1991

On Surface Design with Implicit Algebraic Surfaces (Ph.D. Thesis)

Insung Ihm

Report Number:
91-057

ON SURFACE DESIGN WITH
IMPLICIT ALGEBRAIC SURFACES

Insung Ihm

CSD-TR-91-057
August 1991

ON SURFACE DESIGN WITH IMPLICIT ALGEBRAIC SURFACES

A Thesis
Submitted to the Faculty

of

Purdue University

by

Insung Ihm

In Partial Fulfillment of the
Requirements for the Degree

of

Doctor of Philosophy

August 1991

To the memory of my parents

ACKNOWLEDGMENTS

I would like to thank my advisor, Professor Chanderjit Bajaj, for his inspiration and guidance. He always provided me with an exciting research environment in which this work was possible. My gratitude also goes to the advisory committee members, Professors Shreeram Abhyankar, Robert Lynch, and David Anderson for their high standards. Professor Christoph Hoffmann had served as a committee member even though he could not come to my final exam due to my hectic schedule. Part of this work was done under the supervision of Dr. Bruce Naylor at AT&T Bell Laboratories in Murray Hill, New Jersey. Professor Myeong-Soo Kim at POSTECH, Korea, always cheered me up.

My sincere appreciation goes to Bill Bouma and Andrew Royappa. Bill was an endless source of computer graphics whose utility programs made much easier the picture-making process that comprises an important part of my four years education at Purdue. Andrew, my colleague and roommate, was always patient in answering to my various, sometimes boring, questions on Unix systems to "how to get ALL of our deposits back from the seemingly tough landlords". Also, his meticulous proofreading enhanced the quality of this thesis. Learning *Nihon Go* from Kunihiko Okamura *San* let me feel somewhat shorter my last long summer in West Lafayette. I'd also like to thank all the friends who made the G16 lab an enjoyable work place.

This research was supported in part by the David Ross Fellowship from Purdue University, NSF grant DMS 88-16286, CCR 90-0228, ONR contract N00014-88-K0402, and AFOSR contract 91-0276.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Ihm, Insung. Ph.D., Purdue University, August 1991. On Surface Design with Implicit Algebraic Surfaces. Major Professor: Chanderjit Bajaj.

Computer Aided Geometric Design (CAGD) is a rapidly growing area that involves theories and techniques from many disciplines such as computer science and mathematics as well as engineering. One of the most important subjects in CAGD is to efficiently model physical objects with a surface or collection of surfaces for many applications of CAD/CAM, computer graphics, medical imaging, robotics and etc. Most research in surface modeling has been largely dominated by the theory of parametrically represented surfaces. While they have been successfully used in representing physical objects, parametric surfaces are confronted with some problems when objects represented with them are manipulated in geometric modeling systems.

In recent years, increasing attention has been paid to algebraic surfaces that are implicitly defined by a polynomial equation, and provide a more general class of surfaces at lower degrees. In this thesis, we consider the problem of modeling complex geometric objects with smooth piecewise algebraic surface patches. We present an interpolation algorithm, called Hermite interpolation, which characterizes a class of all algebraic surfaces of a specified degree that interpolate given points and space curves with tangent plane continuity. The Hermite interpolation algorithm with least squares approximation transforms the geometric problem of algebraic surface design into a linear algebra problem which can be solved efficiently. Based on this algebraic model, we explore the class of quintic algebraic surfaces to smooth convex polyhedra with a mesh of smooth piecewise algebraic surface patches. Degrees of freedom in constructing wire frames for polyhedra are used to control shapes of curved models of polyhedra. The open problem of modeling polyhedra having arbitrary shapes

with quintic triangular algebraic surface patches is considered. Finally, we present a heuristic algorithm which quickly computes a good piecewise linear approximation of a given digitized space curve. This algorithm serves as a primary tool in polygonizing triangular algebraic surface patches.

1. INTRODUCTION

Computer Aided Geometric Design or CAGD is a rapidly growing area that involves theories and techniques from many disciplines such as computer science and mathematics as well as engineering. The primary goal of CAGD is to create geometric models of physical objects, and to automate the process of design, analysis, and manufacturing. Facilitating such modeling and analysis is getting more attention in industry because constructing computer prototypes and analyzing them saves time and money in the manufacturing process. Efficient construction and manipulation of geometric objects is necessary in many applications of CAD/CAM, computer graphics, medical computing, pattern recognition, robotics, vision, and etc.

One of the most important subjects in CAGD is to model or represent physical objects with a surface or collection of surfaces. The tools from areas of mathematics like algebraic and differential geometry, and approximation theory have played key roles in exploring the mathematical concepts of surfaces, and exploiting them in implementing geometric modeling systems.

Most research in surface modeling has been largely dominated by the theory of parametrically represented surfaces, such as Bézier surfaces, Coons patches, and B-spline surfaces, due to their highly desirable properties in modeling [13, 14, 25]. While they have been successfully used in representing physical objects, parametric surfaces are confronted with some problems when objects represented with them are manipulated in geometric modeling systems. The flexibility of parametric surfaces comes with the cost of high degrees of surfaces. For instance, computing the intersection of two parametric surfaces of even moderately low degrees is expensive. Since a bidegree n parametric surface can be an algebraic surface of a degree up to $2n^2$, two bicubic parametric surfaces intersect in a curve of a degree up to 324.

In recent years, increasing attention has been paid to algebraic surfaces that are implicitly defined by a polynomial equation $f(x, y, z) = 0$. Algebraic surfaces provide a more general class of surfaces which is closed under geometric operations like offsetting [6], while the class of parametric surfaces is not. In fact, all rational parametric surfaces can be represented in implicit form, although the reverse is not true.

In CAGD, keeping the degrees of surfaces low is important because high degree surfaces entail various computational problems although they give more flexibility in surface design. The rendition of surfaces frequently included in CAGD applications [28] presents a view on the hierarchy of surfaces. Planes and quadrics, two simple classes of algebraic surfaces, are well known, and comprise important primitives in geometric modeling systems due to their simplicity. The limited flexibility of planes and quadrics leads to an investigation of the class of cubic algebraic surfaces, searching for more flexibility. Tori, that are a type of quartic algebraic surfaces, are frequently used as modeling primitives since they are adequate for some applications like joining two pipes.

The next classes of surfaces in the hierarchy of algebraic surfaces that are used in CAGD are parametric quadrics and biquadrics which are included in the classes of algebraic surfaces of degrees 4 and 8, respectively. Although they are able to model more complex geometric objects, their flexibility is also severely limited [28]. Parametric cubics and bicubics reside in the classes of degrees 9 and 18, respectively. Biquartic surfaces belong to the class of even higher degree in the hierarchy. (See Figure 1.1.)

Now, we observe gaps between the classes of algebraic surfaces typically used in CAGD. The gaps become more prominent considering that the class of parametric surfaces of algebraic degree n is a proper subset of the whole class of degree n. Then, we naturally arrive at the following questions: what about quartic and quintic algebraic surfaces? Are sextic algebraic surfaces inadequate as geometric modeling tools? The work in this thesis has originated from such questions in the hope of filling the gaps in the hierarchy with algebraic surfaces having moderately low degrees.

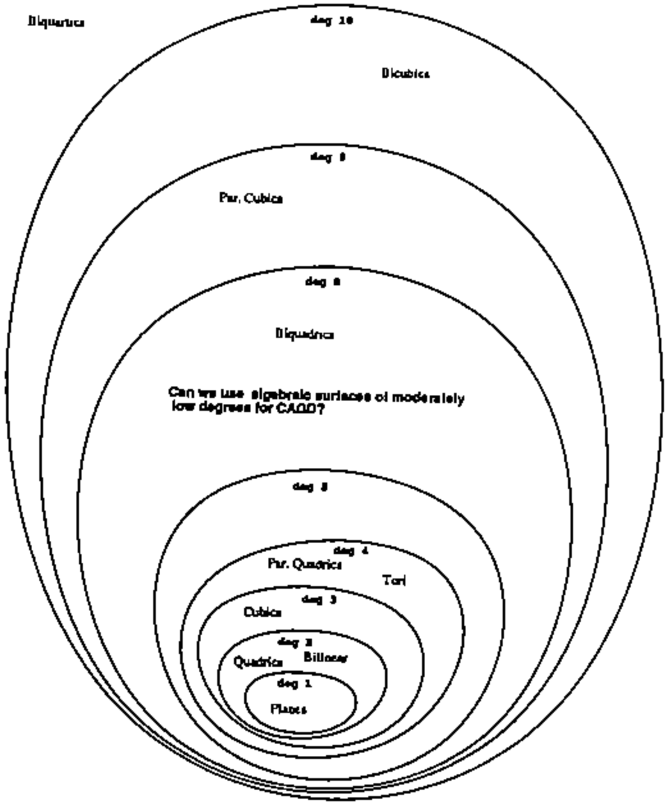

Figure 1.1 The Hierarchy of Algebraic Surfaces

There have been several noticeable works in computing algebraic surfaces for geometric modeling. Dahmen [20] presented an algorithm that constructs a mesh of smooth piecewise quadratic surface patches for some special type of polyhedron. Sederberg [66, 67] discussed some techniques for free form algebraic surface modeling, paying special attention to cubic surfaces. He introduced the concept of control points in barycentric coordinates as a way of defining and controlling a piecewise algebraic surface patch, though a more concrete controlling scheme needs to be developed. Guo [32] used cubic surface patches to smooth a polyhedron where cubic patches for faces are connected with two extra cubic patches. Quartic surfaces were used by Hoffmann et al. [36] and Middleditch et al. [45] to blend two primary quadric surfaces. In his thesis, Warren [75] investigated the mathematical structures of algebraic surfaces that meet a given surface with a specified order of geometric continuity, and applied the theory to the blending problem with low degree algebraic surfaces. Kosters [40] studied high-order continuous blending of algebraic surfaces.

In spite of all the previous results, much work remains to be done to explore the potential of algebraic surfaces as effective tools of CAGD. In particular, the capability of modeling a mesh of three dimensional data with smooth piecewise algebraic surface patches is essential because once it is achieved, physical objects can be modeled using algebraic surfaces, and can be included in geometric modeling systems for further manipulation. Hence, creating complex geometric objects with smooth piecewise algebraic surface patches is the first step toward construction of a geometric modeling system with algebraic surfaces as primitive tools.

In this thesis, we propose a direction of exploration of moderately low degree algebraic surfaces as tools of CAGD. In Chapter 2, we devise an interpolation algorithm, called Hermite interpolation, for algebraic surfaces. This algorithm takes as input positional and (optionally) first derivative information of points and space curves, parametrically or implicitly given, and characterizes, in terms of the nullspace of a matrix, the space of all the algebraic surfaces of a specified degree that smoothly interpolate the specified geometric data. Given input data, it produces a homogeneous

linear system, where unknowns are coefficients of algebraic surfaces, such that any algebraic surfaces with coefficients that are solutions of the system interpolate the input data. The Hermite interpolation algorithm serves as a fundamental tool for finding algebraic surfaces of a specified degree meeting with tangent plane continuity.

In Chapter 3, we consider how to choose an instance surface from a family of algebraic surfaces, resulting from Hermite interpolation. In geometric modeling systems, a user must be able to select a surface interactively with geometric intuition. The class or family of algebraic surfaces, computed with Hermite interpolation, is expressed in terms of the nullspace of a matrix that can be spanned by a set of basis vectors. The dimension of the nullspace equals the number of degrees of freedom left after consuming, for interpolation, some of the degrees of freedom of the class of algebraic surfaces of a given degree. We apply least square approximation to selection of a surface from the family by consuming the remaining degrees of freedom properly. Through Hermite interpolation and least square approximation, the geometric problem of algebraic surface design is transformed into a linear algebra problem which can be solved efficiently. A scheme of controlling shapes of algebraic surfaces in the family computed by the Hermite interpolation algorithm is investigated in the barycentric coordinate system.

Then, we attempt to generate a mesh of smooth piecewise algebraic surface patches. In Chapter 4, triangular surface patches are taken from the class of quintic algebraic surfaces to smooth a given convex polyhedron. Each edge of a polyhedron is replaced by a conic curve with associated normal directions, and then each face is replaced by a quintic algebraic patch that fleshes the three boundary curves. Each conic curve can be selected with a degree of freedom, and its shape or sharpness can be used to control the shape of triangular surface patches. Then, we consider the more general problem of smoothing an arbitrary polyhedron. We present some ideas for coping with nonconvexity of a polyhedron, and discuss open problems that need to be resolved in smoothing arbitrary polyhedra with algebraic surfaces.

In Chapter 5, we consider how to approximate an arbitrary three dimensional space curve, made of $n+1$ points, with m line segments. Generating piecewise linear approximations of digitized or densely sampled curves is an important problem in image processing, pattern recognition, geometric modeling, and computer graphics. Even though much attention has been paid to the planar curve case, little work has addressed space curve approximation. The heuristic algorithm we present in this chapter consumes $O(N_{iter} n)$ time and $O(n)$ space. It is based upon the notions of curve length and spherical image which are the fundamental concepts describing intrinsic properties of space curves. In this work, this heuristic piecewise linear segmentation algorithm provides a basic tool for generating an adaptive polygonization of algebraic triangular surface patches, computed in Chapter 4.

Finally, this thesis is summarized, and open problems for future research are discussed in Chapter 6.

2. HERMITE INTERPOLATION FOR ALGEBRAIC SURFACES

The primary objective of this work is to construct or approximate physical objects using meshes of algebraic surface patches. For aesthetic or functional reasons, it is usually required that the surface patches meet with geometric continuity. In many applications, C^1 or tangent plane continuity is sufficient. In his thesis, Warren [75] investigated algebraic structures of all surfaces meeting a given algebraic surface smoothly at a point or along a curve on that surface. He applied ideal theory to characterize the class of such surfaces in terms of polynomial expressions.

In this chapter, we present an algorithm, called Hermite interpolation, which algorithmically characterizes the class of all algebraic surfaces of a fixed degree which satisfy given geometric specifications. Input to this algorithm is a combination of points and algebraic space curves that are expressed either implicitly or parametrically. The points and space curves may have associated first derivative information in the form of normal vectors that define tangent planes at the points and space curves. Given an algebraic surface $S : f(x, y, z) = 0$ of degree n, the Hermite interpolation algorithm constructs a homogeneous linear system $\mathbf{M_I x} = 0$, $\mathbf{M_I} \in \mathrm{R}^{n_i \times n_v}$, $\mathbf{x} \in \mathrm{R}^{n_v}$ of n_i equations and n_v unknowns where the unknowns $\mathbf{x}$ are $n_v (= \binom{n+3}{3})$ coefficients of S. [1] Only when the rank r of $\mathbf{M_I}$ is less than the number of the coefficients n_v, does there exist a nontrivial solution to the system. All the vectors except 0 in the nullspace of $\mathbf{M_I}$ form a family of algebraic surfaces of degree n, satisfying the given input specifications, whose coefficients are expressed by homogeneous combinations of $q (= n_v - r)$ free parameters where q is the dimension of the nullspace.

As a result, the Hermite interpolation algorithm characterizes the family of algebraic surfaces with specified geometric properties in terms of the nullspace of a

[1] An algebraic surface of degree n has $\binom{n+3}{3}$ terms.

matrix. The algorithm is also useful in proving the existence or nonexistence of algebraic surfaces of degree n satisfying the input specifications since, when the rank of M_I is n_v, there is only the trivial solution 0 which does not correspond to an algebraic surface.

This chapter is organized as follows. First, in Section 2.1 we present some fundamental definitions and a key theorem used throughout the thesis. In Section 2.2 and 2.3, the Hermite interpolation algorithm is described. In Section 2.4, we briefly consider geometric continuity, and prove that our algorithm finds a family of all the desirable surfaces with G^1 rescaling continuity. Then, some computational aspects of Hermite interpolation are considered along with several examples of computing low degree algebraic surfaces.

2.1 Preliminaries

We give brief definitions of certain terms we need and also state a form of Bezout theorem. For detailed and additional definitions, refer to [1, 73]. For any multivariate polynomial f, partial derivatives are written by subscripting, for example, $f_x = \partial f/\partial x$, $f_{xy} = \partial^2 f/(\partial x \partial y)$, and so on. An algebraic surface of degree n in $\mathbb{R}^3$ is implicitly defined by a single polynomial equation $f(x, y, z) = \sum_{i+j+k \le n} c_{ijk} x^i y^j z^k = 0$ where the coefficients c_{ijk} of f are real numbers. The normal or gradient of $f(x, y, z) = 0$ is the vector function $\nabla f = (f_x, f_y, f_z)$. A point $\mathbf{p} = (x_0, y_0, z_0)$ on a surface is a regular point if the gradient at $\mathbf{p}$ is not null. Otherwise, the point is singular. An algebraic surface $f(x, y, z) = 0$ is irreducible if $f(x, y, z)$ does not factor over the field of complex numbers. An algebraic space curve is defined by the common intersection of two or more algebraic surfaces. Although it is not known if a complete algebraic space curve can be completely determined by the intersection of only two surfaces, in geometric design, we often restrict our consideration to a specific curve segment which is contained in the intersection of two algebraic surfaces. A rational parametric space curve is represented by the triple

$G(s) = (x = G_1(s), y = G_2(s), z = G_3(s))$, where G_1, G_2 and G_3 are rational functions in s. The degree of an algebraic surface is the number of intersections between the surface and a line, properly counting complex, infinite and multiple intersections. This degree is also the same as the degree of the defining polynomial. The degree of an algebraic space curve is the number of intersections between the curve and a plane, properly counting complex, infinite and multiple intersections. The degree of an algebraic curve segment given as the intersection curve of two algebraic surfaces is also no larger than the product of the degrees of the two surfaces. Furthermore, the degree of a rational parametric curve is the same as the maximum degree of the numerator and denominator polynomials in the defining triple of rational functions.

The following definitions are pertinent to our Hermite interpolation algorithm:

Definition 2.1 Let $\mathbf{p} = (p_x, p_y, p_z)$ be a point with an associated normal vector $\mathbf{n} = (n_x, n_y, n_z)$ in $\mathbb{R}^3$. An algebraic surface $S : f(x, y, z) = 0$ is said to contain $\mathbf{p}$ with C^1 or tangent plane continuity if
(1) $f(\mathbf{p}) = f(p_x, p_y, p_z) = 0$ (containment condition), and
(2) $\nabla f(\mathbf{p})$ is not zero and $\nabla f(\mathbf{p}) = \alpha \mathbf{n}$ for some nonzero α (tangency condition).

Definition 2.2 Let C be an algebraic space curve with an associated varying normal vector $\mathbf{n}(x, y, z) = (n_x(x, y, z), n_y(x, y, z), n_z(x, y, z))$, defined for all points on C. An algebraic surface $S : f(x, y, z) = 0$ is said to contain C with C^1 or tangent plane continuity if
(1) $f(\mathbf{p}) = 0$ for all points $\mathbf{p}$ of C (containment condition), and
(2) $\nabla f(\mathbf{p})$ is not identically zero and $\nabla f(\mathbf{p}) = \alpha \mathbf{n}(\mathbf{p})$ for some α and for all points $\mathbf{p}$ of C (tangency condition).

Definition 2.3 An algebraic surface $S : f(x, y, z) = 0$ is said to Hermite interpolate a given collection of points and space curves with associated normal vectors, if S contains all the points and space curves with C^1 continuity.

The following is one form of Bezout theorem, the oldest theorem of algebraic geometry. As will be seen, this theorem plays an important role in proving the correctness of the Hermite interpolation algorithm.

Theorem 2.1 (Bezout) An algebraic curve C of degree d intersects an algebraic surface S of degree n in exactly nd points, properly counting complex, infinite, and multiple intersections, or C intersects S infinitely often, that is, a component of C lies entirely on S.

2.2 Interpolation of Points with Normals

2.2.1 Containment

From the containment condition of Definition 2.1, it follows that any algebraic surface $S : f(x, y, z) = 0$, whose coefficients satisfy the linear equation $f(\mathbf{p}) = 0$ will contain the point $\mathbf{p}$. For a set of k data points, this yields k homogeneous linear equations. Since division of $f(x, y, z) = 0$ by a nonzero number does not change the surface the polynomial $f(x, y, z)$ represents, an algebraic surface of degree n has, in fact, $F = \binom{n+3}{3} - 1$ degrees of freedom. Interpolation of all the points is achieved by selecting an algebraic surface of degree n such that $F \geq r$, where r ($\leq k$) is the rank of a system of k homogeneous linear equations. Similar approaches for constructing algebraic surfaces that interpolate points can be found in [59].

2.2.2 Containment with Tangency

A point $\mathbf{p} = (p_x, p_y, p_z)$ with a normal vector $\mathbf{n} = (n_x, n_y, n_z)$ determines a unique plane $P : n_x x + n_y y + n_z z - (n_x p_x + n_y p_y + n_z p_z) = 0$ at the point $\mathbf{p}$. An algebraic surface $S : f(x, y, z) = 0$ of degree n that Hermite interpolates the point $\mathbf{p}$, can be constructed by setting up a linear system of equations as follows:
For each point $\mathbf{p}$ with a normal vector $\mathbf{n} = (n_x, n_y, n_z)$,

1. containment condition Use the linear equation $f(\mathbf{p}) = 0$ in the unknown coefficients of S.

2. tangency condition Select one of the following:
   (a) If $n_x \neq 0$, use the equations $n_x f_y(\mathbf{p}) - n_y f_x(\mathbf{p}) = 0$ and $n_x f_z(\mathbf{p}) - n_z f_x(\mathbf{p}) = 0$.
   (b) If $n_y \neq 0$, use the equations $n_y f_x(\mathbf{p}) - n_x f_y(\mathbf{p}) = 0$ and $n_y f_z(\mathbf{p}) - n_z f_y(\mathbf{p}) = 0$.
   (c) If $n_z \neq 0$, use the equations $n_z f_x(\mathbf{p}) - n_x f_z(\mathbf{p}) = 0$ and $n_y f_z(\mathbf{p}) - n_z f_y(\mathbf{p}) = 0$.
3. Next, ensure that the coefficients of $f(x, y, z) = 0$ satisfying the above three linear equations, additionally satisfy the constraints $\nabla f(\mathbf{p}) \neq 0$, since nontangency at $\mathbf{p}$ may occur if S turns out to be singular at $\mathbf{p}$.

The proof of correctness of the above algorithm follows from the following lemma.

Lemma 2.1 The equations of the above algorithm satisfy Definition 2.1 of point containment and tangency.

Proof: The first linear equation $f(\mathbf{p}) = 0$ satisfies containment by definition. We now show that the remaining equations satisfy $\nabla f(\mathbf{p}) = \alpha \cdot \mathbf{n}$ for a nonzero α. Since $\mathbf{n}$ is not a null vector, without loss of generality, we may assume that $n_x \neq 0$ in step 2 above. Other cases of $n_y \neq 0$ or $n_z \neq 0$ can be handled analogously. Now let $\alpha = \frac{f_x}{n_x}$, assuming $n_x \neq 0$. Then $f_x = \alpha \cdot n_x$ and substituting it in the selected linear equation $n_x f_y - n_y f_x = 0$ yields $f_y = \alpha \cdot n_y$ and substituting it again in the other selected linear equation $n_x f_z - n_z f_x = 0$ yields $f_z = \alpha \cdot n_z$. Hence $\nabla f(\mathbf{p}) = \alpha \cdot \mathbf{n}$. Finally, note that $f_x = 0$ for $n_x \neq 0$, in the selected linear equations of step 2(a), would cause $\nabla f(\mathbf{p}) = 0$, which we ensured would not happen in step 3 of the algorithm. Hence $f_x \neq 0$ and so $\alpha \neq 0$ and the lemma is proved. □

2.3 Interpolation of Curves with Normals

The varying normal vector associated with a space curve C can be defined implicitly by the triple $\mathbf{n}(x, y, z) = (n_x(x, y, z), n_y(x, y, z), n_z(x, y, z))$ where n_x, n_y and

n_z are polynomials of maximum degree m and defined for all points $p = (x, y, z)$ along the curve C. For the special case of a rational curve which we shall treat separately in Subsections 2.3.1.2 and 2.3.2.2, the varying normal vector can be also defined parametrically as $n(s) = (x = n_x(s), y = n_y(s), z = n_z(s))$, with n_x, n_y and n_z now rational functions in s.

