geometry.* Birkhauser, 1985.

[LBMSFB88] E. Le Bras-Mehlman, M. Schmitt, O. Faugeras, and J. Boissonnat. How the delaunay triangulation can be used for representing stereo data. In *International Conference on Computer Vision*, pages 54–63, 1988.

[Lee82] D. Lee. Medial axis transformation of planar shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 4:363–369, 1982.

[Lev76] J. Levin. A parametric algorithm for drawing pictures of solid objects bounded by quadrics surfaces. *Communications of the ACM*, pages 555–563, November 1976.

[LH74] C. Lawson and R. Hanson. *Solving Least Square Problems.* Prentice-Hall, 1974.

[Man88] M. Mantyla. *An Introduction to Solid Modeling.* Computer Science Press, Rockville, Maryland, 1988.

[Mas77] W. Massey. *Algebraic Topology: an Introduction.* Springer-Verlag, New York, 1977.

[MLL+90] S. Mann, C. Loop, M. Lounsbery, D. Meyers, J. Painter, T. DeRose, and K. Sloan. *A survey of parametric scattered data fitting.* Department of Computer Science, University of Washington, 1990. Preprint.

[Mor87] M. Mortenson. *Geometric Modeling.* John Wiley & Sons, New York, 1987.

[MS85] A. Middleditch and K. Sears. Blend surfaces for set volume modeling systems. *Computer Graphics*, 19:161–170, 1985.

[MW90] D. Moore and J. Warren. *Adaptive approximation of scattered contour data using piecewise implicit surfaces.* Department of Computer Science, Rice University, 1990. Preprint.

[OBW87] J. O'Rourke, H. Booth, and R. Washington. Connect-the-dots: A new heuristic. *Computer Vision, Graphics, and Image Processing*, 39:258–266, 1987.

[OSS83] S. Ocken, J. Schwartz, and M. Sharir. Piecewise implementation of cad primitives using rational parameterizations of standard surfaces. In *Planning, Geometry, and Complexity of Robot Motion*, pages 245–266, Norwood, NJ, 1983. Ablex Publishing.

[PG86] M. Pratt and A. Geisow. Surface/surface intersection problem. In J. Gregory, editor, *The Mathematics of Surfaces*, pages 285–307, 1986.

[PK89] N. Patrikalakis and G. Kriezis. Representation of piecewise continuous algebraic surfaces in terms of b-splines. *The Visual Computer*, 5:360–374, 1989.

[Pra79] M. Pratt. *Computational Geometry for Design and Manufacture.* Halsted Press, 1979.

[Pre77] F. Preparata. Steps into computational geometry. Technical report, Coordinated Science Laboratory, University of Illinois, 1977.

[PS85] F. Preparata and M. Shamos. *Computational Geometry.* Springer-Verlag, 1985.

[RO87] A. Rockwood and J. Owen. Blending surfaces in solid geometric modeling. In G. Farin, editor, *Geometric Modeling: Algorithms and New Trends*, 1987.

[Ros85] J Rossignac. *Blending and Offsetting Solid Models.* Ph.D. dissertation, Department of Electrical Engineering, University of Rochester, 1985.

[RV83] A.A.G. Requicha and H.B. Voelcker. Solid modeling: Current status and research directions. *IEEE Journal on Computer Graphics and Its Application*, 3:25–37, 1983.

[RVC88] A.A.G. Requicha, H.B. Voelcker, and R.W. Conway. Computer applications in manufacturing. *Annual Review in Computer Science*, 3:349–387, 1988.

[Sed83] T.W. Sederberg. *Implicit and Parametric Curves and Surfaces for Computer Aided Geometric Design.* Ph.D. dissertation, Mechanic Engineering, Purdue University, 1983.

[Sed85] T.W. Sederberg. Piecewise algebraic surface patches. *Computer Aided Geometric Design*, 2:53–59, 1985.

[Sed87] T.W. Sederberg. Algebraic geometry for surface and solid modeling. In G. Farin, editor, *Geometric Modeling: Algorithms and New Trends*, pages 171–180, 1987.

[Sed90] T. Sederberg. Techniques for cubic algebraic surfaces. *IEEE Computer Graphics and Its Applications*, pages 14–21, July 1990.

[Som34] D.M.Y. Sommerville. *Analytical Geometry of Three Dimensions.* Cambridge University Press, 1934.

[War86] J. Warren. *On algebraic surfaces meeting with geometric continuity.* Ph.D. dissertation, Department of Computer Science, Cornell University, 1986.

[War87] J. Warren. Blending quadric surfaces with quadric and cubic surfaces. In *Proceedings of the ACM Symposium on Computational Geometry*, pages 341–347, 1987.

[War89] J. Warren. Blending algebraic surfaces. *ACM Transactions on Graphics*, 8(4):263–278, 1989.

[WLPL+80] M. Wesley, T. Lozano-Perez, L. Lieberman, M. Lavin, and D. Grossman. A geometric modeling system for automated mechanical assembly. *IBM Journal on Research and Development*, 24(1):64–74, 1980.

[Woo87] J. Woodwark. Blends in geometric modeling. In R. Martin, editor, *The Mathematics of Surfaces II*, pages 255–297, 1987.

[Zar71] O. Zariski. *Algebraic Surfaces.* Springer-Verlag, New York, 1971.