2.3.1 Containment

2.3.1.1 Algebraic Curves: Implicit Definition

Let $C : (f_1(x, y, z) = 0, f_2(x, y, z) = 0)$ implicitly define an algebraic space curve of degree d. The irreducibility of the curve is not a restriction, since reducible curves can be handled by treating each irreducible curve component separately. For precise definitions of irreducible components of an algebraic curve, see [73]. The containment condition (as well as the tangency condition) requires the interpolating surface to be zero at a finite number of points on the curve. To ensure containment of a specific irreducible component requires choosing this finite number of points on that component. The precise number, derived from Bezout theorem, is a linear function of the degree of that curve component.

The situation is more complicated in the real setting, if we wish to achieve separate containment of one of possibly several connected real ovals of a single irreducible component of the space curve. There is a nontrivial problem of specifying a single isolated real oval of a curve. See [5] where a solution is derived in terms of a decomposition of space into cylindrical cells which separates out the various components of any real curve (or any real algebraic or semi-algebraic set).

An interpolating surface $S : f(x, y, z) = 0$ of degree n for containment of an irreducible curve component C, is computed as follows:

1. Choose a set L_c of $nd + 1$ points on C, $L_c = \{p_i = (x_i, y_i, z_i) | i = 1, \cdots, nd + 1\}$. The set L_c may be computed, for example, by tracing the intersection of $f_1 = f_2 = 0$ [7]. Thus, alternatively, an algebraic curve may be given as a list of points.
2. Next, set up $nd + 1$ homogeneous linear equations $f(p_i) = 0$, for all $p_i \in L_c$. Any nontrivial solution of this linear system will represent an algebraic surface which interpolates the entire curve C.

The proof of correctness of the above algorithm is captured in the following Lemma.

Lemma 2.2 To satisfy the containment condition of an algebraic curve C of degree d by an algebraic surface S of degree n, it suffices to satisfy the containment condition of $nd + 1$ points of C by S.

Proof: This is essentially a restatement of Bezout theorem in Section 2.1. Making S contain $nd + 1$ points of C ensures that S must intersect C infinitely often and hence, S must contain the entire curve. □

Recall that $S : f(x, y, z) = 0$ of degree n has $F = \binom{n+3}{3} - 1$ degrees of freedom. Let r be the rank of the system of $nd + 1$ linear equations. There are nontrivial solutions to this homogeneous system if and only if $F > r$ and a unique nontrivial solution when $F = r$. Again, an interpolating surface can be obtained by choosing a degree n such that $F \geq r$.

2.3.1.2 Rational Curves : Parametric Definition

When a curve is given in rational parametric form, its equations can be used directly to produce a linear system for interpolation, instead of first computing $nd + 1$ points on the curve. Let $C : (x = G_1(t), y = G_2(t), z = G_3(t))$ be a rational curve of degree d. An interpolating surface $S : f(x, y, z) = 0$ of degree n which contains C is computed as follows:

1. Substitute $(x = G_1(t), y = G_2(t), z = G_3(t))$ into the equation $f(x, y, z) = 0$.
2. Simplify and rationalize the expression from step 1 to obtain the numerator $Q(t) = 0$, where Q is a polynomial in t of degree at most nd with coefficients

which are homogeneous linear expressions in the coefficients of f. For Q to be identically zero, each of its coefficients must be zero, and hence we obtain a system of at most $nd+1$ linear equations, where the unknowns are the coefficients of f. Any nontrivial solution of this linear system will represent a surface S which interpolates C.

Lemma 2.3 The containment condition is satisfied by step 2 of the above algorithm.

Proof: Obvious. □

2.3.2 Containment with Tangency

In order to Hermite interpolate an algebraic curve C with a normal vector $\mathbf{n}$ by an algebraic surface S, we again need to solve a homogeneous linear system, whose equations stem from both the containment condition and the tangency conditions of Definition 2.2.

2.3.2.1 Algebraic Curves with Normals: Implicit Definition

As before, let $C : (f_1(x,y,z) = 0, f_2(x,y,z) = 0)$ implicitly define an irreducible algebraic space curve of degree d, together with an associated normal vector defined implicitly by the triple $\mathbf{n}(x,y,z) = (n_x(x,y,z), n_y(x,y,z), n_z(x,y,z))$ where n_x, n_y and n_z are polynomials of maximum degree m and defined for all points $\mathbf{p} = (x,y,z)$ along the curve C. A Hermite interpolating surface $S : f(x,y,z) = 0$ of degree n which contains C with C^1 continuity is then computed as follows:

1. Choose a set L_c of $nd+1$ points on C, $L_c = \{\mathbf{p}_i = (x_i, y_i, z_i) | i = 1, \cdots, nd+1\}$. The set L_c may be computed, as before, by tracing the intersection of $f_1 = f_2 = 0$.

2. Construct a list L_t of $(n+m-1)d+1$ point-normal pairs on C, $L_t = \{[(x_i, y_i, z_i), (n_{xi}, n_{yi}, n_{zi})] | i = 1, \cdots, (n+m-1)d+1\}$, where $(n_{xi}, n_{yi}, n_{zi}) = \mathbf{n}(x_i, y_i, z_i)$ for all i. Thus, alternatively, an algebraic curve C and its associated normal vector $\mathbf{n}$ may (either or both) be given as a list of points or point-normal pairs.

3. <u>containment condition</u> Next, set up $nd+1$ homogeneous linear equations $f(\mathbf{p}_i) = 0$, for $\mathbf{p}_i \in L_c$, $i = 1, \cdots, nd+1$.

4. <u>tangency condition</u>

   (a) Compute $\mathbf{t}(x,y,z) = \nabla f_1(x,y,z) \times \nabla f_2(x,y,z)$. Note $\mathbf{t} = (t_x, t_y, t_z)$ is the tangent vector to C.

   (b) Select one of the following:

      i. If $t_x \neq 0$, use the equation $f_y \cdot n_z - n_y \cdot f_z = 0$.

      ii. If $t_y \neq 0$, use the equation $f_x \cdot n_z - n_x \cdot f_z = 0$.

      iii. If $t_z \neq 0$, use the equation $f_x \cdot n_y - n_x \cdot f_y = 0$.

   Substitute each point-normal pair in L_t into the above selected equation to yield $(n+m-1)d+1$ additional homogeneous linear equations in the coefficients of $f(x,y,z)$.

5. In total, we obtain a homogeneous system of $(2n+m-1)d+2$ linear equations. Any nontrivial solution of the homogeneous linear system, for which, additionally, ∇f is not identically zero for all points of C (that is, the surface S is not singular at all points along the curve C), will represent a surface which Hermite interpolates C.

The proof of correctness of the above algorithm follows from Lemma 2.2 and the following lemma, which shows why the selected equation of step 4(b), evaluated at $(n+m-1)d+1$ point-normal pairs, is sufficient.

Lemma 2.4 To satisfy the tangency condition of an algebraic curve C of degree d with a normal vector $\mathbf{n}$ of degree m, by an algebraic surface S of degree n, it suffices to satisfy the tangency condition at $(n+m-1)d+1$ points of C by S as in step 4 of the above algorithm.

Proof: In step 4(b), assume, without loss of generality, that $t_x \neq 0$. Then the selected equation

$$f_y \cdot n_z - n_y \cdot f_z = 0 \tag{2.1}$$

We first show that if equation (2.1) is evaluated at only $(n+m-1)d+1$ points of C in step 4(b) above, it holds for all points on C. Equation (2.1) defines an algebraic surface H of degree $(n+m-1)$ which intersects C of degree d at at most $(n+m-1)d$ points. Invoking Bezout theorem, it follows that C must lie entirely on the surface H. Hence equation (2.1) is valid along the entire curve C.

We now show that step 4 of the above algorithm satisfies the tangency condition as specified in Definition 2.2. Since $\mathbf{t}$ of step 4(a) is a tangent vector at all points of C, and the surface $S : f = 0$ contains C, the gradient vector ∇f is orthogonal to $\mathbf{t}$, which yields the equation:

$$f_x \cdot t_x + f_y \cdot t_y + f_z \cdot t_z = 0 \tag{2.2}$$

valid for all points of C. Next, from the definition of a normal vector of a space curve,

$$n_x \cdot t_x + n_y \cdot t_y + n_z \cdot t_z = 0 \tag{2.3}$$

valid for all points of C. Now it is impossible that both $n_y(x,y,z)$ and $n_z(x,y,z)$ are identically zero along C, since if they were, then equation (2.3) would imply that $n_x \cdot t_x = 0$, and as we assumed that $t_x \neq 0$, would in turn imply that also $n_x = 0$ along C, which would contradict the earlier assumption that $\mathbf{n}$ is not identically zero. Hence, at least, one of $\mathbf{n}_y$ and $\mathbf{n}_z$ must also be nonzero. Without loss of generality, let $n_y \neq 0$. Also, let $\alpha(x,y,z) = \frac{f_y}{n_y}$. Then,

$$f_y = \alpha \cdot n_y \tag{2.4}$$

and substituting it into equation (2.1) yields

$$f_z = \alpha \cdot n_z \tag{2.5}$$

for all points on C. From equations (2.2), (2.4) and (2.5) we obtain,

$$f_x \cdot t_x + \alpha \cdot n_y \cdot t_y + \alpha \cdot n_z \cdot t_z = 0 \tag{2.6}$$

By multiplying α to equation (2.3) and subtracting equation (2.6) from it, we obtain

$$f_x \cdot t_x = \alpha \cdot n_x \cdot t_x \tag{2.7}$$

and since $t_x \neq 0$, finally obtain

$$f_x = \alpha \cdot n_x \tag{2.8}$$

valid at all points of C. Hence equations (2.4), (2.5), and (2.8) together imply that $\nabla f(x,y,z) = \alpha \cdot \mathbf{n}$ for all points C and some nonzero α.[2] Hence, the tangency condition of Definition 2.2 is met. □

2.3.2.2 Rational Curves with Normals : Parametric Definition

When both a space curve and its associated normal vector are given in rational parametric form, their equations can be used directly to produce a linear system for interpolation, instead of first computing points and point-normal pairs of the curve. Let $C : (x = G_1(s), y = G_2(s), z = G_3(s))$ be a rational curve of degree d with a normal vector $\mathbf{n}(s) = (n_x(s), n_y(s), n_z(s))$ of degree m. A Hermite interpolating surface $S : f(x,y,z) = 0$ of degree n which contains C with C^1 continuity is computed as follows:

1. containment condition Substitute $(x = G_1(s), y = G_2(s), z = G_3(s))$ into the equation $f(x,y,z) = 0$. This results in, at most, $nd+1$ homogeneous linear equations as in Subsection 2.3.1.2.

2. tangency condition

   (a) Compute $\nabla f(s) = \nabla f(G_1(s), G_2(s), G_3(s))$ and $\mathbf{t}(s) = (\frac{dx}{ds}, \frac{dy}{ds}, \frac{dz}{ds})$. Note that $\mathbf{t} = (t_x, t_y, t_z)$ is the tangent vector to C.

   (b) Select one of the following:

      i. If $t_x \neq 0$, use the equation $f_y(s) \cdot n_z(s) - n_y(s) \cdot f_z(s) = 0$.

[2] From the equation (2.6) we see that $\alpha(x,y,z)$ must not be identically zero along C, for otherwise, $\nabla f = (0,0,0)$ for points along C and would contradict the fact that we chose a nontrivial solution for the surface $S : f = 0$ where ∇f is not identically zero.

ii. If $t_y \neq 0$, use the equation $f_x(s) \cdot n_z(s) - n_x(s) \cdot f_z(s) = 0$.

iii. If $t_z \neq 0$, use the equation $f_x(s) \cdot n_y(s) - n_x(s) \cdot f_y(s) = 0$.

In each case, the numerator of the simplified rational polynomial is set to zero. This yields at most, $(n-1)d + m + 1$ additional homogeneous linear equations in the coefficients of the surface $S : f(x, y, z) = 0$.

3. In total, we obtain a homogeneous system of at most $(2n-1)d + m + 2$ linear equations. Any nontrivial solution of the linear system, for which additionally ∇f is not identically zero for all points of C (that is, the surface S is not singular along the curve C), will represent a surface which Hermite interpolates C.

The proof of correctness of the above algorithm follows from Lemma 2.3 and the following lemma.

Lemma 2.5 If we choose a nontrivial solution for which the resulting Hermite interpolating surface S is not singular along the entire curve C, step 2 guarantees that the tangency condition of Definition 2.2 is met.

Proof: The proof is very similar to that of Lemma 2.4 with minor modifications and is omitted. □

2.4 Geometric Continuity

In the Hermite interpolation algorithm, tangent plane continuity between two surfaces is achieved by making the tangent planes of the two surfaces identical at a point or at all points along a common curve of intersection. This definition of continuity agrees with several other definitions of G^1 geometric continuity given for parametric and implicit algebraic surfaces. De Rose [63] gave a definition of higher orders of geometric continuity between parametric surfaces where two surfaces F_1 and F_2 meet with order k geometric continuity or G^k continuity along a curve C if and only if there exist reparameterizations F_1' and F_2' of F_1 and F_2, respectively, such that all partial derivatives of F_1' and F_2' up to degree k agree along C.

Warren [75] formulated an intuitive definition of G^k continuity between implicit surfaces as following:

Definition 2.4 Two algebraic surfaces $f(x, y, z) = 0$ and $g(x, y, z) = 0$ meet with G^k rescaling continuity at a point p or along an algebraic curve C if and only if there exists two polynomials $a(x, y, z)$ and $b(x, y, z)$, not identically zero at p or along C, such that all derivatives of $af - bg$ up to degree k vanish at p or along C.

This formulation is more general than just making all the partials of $f(x, y, z) = 0$ and $g(x, y, z) = 0$ agree at a point or along a curve. For example [75], consider the intersection of the cone $f(x, y, z) = xy - (x + y - z)^2 = 0$ and the plane $g(x, y, z) = x = 0$ along the line defined by two planes $x = 0$ and $y = z$. It is not hard to see that these two surfaces meet smoothly along the line since the normals to $f(x, y, z) = 0$ at each point on the line are scalar multiples of those to $g(x, y, z) = 0$. But, this scale factor is a function of z. Situations like this are thus corrected by allowing multiplication by rescaling polynomials, not identically zero along an intersection curve. Note that multiplication of a surface by polynomials nonzero along a curve does not change the geometry of the surface in the neighborhood of the curve. In [26], Garrity et al. showed that both definitions of geometric continuity for a parametric and an implicit surface are equivalent by introducing the concept of a manifold which describes an intrinsic and local property of a surface.

The definition for G^0 rescaling continuity corresponds to the containment definition in Section 2.1. The following lemma shows that the C^1 continuity definition in Section 2.1 agrees with the G^1 rescaling continuity definition.

Lemma 2.6 G^1 rescaling continuity between $f(x, y, z) = 0$ and $g(x, y, z) = 0$ at a common point p or along a common curve C corresponds to $f(x, y, z) = 0$ and $g(x, y, z) = 0$ having common tangent planes at p or along every point of C.

Proof: The requirement for G^1 rescaling continuity is that there exist $a(x, y, z)$ and $b(x, y, z)$, not identically zero at p or along C, such that

$$\frac{\partial(af - bg)}{\partial x} = a_x f + a f_x - b_x g - b g_x$$

$$
\begin{aligned}
&= 0 \quad at\ p\ or\ along\ C, \\
\frac{\partial(af-bg)}{\partial y} &= a_y f + a f_y - b_y g - b g_y \\
&= 0 \quad at\ p\ or\ along\ C, \\
\frac{\partial(af-bg)}{\partial z} &= a_z f + a f_z - b_z g - b g_z \\
&= 0 \quad at\ p\ or\ along\ C.
\end{aligned}
$$

Since p or C is contained in both f and g (that is, $f = g = 0$ at p or along C), the requirement becomes

$$
\begin{aligned}
af_x &= bg_x \\
af_y &= bg_y \\
af_z &= bg_z,
\end{aligned}
$$

which means $(f_x, f_y, f_z) = \frac{b}{a}(g_x, g_y, g_z)$ at p or along C. Hence, f and g are required to have common tangent planes at p or along C. □

The correctness proofs in Section 2.2 and Section 2.3 imply that Hermite interpolation finds all the algebraic surfaces which have common tangent planes at a point or a curve. It also yields the following theorem.

Theorem 2.2 Hermite interpolation finds all the algebraic surfaces F which meet a surface H at a point p or along a curve C on H with G^1 rescaling continuity.

A family of algebraic surfaces F as in the above theorem can be constructed in the Hermite interpolation framework of Section 2.3 as follows. Given a surface H and a point p or curve C on H, defined implicitly or parametrically, the input to the Hermite interpolation algorithm is the point p or the curve C and the normal vector to p or C obtained directly from the ∇H, evaluated at p or along C. The algorithm then yields a solution for the coefficients of the family of algebraic surfaces which meet H at p or along C with C^1, tangent plane, or G^1 rescaling continuity. Several examples of this are provided in the next section.

2.5 Computational Aspects of Hermite Interpolation

The basic mechanics of Hermite interpolation for algebraic surfaces, as presented in the algorithms of Section 2.2 and Section 2.3, are

1. geometric properties of a surface to be designed are described in terms of a combination of points, curves, and possibly associated normal vectors,
2. these properties are translated into a homogeneous linear system of equations with extra surface constraints, and
3. nontrivial solutions of the linear system are computed.

In this section, we discuss some computational aspects of Hermite interpolation, and give several examples of algebraic surface design with Hermite interpolation.

2.5.1 On Computing Nontrivial Solutions

As explained before, the Hermite interpolation algorithm converts geometric properties of a surface into a homogeneous linear system:

$$\mathbf{M_I x} = 0 \ (\mathbf{M_I} \in \mathbf{R}^{n_i \times n_v}, \mathbf{x} \in \mathbf{R}^{n_v}),$$

where n_i is the number of equations generated, n_v is the number of unknown coefficients of a surface of degree n ($n_v = \binom{n+3}{3}$), $\mathbf{M_I}$ is a matrix for the linear equations, and $\mathbf{x}$ is a vector whose elements are unknown coefficients of a surface.

In order to solve the linear system in a computationally stable manner, we compute the singular value decomposition (SVD) of $\mathbf{M_I}$ [31]. Hence, $\mathbf{M_I}$ is decomposed as $\mathbf{M_I} = U\Sigma V^T$ where $U \in \mathbf{R}^{n_i \times n_i}$ and $V \in \mathbf{R}^{n_v \times n_v}$ are orthonormal matrices, and $\Sigma = diag(\sigma_1, \sigma_2, \cdots, \sigma_s) \in \mathbf{R}^{n_i \times n_v}$ is a diagonal matrix with diagonal elements $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_s \geq 0$ ($s = min\{n_i, n_v\}$). It can be proved that the rank r of $\mathbf{M_I}$ is the number of the positive diagonal elements of Σ, and that the last $n_v - r$ columns of V span the nullspace of $\mathbf{M_I}$. Hence, the nontrivial solutions of the homogeneous linear system are compactly expressed as: $\{\mathbf{x}(\neq 0) \in \mathbf{R}^{n_v} \mid \mathbf{x} = \sum_{i=1}^{n_v - r} w_i \cdot \mathbf{v}_{r+i},\ where\ w_i \in$

R, *and* $\mathbf{v}_j$ *is the* j*th column of* V}, or $\mathbf{x} = V_{n_v-r}\mathbf{w}$ where $V_{n_v-r} \in \mathbf{R}^{n_v \times (n_v - r)}$ is made of the last $n_v - r$ columns of V, and $\mathbf{w}$ is a $(n_v - r)$ vector for free parameters.

Example 2.1 Computation of Nontrivial Solutions

Let $C : (\frac{2t}{1+t^2}, \frac{1-t^2}{1+t^2}, 0)$, and $\mathbf{n}(t) = (\frac{4t}{1+t^2}, \frac{2-2t^2}{1+t^2}, 0)$, which is from the intersection of a sphere $x^2 + y^2 + z^2 - 1 = 0$ with the plane $z = 0$. To find a surface of degree 2 which Hermite interpolates C, we let $f(x,y,z) = c_1x^2 + c_2y^2 + c_3z^2 + c_4xy + c_5yz + c_6zx + c_7x + c_8y + c_9z + c_{10}$. From the containment condition, we get 5 equations, $c_{10} - c_8 + c_2 = 0$, $2c_7 - 2c_4 = 0$, $2c_{10} - 2c_2 + 4c_1 = 0$, $2c_7 + 2c_4 = 0$, $c_{10} + c_8 + c_2 = 0$, and from the tangency condition, we also get 5 equations, $-2c_9 + 2c_5 = 0$, $-4c_6 = 0$, $-4c_5 = 0$, $4c_6 = 0$, $2c_9 + 2c_5 = 0$. In matrix form,

$$
\mathbf{M_I x} = \begin{pmatrix}
0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 \\
0 & 0 & 0 & -2 & 0 & 0 & 2 & 0 & 0 & 0 \\
4 & -2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 \\
0 & 0 & 0 & 2 & 0 & 0 & 2 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & -2 & 0 \\
0 & 0 & 0 & 0 & 0 & -4 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -4 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 2 & 0
\end{pmatrix}
\begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \\ c_7 \\ c_8 \\ c_9 \\ c_{10} \end{pmatrix} = 0.
$$

The Σ in the SVD of $\mathbf{M_I}$ is $diag(5.657, 4.899, 4.899, 2.828, 2.828, 2.828, 2.0, 1.414, 0.0, 0.0)$.[3]

[3]The subroutine dsvdc of LINPACK was used to compute the SVD of $\mathbf{M_I}$.

Hence, we see that the rank of $\mathbf{M_I}$ is 8, and the null space of $\mathbf{M_I}$ is

$$
\mathbf{x} = w_1\mathbf{v}_9 + w_2\mathbf{v}_{10} = w_1 \begin{pmatrix} 0.0 \\ 0.0 \\ 1.0 \\ 0.0 \\ 0.0 \\ 0.0 \\ 0.0 \\ 0.0 \\ 0.0 \\ 0.0 \end{pmatrix} + w_2 \begin{pmatrix} 0.57735 \\ 0.57735 \\ 0.0 \\ 0.0 \\ 0.0 \\ 0.0 \\ 0.0 \\ 0.0 \\ 0.0 \\ -0.57735 \end{pmatrix}.
$$

The nontrivial solutions are obtained by making sure that the free parameters w_1 and w_2 do not vanish simultaneously. Hence, the Hermite interpolating surface is $f(x,y,z) = 0.57735w_2x^2 + 0.57735w_2y^2 + w_1z^2 - 0.57735w_2 = 0$ which has one degree of freedom in controlling its coefficients. The surface $f(x,y,z) = 0$ can be made to contain a point, say, $(1,0,1)$. That is, $f(1,0,1) = 0.57735w_2 + w_1 - 0.57735w_2 = w_1 = 0$. So, the circular cylinder $f(x,y,z) = 0.57735w_2(x^2 + y^2 - 1) = 0$ is an appropriate Hermite interpolating surface. □

2.5.2 Bounding the Degree of Surfaces

The total number of linear equations generated for a possible algebraic surface of degree n to Hermite interpolate k points with fixed constant normal directions and also to contain, with C^1 continuity, l space curves of degree d with assigned normal directions, varying as a polynomial of degree m, is $3k + (2n + m - 1)dl + 2l$. This number becomes $3k + (2n - 1)dl + ml + 2l$ when all the space curves and associated normal vectors are defined parametrically.

For a given configuration of points, curves, and normal vectors, the above interpolation scheme allows one to both-upper and lower-bound the degree of Hermite interpolating surfaces.

1. Lower Bound Let $r(n)$ be the rank of a homogeneous system of linear equations, obtained from the given geometric configuration and surface degree n. The rank tells us the exact number of independent constraints on the coefficients of the desired algebraic surface of degree n. Dependencies arise from spatial interrelationships of the given points and curves. From the rank, we can conclude that there exists no algebraic surface of a degree less than or equal to n_0 where n_0 is the largest n such that $F(n) < r(n)$ with $F(n) = \binom{n+3}{3} - 1$.

2. Upper Bound Alternatively, the smallest n can be chosen such that $F(n) \geq r(n)$. The nontrivial solutions of the linear system represents a $(F(n) - r(n) + 1)$-parameter family (with $F(n) - r(n)$ degrees of freedom) of algebraic surfaces of degree n which interpolate the given geometric data. We select suitable surfaces from this family, which additionally satisfy our nonsingularity and irreducibility constraints.[4]

One way to apply the Hermite interpolation technique to computation of a lowest degree algebraic surface which has given geometric properties, is to search through the degrees, i.e., from $n = 1, 2, 3, \cdots$ for an interpolating surface. In Chapter 4, we illustrate how the rank can be predicted *a priori*, without generating a linear system and then actually computing its rank, using only topological interrelationships between input curves and normal directions. However, since the dependencies between linear equations do depend on the specific spatial interrelationships of the given points and curves, it is, in general, quite difficult to bound the degree of interpolating surfaces *a priori*. For example, it is possible to design input data, made of an arbitrary number of degree 4 curves with normal directions, which can be interpolated by a quadric surface.

We now enumerate some results in which we lower-bound the degrees of some Hermite interpolating surfaces.

[4]However, some of these interpolating surfaces still might not be suitable for the design application they were intended to benefit. These problems arise when the given points or curves are smoothly interpolated, but, lie on separate real components of the same nonsingular, irreducible algebraic surface.

1. Two skewed lines in space with constant direction normals cannot be Hermite interpolated with nondegenerate quadric surfaces. The only quadric which satisfies both containment and tangency conditions reduces into two planes.

2. Two lines in space with constant direction normals can be Hermite interpolated with a quadric surface if and only if the lines are parallel or intersect at a point, and the normals are not orthogonal to the plane containing them. The quadric is a cylinder when the lines are parallel, and a cone when the lines intersect.

3. The minimum degree of an algebraic surface, which Hermite interpolates two lines in space, one with a constant direction normal, the other with a linearly varying normal is three.

4. Two lines with linearly varying normals can be Hermite interpolated by a quadric in only some special cases. In general, a surface of at least degree three is needed. When quadric surface interpolation is possible, the quadric is either a hyperboloid of one sheet (the two lines may be parallel, intersecting, or skewed) or a hyperbolic paraboloid (the two lines can only be intersecting or skewed).

2.5.3 Examples

In this subsection, we exhibit the method of Hermite interpolation by constructing lowest degree Hermite interpolating surfaces for joining and blending primary surfaces of solid models as well as for fleshing curved wire frame models of physical objects. [5]

Example 2.2 (JOINING 1) A Cubic Surface for Smoothly Joining Two Elliptic Cylinders

[5]The solutions of all the examples in this subsection were obtained using MACSYMA in which Gaussian elimination algorithm is applied. The reason was to express solutions more clearly, however, the singular value decomposition algorithm was used in our implementation. Of course, the solution spaces are the same whichever method to be used in computing the nullspace, although the bases that span the vector subspace are different.

Consider computing a lowest degree surface which can smoothly join two truncated elliptic cylinders $CYL_1 : (y+1)^2 + \frac{z^2}{4} - 1 = 0$ for $x \leq -2$ and $CYL_2 : 25z^2 + 36y^2 - 96xy + 64x^2 - 100 = 0$ for $3x + 4y \geq 0$. Here, we illustrate the Hermite interpolation technique which not only computes the unique cubic interpolating surface but also proves that degree three is the lowest for an algebraic surface to satisfy the smooth-join requirement for this configuration. We take an ellipse $C_1 : (-2, \frac{-2t^2}{1+t^2}, \frac{4t}{1+t^2})$ on CYL_1 with the associated rational normal $\mathbf{n}_1(t) : (0, \frac{2-2t^2}{1+t^2}, \frac{2t}{1+t^2})$ and another ellipse $C_2 : (\frac{-4+4t^2}{5+5t^2}, \frac{3-3t^2}{5+5t^2}, \frac{4t}{1+t^2})$ on CYL_2 with the associated rational normal $\mathbf{n}_2(t) : (\frac{-160+160t^2}{1+t^2}, \frac{120-120t^2}{1+t^2}, \frac{200t}{1+t^2})$. Both C_1 and C_2's normals are respectively chosen in the same directions as the gradients of their corresponding surfaces CYL_1 and CYL_2. This ensures that any Hermite interpolating surface for C_1 and C_2 will also meet CYL_1 and CYL_2 smoothly along these curves. A degree two algebraic surface does not suffice for Hermite interpolation, since the rank of the constructed linear system is greater than 9 which is the degrees of freedom of a quadric surface. (Note that a quadric surface has 10 coefficients.) Next, as a possible Hermite interpolant, consider a degree three algebraic surface with 20 coefficients. Applying the Hermite interpolation algorithm of Subsection 2.3.2.2 to the curves results in 26 linear equations (28 equations are supposed to be generated, but 2 of the 28 are degenerate.). The rank of this linear system is 19, and thus there is a unique cubic Hermite interpolating surface, which is $f(x,y,z) = r_1(2yz^2 - xz^2 - 5z^2 + 8y^3 - 4xy^2 - 4y^2 + 8x^2y + 24xy - 8y - 4x^3 - 11x^2 + 4x + 20)$. See Figure 2.1. □

Example 2.3 (JOINING 2) A Quartic Surface for Smoothly Joining Three Circular Cylinders

Consider computing a lowest degree surface which smoothly joins three truncated orthogonal circular cylinders $CYL_1 : x^2 + y^2 - 1 = 0$ for $z \geq 2$, $CYL_2 : y^2 + z^2 - 1 = 0$ for $x \geq 2$, and $CYL_3 : z^2 + x^2 - 1 = 0$ for $y \geq 2$.

In [76], a degree five surface is found for joining these cylinders. After applying the Hermite interpolation algorithm, we find out that the minimum degree for such joining surfaces is 4, and we get a 2-parameter (one degree of freedom) family of algebraic surfaces.

As before, we take a circle $C_1 : (\frac{2t}{1+t^2}, \frac{1-t^2}{1+t^2}, 2)$ on CYL_1 with the associated rational normal $\mathbf{n}_1(t) : (\frac{4t}{1+t^2}, \frac{2-2t^2}{1+t^2}, 0)$, the circle $C_2 : (2, \frac{2t}{1+t^2}, \frac{1-t^2}{1+t^2})$ on CYL_2 with the associated rational normal $\mathbf{n}_2(t) : (0, \frac{4t}{1+t^2}, \frac{2-2t^2}{1+t^2})$, and the circle $C_3 : (\frac{2t}{1+t^2}, 2, \frac{1-t^2}{1+t^2})$ on CYL_3 with the associated rational normal $\mathbf{n}_3(t) : (\frac{4t}{1+t^2}, 0, \frac{2-2t^2}{1+t^2})$. Again, all C_1, C_2 and C_3's normals are respectively chosen in the same direction as the gradients of their corresponding surfaces CYL_1, CYL_2, and CYL_3. This ensures that any Hermite interpolating surface for C_1, C_2, and C_3 will also meet CYL_1, CYL_2, and CYL_3 smoothly along these curves. A degree three algebraic surface does not suffice for Hermite interpolation, since the rank of the resulting linear system is greater than 19. Next, as a possible Hermite interpolant, consider a degree four algebraic surface with 35 coefficients, and 34 degrees of freedom. Applying the Hermite interpolation algorithm to the curves results in 52 equations. The rank of this linear system is 33, and thus there is a 2-parameter family of quartic Hermite interpolating surfaces, which is $f(x,y,z) = r_1z^4 + \frac{r_2+10r_1}{12}yz^3 + \frac{r_2+10r_1}{12}xz^3 - \frac{r_2+10r_1}{3}z^3 + 2r_1y^2z^2 + \frac{r_2+10r_1}{12}xyz^2 - \frac{r_2+10r_1}{3}yz^2 + 2r_1x^2z^2 - \frac{r_2+10r_1}{3}xz^2 + r_2z^2 + \frac{r_2+10r_1}{12}y^3z + \frac{r_2+10r_1}{12}xy^2z - \frac{r_2+10r_1}{3}y^2z + \frac{r_2+10r_1}{12}x^2yz - \frac{r_2+10r_1}{3}xyz + \frac{r_2+10r_1}{4}yz + \frac{r_2+10r_1}{12}x^3z - \frac{r_2+10r_1}{3}x^2z + \frac{r_2+10r_1}{4}xz + \frac{r_2+10r_1}{3}z + r_1y^4 + \frac{r_2+10r_1}{12}xy^3 - \frac{r_2+10r_1}{3}y^3 + 2r_1x^2y^2 - \frac{r_2+10r_1}{3}xy^2 + r_2y^2 + \frac{r_2+10r_1}{12}x^3y - \frac{r_2+10r_1}{3}x^2y + \frac{r_2+10r_1}{4}xy + \frac{r_2+10r_1}{3}y + r_1x^4 - \frac{r_2+10r_1}{3}x^3 + r_2x^2 + \frac{r_2+10r_1}{3}x + \frac{5r_1-7r_2}{3}$.

An instance of this family ($r_1 = 1$, $r_2 = 10$) is shown in Figure 2.2. It should be noted that every surface in the computed family is not always appropriate for geometric modeling. The quartic surface in Figure 2.3 is one used in Figure 2.2. On the other hand, the surface in Figure 2.4, which is not useful for geometric modeling, is also in the same family with $r_1 = 1$ and $r_2 = -1$. □

Example 2.4 (JOINING 3) A Quartic Surface for Smoothly Joining Four Circular Cylinders

In this example, we compute a lowest degree surface which smoothly joins four truncated parallel circular cylinders defined by $CYL_1 : y^2 + z^2 - 1 = 0$ for $x \geq 2$,

$CYL_2 : y^2 + z^2 - 1 = 0$ for $x \le -2$, $CYL_3 : (y-4)^2 + z^2 - 1 = 0$ for $x \ge 2$, and $CYL_4 : (y-4)^2 + z^2 - 1 = 0$ for $x \le -2$.

The Hermite interpolation technique indicates that the minimum degree for such a joining surface is 4, and computes a 2-parameter (one degree of freedom) family of algebraic surfaces which is $f(x,y,z) = \frac{r_1}{14}z^4 + \frac{r_1}{7}y^2z^2 - \frac{4r_1}{7}yz^2 + r_1z^2 + \frac{r_1}{14}y^4 - \frac{4r_1}{7}y^3 + r_1y^2 + \frac{4}{7}r_1y + \frac{14r_2+15r_1}{224}x^4 - \frac{14r_2+15r_1}{28}x^2 + r_2$. An instance of this family ($r_1 = 392$, $r_2 = -868$) is shown in Figure 2.5. □

Example 2.5 (BLENDING 1) Hyperboloid Patches for Blending Two Perpendicular Cylinders

The case of two circular cylinders is a common test case for blending algorithms. Various different ways have been given, (for example, see [35, 50, 76]) for computing a suitable surface which smoothes or blends the intersection of two equal radius cylinders, $CYL_1 : x^2 + z^2 - 1 = 0$ and $CYL_2 : y^2 + z^2 - 1 = 0$. We consider an ellipse C_1 on CYL_1 (it is the intersection with the plane $3x + y = 0$), defined parametrically, $C_1 : (\frac{2t}{1+t^2}, \frac{-6t}{1+t^2}, \frac{1-t^2}{1+t^2})$ with the associated rational normal $\mathbf{n}_1(t) = (\frac{4t}{1+t^2}, 0, \frac{2-2t^2}{1+t^2})$, and the ellipse C_2 on CYL_2 defined implicitly, $C_2 : ((y^2 + z^2 - 1 = 0, x + 3y = 0)$ with the associated normal $\mathbf{n}_2(x,y,z) = (0, 2y, 2z)$. As a possible Hermite interpolant, we consider a degree two algebraic surface. Applying the method of Subsection 2.3.2.2, to C_1 results in 8 equations, 5 from the containment condition and 3 from the tangency condition. (5 equations are supposed to be generated, but 2 of these turn out to be degenerate). For C_2, we use the method of Subsection 2.3.2.1, and first compute L_c = {(0, 0, 1), (-3, 1, 0), (3, -1, 0), (-2.4, 0.8, -0.6), (2.4, -0.8, -0.6)} and L_t = {[(0, 0, 1), (0, 0, 2)], [(-3, 1, 0), (0, 2, 0)], [(3, -1, 0), (0, -2, 0)], [(-2.4,0.8,-0.6), (0,1.6,-1.2)], [(2.4,-0.8,-0.6), (0,-1.6,-1.2)]}. From these lists, we get 10 equations, 5 from the containment condition and another 5 from the tangency condition. Hence, overall the linear system consists of 10 unknowns and 18 equations. The rank of this system is 9, and hence we get the unique surface solution $f_1(x,y,z) = r_1(x^2 + y^2 - 8z^2 + 6xy + 8 = 0)$. This quadric satisfies both the nonsingularity and irreducibility constraints. It is a

hyperboloid of one sheet and the lowest degree surface which blends, together with a symmetric hyperboloid $f_2(x,y,z) = r_1(x^2 + y^2 - 8z^2 - 6xy + 8 = 0)$, the intersection of the two cylinders. See Figure 2.6. □

Example 2.6 (BLENDING 2) A Quartic Surface for Blending Two Elliptic Cylinders

In this example, we compute a lowest degree surface which blends two perpendicular elliptic cylinders. We have seen a quadric blending of the circular cylinders in Example 2.5. Here, we try a quartic blending surface by taking different types of input curves.

Input to Hermite interpolation is defined by $CYL_1 : y^2 + 4z^2 - 4 = 0$ for $x \ge 1$, $CYL_2 : y^2 + 4z^2 - 4 = 0$ for $x \le -1$, $CYL_3 : 9x^2 + y^2 - 9 = 0$ for $z \ge 1$, and $CYL_4 : 9x^2 + y^2 - 9 = 0$ for $z \le -1$.

The Hermite interpolation algorithm proves that 4 is the minimum degree for such a blending surface, and generates a linear system with 72 equations of rank 33. The 2-parameter (one degree of freedom) family of algebraic surfaces is $f(x,y,z) = r_1z^4 - \frac{8r_2+81r_1}{72}y^2z^2 - \frac{8r_2+81r_1}{8}x^2z^2 + \frac{8r_2+85r_1}{8}z^2 - \frac{8r_2+99r_1}{288}y^4 - \frac{8r_2+81r_1}{32}x^2y^2 + \frac{104r_2+1053r_1}{288}y^2 + \frac{81r_1}{16}x^4 + r_2x^2 - \frac{10r_2+85r_1}{16}$. An instance of this family ($r_1 = 1$, $r_2 = 2$) is shown in Figure 2.7. □

Example 2.7 (FLESHING 1) A Quartic Surface for Smoothing a Corner of a Table

Interpolation can be useful in generating an algebraic corner blending surface. We look for a quartic surface $S : f(x,y,z) = 0$ which smooths out the corner of a table. In fact, S is a fleshing surface of a wire frame made of the edges of the corner: $C_1 : ((y^2 + z^2 - 25 = 0, x = 0)$, and $C_2 : ((x^2 + z^2 - 25 = 0, y = 0)$. Each wire is associated with a normal vector which is chosen in the same direction as the gradients of the side of table, the cylinder in C_1 and C_2. That is, $\mathbf{n}_1(x,y,z) = (0, 2y, 2z)$, and $\mathbf{n}_2(x,y,z) = (2x, 0, 2z)$. [6]

[6] Of course, a sphere, which is quadratic, can do the job. But, we deliberately chose the degree four to give the idea of a family of interpolating surfaces. Also, this higher degree algebraic surface is more flexible for shape control.

The interpolation matrix $\mathbf{M_I}$, produced by Hermite interpolation, is a 32×35 matrix (32 linear equations and 35 coefficients for a quartic surface) whose rank turns out to be 24. The nullspace of $\mathbf{M_I}$ is of dimension 11 represented by a family of quartic surfaces $f(x,y,z) = r_1 z^4 + (r_2 y + r_6 x + 5r_4) z^3 + (r_3 y^2 + (r_7 x + 5r_8) y + r_{10} x^2 + 5r_{11} x - 25r_9 - 25r_1) z^2 + (r_2 y^3 + (r_6 x + 5r_4) y^2 + (r_2 x^2 - 25r_2) y + r_6 x^3 + 5r_4 x^2 - 25r_6 x - 125r_4) z + (r_3 - r_1) y^4 + (r_7 x + 5r_8) y^3 + (r_5 x^2 + 5r_{11} x - 25r_9 - 25r_3 + 25r_1) y^2 + (r_7 x^3 + 5r_8 x^2 - 25r_7 x - 125r_8) y + (r_{10} - r_1) x^4 + 5r_{11} x^3 + (-25r_9 - 25r_{10} + 25r_1) x^2 - 125r_{11} x + 625r_9$. An instance $f(x,y,z) = -1250 - x^4 - y^4 - x^2z^2 - y^2z^2 + 50z^2 + 75y^2 + 75x^2$ in this family is shown with the table in Figure 2.8. □

2.6 Summary

In this chapter, we presented the Hermite interpolation algorithm for algebraic surfaces. With the algorithm, it was possible to characterize the class of algebraic surfaces of a fixed degree that have given positional and tangential properties, in terms of the nullspace of a matrix. The rank of the matrix, produced by the algorithm, was used in proving existence or nonexistence of algebraic surfaces of a given degree. We also considered computational aspects of the algorithm, and illustrated the usefulness of the algorithm from several examples.

As a result of application of the Hermite interpolation algorithm, a class or family of algebraic surfaces is computed, when the degree is high enough, where the family is expressed as a subspace of $\mathbf{R}^{n_v}$ of dimension $n_v - r$. In geometric design, it is hoped that an appropriate surface can be selected interactively and intuitively from the family. Choosing a proper surface from the family is equivalent to assigning proper values to free parameters. In the Chapter 3, we consider how an instance surface is selected interactively with geometric intuition.

Figure 2.1 Smooth Joining of Two Cylinders with a Cubic Surface

Figure 2.2 Smooth Joining of Three Cylinders with a Quartic Surface

Figure 2.3 A "Good" Quartic Surface

Figure 2.4 A "Bad" Quartic Surface

Figure 2.5 Smooth Joining of Four Cylinders with a Quartic Surface

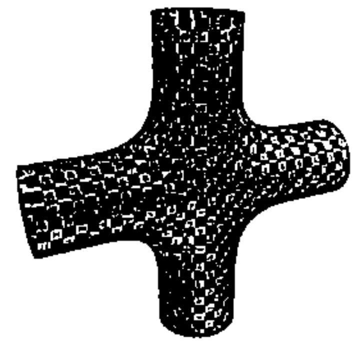

Figure 2.6 Smooth Blending of Two Cylinders with a Quadric Surface

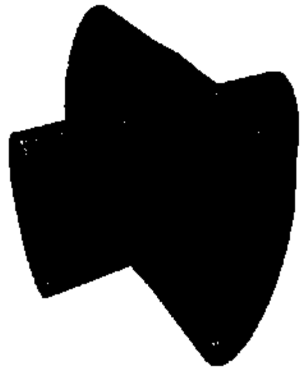

Figure 2.7 Smooth Blending of Two Cylinders with a Quartic Surface

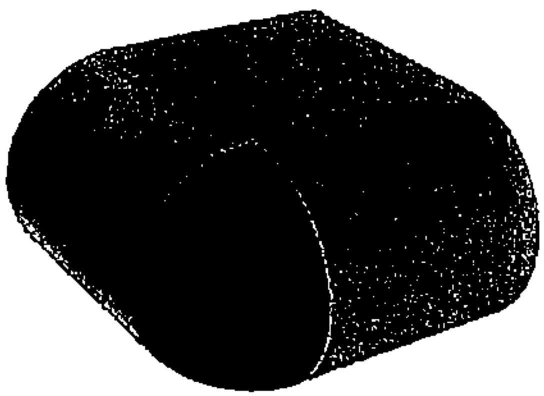

Figure 2.8 Table Corner Blending with a Quartic Surface

3. A COMPUTATIONAL MODEL FOR ALGEBRAIC SURFACE FITTING

In the foregoing chapter, we described how a class of algebraic surfaces with given geometric properties is characterized in terms of the nullspace of a matrix. In practice, an instance surface must be interactively selected from the class with geometric intuition such that the selected surface has desirable properties (for example, no self-intersections) within a necessary region. In this chapter, we present a computational model for the algebraic surface fitting problem. This model is designed to effectively choose an instance surface among many possible ones in the family. We also consider how the Bernstein-Bézier basis can be used to help control the shape of the selected surface intuitively.

Fitting of algebraic curves (primarily lines and conics) has been considered extensively by many authors [3, 15, 17, 27, 51, 65]. A good exposition of exact and least squares fitting of algebraic curves and surfaces through given data points, was presented by Pratt [59]. Sederberg [68] presented the idea of C^0 interpolation of data points and curves with implicit algebraic surfaces. These previous works on interpolation are extended by our Hermite interpolation algorithm which can handle tangential information (C^1) as well as positional information (C^0).

The model we consider in this chapter is based upon a proper normalization of coefficients of algebraic surfaces as well as least squares approximation and Hermite interpolation. The mathematical model we derive is a constrained minimization problem of the form:

$$\begin{aligned} &\textit{minimize} && \mathbf{x}^T \mathbf{M_A}^T \mathbf{M_A} \mathbf{x} \\ &\textit{subject to} && \mathbf{M_I} \mathbf{x} = 0 \\ & && \mathbf{x}^T \mathbf{x} = 1, \end{aligned}$$

where $\mathbf{M_I} \in \mathbb{R}^{n_i \times n_v}$ and $\mathbf{M_A} \in \mathbb{R}^{n_a \times n_v}$ are matrices for interpolation and least squares approximation, respectively, and $\mathbf{x} \in \mathbb{R}^{n_v}$ is a vector containing coefficients of an algebraic surface.

In Section 3.1, we consider interpolation, least squares approximation, and normalization in detail, and explain how the minimization problem is derived. Then, in Section 3.2, compact computational algorithms are described with examples. In Section 3.3, we consider how the geometric properties of the Bernstein-Bézier basis are used for shape control of algebraic surfaces which are contained in a family computed by Hermite interpolation.

3.1 Matrices for Interpolation, Approximation and Normalization

3.1.1 Interpolation

In Chapter 2, we explained how an interpolation matrix $\mathbf{M_I} \in \mathbb{R}^{n_i \times n_v}$ is generated by the Hermite interpolation algorithm, and how its nullspace is computed. Only when the rank r of $\mathbf{M_I}$ is less than the number of the coefficients n_v, does there exist a nontrivial solution to the linear system. All vectors except $\mathbf{0}$ in the nullspace of $\mathbf{M_I}$ form a family of algebraic surfaces, satisfying the given input specification, whose coefficients are expressed by homogeneous combinations of q free parameters where $q = n_v - r$ is the dimension of the nullspace. In Hermite interpolation, tangent plane or C^1 continuity is achieved by forcing normals of tangent planes of a surface to be parallel to those of given points or space curves.

For some applications of geometric modeling, such as ship hull design, however, more than tangent plane continuity may be desirable. As explained in Section 2.4, the concept of smoothness is generalized by defining <u>higher order geometric continuity</u>. In Section 2.4, it was shown that the Hermite interpolation algorithm finds all algebraic surfaces of a given degree meeting each other with C^1 or G^1 rescaling continuity. Even though we are currently unable to translate geometric specifications for G^k rescaling continuity ($k \geq 2$) into a matrix $\mathbf{M_I}$ whose nullspace captures <u>all</u>

G^k rescaling continuous surfaces of a fixed degree, we can generate an interpolation matrix $\mathbf{M_I}$ whose nullspace captures a subset of the whole class. This technique is based on the following theorem whose proof is found in [75].

Theorem 3.1 Let $g(x,y,z)$ and $h(x,y,z)$ be distinct, irreducible polynomials. If the surfaces $g(x,y,z) = 0$ and $h(x,y,z) = 0$ intersect transversally in a single irreducible curve C, then any algebraic surface $f(x,y,z) = 0$ that meets $g(x,y,z) = 0$ with G^k rescaling continuity along C must be of the form $f(x,y,z) = \alpha(x,y,z)g(x,y,z) + \beta(x,y,z)h^{k+1}(x,y,z)$. If $g(x,y,z) = 0$ and $h(x,y,z) = 0$ share no common components at infinity, then the degree of $\alpha(x,y,z)g(x,y,z) \leq$ degree of $f(x,y,z)$ and the degree of $\beta(x,y,z)h^{k+1}(x,y,z) \leq$ degree of $f(x,y,z)$.

For given curves C_i, $i = 1, \cdots, l$, which are the transversal intersection of given algebraic surfaces $g_i(x,y,z) = 0$ and $h_i(x,y,z) = 0$, respectively, a surface $f(x,y,z) = 0$ containing space curves C_i with G^k rescaling continuity can be constructively obtained by the relations

$$f(x,y,z) = \alpha_i(x,y,z)g_i(x,y,z) + \beta_i(x,y,z)h_i^{k+1}(x,y,z), \quad i = 1, \cdots, l. \qquad (3.1)$$

Since g_i and h_i are known surfaces, the unknown coefficients are those of f, α_i and β_i. When the hypothesis of Theorem 3.1 is met, the polynomials α_i and β_i are of bounded degrees. From (3.1), we see that these unknown coefficients form a system of linear equations, yielding an interpolation matrix $\mathbf{M_I}$ for G^k rescaling continuity.

Example 3.1 Algebraic Surfaces with G^2 and G^3 Rescaling Continuity

Consider a space curve C defined by the two equations $f_1(x,y,z) = x^2+2y^2+2z^2-2 = 0$ and $f_2(x,y,z) = x = 0$. We compute a cubic surface $f_3(x,y,z) = 0$ which meets f_1 along C with G^2 rescaling continuity as follows: A general cubic algebraic surface is given by $f_3(x,y,z) = c_1x^3+c_2y^3+c_3z^3+c_4x^2y+c_5xy^2+c_6x^2z+c_7xz^2+c_8y^2z+c_9yz^2+c_{10}xyz+c_{11}x^2+c_{12}y^2+c_{13}z^2+c_{14}xy+c_{15}yz+c_{16}xz+c_{17}x+c_{18}y+c_{19}z+c_{20} = 0$. Equating the generic f_3 for G^2 rescaling continuity as explained, we have $f_3(x,y,z) = (r_1x +$

$r_2y+r_3z+r_4)f_1(x,y,z)+r_5f_2(x,y,z)^3$, yielding the linear equations: $c_1-r_1-r_5=0$, $c_2-2r_2=0$, $c_3-2r_3=0$, $c_4-r_2=0$, $c_5-2r_1=0$, $c_6-r_3=0$, $c_7-2r_1=0$, $c_8-2r_3=0$, $c_9-2r_2=0$, $c_{10}=0$, $c_{11}-r_4=0$, $c_{12}-2r_4=0$, $c_{13}-2r_4=0$, $c_{14}=c_{15}=c_{16}=0$, $c_{17}+2r_1=0$, $c_{18}+2r_2=0$, $c_{19}+2r_3=0$, $c_{20}+2r_4=0$ in the unknowns $c_1,\cdots,c_{20}$ and $r_1,\cdots,r_5$. By eliminating $r_1,\cdots,r_5$ from the equations, we get a homogeneous linear system $\mathbf{M_I x}=0$ in terms of f_3's coefficients $c_1,\cdots,c_{20}$. An instance cubic surface ($r_1=1$, $r_2=-1$, $r_3=1$, $r_4=1$, $r_5=2$) $f_3(x,y,z)=2z^3-2yz^2+2xz^2+2z^2+2y^2z+x^2z-2z-2y^3+2xy^2+2y^2-x^2y+2y+3x^3+x^2-2x-2$ is shown in Figure 3.1.

In the same way, we can compute a quartic surface $f_4(x,y,z)=16z^4-16yz^3+32xz^3+32z^3+16y^2z^2-16xyz^2-16yz^2+24x^2z^2+32xz^2-16y^3z+32xy^2z+32y^2z-8x^2yz+16yz+32x^3z+16x^2z-32xz-32z-9y^4-16xy^3-16y^3+16x^2y^2+32xy^2+16y^2-8x^3y-8x^2y+16xy+16y+24x^4+32x^3-8x^2-32x-16$ which meets f_3 with G^3 rescaling continuity along the curve defined by f_3 and $f_5(x,y,z)=y=0$. □

3.1.2 Normalization

To compute an algebraic surface that approximates given data in the least squares sense, one needs to first define a distance metric which is meaningful and computationally efficient. The Geometric distance of a point p from a surface $S: f(x,y,z)=0$ is the Euclidean distance from p to the nearest point on S. However, computing the geometric distance from a point to an algebraic surface itself entails a computationally expensive procedure, and when the metric is adopted for surface approximation, the problem becomes even more intractable. A commonly used approximation to geometric distance from a point to implicitly represented algebraic curves and surfaces is the value $f(p)$, called algebraic distance. Since $cf(x,y,z)=0$ represents the same surface for all $c\neq 0$, the coefficients of f are first normalized such that $f(x,y,z)=0$ is a representation of the equivalence class $\{cf(x,y,z)=0|c\neq 0\}$.

The normalization we shall use is a quadratic normalization of the form $\mathbf{x}^T\mathbf{x}=1$. While some variations [17, 59, 65] of a quadratic normalization have been proposed in fitting scattered planar data with conic curves, it is not easily seen how different quadratic or nonquadratic normalizations affect surface fitting when the degree of a surface is greater than 2, a case of considerable interest for geometric modeling. The normalization $\mathbf{x}^T\mathbf{x}=1$ is a sphere in the coefficient vector space, and does not have singularities. That is, this normalization eliminates only the degenerate surface with all zero coefficients from possible solutions. This normalization also leads to compact and efficient algorithms for surface fitting. It remains open to determine a generalized quadratic normalization of the form $\mathbf{x}^T\mathbf{M_N x}=1$, where $\mathbf{M_N}$ is no longer the identity matrix, with good qualities for surface fitting.

3.1.3 Least Squares Approximation

When the rank r of an interpolation matrix $\mathbf{M_I}\in\mathbf{R}^{n_r\times n_v}$ is less than n_v, the dimension of the coefficient vector, there exists a family of algebraic surfaces which satisfy the given geometric constraints where the underdetermined coefficients can be homogeneously expressed in terms of q $(=n_v-r)$ free parameters. An important problem is to select a surface interactively and intuitively which is most appropriate for a given application. Selecting an instance surface from the family is equivalent to assigning values to each of the q parameters.

Least squares approximation can help choose a surface and control its shape. When there are some degrees of freedom left, we may additionally specify a set of points or curves around given input data, which approximately describes a desirable surface. The final fitting surface can be obtained by consuming the remaining degrees of freedom via least squares approximation to the additional data set.

The algebraic distance $f(p)$ is straightforward to compute and, in case the data point is close to a surface, approximates its geometric distance quite well. When the sum of squares of the algebraic distances of all points is minimized, one obtains algebraically nice solutions. Each row of an approximation matrix $\mathbf{M_A}$ is computed by evaluating each term in $f(x,y,z)$ at the corresponding point. Then, the sum of squares, minimized in least squares approximation, is expressed as $\| \mathbf{M_A x} \|^2 = \mathbf{x}^T\mathbf{M_A}^T\mathbf{M_A x}$.

In addition to the algebraic distance, we also consider a nonalgebraic distance metric $\frac{|f(p)|}{\|\nabla f(p)\|}$. Sampson [65] proposed its use, in conic curve fitting, as a distance measure which is, in fact, the first order approximation to geometric distance. With this metric, a better approximation to geometric distance is achievable, however, only at the expense of several iterative applications of least squares approximation. We give an example of application of this metric to a quadratic surface fitting problem.

Containment of points and curves is not the only way to produce $\mathbf{M_A}$. The matrix for higher order interpolation can be used as an approximation matrix when a surface is not flexible enough for the higher order interpolation. For instance, in Example 3.1, suppose that there are more points that must be contained in a fitting surface. Then, it may not be possible that a cubic surface $f_3(x, y, z) = 0$ not only meets f_1 with G^2 rescaling continuity but also contains the extra points. If C^1 continuity is permissible, we can generate $\mathbf{M_I}$ for containment of the intersection curve (C^1), and the points (C^0) using the Hermite interpolation technique, and the matrix produced in the example can be used as $\mathbf{M_A}$. That is, the remaining degrees of freedom, after C^1 interpolation, are used so that the G^2 rescaling continuity requirement is satisfied as much as possible. However, more effort must be made to see clearly how this algebraic interpolation and approximation technique affects the resulting algebraic surface geometrically.

3.2 Computing Optimum Solutions

In the foregoing section, we explained how the algebraic surface fitting problem is transformed into a constrained minimization problem of the form:

$$\begin{array}{ll} \textit{minimize} & \mathbf{x}^T\mathbf{M_A}^T\mathbf{M_A x} \\ \textit{subject to} & \mathbf{M_I x} = 0 \\ & \mathbf{x}^T\mathbf{x} = 1, \end{array}$$

where $\mathbf{M_A} \in \mathbf{R}^{n_a \times n_v}$, $\mathbf{M_I} \in \mathbf{R}^{n_i \times n_v}$ and $\mathbf{x} \in \mathbf{R}^{n_v}$. This minimization problem appears in some applications [30]. In [29], a solution was obtained by applying Householder transformations to $\mathbf{M_I}$ to obtain its orthogonal decomposition, and then directly computing eigenvalues and eigenvectors of a reduced matrix. In this section, we consider some cases of the surface fitting problems which arise in geometric design, and describe different algorithms where the singular value decomposition (SVD) algorithm is applied to computation of eigenvalues and eigenvectors. In each case, we assume a quadratic normalization constraint which always guarantees a nontrivial solution.

3.2.1 Interpolation and Approximation

In Subsection 2.5.1, the nullspace is expressed as $\mathbf{x} = V_{n_v-r}\mathbf{w}$ where $V_{n_v-r} \in \mathbf{R}^{n_v \times (n_v - r)}$ is made of the last $n_v - r$ columns of the right singular vectors V, and $\mathbf{w}$ is a $(n_v - r)$-vector whose elements are free parameters appearing as coefficients of a family of algebraic surfaces. A final surface is selected by providing proper values for $\mathbf{w}$, by a shape control process. One method is for a user to specify an approximate shape of a desired surface with an additional set of points or curves and let a geometric modeling system automatically find a solution vector $\mathbf{w}$. Then what the system needs to solve efficiently is a constrained least squares problem: *minimize* $\mathbf{x}^T\mathbf{M_A}^T\mathbf{M_A x}$ *subject to* $\mathbf{M_I x} = 0$ *and* $\mathbf{x}^T\mathbf{x} = 1$.

The solution to this minimization problem can be expressed analytically in closed form. From the interpolation requirement, we get $\mathbf{x} = V_{n_v-r}\mathbf{w}$ as before. Hence, after removing the linear constraints, we get to the problem *minimize* $\mathbf{w}^T V_{n_v-r}^T \mathbf{M_A}^T\mathbf{M_A} V_{n_v-r}\mathbf{w}$ *subject to* $\mathbf{w}^T\mathbf{w} = 1$. Note that $V_{n_v-r}^T \mathbf{M_A}^T\mathbf{M_A} V_{n_v-r}$ is a positive definite matrix, and this problem is equivalent to minimizing the ratio of two quadratics $R(\mathbf{w}) = (\mathbf{w}^T V_{n_v-r}^T \mathbf{M_A}^T\mathbf{M_A} V_{n_v-r}\mathbf{w})/(\mathbf{w}^T\mathbf{w})$. $R(\mathbf{w})$, which is known as Rayleigh's quotient, is minimized by the first eigenvector $\mathbf{w} = \mathbf{w}_{min}$ of $V_{n_v-r}^T \mathbf{M_A}^T\mathbf{M_A} V_{n_v-r}$, and its minimum value is the smallest eigenvalue λ_{min} [71].

Instead of computing the eigenvectors and eigenvalues of $V_{n_v-r}^T \mathbf{M_A}^T \mathbf{M_A} V_{n_v-r}$ directly as in [29], we apply singular value decomposition to $\mathbf{M_A} V_{n_v-r}$ without computing $V_{n_v-r}^T \mathbf{M_A}^T \mathbf{M_A} V_{n_v-r}$ explicitly [43]. This leads to a numerically cheaper computation. Here, we assume that $n_a \geq n_v - r$, and that the rank of $\mathbf{M_A} V_{n_v-r}$ is $n_v - r$. (That is, there are enough linear constraints to consume the remaining degrees of freedom.) Then, $\mathbf{M_A} V_{n_v-r} = P \Omega Q^T$ where $P \in \mathbf{R}^{n_a \times n_a}$ and $Q \in \mathbf{R}^{(n_v-r)\times(n_v-r)}$ are orthonormal matrices, and $\Omega = diag(\omega_1, \omega_2, \cdots, \omega_{n_v-r}) \in \mathbf{R}^{n_a \times (n_v-r)}$ with $\omega_1 \geq \omega_2 \geq \cdots \geq \omega_{n_v-r} > 0$.

Now,

$$
\begin{aligned}
\lambda \mathbf{w} &= V_{n_v-r}^T \mathbf{M_A}^T \mathbf{M_A} V_{n_v-r} \mathbf{w} \\
&= Q \Omega^T P^T P \Omega Q^T \mathbf{w} \\
&= Q \Omega^T \Omega Q^T \mathbf{w}.
\end{aligned}
$$

Here, $\Omega^T \Omega$ is a $(n_v - r) \times (n_v - r)$ diagonal matrix with a diagonal entry $\omega_i^2 > 0$, $i = 1, 2, \cdots, (n_v - r)$. Then, from the above equation, $\Omega^T \Omega (Q^T \mathbf{w}) = \lambda (Q^T \mathbf{w})$ which implies that the first eigenvector $\mathbf{w}_{min}$ of $V_{n_v-r}^T \mathbf{M_A}^T \mathbf{M_A} V_{n_v-r}$ is such that $Q^T \mathbf{w}_{min} = e_{n_v-r}$ where $e_{n_v-r} = (0, 0, \cdots, 0, 1)^T$ is a $(n_v - r)$-vector , and its minimum value λ_{min} is $\omega_{n_v-r}^2$. Hence, $\mathbf{w}_{min}$ is the last column of Q. Once we compute Q, we get the coefficients of the algebraic surface $\mathbf{x} = V_{n_v-r} Q e_{n_v-r}$, which is not a zero vector, and hence satisfies the normalization constraint.

Example 3.2 Quartic Surfaces for Smoothly Joining Four Cylindrical Surfaces

In this example, we determine a surface $S : f(x, y, z) = 0$ which smoothly joins four cylinders which are given as $CYL_1 : y^2 + z^2 - 1 = 0$ for $x \geq 2$, $CYL_2 : y^2 + z^2 - 1 = 0$ for $x \leq -2$, $CYL_3 : x^2 + y^2 - 1 = 0$ for $z \geq 2$, and $CYL_4 : x^2 + y^2 - 1 = 0$ for $z \leq -2$.

The interpolation requirement is for S to meet the four curves on the cylinders with C^1 continuity. Hermite interpolation for a quartic surface S generates $\mathbf{M_I} \in$

$\mathbf{R}^{64 \times 35}$ (64 linear equations and 35 coefficients) whose rank is 33. [1] This implies a 2-parameter family of quartic surfaces satisfying the interpolation constraints.

Then we need to select, from this family, a surface with desired shape. We use least squares approximation during this process. To illustrate the effect of approximation, two sets of points are chosen: S_1 = { (0, 1.75 ,0), (0, -1.75, 0), (-1, 1.25, 0), (-1, -1.25, 0), (1, 1.25, 0), (1, -1.25, 0) } and S_2 = { (0, 1.25, 0), (0, -1.25, 0), (-0.5, 1.125, 0), (-0.5, -1.125, 0), (0.5, 1.125, 0), (0.5, -1.125, 0)}. (See Figure 3.2.)

For the least squares approximation with a normalization, the eigenvalues and eigenvectors for S_1 and S_2 are computed. As a result, we obtain $\lambda_{mins_1} = 1.265429 \cdot 10^{-1}$, $\lambda_{mins_2} = 5.097809 \cdot 10^{-3}$, $f_{S_1}(x, y, z) = 0.315034x^2 + 0.273947y^2 + 0.315034z^2 - 0.649216 - 0.035612x^4 - 0.030137x^2y^2 - 0.030137x^2z^2 + 0.005474y^4 - 0.030137y^2z^2 - 0.035612z^4$, and $f_{S_2}(x, y, z) = 0.281104x^2 + 0.615461y^2 + 0.281104z^2 - 0.201225 + 0.005325x^4 - 0.323706x^2y^2 - 0.323706x^2z^2 - 0.329031y^4 - 0.323706y^2z^2 + 0.005325z^4$. The two computed surfaces are shown in Figure 3.3. □

3.2.2 Least Squares Approximation Only

At times one desires a surface which is only the least squares approximation from given geometric data. This is often the case when straightforward interpolation leads to a prohibitively high algebraic degree of the resulting surface. This least squares problem is just a special case ($\mathbf{M_I} = 0$) of the minimization problem in the previous subsection. In this case, V_{n_v-r} disappears in the solution, which results in $\mathbf{x} = Q e_{n_v}$.

Example 3.3 Least Squares Approximation to Given Points : Algebraic Distance

Consider that we are computing a quadric surface $f(x, y, z) = 0$ which approximates the following collection of points in the least squares sense: S = { (0.45166, -0.62397,

[1] As a by-product of this interpolation process, it is found out that degree 4 is the minimum degree required.

0.06839), (0.32834, -0.67743, -0.05892), (0.43922, -0.59102, -0.11233), (0.20366, -0.71340, -0.17960), (0.31615, -0.64298, -0.23526), (0.41615, -0.54803, -0.28378), (-0.01352, -0.72637, -0.35770), (0.09109, -0.68925, -0.41407), (0.08685, -0.72867, -0.27968), (0.18959, -0.62765, -0.46521), (0.19834, -0.67449, -0.33577), (0.35025, -0.44569, -0.53594), (0.38550, -0.49832, -0.42532), (0.27772, -0.54493, -0.50640), (0.29999, -0.59612, -0.38273) }.

Each row of $\mathbf{M_A}$ is obtained by simply evaluating, at each point, the basis of quadrics: $\{ x^2, y^2, z^2, xy, yz, zx, x, y, z, 1 \}$. After applying SVD to $\mathbf{M_A}$, we get a quadric surface whose error-of-fit is $\lambda_{min} = 2.281646 \cdot 10^{-7}$. □

In the previous example, the sum of squares of algebraic distances is minimized, which are, in fact, contour levels of the function $w = f(x, y, z)$. The algebraic distances are not always the same as the corresponding geometric distances, which are the actual distances from the points to the surface. Sometimes, it may be more desirable to minimize the sum of squares of real distances. Unfortunately, this nonalgebraic, geometric metric entails an intractable minimization problem whose solution can not be expressed analytically in closed form. Sampson [65] used a nonalgebraic distance metric, which approximates geometric distance, in fitting conic curves. This concept can be naturally extended to the surface fitting problem. We get to this nonalgebraic metric via a different derivation as follows.

First, let us recall that the distance from a point p to a surface $f(x, y, z) = 0$ is the distance from p to a nearest point on the surface. Let q be the point on the surface which results in the distance. Then, the line in the direction of the normal of f at q must pass through p, and $q = p + t\frac{\nabla f(q)}{\|\nabla f(q)\|}$ where the absolute value of t is the geometric distance. From Taylor's expansion,

$$0 = f(q) = f(p) + \nabla f(p) \cdot (t\frac{\nabla f(q)}{\| \nabla f(q) \|}) + \cdots.$$

Hence,

$$| t | \approx | \frac{-f(p) \| \nabla f(q) \|}{\nabla f(p) \cdot \nabla f(q)} | \qquad (3.2)$$

is the first order approximation to the distance from p to f. When p is close to the surface, $\nabla f(p)$ is a good approximation to $\nabla f(q)$. In this case, the expression (3.2) becomes

$$\begin{aligned} | t | &\approx | \frac{-f(p) \| \nabla f(p) \|}{\nabla f(p) \cdot \nabla f(p)} | \\ &= | \frac{-f(p) \| \nabla f(p) \|}{\| \nabla f(p) \|^2} | \\ &= \frac{| f(p) |}{\| \nabla f(p) \|} \stackrel{\text{def}}{=} dist_f(p). \end{aligned}$$

This argument suggests that $dist_f(p)$, the weighted algebraic distance, is a good approximation to the geometric distance, and that

$$\sum_{for\ all\ p} dist_f(p)^2 = \sum_{for\ all\ p} \frac{f(p)^2}{\| \nabla f(p) \|^2} \qquad (3.3)$$

is minimized instead of

$$\sum_{for\ all\ p} f(p)^2. \qquad (3.4)$$

However, the solution which minimizes the expression (3.3) can not be easily expressed in closed form due to introduction of the weight $\| \nabla f(p) \|$.

This numerical intractability can be avoided by an iterative refinement algorithm. First, we compute $\mathbf{x}_{(0)}$, coefficients of a surface $f_{(0)}$, such that (3.4), the sum of squares of algebraic distances, is minimized. To do this, $\mathbf{M_A} = \mathbf{M_{A(0)}}$ is obtained as before. The gradient of $f_{(0)}$ gives an initial approximation of $\nabla f(p)$. Then, dividing each row of $\mathbf{M_A}$ by $\| \nabla f_{(0)}(p) \|$ for each corresponding p results in $\mathbf{M_{A(1)}}$ which is, then, singular-value-decomposed to compute $\mathbf{x}_{(1)}$ and $f_{(1)}$. This process is repeated further producing a sequence of $f_{(k)}$ which refines the solution. In each iteration, $f_{(k)}$ is expected to be a better approximation to the surface we are trying to find.

Example 3.4 Iterative Weighted Least Squares Approximation to the Points : Nonalgebraic Distance

In Example 3.3, we have computed $\mathbf{M_A} = \mathbf{M_{A(0)}}$, and $f_{(0)}$. The Table 3.1 illustrates the result of application of the iterative algorithm to the points used in

Table 3.1 The Geometric and Algebraic Distances

k	*geo. distance*	*alg. distance*
0	3.925480319e-05	2.281646641e-07
1	2.870799913e-05	2.497249375e-07
2	2.762911566e-05	2.472207775e-07
3	2.696617975e-05	2.465526346e-07
4	2.661304527e-05	2.461413816e-07
5	2.642308921e-05	2.459224774e-07
6	2.632187346e-05	2.458047987e-07
7	2.626807583e-05	2.457421127e-07
8	2.623953195e-05	2.457087993e-07
9	2.622440016e-05	2.456911254e-07
...	...	...
26	2.620735209e-05	2.456712015e-07
27	2.620735193e-05	2.456712014e-07
28	2.620735184e-05	2.456712013e-07

Example 3.3. The geo. distance column shows the sum of squares of the real geometric distances [3] for $f_{(k)}$, and the alg. distance column shows the value of the expression (3.4), the sum of squares of the algebraic distances for $f_{(k)}$. It is observed that the sum of squares of the geometric distances decreases as iterations proceed, which implies that $f_{(k)}$ converges to a surface which is expected to best-fit the given point data. It is also interesting to notice that the sum of squares of the algebraic distances makes a quantum jump at the first iteration, and then converges to a local minimum. □

3.3 Interactive Shape Control of Hermite Interpolating Surfaces

As mentioned before, the result of Hermite interpolation is a q parameter family of algebraic surfaces $f(x,y,z) = 0$ of a given degree that satisfy given geometric properties. The equation of the family has the generic form

$$f(x,y,z) = \sum_{i=0}^{n}\sum_{j=0}^{n-i}\sum_{k=0}^{n-i-j} c_{ijk} \cdot x^i y^j z^k = 0, \tag{3.5}$$

where each c_{ijk} is a homogeneous linear combination of q-parameters $r_1, r_2, \cdots, r_q$.

In the previous sections, we proposed to use least squares approximation to select an initial instance surface from the family obtained from Hermite interpolation. Even though we can get some geometric intuition from least squares approximation, we may want to change the shape of the computed surface interactively by modifying the values of the free parameters. However, since the computed surface $f(x,y,z) = 0$ is a polynomial in the standard power basis, its coefficients are algebraic, not geometric. That is, they contain little intuitive geometric information, hence they do not provide a convenient tool with which the shape of an algebraic surface can be controlled intuitively.

Sederberg [66] presented an idea in which free form piecewise algebraic surface patches are defined in trivariate barycentric coordinates using a reference tetrahedron

[3] The geometric distances were calculated by solving a 4-by-4 system of nonlinear equations, derived using the Lagrange multiplier method.

and a regular lattice of control points imposed on the tetrahedron. The coefficients of a surface defined in this way are assigned to the control points, and there is a meaningful relationship between the coefficients and the shape of the surface.

The essence of his idea is to consider an algebraic surface $f(x,y,z)=0$ as the zero contour of the trivariate function $w=f(x,y,z)$. Note that the surface equation of the family of Hermite interpolating algebraic surfaces contains q free variables r_i in its coefficients. A specific portion of a surface can be selected for shape control by defining a tetrahedron which encloses that portion. Given a tetrahedron, the polynomial $f(x,y,z)$ in power basis can be symbolically converted into a polynomial $F(s,t,u)$ in barycentric coordinates, defined with respect to the tetrahedron.

Let a tetrahedron be specified by the four noncoplanar vertices P_{n00}, P_{0n0}, P_{00n}, and P_{000}. Then, the coordinates $P=(x,y,z)$ of a point inside the tetrahedron are related to the barycentric coordinates (s,t,u) by $P = sP_{n00} + tP_{0n0} + uP_{00n} + (1-s-t-u)P_{000}$, $s,u,t,(1-s-t-u) > 0$. Control points on the tetrahedron are defined by $P_{ijk} = \frac{i}{n}P_{n00} + \frac{j}{n}P_{0n0} + \frac{k}{n}P_{00n} + \frac{n-i-j-k}{n}P_{000}$ for nonnegative integers i,j,k such that $i+j+k \le n$. Each control point is associated with a weight w_{ijk}, which is a linear combination of r_i, $i=1,2,\cdots,q$. All these together define the q-parameter algebraic surface family in barycentric coordinates,

$$F(s,t,u) = \sum_{i=0}^{n}\sum_{j=0}^{n-i}\sum_{k=0}^{n-i-j} w_{ijk}\cdot\binom{n}{i,j,k}\cdot s^i t^j u^k (1-s-t-u)^{n-i-j-k} = 0. \qquad (3.6)$$

Example 3.5 Conversion from Power to Bernstein

Consider, as a simple example, a quadric surface which Hermite interpolates a line $LN:(1-t,t,0)$ with a normal $(0,0,1)$. The Hermite interpolation algorithm returns a 5 parameter family $f(x,y,z)=0$ of algebraic surfaces, as in (3.5) with $n=2$, where $c_{200}=r_1$, $c_{110}=2r_1$, $c_{101}=r_4$, $c_{100}=-2r_1$, $c_{020}=r_1$, $c_{011}=r_5$, $c_{010}=-2r_1$, $c_{002}=r_3$, $c_{001}=r_2$,and $c_{000}=r_1$. For a given tetrahedron with vertices $P_{n00}=(2,0,0)$, $P_{0n0}=(0,2,0)$, $P_{00n}=(0,0,2)$, and $P_{000}=(0,0,0)$, the surface $f(x,y,z)=0$ is transformed to $F(s,t,u)=0$, as in (3.6) with $n=2$, where $w_{000}=r_1$, $w_{001}=r_1+r_2$,

$w_{002}=r_1+2r_2+4r_3$, $w_{010}=-r_1$, $w_{011}=-r_1+r_2+2r_5$, $w_{020}=r_1$, $w_{100}=-r_1$, $w_{101}=-r_1+r_2+2r_4$, $w_{110}=r_1$, and $w_{200}=r_1$. □

Since the weights w_{ijk} of $F(s,t,u)=0$ for a q-parameter family of algebraic surfaces have only q degrees of freedom, they can't be selected or modified independently. For example, suppose $w_1 = r_1+r_2+r_3+2r_4-1$, $w_2 = r_1+r_2+r_4+5$, and $w_3 = r_3+r_4$. From these, we can derive the linear relation $w_1-w_2-w_3-6=0$ between the weights, and then an invariant $\Delta w_1 - \Delta w_2 - \Delta w_3 = 0$ which must be satisfied each time some of the weights are modified. (For notational simplicity, we assume the weights are indexed by a single number instead of a triple.)

In general, using Gaussian elimination, we can derive a system of invariant equations

$$\begin{aligned} I_1(\Delta w_1, \Delta w_2, \cdots, \Delta w_c) &= 0 \\ I_2(\Delta w_1, \Delta w_2, \cdots, \Delta w_c) &= 0 \\ &\vdots \\ I_l(\Delta w_1, \Delta w_2, \cdots, \Delta w_c) &= 0 \end{aligned}$$

from the linear expressions of the weights

$$\begin{aligned} w_1(r_1, r_2, \cdots, r_p) &= w_1 \\ w_2(r_1, r_2, \cdots, r_p) &= w_2 \\ &\vdots \\ w_c(r_1, r_2, \cdots, r_p) &= w_c. \end{aligned}$$

Changing the weights can now be considered as moving from a weight vector $W=(w_1,w_2,\cdots,w_c)$ to another $W'=(w'_1,w'_2,\cdots,w'_c)$, with the constraint that $\Delta W = W' - W$ is a solution of the system of invariant equations.

Example 3.6 Shape Control of a Family of Quadric Surfaces

The invariant system for the family of algebraic surfaces in Example 3.5 is $\Delta w_{010} + \Delta w_{000} = 0$, $\Delta w_{020} - \Delta w_{000} = 0$, $\Delta w_{100} + \Delta w_{000} = 0$, $\Delta w_{110} - \Delta w_{000} = 0$, $\Delta w_{200} - \Delta w_{000} = 0$. Figure 3.4 (upper left) shows an instance from the family where $w_{000} = -4$, $w_{001} = 4$, $w_{002} = 8$, $w_{010} = 4$, $w_{011} = 14$, $w_{020} = -4$, $w_{100} = 4$, $w_{101} = 12$, $w_{110} = -4$, and $w_{200} = -4$.

Now, suppose we want to pull the surface patch toward the control points P_{002} (the leftmost vertex in the figure). This can be achieved by decreasing the value of w_{002}, say, $\Delta w_{002} = -7$.

Other Δw_{ijk} can be arbitrarily chosen as long as they satisfy the equations in the invariant system. Let $\Delta w_{000} = \Delta w_{200} = \Delta w_{110} = \Delta w_{020} = -1$, $\Delta w_{100} = \Delta w_{010} = 1$, $\Delta w_{001} = -4$, $\Delta w_{101} = \Delta w_{011} = -2$. The new instance surface is shown in Figure 3.4 (upper right). □

Example 3.7 Shape Control of a Family of Quartic Surfaces

Figure 3.4 (bottom left) illustrates three different instances of the family computed in Example 2.3, corresponding to the three different values of w_{000} for $P_{000} = (0,0,0)$. As a weight w_{000} increases from a negative value, the surface approaches to P_{000}. The surface passes through P_{000} when $w_{000} = 0$, and gets separated into three irreducible components as w_{000} becomes positive. (See also Figure 2.3 and 2.4.) □

Sometimes, we may want to see how the shape of a surface changes as a specific weight is modified. However, if a weight, say, w_1 is modified, then this modification affects other weights as related in the invariant system. Usually, the linear system of invariant equations is underdetermined, yielding an infinite number of choices of Δw_i $(i = 2, 3, \cdots, c)$. Then, how can we select the other weights such that their effects to w_1 are minimized?

One possible heuristic is to minimize the 2-norm of $(\Delta w_2, \cdots, \Delta w_c)$, and hence the 2-norm $\|\Delta W\|_2 = (\Delta w_1^2 + \Delta w_2^2 + \cdots + \Delta w_c^2)^{\frac{1}{2}}$ of ΔW. For $\Delta w_1 = d$, we know that the linear system

$$
\begin{aligned}
I_1(d, \Delta w_2, \cdots, \Delta w_c) &= 0 \\
I_2(d, \Delta w_2, \cdots, \Delta w_c) &= 0 \\
&\vdots \\
I_l(d, \Delta w_2, \cdots, \Delta w_c) &= 0
\end{aligned}
$$

has a solution $\Delta W^0 = (d, \Delta w_2^0, \cdots, \Delta w_c^0)$ where Δw_i^0's are expressed linearly through another set of free parameters $p_1, p_2, \cdots, p_s$. Hence, $\|\Delta W^0\|_2^2$ is a quadratic function $Q(p_1, p_2, \cdots, p_s)$ of the new parameters.

Since Q is quadratic, $Q(p_1, p_2, \cdots, p_s)$ is minimized at the solution of the linear system $\nabla Q(p_1, p_2, \cdots, p_s) = 0$. If the minimum of Q occurs at a point $(p_1^0, p_2^0, \cdots, p_s^0)$, then $\Delta W^0 = (d, \Delta w_2^0, \cdots, \Delta w_c^0)$ corresponding to the point defines the desired change of weights $w_2, \cdots, w_c$ having the minimum effect, in the least squares sense, on the shape of the surface. The instance surface corresponding to the new weights $W' = W + \Delta W^0$ will then reflect predominantly the effect of the change of w_1 by $\Delta w_1 = d$.

Example 3.8 Heuristic Approach to Shape Control Using 2-Norm

Consider the surface in Example 3.6 again. This time we wish to pull the patch more toward P_{002}, and hence set $\Delta w_{002} = -15$. From the invariant system in which Δw_{002} is replaced by -15, $\Delta w_{000} = \Delta w_{020} = \Delta w_{110} = \Delta w_{200} = p_1$, $\Delta w_{010} = \Delta w_{100} = -p_1$, $\Delta w_{001} = p_2$, $\Delta w_{101} = p_3$, $\Delta w_{011} = p_4$, and we obtain the quadratic function $Q(p_1, p_2, p_3, p_4) = 225 + 6p_1^2 + p_2^2 + p_3^2 + p_4^2$. Q has the global minimum at $p_1 = p_2 = p_3 = p_4 = 0$. Hence, the influence of the change of all the weights other than w_{002}, is minimized by setting to zero their Δw, that is, not changing them at all.

This new instance is shown in Figure 3.4 (bottom right). Note that the overall shape of the new surface patch, other than close to P_{002}, has not changed as much as the surface patch in Figure 3.4 (upper right), even though w_{002} has decreased by a larger amount. □

3.4 Summary

In this chapter, we considered how the geometric problem of finding algebraic surfaces is transformed into a linear algebra problem through Hermite interpolation, least squares approximation, and normalization. A compact algorithm for solving the algebraic model efficiently was presented with examples. Although the algebraic formulation of the geometric problem presented in this chapter, results in a compact computation, it must be enhanced by adding more geometric constraints such that the algebraic formulation can also handle undesirable phenomena appearing in algebraic surface design. In Chapter 6, we discuss such unfavorable phenomena of algebraic surfaces in details. We also proposed a surface control scheme in which the shape of a specific portion of an algebraic surface is controlled with geometric intuition in the barycentric coordinate system.

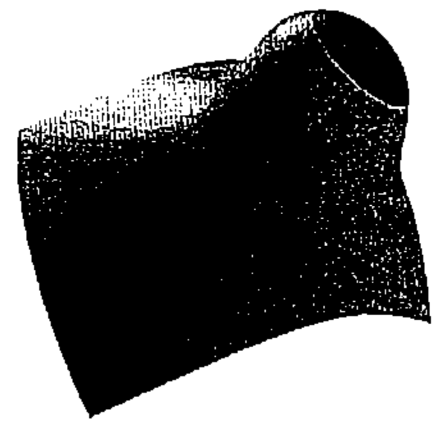

Figure 3.1 A Cubic C^2 Continuous Surface

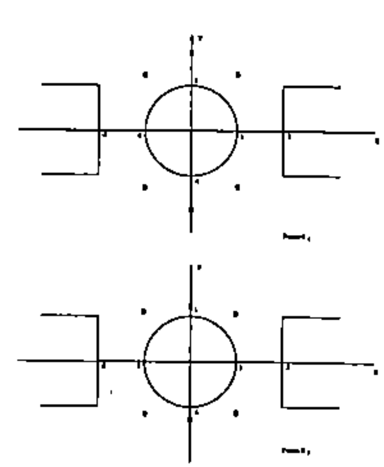

Figure 3.2 Points to be Approximated

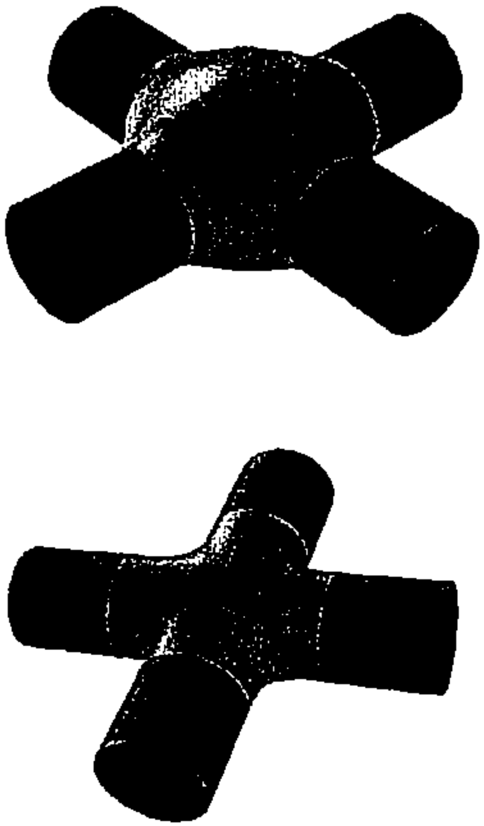

Figure 3.3 Two Different Least squares Approximations

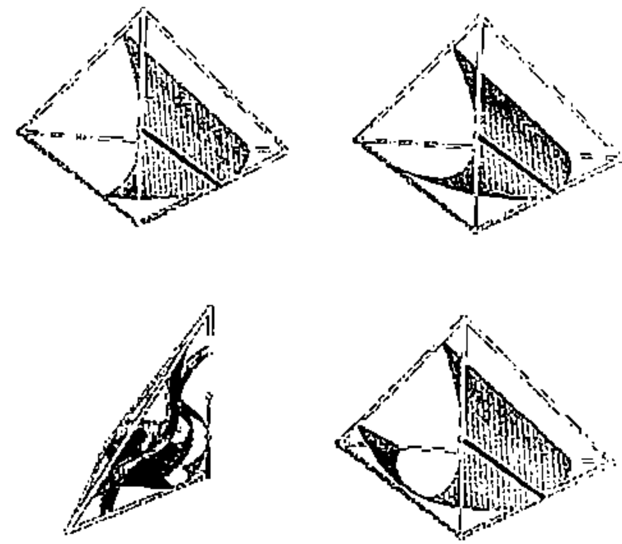

Figure 3.4 Interactive Shape Control Using Barycentric Coordinates

4. SMOOTHING CONVEX POLYHEDRA WITH QUINTIC SURFACES

Modeling a triangular mesh of three dimensional data with smooth piecewise surfaces is a well studied problem. Traditionally, most of the previous works have used parametrically represented surfaces. (See the recent survey paper [23].) On the other hand, there are only few results where implicitly represented algebraic surfaces are used as modeling tools [20, 32, 67]. In [20], Dahmen used quadric algebraic surfaces to smooth a special type of polyhedra splitting a face into several subfaces which, sometimes, produce wave-like oscillations between patches due to the limitations of quadric surfaces. Guo [32] presented an algorithm which constructs a mesh of cubic algebraic surface patches smoothing a polyhedron where cubic patches for faces are connected by two extra cubic patches between faces. It seems that there has been little success in generating a mesh of smooth piecewise patches of algebraic surfaces of nontrivial degrees greater than three which provide more flexibility.

In this chapter, we investigate how the class of quintic algebraic surfaces can be utilized in CAGD. In particular, we show how this class of surfaces can be used to smooth a convex polyhedron. Sometimes it has been stated that degree 5 is so high that quintic algebraic surfaces may not be appropriate for geometric modeling. While it is true that a whole quintic algebraic surface behaves unpredictably, we only need a patch of a surface for geometric modeling, not a whole surface. The work in this chapter shows that triangular patches of quintic algebraic surfaces are flexible enough to be used as effective modeling tools while degree 5 is low enough to allow efficient computations.

We assume, without loss of generality, that a given convex polyhedron has only triangular faces. To smooth a convex polyhedron with tangent plane continuous triangular surface patches, we first construct a quadric wire frame by replacing each

edge of the polyhedron with a conic curve whose shape is controlled by a single parameter. The normals along the curves are derived from the polyhedron, and are associated with the curves. The normals along the curves then provide tangent plane information along the conic curves that must be satisfied by incident surface patches. Then, the triple of boundary curves and normals for each face is Hermite interpolated and approximated in the least squares sense using our computational model, which is slightly different from the model explained in the previous chapter:

$$\begin{aligned} &\textit{minimize} && \| \mathbf{M_A x} - \mathbf{b} \|^2 \\ &\textit{subject to} && \mathbf{M_I x} = 0. \end{aligned}$$

Quintic algebraic surface patches obtained by solving the above problem give a curved model for the polyhedron.

This chapter is organized as follows: In Section 4.1, we explain, step-by-step, how a convex polyhedron is smoothed, and how each triangular implicit patch is displayed interactively. Section 4.2 is devoted to explaining why singularities usually occur at the three vertices of triangular patches. As an example, a given convex polyhedron is smoothed with three different shape control parameters in Section 4.3. In Section 4.4, we discuss open problems on smoothing a nonconvex polyhedron with quintic algebraic surfaces. Finally, this chapter is summarized in Section 4.5.

4.1 Generation of a Quintic Algebraic Triangular Patch

In this section, we show step-by-step how to compute each triangular quintic algebraic surface patch. Each edge of a triangular face of a polyhedron is replaced by a quadric curve with quadric parametric rational normal directions that conform to the positional and tangential information derived from the polyhedron. The three boundary curves with normals are smoothly fleshed by a quintic implicit algebraic surface. Then, the computed triangular patch is polygonized for interactive display.

4.1.1 Generation of a Quadric Wire

First, we give a definition of a quadric wire.

Definition 4.1 Let $C(t) = (\frac{x(t)}{w(t)}, \frac{y(t)}{w(t)}, \frac{z(t)}{w(t)})$ and $N(t) = (\frac{nx(t)}{w(t)}, \frac{ny(t)}{w(t)}, \frac{nz(t)}{w(t)})$ be two triples of quadratic rational parametric polynomials. Then, the pair $W(t) = (C(t), N(t))$ is called a quadric wire if there exists a quadric surface $q(x, y, z) = 0$ such that $q(C(t)) = 0$ and $\nabla q(C(t))$ is proportional to $N(t)$ for all t.

In other word, we take, from a quadric surface, a conic curve and gradient vectors restricted along the curve, and use them as boundary curves of a quadric wire frame.

The first step towards smoothing a convex polyhedron is to compute a conic curve $C(t)$ given two point and unit normal vector pairs (p_0, n_0), (p_1, n_1) and a normal vector npl of a plane containing p_0 and p_1 such that

- a computed conic curve passes through p_0 and p_1,
- its tangents at p_0 and p_1 are perpendicular to n_0 and n_1, respectively, and
- the curve is contained in the plane which contains p_0 and p_1, and has the plane normal npl.

Furthermore, we force $C(0) = p_0$ and $C(1) = p_1$ so that the segment of $C(t)$ for $0 \le t \le 1$ is used as a wire. To compute $C(t)$, the normal vectors n_0 and n_1 are projected onto the plane P on which $C(t)$ will lie. (See Figure 4.1.) This projection results in a control triangle $p_0 - p_2 - p_1$. (p_2 is computed as the intersection of the two tangent lines orthogonal to the projections of n_0 and n_1.) Lee [42] presented a compact method for computing a conic curve $C(t)$ from the control triangle. In his formulation, the conic is expressed in Bernstein-Bézier form:

$$C(t) = \frac{w_0 p_0 (1-t)^2 + 2 w_2 p_2 t(1-t) + w_1 p_1 t^2}{w_0 (1-t)^2 + 2 w_2 t(1-t) + w_1 t^2},$$

where $w_i > 0$, $i = 0, 1, 2$ are shape control parameters. An often used parameterization, called the rho-conic parameterization, is given by the special choice of

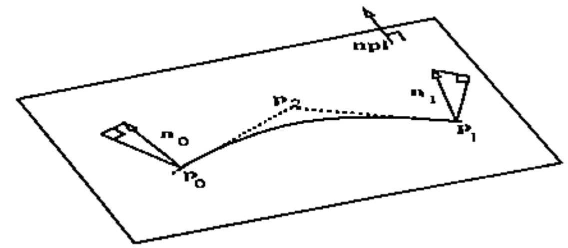

Figure 4.1 Computation of a Conic Curve

$w_0 = w_1 = 1 - \rho$, $w_2 = \rho$, $0 < \rho < 1$. Then the parameter ρ measures the sharpness of the conic curve. Let $p_{01} = (p_0 + p_1)/2$ be the midpoint of the chord $p_0 p_1$. Then, ρ has a property that $C(0.5) - p_{01} = \rho(p_2 - p_{01})$. From this, we can see that as ρ is increased, the conic gets more curved. In particular, it is known that $\rho = 0.5$ for parabolas, $0 < \rho < 0.5$ for ellipses and $0.5 < \rho < 1.0$ for hyperbolas.

Once $C(t)$ is computed, we find a quadratic surface $q(x, y, z) = 0$ such that $N(t)$, which is a restriction of $\nabla q(x, y, z)$ along $C(t)$, interpolates n_0 and n_1 at $t = 0$ and $t = 1$, respectively. Consider a quadric surface $q(x, y, z) = c_1 x^2 + c_2 y^2 + c_3 z^2 + c_4 xy + c_5 yz + c_6 zx + c_7 x + c_8 y + c_9 z + c_{10} = 0$. A quadric surface has 9 degrees of freedom in its coefficients. The first constraint on $q(x, y, z) = 0$ is that it must contain $C(t)$. The Hermite interpolation algorithm gives 5 linear equations where the unknowns are c_i, $i = 1, 2, \cdots, 10$. It is obvious that 5 equations are required considering Bezout theorem which implies that if a conic intersects with a quadric surface at more than 4 points, the curve is contained in the surface.

Hence, 4 ($= 9 - 5$) degrees of freedom remains, and these must be used to interpolate the normal vectors at the two end points. Interpolating n_0 and n_1 at p_0 and p_1, respectively, results in 2 more linear constraints which leaves 2 degrees of freedom in choosing a quadric surface. In fact, we observe that fixing one more normal vector at a point on the curve fixes normal vectors along the whole curve. Consider the special case of Theorem 3.1 where given a quadric wire defined by a quadric surface $a(x,y,z)=0$ and a plane $b(x,y,z)=0$, there is a family of quadric surfaces $c(x,y,z)=0$ with one degree of freedom such that $c(x,y,z)=\alpha a(x,y,z)+\beta b(x,y,z)^2$. This implies that the rank of the linear system for Hermite interpolation of the quadric wire is at most 8. Hence, three normal vectors determine normal vectors along the entire quadric curve.

In our implementation we specify the third normal vector as follows: first, the average $n_{01}=(n_0+n_1)/2$ is computed, and then n_{01} is projected into the plane which contains $C(0.5)$, and is orthogonal to the tangent vector $C'(0.5)$. Then, we use the projected vector as $N(0.5)$. The family of quadric surfaces $q(x,y,z)=0$ computed this way gives the gradient vector $\nabla q(C(t))$ that is used as $N(t)$.

4.1.2 Hermite Interpolation of a Quadric Triangle

As mentioned before, each triangular face of a polyhedron is replaced by a triangular surface patch. To do so, each edge is replaced by a quadric wire forming a wire frame for the polyhedron. To clarify our description, we give the following definitions:

Definition 4.2 An augmented triangle is a 9-tuple $T=(p_0,p_1,p_2,n_0,n_1,n_2,npl_{01},npl_{12},npl_{20})$ where p_i, $i=0,1,2$, are three vertices of a triangle with the corresponding unit normal vectors n_i, $i=0,1,2$, and npl_{ij} are normal vectors of the planes which will contain the quadric wires computed from (p_i,n_i) and (p_j,n_j).

Definition 4.3 A quadric triangle is a triple $QT=(W_0(t),W_1(t),W_2(t))$ of quadric wires such that $W_0(1)\equiv W_1(0)$, $W_1(1)\equiv W_2(0)$, and $W_2(1)\equiv W_0(0)$. [1]

Given an augmented triangle, each quadric wire is computed as described previously. Then, the quadric triangle is fleshed using an algebraic surface $f(x,y,z)=0$ of degree n. The surface $f(x,y,z)=0$ must be flexible enough to interpolate the three quadric wires smoothly, i.e., with tangent plane continuity. Though higher degree algebraic surfaces provide more flexibility, the number $\binom{n+3}{3}$ of coefficients of

[1] By $\equiv$, we mean the points are the same, and the normal vectors are proportional.

an algebraic surface of degree n grows dramatically as n increases. Hence, for a fast computation and less numerical errors, keeping n in a reasonable range is very important. In the below, we discuss a lower bound of degree of algebraic surfaces that can Hermite interpolate a quadric triangle.

Consider an algebraic surface of degree n for smooth interpolation of a quadric wire $W(t)=(C(t),N(t))$. According to Bezout theorem, $2n+1$ constraints on the coefficients of f are required for f to contain $C(t)$ which is quadratic. For tangent plane continuity, the Hermite interpolation algorithm in Subsection 2.3.2.2 produces $(n-1)d+m+1=2(n-1)+2+1$ additional linear equations. In total, $4n+2$ linear equations are generated for smooth interpolation. However, it is uniformly observed that the rank of the linear system is $4n$, while we can not prove this algebraically as of now. We are led to the following conjecture:

Conjecture 4.1 Let $W(t)=(C(t),N(t))$ be a quadric wire. Given an algebraic surface $f(x,y,z)=0$ of degree n, the rank of the linear system generated by Hermite interpolation to to smoothly interpolate $W(t)$ is $4n$.

Since there are three quadric wires, $12n$ linear constraints on the coefficients of a surface are produced according to the above conjecture. On the other hand, we observe some geometric dependency between the three quadric wires which leads to algebraic dependency. First, since the conics intersect pairwise, there must be three rank deficiencies between the equations from the containment conditions. [2] Secondly, at each vertex of a quadric triangle, two incident conics automatically determine the normal at the vertex. It is obvious, from the method of quadric wire construction, that this vector is proportional to the given unit normal vector. So, we see that satisfying the containment conditions for the three conics guarantees that any interpolating surface has gradient vectors at the three points as required. This fact implies that, for each conic, there are two rank deficiencies between the linear equations for the

[2] This dependency gets more evident when considering the Hermite interpolation algorithm. In the algorithm in Subsection 2.3.2.1, if we always choose the intersection points for the list L_r of each conic, three equations are generated twice.

containment conditions, and the equations for its tangency condition.[3] Hence, six additional rank deficiencies with the previous three indicate that the minimum of $12n-9$ and $\binom{n+3}{3}$ is believed to be the maximum possible rank of the linear system that is generated by Hermite interpolation algorithm.

Since an algebraic surface $f(x,y,z)=0$ of degree n has $\binom{n+3}{3}$ coefficients, and the rank of the linear system should be less than the number of coefficients for a nontrivial surface to exist, we find out that 5 is the minimum degree required. In the quintic case, there are 56 coefficients (55 degrees of freedom) and the rank is at most 51, which results in a family of interpolating surfaces with at least four degrees of freedom in selecting an instance surface from the family. Even though some special combinations of three quadric wires can be interpolated by a surface of degree less than 5, for example, three quadric wires from a sphere, the probability that such spatial dependency occurs, given an arbitrary triple of conics with normals, is infinitesimal. Hence, the conjectured lower degree bound is tight.

4.1.3 Least Squares Approximation to Contour Levels

As a result of Hermite interpolation of a quadric triangle QT, a family of quintic algebraic surfaces $f(x,y,z)=0$ with at least 4 degrees of freedom is usually obtained. Then, we need to use these remaining degrees of freedom appropriately to select an instance quintic surface from this family. We can additionally specify a set of points inside the quadric triangle, which approximately describe a desirable surface patch. A final fitting surface may be obtained by consuming the remaining degrees of freedom through least squares approximation to this set of points.

When chosen from the family via least squares approximation, the selected quintic surface is not always good in the light of geometric modeling. For example, a surface which self-intersects inside the quadric triangle is not practically useful though it approximates the additional points best as well as satisfies the smooth interpolation

[3] Again, for each curve, we can choose point-normal pairs at the two end points. The resulting two linear equations should be linearly dependent on the equations from the containment requirement.

requirement. Hence, in the approximation step, we must be careful not to select a surface which is singular inside the quadric triangle. First of all, we observe that, in general, any surface which smoothly interpolates the quadric triangle, that is, three conics with normal directions, is singular at the three vertices. In Section 4.2, we show that just making the normal vectors of three conics consistent at the intersection points is not enough to have a surface that is regular at the three points. In fact, the rates of changes in the normal vectors at the intersection points affect the regularity of a surface. However, the singularities only at the three vertices, not along the whole curve, do not harm the smooth continuity requirement between surface patches. A more serious problem is the singularity of a surface inside a quadric triangle.

Let $S_0 = \{v_i \in \mathbf{R}^3 | i = 1,\cdots,l\}$ be a set of points which approximately describes a desirable surface patch. Then, we can get a linear system $\mathbf{M_A x} = 0$, where each row of $\mathbf{M_A}$ is obtained from $f(v_i)=0$. Then the conventional least squares approximation is to minimize $\| \mathbf{M_A x} \|^2$ over the nullspace of $\mathbf{M_I}$. However, our experiments show that, in many cases, singularities occur inside the quadric triangle. Minimizing $\| \mathbf{M_A x} \|^2$ makes a resulting surface approximate the set of points closely, however, this simple algebraic approximation can not prevent the surface from self-intersecting inside the triangle.

To provide more geometric control in least squares approximation, we suggest that the contour levels of a surface are approximated rather than only the surface. In fact, the implicit surface $f(x,y,z)=0$ is the zero contour of the function $w=f(x,y,z)$. Consider some smooth region of a surface. Since all the partial derivatives of $w=f(x,y,z)$ are well defined in the region, the contour levels behave well in the proximity of the zero contour. In our scheme, we first generate $S_0 = \{(v_i, n_i) | i = 1,\cdots,l\}$ where v_i is an approximating point, and n_i is an approximating gradient vector at v_i. Then, from this set, we construct two more sets $S_1 = \{u_i | u_i = v_i + \alpha n_i, i = 1,\cdots,l\}$, and $S_{-1} = \{w_i | w_i = v_i - \alpha n_i, i = 1,\cdots,l\}$ for some small $\alpha > 0$. Then, we get the least squares system $\mathbf{M_A} = \mathbf{b}$ from three kinds of equations: $f(v_i)=0$, $f(u_i)=1$, and $f(w_i)=-1$. Approximating these three kinds of equations forces the function

$w = f(x, y, z)$ to have well structured contour levels as much as possible near the inside of a quadric triangle, and we notice that this heuristic removes self-intersections in the region significantly. In Subsection 4.1.5, we give a heuristic algorithm for generation of the point-normal set S_0.

4.1.4 Fleshing a Wire Frame

Now, we are led to the following computational model:

$$\begin{array}{ll} \textit{minimize} & \| \mathbf{M_A x} - \mathbf{b} \|^2 \\ \textit{subject to} & \mathbf{M_I x} = 0, \end{array}$$

where $\mathbf{M_I} \in \mathbf{R}^{n_i \times 56}$ is a Hermite interpolation matrix, and $\mathbf{M_A} \in \mathbf{R}^{n_a \times 56}$ and $\mathbf{b} \in \mathbf{R}^{n_a}$ are a matrix and a vector, respectively, for contour level approximation, and $\mathbf{x} \in \mathbf{R}^{56}$ is a vector containing coefficients of a quintic algebraic surface $f(x, y, z) = 0$.

Again, the nullspace of $\mathbf{M_I}$ is compactly expressed as $\mathbf{x} = V_{56-r}\mathbf{w}$ where $\mathbf{w} \in \mathbf{R}^{56-r}$.[4] After substituting for $\mathbf{x}$, we are led to $\| \mathbf{M_A x} - \mathbf{b} \| = \| \mathbf{M_A} V_{56-r}\mathbf{w} - \mathbf{b} \|$. Then, an orthogonal matrix $Q \in \mathbf{R}^{n_a \times n_a}$ is computed such that

$$Q^T \mathbf{M_A} V_{56-r} = R = \begin{pmatrix} R_1 \\ 0 \end{pmatrix}$$

where $R_1 \in \mathbf{R}^{(56-r)\times(56-r)}$ is upper triangular. (This factorization is called a <u>Q-R factorization</u> [31]). Now, let

$$Q^T \mathbf{b} = \begin{pmatrix} c \\ d \end{pmatrix}$$

where c is the first $56 - r$ elements. Then, $\| \mathbf{M_A} V_{56-r}\mathbf{w} - \mathbf{b} \|^2 = \| Q^T \mathbf{M_A} V_{56-r}\mathbf{w} - Q^T\mathbf{b} \|^2 = \| R_1\mathbf{w} - c \|^2 + \| d \|^2$. The solution $\mathbf{w}$ can be computed by solving $R_1\mathbf{w} = c$, from which a fleshing surface is obtained by applying $\mathbf{x} = V_{56-r}\mathbf{w}$.

[4] As mentioned before, in most cases, the rank r of $\mathbf{M_I}$ is 51. However, we use the variable r for the rank because it is possible that there are more dependencies between boundary curves and normal vectors, although the chances are rare.

4.1.5 Display of the Triangular Algebraic Patch

As implicitly defined algebraic surfaces have become increasingly important in geometric modeling, several algorithms for displaying them have emerged. Implicit algebraic surfaces lend themselves naturally to ray tracing [34]. Sederberg et al. [69] used a scan line display method which offers improvement in speed and correctly displays singularities. Even though both approaches produce images of good qualities, the computational cost is high. Also, these static processes do not allow interactive display of surfaces. On the other hand, polygonization of implicit surfaces [4, 16] can use the capability of the graphics hardware which provides very fast interactive rendering. Allgower [4] used simplices to approximate a surface with polygonal meshes. In [16], Bloomenthal presented a numerical technique that approximates an implicit surface with a polygonal representation. The technique is to surround an implicit surface with an octree, at whose corners the implicit function is sampled to generate polygons. Although, in general, they sample implicit surfaces well, these polygonization methods are not well suited to our purpose which is to draw an implicit triangular surface patch with singular vertices. A major problem is how to isolate only a necessary part. Clipping surfaces might be added to the polygonization algorithms, however, the current polygonization algorithms do not handle singularities well.

In our display routine, we <u>walk over</u> implicit quintic surfaces only around the necessary regions producing polygons which approximate triangular patches. Since smooth segments of intersection curves of two algebraic surfaces are easily traced [7], and we are to display smooth portions of implicit surfaces, the algebraic space curve tracing routine performs well for this walk-over. Note that although the fleshing quintic surfaces are usually singular at the three vertices of a quadric triangle, the boundary curves can be traced easily from their parametric equations.

The following simple recursive procedure produces an adaptive polygonization of a triangular algebraic surface patch. Let $f(x, y, z) = 0$ be a primary surface whose triangular portion, clipped by three planes $h_i(x, y, z) = 0$, $i = 1, 2, 3$, is to be polygonized.

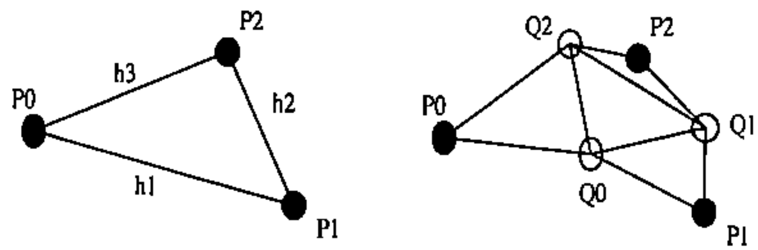

Figure 4.2 Recursive Refinement of a Triangle

(See Figure 4.2.) Initially, the triangle $T_0 = \{P_0, P_1, P_2\}$ is a rough approximation of the surface patch. Each boundary curve, obtained from the intersection of f and h_i, is traced producing a sequence of points on the curve, then an adaptive piecewise linear approximation of order 2^d for some given d is computed. In Chapter 5, we present an algorithm that quickly generates such a piecewise linear approximation given a sequence of points in 3D space. Then, T_0 is refined into four triangles by introducing the 3 points Q_0, Q_1, and Q_2 where Q_i, $i = 0, 1, 2$ is the middle point of each adaptive segmentation of order 2^d. The clipping planes for the subdivided triangles can be computed from the normals of the two triangles incident to the edge. Then, each new edge is traced, and its adaptive piecewise linear approximation of order 2^{d-1} is produced. In this way, this new approximation is further refined by recursively subdividing each triangle until some stopping criterion is met.

While the method produces a regular, but adaptive, network of polygons, it could be improved further to generate more adaptive polygonization. Rather than subdividing all the triangles up to the same level, each triangle is examined to see if it is already a good approximation to the surface portion it is approximating. It is refined only when the answer is no. Some criterions for such local refinement are suggested in [4, 16]. However, to design an irregular adaptive polygonization algorithm with robust local refinement criterions, remains open.

We also use the above recursive subdivision scheme to produce $S_0 = \{(v_i, n_i) | i = 1, \cdots, l\}$ used in Subsection 4.1.3. Initially, only the boundary curves are known, and each time a new curve is to be traced in the above algorithm, a quadric wire is computed as explained in Subsection 4.1.1 from the information on the initial and final points, their normals and clipping plane. The generated quadric wire gives approximate curve and normal information, and is traced to generate points and normals. The final polygonal approximation obtained in this way gives a set of points which are used in least squares approximation. We observe that this heuristic method work quite well when the ρ value is in the reasonable range, say, $0.25 \le \rho \le 0.75$. Figure 4.3 displays a polygonization of a triangular algebraic surface patch, and the points used for least squares approximation.

4.2 Why Singularities?

In this section, we consider why the quintic surfaces which Hermite interpolate quadric triangles are usually singular at three vertices.

4.2.1 A Review of Differential Geometry

We first review some basic concepts of differential geometry [18, 49]. A surface $S \subset R^3$ is regular at a point $p \subset S$ if there exists a neighborhood $V \subset R^3$ and a map $\mathbf{x} : U \longrightarrow V \cap S$ of an open set U in R^2 onto $V \cap S \subset R^3$ such that $\mathbf{x}(u, v) = (x(u, v), y(u, v), z(u, v))$ is differentiable, homeomorphic, and its differential $d\mathbf{x}_q : R^2 \longrightarrow R^3$ is one-to-one for each $q \in U$. A surface S is regular if, at each point on S, S is regular. Intuitively speaking, a surface is regular at a point if a neighborhood of the point in the surface can be obtained by taking a piece of a plane, deforming it in a not too violent fashion, in such a way that the resulting surface has no singularities like sharp points, edges, or self-intersections at the point and it makes sense to speak of a tangent plane at the point. By taking such a map from a plane to a surface, it becomes possible for a regular surface to have a differential and integral calculus which is strictly comparable with the calculus on the Euclidean plane R^2.

A tangent vector to a regular surface S at a point $p \in S$ is the tangent vector $\alpha'(0)$ of a differentiable curve $\alpha : (-\epsilon, \epsilon) \longrightarrow S$ with $\alpha(0) = p$. The plane $T_p(S)$ spanned by all tangent vectors to S at p, is called the tangent plane to S at p which is, in fact, a two dimensional vector space. For a regular point $p \in S$, a unit vector which is perpendicular to $T_p(S)$ is called a unit normal vector at p. For each $q \in \mathbf{x}(U)$, we define a differentiable field of unit normal vectors $N : \mathbf{x}(U) \longrightarrow R^3$ such that $N(q) = \frac{\mathbf{x}_u \times \mathbf{x}_v}{\|\mathbf{x}_u \times \mathbf{x}_v\|}(q)$, where $\mathbf{x}_u = \frac{\partial \mathbf{x}}{\partial u}$ and $\mathbf{x}_v = \frac{\partial \mathbf{x}}{\partial v}$. The map $N : S \longrightarrow S^2$, taking its values in the unit sphere, is called the Gauss map of S, where S^2 is a unit sphere. Then the Gauss map is differentiable, and its differential dN_p of N at p is a linear map from $T_p(S)$ to $T_p(S)$. It measures the rate of the normal vector N in a neighborhood of p.

The following lemma provides a condition which must be satisfied when the unit normal vectors of a surface S change in the neighborhood of regular points. Its proof is found in Chapter 3, pp. 140 of [18].

Lemma 4.1 The differential $dN_p : T_p(S) \longrightarrow T_p(S)$ of the Gauss map is a self-adjoint linear map, that is, $(dN_p(w_1), w_2) = (w_1, dN_p(w_2))$ where w_1 and w_2 are two independent tangent vectors at a regular point p, and $(\cdot,\cdot)$ is the inner product of two vectors.

4.2.2 Interpolation of Two Parametric Curves

The symmetry of the linear map dN_p, implied by Lemma 4.1, entails a necessary condition that must be satisfied between tangent vectors and the rates of changes in normal vectors at a regular point. It implies that, given two regular curves passing through a regular point on a surface, the unit normal vector must change along each curve satisfying the equality in Lemma 4.1.

Consider the problem of Hermite interpolation of two parametric space curves with normal directions, meeting at a point. Let $C_1(u)$ and $C_2(v)$ be the two parametric curves with parametrically represented normal directions $N_1(u)$ and $N_2(v)$ such that $C_1(0) = C_2(0) = p$, and $N_1(0)$ and $N_2(0)$ are proportional, that is, the two curves meet at p and they share the same normal direction at p. We look for a surface S which smoothly interpolates the curves, that is,

- S must contain $C_1(u)$ and $C_2(v)$,
- the normals of tangent planes of S along the curves must coincide with the normals of the curves, and
- S is regular at p.

A straightforward application of Lemma 4.1 yields the following theorem which presents a necessary regularity condition on the curves and normal directions.

Theorem 4.1 Let $C_1(u)$ and $C_2(v)$ be two parametric curves with parametric normal directions $N_1(u)$ and $N_2(v)$ such that $C_1(0) = C_2(0) = p$, and that $N_1(0)$ and $N_2(0)$ are proportional. Then, any surface S, which interpolates the curves with tangent plane continuity, is singular at p unless $\frac{(N_1'(0), C_2'(0))}{\|N_1(0)\|} = \frac{(C_1'(0), N_2'(0))}{\|N_2(0)\|}$.

Proof: Suppose that S is a surface which smoothly interpolates the curves, and is regular at p. Then, we have a local parametrization $\mathbf{x} : U \longrightarrow V \cap S$ of an open set U in R^2 onto $V \cap S \subset R^3$ for a neighborhood V of p such that

- $\mathbf{x}(0,0) = p$,
- $\mathbf{x}_u = \frac{\partial \mathbf{x}}{\partial u}(0,0) = C_1'(0)$ and $\mathbf{x}_v = \frac{\partial \mathbf{x}}{\partial v}(0,0) = C_2'(0)$, and
- the Gauss map N of S is such that $N(C_1(u)) = \frac{N_1(u)}{\|N_1(u)\|}$ and $N(C_2(v)) = \frac{N_2(v)}{\|N_2(v)\|}$.

Then, by Lemma 4.1, for S to be regular at p, it should be that

$$(dN_p(\mathbf{x}_u), \mathbf{x}_v) = (\mathbf{x}_u, dN_p(\mathbf{x}_v)). \tag{4.1}$$

By the definition of the differential,

$$\begin{aligned} dN_p(\mathbf{x}_u) &= \frac{dN(C_1(u))}{du}\Big|_{u=0} \\ &= \frac{d\left(\frac{N_1(u)}{\|N_1(u)\|}\right)}{du}\Big|_{u=0} \end{aligned}$$

$$= \frac{N_1'(u) \| N_1(u) \| - N_1(u) \| N_1(u) \|'}{\| N_1(u) \|^2} |_{u=0}$$
$$= \frac{N_1'(0) \| N_1(0) \| - N_1(0) \| N_1(u) \|'_{u=0}}{\| N_1(0) \|^2}.$$

Since $(N_1(0), \mathbf{x}_v) = 0$, $(dN_p(\mathbf{x}_u), \mathbf{x}_v) = \frac{(N_1'(0), \mathbf{x}_v)}{\|N_1(0)\|} = \frac{(N_1'(0), C_2'(0))}{\|N_1(0)\|}$. In the same way, we get $(\mathbf{x}_u, dN_p(\mathbf{x}_v)) = \frac{(C_1'(0), N_2'(0))}{\|N_2(0)\|}$. Hence, the equation (4.1) becomes

$$\frac{(N_1'(0), C_2'(0))}{\| N_1(0) \|} = \frac{(C_1'(0), N_2'(0))}{\| N_2(0) \|}. \qquad \square$$

The above theorem implies that enforcing two curves to have the same normal vectors at an intersection point does not guarantee the regularity of an interpolating surface at the point. The equation in the theorem is a necessary condition for regularity, indicating that, if the given curves and their normals do not satisfy the equation, any smoothly interpolating surface must be singular at p.

This issue has been also addressed in the literature of parametric surface fitting. Peters [56] showed that not every mesh of parametric curves with well-defined tangent planes at the mesh points can be interpolated by smooth regularly parametrized surfaces with one surface patch per a mesh face. In [55], he used singularly parametrized surfaces to enclose a mesh point when mesh curves emanating from the point do not satisfy a constraint, called the vertex enclosure constraint.

4.3 Smoothing a Convex Polyhedron

In Section 4.1, we described how to compute a quintic triangular algebraic surface patch from a given augmented triangle. A convex polyhedron is smoothed by replacing its faces with the triangular patches meeting each other with tangent plane continuity. For augmented triangles $T = (p_0, p_1, p_2, n_0, n_1, n_2, npl_{01}, npl_{12}, npl_{20})$ of faces of a polyhedron, the normal data, i.e., three vertex normals and three edge normals, must be provided as well as the given three vertices. In some applications, the normal data may come with a solid, but only vertices and their facial information are usually provided. The vertex normal n_i at each vertex p_i can be computed by averaging the normals of the faces incident to the vertex. Also, we average the normals

of the faces incident to each edge (p_i, p_j), and take its cross product with the vector $p_j - p_i$ to get the edge normal vector npl_{ij}. After the normal data are computed, quadric wires are generated for a ρ value which is interactively selected by a user.

Example 4.1 Construction of Quadric Wire Frames

Figure 4.4 and 4.5 show two quadric wire frames for the same convex polyhedron[5] with the ρ values 0.4 and 0.75, respectively. $\square$

Example 4.2 Smoothed Polyhedra with Quintic Algebraic Surface Patches

Each of the 32 faces of the polyhedron in Example 4.1 is replaced by a quintic implicit algebraic surface which smoothly fleshes its quadric triangle. The result is the piecewise tangent plane continuous quintic algebraic surface mesh that smooths the given polyhedron. Figures 4.6, 4.7, and 4.8 illustrate the mesh for $\rho = 0.4, 0.5$, and 0.75, respectively. As mentioned before, ellipses, parabolas, and hyperbolas are used as quadric wires for $\rho = 0.4$, 0.5, and 0.75, respectively. $\square$

4.4 Towards Smoothing an Arbitrary Polyhedron

In the foregoing sections, we showed that algebraic surfaces of degree five, moderately low, can be effectively used in smoothing convex polyhedra. It is natural that we continue to work on smoothing an arbitrary polyhedron, not necessarily convex. In this section, we consider an approach to this open problem which is based on face subdivision.

4.4.1 A Characterization of Existence of Conic Curves

The convex polyhedron smoothing scheme presented in the previous sections consists of two steps. The first step is to construct a quadric wire frame of a given polyhedron, and then each quadric triangle is fleshed by a quintic algebraic surface. As will be seen, edges of a convex polyhedron can be always replaced by quadric wires

[5]This polyhedron is a gyroelongated triangular bicupola with its rectangular faces triangulated.

without creating cusp-like connections. However, in case of a nonconvex polyhedron, conic curves, that do not have inflection points, are not flexible enough to model nonconvex shapes.

In CAGD, piecewise polynomial cubic curves have been useful in fitting arbitrary shapes due to their desirable properties such as their capability of having C^2 continuity between curves, and zero curvature at inflection points [58]. On the other hand, conic splines have been found adequate in representing arbitrary shapes [53, 58]. Conics in rational polynomial form have the advantages, over cubics, of low computational cost and a rich body of mathematical results. In particular, adhering to conics allows us to continue to explore the class of quintic algebraic surfaces as geometric modeling tools.

In this subsection, we derive a criterion which determines if a quadric wire $W(t) = (C(t), N(t))$ can interpolate given two point and unit normal vector pairs (p_0, n_0), (p_1, n_1) in 3D space. Here, by interpolation of normal vectors, we mean strict interpolation where $W(0)$ and $W(1)$ have the same directions as n_0 and n_1, respectively. This restriction guarantees construction of quadric wire frames which are free of cusp-like connections.

We first consider the planar case, and derive a criterion which tells if there exists a conic curve that interpolates two given point and unit normal vector pairs $P_0 = (p_0, n_0)$ and $P_1 = (p_1, n_1)$ where $p_0 = (p_{0x}, p_{0y})$ and $p_1 = (p_{1x}, p_{1y})$ are two points on a plane with associated unit normal vectors $n_0 = (n_{0x}, n_{0y})$ and $n_1 = (n_{1x}, n_{1y})$. To be more precise, we give the following definition:

Definition 4.4 Let $P_0 = (p_0, n_0)$ and $P_1 = (p_1, n_1)$ be two given pairs. A conic segment $S(P_0, P_1)$ is said to smoothly interpolate P_0 and P_1 if there exists a nondegenerate conic curve $f(x, y) = ax^2 + 2hxy + by^2 + 2gx + 2fy + c$ such that

- $S(P_0, P_1)$ is a continuous segment of $f(x, y) = 0$,
- p_0 and p_1 are the end points of $S(P_0, P_1)$, and
- the gradients of $f(x, y) = 0$ at p_0 and p_1 have the same directions as n_0 and n_1, respectively. In other words, $\frac{\langle \nabla f(p_0), n_0 \rangle}{\|\nabla f(p_0)\| \cdot \|n_0\|} = 1$, and $\frac{\langle \nabla f(p_1), n_1 \rangle}{\|\nabla f(p_1)\| \cdot \|n_1\|} = 1$.

Given a pair $P = ((p_x, p_y), (n_x, n_y))$, we can define $T_P(x, y) = n_x(x - p_x) + n_y(y - p_y) = 0$ which is the equation of the tangent line that passes through (p_x, p_y) and has a normal direction (n_x, n_y). Note that the normal of the tangent line $T_P(x, y) = 0$ has the same direction as (n_x, n_y), and divides a plane into a positive halfspace $\{(x, y) \in \mathbb{R}^2 | T_P(x, y) > 0\}$, and a negative halfspace $\{(x, y) \in \mathbb{R}^2 | T_P(x, y) < 0\}$.

Lemma 4.2 Let p_0 and p_1 be on a nondegenerate conic $f(x, y) = ax^2 + 2hxy + by^2 + 2gx + 2fy + c = 0$. Then, $T_{(p_0, \nabla f(p_0))}(p_1) \cdot T_{(p_1, \nabla f(p_1))}(p_0) > 0$.

Proof: Without loss of generality, we assume that $p_0 = (0, 0)$, and $p_1 = (1, 0)$. Since $\nabla f(x, y) = (2ax + 2hy + 2g, 2hx + 2by + 2f)$, $\nabla f(0, 0) = (2g, 2f)$ and $\nabla f(1, 0) = (2a + 2g, 2h + 2f)$. Hence, $T_{(p_0, \nabla f(p_0))}(x, y) = 2gx + 2fy$, and $T_{(p_1, \nabla f(p_1))}(x, y) = (2a + 2g)(x - 1) + (2h + 2f)y$. From the containment conditions of the two points, $f(0, 0) = c = 0$, and $f(1, 0) = a + 2g + c = 0$. Then, $T_{(p_0, \nabla f(p_0))}(p_1) \cdot T_{(p_1, \nabla f(p_1))}(p_0) = 2g(-2a - 2g) = 2g(-2(-2g) - 2g) = 4g^2 \geq 0$. If $g = 0$, it follows that $a = c = g = 0$ in which case $f(x, y)$ reduces into two lines. Since we assume that $f(x, y)$ is nondegenerate, $g \neq 0$, and we have proven the lemma. □

The geometric interpretation of the inequality $T_{(p_0, \nabla f(p_0))}(p_1) \cdot T_{(p_1, \nabla f(p_1))}(p_0) > 0$ is that p_0 is on the positive (negative) halfspace of T_{P_1} if and only if p_1 is on the positive (negative) halfspace of T_{P_0}. The following theorem shows that this condition is, in fact, a sufficient and necessary condition.

Theorem 4.2 There exists a conic segment $S(P_0, P_1)$ that smoothly interpolates two pairs $P_0 = (p_0, n_0)$ and $P_1 = (p_1, n_1)$ if and only if $T_{P_0}(p_1) \cdot T_{P_1}(p_0) > 0$.

Proof: ($\Rightarrow$) Let $f(x, y) = 0$ be the conic that contains $S(P_0, P_1)$. From our definition of smooth interpolation, it follows that $T_{P_0}(p_1) \cdot T_{P_1}(p_0) = T_{(p_0, \nabla f(p_0))}(p_1) \cdot T_{(p_1, \nabla f(p_1))}(p_0)$ which is positive according to Lemma 4.2.
($\Leftarrow$) If $T_{P_0}(p_1) \cdot T_{P_1}(p_0) > 0$, then the conic segment on $q(x, y) = L(x, y)^2 - \kappa \cdot$

$T_{P_0}(x,y) \cdot T_{P_1}(x,y) = 0$ or $-q(x,y) = 0$ will smoothly interpolate the two pairs where $L(x,y) = 0$ is the line connecting p_0 and p_1, and κ is a constant [64].[6] □

Now, back to the original problem of computing a quadric wire that smoothly interpolates two given point and unit normal vector pairs $P_0 = (p_0, n_0)$ and $P_1 = (p_1, n_1)$ in $\mathbb{R}^3$. The concept of the tangent line in a plane is naturally extended to an oriented tangent plane $T_P(x,y,z) = n_x(x - p_x) + n_y(y - p_y) + n_z(z - p_z) = 0$ given a pair $P = ((p_x, p_y, p_z), (n_x, n_y, n_z))$ in 3D space, and this tangent plane divides 3D space into two halfspaces. In fact, we see that the inequality $T_{P_0}(p_1) \cdot T_{P_1}(p_0) > 0$ is also a criterion which determines if a quadric wire can smoothly interpolate two given pairs of points and normal vectors.

Corollary 4.1 Given two point and unit normal vector pairs $P_0 = (p_0, n_0)$ and $P_1 = (p_1, n_1)$ in 3D space, there exists a quadric wire $W(t) = (C(t), N(t))$, contained in a plane determined by a given plane normal vector npl_{01}, that smoothly interpolates the pairs if and only if $T_{P_0}(p_1) \cdot T_{P_1}(p_0) > 0$.

Proof: Consider the two pairs P_0 and P_1, their two tangent planes T_{P_0} and T_{P_1}, and the plane H which is defined by npl_{01}. Then, the intersection lines of H and T_{P_0}, T_{P_1} become the tangent lines in H to which a conic curve must be tangent. That is, the normal vectors of the tangent lines are the projections of the normal vectors of the tangent planes. Note that the positiveness and negativeness of halfspaces are inherited from 3D space to the plane H. Hence, we see that the inequality $T_{P_0}(p_1) \cdot T_{P_1}(p_0) > 0$ holds in 3D space if and only if its 2D version holds in H.

If there exists a conic curve in H, we can find a quadric surface which smoothly interpolates the given pairs, as explained before, and take $W(t)$ from this quadric surface that has the same gradient directions as the given two normal vectors. □

It is not difficult to imagine that when the averaging technique is used to compute vertex and edge normals of a convex polyhedron, the inequality criterion is always true for each edge, hence, a quadric wire frame can be easily computed.

[6] Thanks to Jia Xun Yu for pointing me to this.

4.4.2 Iterative Subdivision of Faces

In this subsection, we consider a procedure that tries to produce a quadric wire frame for a given polyhedron with arbitrary shape. This procedure checks if the inequality criterion holds for each edge, and if it does, the edge is replaced by a quadric wire, as before. If the criterion does not hold, it is inevitable that a curve with an inflection point must be used. In this procedure, we break the edge, and use two quadric wires meeting with C^1 continuity rather than using a cubic curve which would require a higher degree algebraic surface for Hermite interpolation. When an edge is broken, the triangular face incident to the edge is subdivided into a few subfaces depending on how many edges of the face are broken.

The following procedure subdivides each face of a given polyhedron iteratively until the criterion is met for all the edges.

Procedure 4.1 (Iterative Face Subdivision)

> *compute the vertex normals and edge normals;*
> do
>   /* beginning of a new phase */
>   for each *face of the current polyhedron* do
>   if *the face must be subdivided* then
>   *subdivide the face;*
>   endif
>   endfor
>   *update the polyhedron;*
> until *no face is subdivided in the current phase*

In each phase of the iteration, an edge is broken into two subedges when necessary, and a proper normal vector is associated with the new vertex. As of now, we do not know which way of associating normal vectors with the new vertices is the best, however, it must be such that the resulting new edges and their point and normal vector pairs satisfy the inequality criterion as much as possible without harming the aesthetics of a quadric wire frame to be computed. Once a quadric wire frame is constructed, we can apply the same technique to flesh each quadric triangle of the wire frame.

Much work remains in converting the above experimental procedure into a robust face subdivision algorithm. First, there are many degrees of freedom in replacing each edge with two quadric wires when necessary. The inflection point where two wires meet with each other and a normal vector at the point must be specified. Also, the two quadric wires need not be on the same planes but can be on different planes as long as the normal condition is satisfied at the inflection point. Also, there are two degrees of freedom in selecting the ρ values of the two quadric wires, although they can be used to achieve C^2 continuity at the inflection vertex [58]. Secondly, as the face subdivision process proceeds, some faces with bad shapes may be generated. The aspect ratio of a face or a triangle is defined as the ratio of the radius of the circumscribed circle to the radius of the inscribed circle. Triangles with large aspect ratios tend to produce more numerical errors in computation [78] as well as being inappropriate for display techniques such as Gouraud shading [33]. In our polyhedron smoothing scheme, it appears to be more difficult to remove self-intersections inside quadric triangles when the aspect ratios of faces are large. Hence, it is important to maintain aspect ratios of faces in a reasonable range by adaptively subdividing them.

Figure 4.9 shows a nonconvex polyhedron, and Figure 4.10 illustrates a curved object obtained as a result of smoothing the polyhedron. We observe wave-like oscillations between the surface patches of the subdivided faces, while the surface patches of the convex region produce pleasing curved approximations.

4.5 Summary

In this chapter, we explored the class of quintic algebraic surfaces to smooth a given convex polyhedron with triangular faces. In the presented scheme, a wire frame made of quadric curves and quadric normals was constructed first, and then the triple

of quadric curves corresponding to each face was fleshed with a quintic algebraic surface through Hermite interpolation and contour level least squares approximation. We observed that the minimum degree of algebraic surfaces for this polyhedron smoothing scheme is at least quintic, and also discussed how a triangular algebraic surface patch, whose vertices may be singular, is polygonized. The problem of smoothing a polyhedron with an arbitrary shape with quintic algebraic surfaces is still open. We need to devise an algorithm for constructing good quadric wire frames for nonconvex polyhedra.

There are some more open problems that need to be mentioned. First, a more robust method of generating the points and contour levels for least squares approximation is desirable. While the heuristic for least square approximation usually works well, sometimes we may have to change the value of α in S_1 and S_{-1} manually. Secondly, although the singularities at the vertices of triangular patches do not harm geometric continuity between them, it will be interesting, at least theoretically, to try to produce triangular patches which are regular at their vertices. This might be possible via subdivision techniques used in parametric surface fitting.

Our ultimate goal is to construct curved solids with quintic algebraic surface patches, and then to manipulate them through geometric operations such as boolean set operations. This ability will provide a geometric modeling system with a complex way of creating and manipulating models of physical objects with various geometries. Also, this research can be fully applied to visualization of three dimensional imaging data obtained from computed tomography (CT) and magnetic resonance imaging (MRI) techniques in medical imaging.

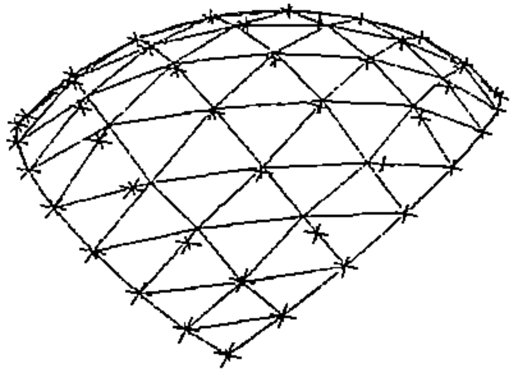

Figure 4.3 A Polygonization and Points Generated

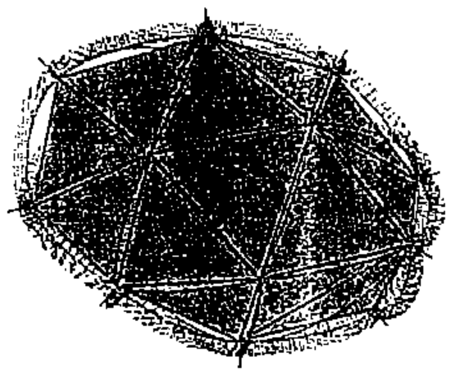

Figure 4.4 A Convex Polyhedron with Quadric Wires : $\rho = 0.4$

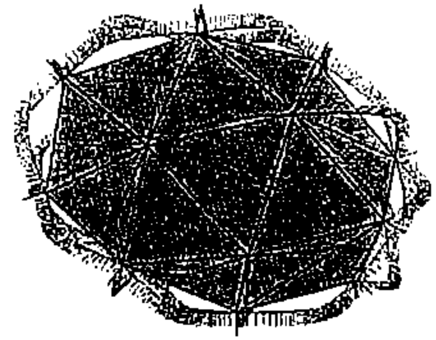

Figure 4.5 A Convex Polyhedron with Quadric Wires : $\rho = 0.75$

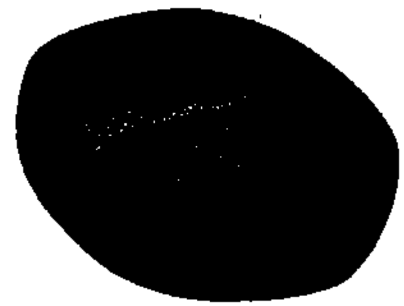

Figure 4.6 A Quintic Algebraic Surface Mesh : $\rho = 0.4$

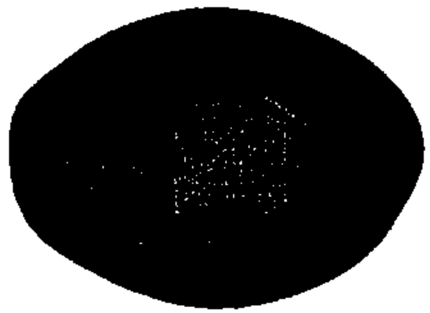

Figure 4.7 A Quintic Algebraic Surface Mesh : $\rho = 0.5$

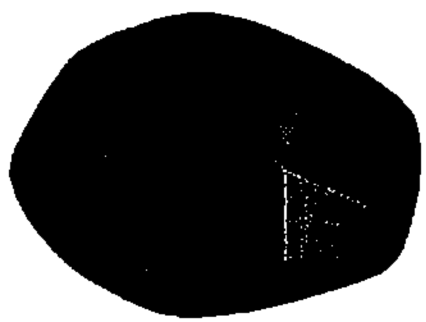

Figure 4.8 A Quintic Algebraic Surface Mesh : $\rho = 0.75$

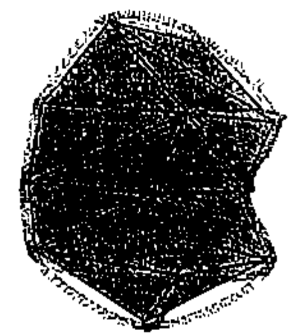

Figure 4.9 A Nonconvex Polyhedron after Faces Subdivided

Figure 4.10 A Quintic Algebraic Surface Mesh : $\rho = 0.5$

5. PIECEWISE LINEAR APPROXIMATION OF SPACE CURVES

Finding piecewise linear approximation of a digitized or densely sampled curve is an important problem in image processing, pattern recognition, geometric modeling, and computer graphics. Digitized curves occur as boundaries of regions or objects. Such curves, usually represented as sequences of points, may be measured by devices such as scanning digitizers or may be generated by evaluating parametric equations of space curves, or by tracing intersection curves given by implicit surface equations. They can also be obtained from experiments. For efficient manipulation of digitized curves, they are typically represented in the form of sequences of line segments. While the original curves are made of large sequences of points, their approximations are represented by a small number of line segments that are visually acceptable.

The piecewise linear approximation problem has received much attention, and there exist many approximation algorithms for this problem. The literature in related areas contains many heuristic methods that are direct and efficient even though, in general, they do not find an optimal approximation [22, 24, 52, 54, 60, 62, 61, 70, 74, 77]. This problem was also treated more theoretically in the area of computational geometry. Imai et al. [38] presented an $O(n^3)$ time algorithm for approximating a polygonal chain of length n with a minimal number of line segments within a given tolerance. The time complexity is reduced to $O(n^2 \log n)$ in [44, 72]. However, most of these works consider only planar curves as their input data, and little work has addressed space curve approximation. In many applications, a three dimensional (3D) object is designed with a set of boundary curves in 3D space which are represented as a set of equations or as a sequence of points. Hence, having a good approximation method for digitized space curves is essential. In [39], which is one of few works on 3D space curve approximation, a quintic B-spline is constructed for noisy data, and the length of the Darboux vector, also known as total curvature, is used as the criterion for segmentation of 3D curves. This method requires construction of quintic B-splines, explicit computation of curvature and torsion, and polynomial root isolation.

In this chapter, we consider how to quickly produce a good piecewise linear approximation of a digitized space curve with a smaller number of line segments. Both speed and quality are very important in most applications, and in particular, the heuristic algorithm presented in this chapter is used to polygonize implicit triangular algebraic surface patches computed in Chapter 4.

Our algorithm is based on the notions of curve length and spherical image, which are fundamental concepts in differential geometry [18, 41, 49]. In Section 5.1, we first define some terminology and give a mathematical formulation of the specific problem we are dealing with. This approximation problem is naturally reduced to a combinatorial minimax problem which can be stated as "Given n points, choose a smaller number m of points such that the maximum error of approximation is minimized." In Section 5.2, optimal approximation is found by an algorithm that runs in $O(n^3 \log m)$ time and $O(n^2 \log m)$ space. We describe, in Section 5.3, a fast heuristic iterative algorithm which requires $O(N_{iter} n)$ time and $O(n)$ space, where N_{iter} is a number of iterations carried out. Also, the performance of the heuristic algorithm on some test cases is analyzed. In Section 2.5, we illustrate applications of this fast heuristic algorithm in which space curves and implicit surfaces are adaptively linearized. We also apply the heuristic approximation algorithm to construction of adaptive binary space partitioning (BSP) trees for a class of objects made by revolution, where the linear approximation of a curve is naturally extended to linearly approximate the class of three dimensional curved objects, made by revolution, with BSP trees that are well-balanced.

5.1 Preliminaries

Definition 5.1 Let C be a space curve in $\mathbf{R}^3$. A space curve segment $C(a, b)$ is a connected portion of a curve C with end points $a, b \in \mathbf{R}^3$.

In order to define a curve segment without ambiguity, a tangent vector at a may be required. But we assume this vector is implicitly given.

Definition 5.2 A digitized space curve segment $\bar{C}(a,b,n)$ of order n is an ordered sequence $\{a = p_0, p_1, p_2, \cdots, p_n = b\}$ of points $p_i \in \mathbf{R}^3$, $i = 0, 1, \cdots n$, which approximates $C(a,b)$.

Approximation of a digitized space curve with a small number of line segments inevitably results in an approximation error. The quality of approximation is measured in terms of a given error norm that can be defined in many ways. Some commonly used ones are

1. maximum norm :
$$L_\infty = max\ e_i$$
2. 2-norm :
$$L_2 = (\sum e_i^2)^{\frac{1}{2}}$$
3. area norm : L_{area} = absolute area between curve segment and approximating line segment.

In this chapter, we use L_∞ as an error norm to measure a goodness of an approximation, although our algorithms in the later sections are also compatible with L_2.

Definition 5.3 A piecewise linear approximation $LA(\bar{C}, a, b, m)$ of order m to $\bar{C}(a,b,n)$ is an increasing sequence $\{0 = q_0, q_1, q_2, \cdots, q_m = n\}$ of indices to points in $\bar{C}$. An error $E(LA(\bar{C}, a, b, m))$ of a piecewise linear approximation LA is defined as $\max_{0 \le i \le m-1} E_{seg}(i)$ where the i-th segment error $E_{seg}(i)$ is $\max_{q_i \le j \le q_{i+1}} dist(p_j, line(p_{q_i}, p_{q_{i+1}}))$, and $dist(x, line(y,z))$ is the Euclidean distance from a point x to a line, determined by two points y and z. [1]

[1] For any point $x \in \mathbf{R}^3$, and two other points $y, z \in \mathbf{R}^3$, $(y \neq z)$, $dist(x, line(y,z))$ can be compactly expressed as $\| y - x + \frac{(x-y,z-y)}{\|z-y\|^2}(z-y) \|_2$ where $(\cdot,\cdot)$ is the dot product of two vectors and $\|\cdot\|$ is the length of a vector.

As pointed out in Pavlidis et al. [54], the problem of finding a piecewise linear approximation LA can be expressed in two ways :

1. find a $LA(\bar{C}, a, b, m)$ such that $E(LA) < \epsilon$ for a given bound ϵ and m is minimized.
2. find a $LA(\bar{C}, a, b, m)$ that minimizes $E(LA)$ for a given m.

We focus mainly on the second type of problem. However, we will also discuss briefly the first type of problem in Section 5.3.3.5.

Definition 5.4 The optimal piecewise linear approximation $LA^*(\bar{C}, a, b, m)$ of order m, given $\bar{C}(a,b,n)$ and an integer m $(n \ge m)$, is a piecewise linear approximation, not necessarily unique, such that $E(LA^*) \le E(LA)$ for any piecewise linear approximation LA of order m.

Given these definitions, the problem can be stated as :

Problem 5.1 Given $\bar{C}(a,b,n)$ and m, find $LA^*(\bar{C}, a, b, m)$.

5.2 An Optimal Solution

5.2.1 An Algorithm

A naive algorithm for Problem 5.1 would be as following :

Algorithm 5.1 (NAIVE)

> *temp* = ∞;
> *for all the possible* $\binom{n-1}{m-1}$ $LA(\bar{C}, a, b, m)$ do
>   *compute* $E(LA)$;
>   if $E(LA) <$ *temp* then $LA^* = LA$; *temp* $= E(LA)$;
> endfor

Note that this problem has a recursive nature, that is, it can be naturally subdivided into two subproblems of the same type. Dynamic programming, which is a general problem solving technique widely used in many disciplines [2], can be applied to this problem to produce a rather straightforward algorithm. We first give an algorithm which works in case m is a power of 2. Then the algorithm is slightly modified for an arbitrary m.

Define E_{ij}^l to be the error of $LA^*(\bar{C}, p_i, p_j, l)$, that is, the smallest error of all piecewise linear approximations with l $(= 2^d)$ segments to the portion of $\bar{C}$ from p_i to p_j. Then E_{ij}^l can be expressed in terms of $E_{ik}^{\frac{l}{2}}$ and $E_{kj}^{\frac{l}{2}}$ as following :

$$E_{ij}^{2^d} = \min_{i<k<j} \max\{E_{ik}^{2^{d-1}}, E_{kj}^{2^{d-1}}\} \quad \mathit{for} \;\; 0 \leq i < j \leq n \; \mathit{and} \; d > 0, \tag{5.1}$$

where $E_{ij}^{2^d} = 0$ if $j - i \leq 2^d$.

The recursive relation gives rise to the following dynamic programming algorithm which computes the minimum error E_{0n}^m and its corresponding LA^* :

Algorithm 5.2 (DYNAMIC)

```
/* basis step */
for i = 0 to n − 1 do
  for j = i + 1 to n do
    compute E_ij^1;
  endfor
endfor
/* inductive step */
for d = 1 to log m do
  for i = 0 to n − 2^d − 1 do
    for j = i + 2^d + 1 to n do
      E_ij^(2^d) = max{E_ik'^(2^(d-1)), E_k'j^(2^(d-1))} = min_{i<k<j} max{E_ik^(2^(d-1)), E_kj^(2^(d-1))};
      K_ij^d = k';
    endfor
  endfor
endfor
construct LA* from K_ij^d;
```


In the basis step, E_{ij}^1 is computed by calculating the distances from the points p_k, $i < k < j$ to the line passing through p_i and p_j, and taking their maximum. K_{ij}^d is needed to recursively construct the optimal piecewise linear approximation once E_{0n}^m is computed. Note the recursive relation $LA^*(\bar{C}, p_i, p_j, 2^d) = LA^*(\bar{C}, p_i, p_{K_{ij}^d}, 2^{d-1}) \cup LA^*(\bar{C}, p_{K_{ij}^d}, p_j, 2^{d-1})$.

5.2.2 Time and Space Complexity

Since E_{ij}^1 is computed in $O(j-i)$ time, the basis step requires $O(\sum_{i=0}^{n-1}\sum_{j=i+1}^{n}(j-i)) = O(n^3)$ time. Similarly, $E_{ij}^{2^d}$ can be computed in $O(j-i)$ time. So, the inductive step needs $O(n^3 \log m)$ time. Also, LA^* can be constructed in $O(m)$ time. These three time bounds are combined into $O(n^3 \log m)$.

The algorithm needs $O(n^2)$ space for storing a table for $E_{ij}^{2^d}$. Also, $O(n^2 \log m)$ space is required to save K_{ij}^d, $d = 1, 2, \cdots, \log m$. Hence, the space complexity is $O(n^2 \log m)$.

5.2.3 An Algorithm for Arbitrary m

In the algorithm $DYNAMIC$, l in E_{ij}^l is doubled in each step. We can imagine a computation tree for this recursive computation where its root has value m, and each node with value x has two children with values $\frac{x}{2}$. The nodes of any path from a leaf to a root have values, $1, 2, 2^2, 2^3, \cdots, m$, and we can view $DYNAMIC$ as traversal of the path from a leaf to a root computing, by merging two children, E_{ij}^l, where l is a value of a node.

When m is not a power of 2, we can also think of an imaginary computation tree which is constructed as following. First, m is a root of the tree. The root has two children with values, m' and $m - m'$ where m' is the largest power of 2 less than m.

Then, a complete subtree for m' is built as when m' is a power of 2, and $m - m'$ is divided in the same way as m was. In this tree, there are two different types of paths from a leaf to a root. One is a power path of nodes whose values are powers of 2, and the other is a nonpower path of nodes whose values are not the powers of 2. In this case, those two paths should be traversed in parallel. By synchronizing the order of traversal of each path, and using two tables, one for each path, we can compute E_{ij}^m.

For example, let $m = 27$. The power path is $1 \to 2 \to 4 \to 8 \to 16$, and the nonpower path is $1 \to 3 \to 11 \to 27$. First, E_{ij}^1 is copied into each table. Then, E_{ij}^2 in the power path is computed and is stored in its table. Since E_{ij}^2 and E_{ij}^1 are available, E_{ij}^3 in the nonpower path can be computed and is stored in its table. Then, E_{ij}^4 and E_{ij}^8 are computed in the power table. E_{ij}^8 and E_{ij}^3 are used to compute E_{ij}^{11} in the nonpower table. In this way, the tree is traversed to compute E_{ij}^{27}. It is not difficult to see that this modification only increases both time and space complexities by constant factors.

5.3 A Heuristic Solution

Even though the algorithm *DYNAMIC* finds an optimal approximation, the time and space requirement is excessive. As stated in Section 5.3.3.4, the algorithm is extremely slow even for modest n, for example, $n = 400$. In many applications, it is more desirable to generate quickly a good, but not necessarily optimal, approximation. In this section, we describe a heuristic algorithm which consists of two parts, computation of an initial approximation and iterative refinement of the approximation. Our heuristic algorithm is based upon the observation that the error of a segment is a function of the length of the curve segment, and the total absolute change of the angles of tangent vectors along the curve segment. The longer curve segment tends to have the larger segment error. Also, the total angle change is a measure of how much a curve segment is bent. However, it is

illustrated in the next two subsections that neither measure alone is a good heuristic. Our heuristic in Section 5.3.3 is a weighted sum of these two measures, and this simple combined measure yields a good initial guess.

5.3.1 Curve Length Subdivision

Assume we have a parametric representation $C(t)$ of a curve C. The first heuristic is to divide a curve segment into subsegments with the same curve length where the curve length is defined to be $\int_a^b \| \frac{dC(t)}{dt} \| \, dt$. This quantity is usually approximated by the chord length as following.

Given a digitized curve $\bar{C}(a, b, n) = \{a = p_0, p_1, \cdots, p_n = b\}$, consider a parametric curve $C(t)$ where $C(0) = p_0$ and $C(l) = p_n$. Then,

$$
\begin{aligned}
\int_0^l \| \frac{dC(t)}{dt} \| \, dt &\approx \sum_{i=0}^{n-1} \| \frac{p_{i+1} - p_i}{d(p_i, p_{i+1})} \| \, d(p_i, p_{i+1}) \\
&= \sum_{i=0}^{n-1} \| p_{i+1} - p_i \| \\
&= \sum_{i=0}^{n-1} d(p_i, p_{i+1})
\end{aligned}
$$

where $d(p_i, p_{i+1})$ is the Euclidean distance between two points p_i and p_{i+1} in $\mathbf{R}^3$.

Algorithm 5.3 (LENGTH)

> /* let $L_{seg}(i, j)$ be $\sum_{k=i}^{j-1} d(p_k, p_{k+1})$ */
> *compute total* $= \sum_{k=0}^{n-1} d(p_k, p_{k+1})$;
> *seglength* $=$ *ceil(total/m)*;
> $q_0 = 0$; $i = 0$;
> while $i < m - 1$ do
>   *find the largest* j *such that* $L_{seg}(q_i, j) <$ *seglength*;
> $q_{i+1} = j$; $i = i + 1$;
> endwhile
> $q_m = n$;

Figure 5.2 (upper left) indicates that this algorithm produces a LA which approximates $\tilde{C}$ quite well in flat regions of a curve, and poorly in highly curved regions.

5.3.2 Spherical Image Subdivision

Consider a curve $C(s)$ with an arc length parameter s [41, 49]. When all unit tangent vectors $T(s)$ of $C(s)$ are moved to the origin, their end points will describe a curve on the unit sphere. This curve is called the spherical image or spherical indicatrix of $C(s)$. Given a curve segment, the length of the corresponding spherical image implies how much the unit tangent vector changes its direction along the curve segment. Hence, it provides us with a measure of the degree to which a curve segment is bent. It is easily shown that the curvature $\kappa(s)$ is equal to the ratio of the arc length of the spherical image, and the arc length of $C(s)$. So, the length of the spherical image corresponding to a curve segment $C(s) : [0, l]$ is $\int_0^l \kappa(s)\,ds$. $\int_0^l \kappa(s)\,ds$ is sometimes called the total curvature [49], while it also can mean the length of the Darboux vector [41]. In practice, this quantity must be approximated.

Given a digitized curve $\tilde{C}(a, b, n) = \{a = p_0, p_1, \cdots, p_n = b\}$, consider an imaginary parametric curve $C(s)$ of arc length parameter s where $C(0) = p_0$ and $C(l) = p_n$. At a point p_i, $s \approx cl(p_0, p_i)$ such that $C(s) = p_i$, where $cl(p_0, p_i) = \sum_{j=0}^{i-1} d(p_j, p_{j+1})$. Then, the curvature is approximated as follows :

$$\begin{aligned} \kappa(s) &= \| \lim_{\delta s \to 0} \frac{T(s+\delta s) - T(s)}{\delta s} \| \\ &\approx \| \frac{t_{i+1} - t_i}{d(p_i, p_{i+1})} \| \qquad (1) \end{aligned}$$

where t_i is an approximated unit tangent vector. (We will discuss how to get t_i shortly.) Then,

$$\begin{aligned} \int_0^l \kappa(s)\,ds &\approx \sum_{i=0}^{n-1} \| \frac{t_{i+1} - t_i}{d(p_i, p_{i+1})} \| \, d(p_i, p_{i+1}) \\ &= \sum_{i=0}^{n-1} \| t_{i+1} - t_i \| \\ &= \sum_{i=0}^{n-1} d(t_i, t_{i+1}). \end{aligned}$$

The simple forward-difference approximation (1) to $\kappa(s)$ can be replaced by the popular central-difference approximation $\frac{d(t_{i-1}, t_{i+1})}{d(p_{i-1}, p_i) + d(p_i, p_{i+1})}$ which is a much better approximation when the points are close together. Integration can be also replaced by a better approximation formula. See [19] for more numerical techniques.

In this second heuristic method, $\tilde{C}(a, b, n)$ is subdivided into $LA(\tilde{C}, a, b, m) = \{0 = q_0, q_1, q_2, \cdots, q_m = n\}$ such that each subsegment has the same length of the spherical image.

Algorithm 5.4 (IMAGE)

> /* let $I_{seg}(i, j)$ be $\sum_{k=i}^{j-1} d(t_k, t_{k+1})$ */
> *compute total* $= \sum_{k=0}^{n-1} d(t_k, t_{k+1})$;
> *segind* $=$ *ceil*(*total*/m);
> $q_0 = 0$; $i = 0$;
> while $i < m - 1$ do
>   *find the largest* j *such that* $I_{seg}(q_i, j) <$ *segind*;
> $q_{i+1} = j$; $i = i + 1$;
> endwhile
> $q_m = n$;

The quantity $I_{seg}(q_i, q_j)$ is an approximating measure of the length of the spherical image of the segment from p_{q_i} to $p_{q_{i+1}}$, that is, $I_{seg}(q_i, q_j)$ is the total absolute change in the angles of the tangent vectors. Hence, this algorithm is sensitive to high curvature. In Figure 5.2 (upper right), we can see $IMAGE$ returns a LA which approximates $\tilde{C}$ poorly in flat portions of a curve, and very well in highly curved portions.

In the above algorithm, tangent vector information is used to subdivide a curve. If the digitized space curve has been generated from equations, say a parametric equation or two implicit equations, the tangent vector at each sample point can be computed directly from them. When instead a digitized curve has been given in terms of a sequence of points, or direct computation of tangent vectors from given equations is expensive, the tangent vector t_k to a curve C at p_k still can be approximated by averaging the directions of the neighboring lines of p_k in $\vec{C}$. In our implementation, the tangent vector is approximated by 5 successive points as follows [57]:

$$t_k = (1 - \frac{\alpha}{\alpha+\beta}) \cdot v_i + \frac{\alpha}{\alpha+\beta} \cdot v_{i+1},$$

where $\alpha = \| v_{i-1} \times v_i \|, \beta = \| v_{i+1} \times v_{i+2} \|$, $v_i = p_i - p_{i-1}$, and $\times$ means a cross product of two vectors. In case the digitized curve is open, the Bessel conditions are applied for the tangents at the end points as follows [21]:

$$v_0 = 2v_1 - v_2, \qquad v_{-1} = 2v_0 - v_1,$$
$$v_{n+1} = 2v_n - v_{n-1}, \quad v_{n+2} = 2v_{n+1} - v_n.$$

5.3.3 Heuristic Subdivision

Now, we give a heuristic algorithm which combines the two techniques. It consists of two steps: generation of an initial piecewise linear approximation LA_0, and iterative refinement of the piecewise linear approximation LA_k to produce LA_{k+1}.

5.3.3.1 Computation of an Initial Approximation : LA_0

An initial LA_0 is computed by an algorithm which is a combination of $LENGTH$ and $IMAGE$.

The weight, α is a parameter which controls the relative emphasis between curve length and spherical image, and is empirically chosen.

Algorithm 5.5 (INIT)

select some value of α $(0 \le \alpha \le 1)$;
compute $total = \sum_{k=0}^{n-1}(\alpha \cdot d(p_k, p_{k+1}) + (1-\alpha) \cdot d(t_k, t_{k+1}))$;
$segsum = ceil(total/m)$;
$q_0 = 0$; $i = 0$;
while $i < m - 1$ do
  find the largest j *such that* $\alpha \cdot L_{seg}(q_i, j) + (1-\alpha) \cdot I_{seg}(q_i, j) < segsum$;
  $q_{i+1} = j$; $i = i + 1$;
endwhile
$q_m = n$;

See Figure 5.2 (bottom left).

5.3.3.2 Iterative Refinement of Approximations : LA_k

The hybrid algorithm $INIT$ generally produces a good piecewise linear approximation. The next step is to diffuse errors iteratively in order to refine the initial approximation. Note that each segment is made of a sequence of consecutive points of a digitized curve, and it is approximated by a line connecting its end points. Usually, the error of a segment decreases as either of its end points is assigned to its neighboring segment. Hence, the basic idea in the following iterative algorithm is to move one of the end points of a segment with larger error to its neighboring segment with less error, expecting a decrease of the total error of the new LA. In the kth step of the following algorithm $ITER$, each segment of LA_k is examined , diffusing, if possible, its error to one of its neighbors. LA_k tends to quickly converge to a minimal LA which is a local minimum. See Figure 5.2 (bottom right) and Figure 5.3.

Algorithm 5.6 (ITER)

compute LA_0 *from* $INIT$;
$k = 0$;
do until *(satisfied)*
  compute errors of segments in LA_k;
  $curmax = E(LA_k(\vec{C}, a, b, m))$;
  for $i = 0$ to $m - 1$ do
    if the error of i-th segment is larger than
      that of either of its neighboring segments
    then *move the i-th segment's end points to the neighbor*

only if this change does not result in segment errors
larger than curmax;
endif
endfor
$LA_{k+1} = LA_k$;
enddo

5.3.3.3 Time and Space Complexity

First, $O(n)$ time is needed to approximate the tangent vector at each point. The algorithm *INIT* needs to scan the points and tangent vectors to compute L_{seg} and I_{seg} first, and then L_{seg} and I_{seg} are scanned to divide the digitized curve. Hence, it takes $O(n)$ time. Now, consider the algorithm *ITER*. First, the segment errors of LA_k are computed in $O(n)$ time. In the for loop, each segment and its two neighbors are examined; hence, each segment is examined twice. Since the segment error must be computed for each segment, the for loop requires $O(n)$ computation. Therefore, *ITER* takes $O(N_{iter}n)$ time where N_{iter} is the number of iterations. So, the time complexity of the heuristic algorithm is $O(N_{iter}n)$, and it is easy to see $O(n)$ space is sufficient for storing input data and intermediate data.

5.3.3.4 Performance

We have implemented both the optimal and heuristic algorithms on a Sun 4 workstation and a Personal Iris workstation, experimented with test data. [2]

[2]

1. Figure 5.2 : Folium of Descartes
   (a) equation : $C(t) = (\frac{3t}{1+t^3}, \frac{3t^2}{1+t^3}, 0)$ or $(f(x,y,z) = x^3 - 3xy + y^3,\ g(x,y,z) = z)$
   (b) $n = 109$, $m = 20$
2. Figure 5.3 : A Human Profile and a Goblet
   (a) Points were generated from 12 rational Bezier curves in [57], and then slightly disturbed.
   (b) (profile) $n = 169$, $m = 20$
   (c) (goblet) $n = 237$, $m = 20$

Tables 5.1, 5.2, 5.3, and 5.4 show their performance for selected test data. The integer in parentheses is the number of iterations needed to arrive at the local minimum. The bottom row (LA_k/LA^*) of each table indicates the performance of our heuristic algorithm, and it is observed that the optimal solution is approximated reasonably well. The program for the heuristic algorithm computes the approximate solution quickly (immediately or in a few seconds depending on how many iterations are needed.) On the other hand, it takes about 45 minutes to compute the optimal solution for the ($n = 404, m = 64$) example of Table 5.3.

5.3.3.5 The Center of Curvature

We now briefly consider the following variant of the piecewise linear approximation problem : "find a $LA(\bar{C}, a, b, m)$ such that $E(LA) < \epsilon$ for a given bound ϵ and m is minimized." Even though our heuristic algorithm was invented for an arbitrary number of subsegments, we can use it for dividing a segment into 2 subsegments. One simple algorithm would be to recursively divide a curve segment until the error in each subsegment is less than ϵ.

If a curve segment is to be divided into only two subsegments, the notion of the center of curvature can be applied. As before, assume we have a parametric representation $C(s)$ of a curve C, where s is an arc length parameter, and $\kappa(s)$ is its curvature. Consider a curve segment defined by an interval $[0, l]$. Then the

3. Figure 5.4 : A Four Leaved Rose
   (a) equation : $(f(x,y,z) = x^6 + 3x^4y^2 - 4x^2y^2 + 3x^2y^4 + y^6,\ g(x,y,z) = z)$
   (b) $n = 400$, $m = 64$
4. Figure 5.5 : A Nonplanar Quartic Curve
   (a) equation : $(f(x,y,z) = 36x^2 + 81y^2 + 9z^2 - 324,\ g(x,y,z) = x^2 + y^2 - 3.94)$
   (b) $n = 404$, $m = 32$
5. Figure 5.6 : A Nonplanar Sextic Curve
   (a) equation : $(f(x,y,z) = y^2 - x^2 - x^3,\ g(x,y,z) = z - x^2 + x - 2)$
   (b) $n = 234$, $m = 20$

center of curvature, defined by $c_\kappa = \int_0^l s\kappa(s)\,ds / \int_0^l \kappa(s)\,ds$, can be used as a heuristic that divides a curve segment $C(s) : [0, l]$ into two subsegments $C(s) : [0, c_\kappa]$ and $C(s) : [c_\kappa, l]$.

Again, c_κ needs to be approximated. For a digitized curve $\bar{C}(a, b, n) = \{a = p_0, p_1, \cdots, p_n = b\}$, consider an imaginary parametric curve $C(s)$ of arc length parameter s where $C(0) = p_0$ and $C(l) = p_n$. Then, at a point p_i, $s \approx cl(p_0, p_i)$ such that $C(s) = p_i$, where $cl(p_0, p_i) = \sum_{j=0}^{i-1} d(p_j, p_{j+1})$. Together with the approximation of the denominator given before, the following expression results in an approximation of c_κ :

$$\begin{aligned}
\int_0^l s\kappa(s)\,ds &\approx \sum_{i=0}^{n-1} cl(p_0, p_i) \left\| \frac{t_{i+1} - t_i}{d(p_i, p_{i+1})} \right\| d(p_i, p_{i+1}) \\
&= \sum_{i=0}^{n-1} cl(p_0, p_i) \| t_{i+1} - t_i \| \\
&= \sum_{i=0}^{n-1} cl(p_0, p_i) d(t_i, t_{i+1}).
\end{aligned}$$

5.4 Applications

5.4.1 Adaptive Display of Space Curve Segments

Our heuristic algorithm is well suited to producing a piecewise linear approximation of a space curve segment given in parametric or implicit form. First, the curve segment is densely sampled, and then the linear approximation algorithm filters the sampled points, producing a good approximation to the curve segment. Points on a parametric curve are easily generated. A curve represented by two implicit surfaces or an implicit surface and a parametric surface, can be traced using a surface intersection algorithm [7]. The space curve tracing algorithm is fast when the degrees of curves are in a reasonable range and there are no singular points along the curve segment. As seen clearly in the examples, a small number of line segments, adaptively filtered, can approximate a curve segment well, resulting in fast display. Figure 5.2 and 5.4 are two examples of planar curves, and Figure 5.5 and 5.6 are examples of nonplanar space curves.

5.4.2 Adaptive Display of Implicit Surface Patches

In Section 4.1.5, we showed how our heuristic algorithm could be used to generate adaptive polygonizations of implicit triangular quintic patches in the hope of placing more triangles in the highly curved portions. Figure 5.7 is another example of the adaptive polygonizations of a triangular patch of a quartic algebraic surface $f(x, y, z) = 0.01853292z^4 - 1.14809166y^2z^2 - 1.14809166x^2z^2 + 0.99982830z^2 - 1.16662458y^4 - 1.14809166x^2y^2 + 2.1849858y^2 + 0.01853292x^4 + 0.99982830x^2 - 0.72183450$.

5.4.3 Construction of Binary Space Partitioning Trees

The Binary Space Partitioning (BSP) tree has been shown to provide an effective representation of polyhedra through the use of spatial subdivision, and is an alternative to the topologically based B-reps. It represents a recursive, hierarchical partitioning, or subdivision, of d dimensional space. It is most easily understood as a process which takes a subspace and partitions it by any hyperplane that intersects the subspace's interior. This produces two new subspaces that can be partitioned further.

An example of a BSP tree in 2D can be formed by using lines to recursively partition the plane. Figure 5.1(a) shows a BSP tree induced partitioning of the plane and (b) shows the corresponding binary tree. The root node represents the entire plane. A binary partitioning of the plane is formed by the line labeled u, resulting in a negative halfspace and a positive halfspace. These two halfspaces are represented by the left and right children of the root, respectively. A binary partitioning of each halfspace may then be performed, as in the figure, and so on recursively. When subdivision terminates, the leaf node will correspond to an unpartitioned region, called a cell.

The primary use of BSP trees to date has been to represent polytopes. This is accomplished by simply associating with each cell of the tree a single boolean attribute in or out. If, in Figure 5.1, we choose cells 1 and 5 to be in cells, and the rest to be

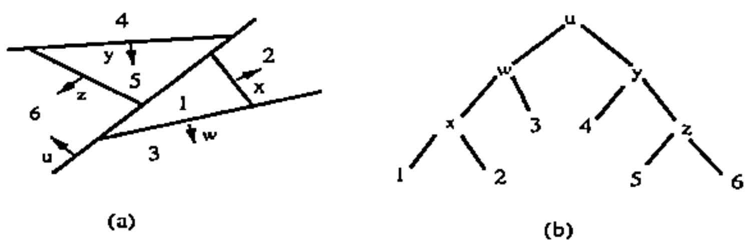

Figure 5.1 Partitioning of the Plane (a), and its BSP Tree (b)

out cells, we will have determined a concave polygon of six sides. This method, while conceptually very simple, is capable of representing the entire domain of polytopes, including unbounded and nonmanifold varieties. Moreover, the algorithms that use the BSP tree representation of space are simple and uniform over the entire domain. This is because the algorithms only operate on the tree one node at a time and so are insensitive to the complexity of the tree.

A number of BSP tree algorithms are known, including affine transformations, set operations, and rendering [48]. The computational complexity of these algorithms depends upon the shape and size of each tree. For example, consider point classification. The tree is simply traversed, and at each node the location of the point with respect to the node's hyperplane determines whether to take the left or right branch; this continues until a leaf is reached. The cost of this is the length of the path taken. Now, if this point is chosen from a uniform distribution of points over some sample space of volume v, then for any cell c with volume v_c at tree depth d_c, the probability p_c of reaching c is simply $\frac{v_c}{v}$ and the cost is d_c. So an optimal expected case BSP tree for point classification would be a tree for which the sum of $p_c d_c$ over all c is minimized. If the embedding space is one dimensional, then this is the classic problem of constructing an optimal binary search tree; a problem solved by dynamic programming.

The essential idea here is that the largest cells should have the shortest paths and smallest cells the longest. For example, satisfying this objective function globally generates bounding volumes as a by-product: if a polytope's volume is somewhat smaller than the sample space's volume, constructing a bounding volume with the first hyperplanes of the tree results in large out cells with very small depths. Now, in the general case in which the query object q has extent, i.e. is not a point, then q will lie in more than one cell, and a subgraph of the tree will be visited. Thus the cost of the query is the number of nodes in this subgraph. This leads to a more complicated objective function, which we do not intend to examine here, but the intuition taken from point classification remains valid.

We use these ideas in conjunction with the linear approximation methods, described before, to build good expected case trees for solids defined as surfaces of revolution (that is, we expect these trees to be good). First, we orthogonally project the curve to be revolved onto the axis of rotation, which is taken to be the vertical z-axis. We then partition space with horizontal planes where each plane contains one of the linearly approximated curve points. The BSP tree representing this is a nearly balanced tree, and each cell will contain the surface resulting from the revolution of a single curve segment.

Now the revolution of the curve need not be along a circle, but can be any convex path for which we have constructed a linear approximation. Thus each face of the solid will be a quadrilateral in which the upper and lower edges lie in consecutive horizontal partitioning planes, are parallel, and are instances of a single path edge at some distance from the axis of revolution that is determined by the revolved curve. Now the BSP subtree for the surface between horizontal planes is obtained by recursively partitioning the path of revolution to form a nearly balanced tree.

The method we use is one that in 2D generates for any n-sided convex polygon a corresponding nearly balanced BSP tree of size $O(n)$ and height $O(\log n)$. The path curve is first divided into four sub-curves, one for each quadrant, and a hyperplane containing the first and last points is constructed. By convexity, a sub-curve lies entirely in one halfspace of its corresponding hyperplane, and we call that halfspace the outside halfspace and the opposite halfspace the inside halfspace. The intersection of the four inside halfspaces is entirely inside the polygon, and so forms an in cell of the BSP tree. We then construct independently a tree for each subcurve, recursively.

We first choose the median segment of the subcurve and partition by the plane of the corresponding face. Since the path curve is convex, all of the faces will be in the inside halfspace of this plane and an out cell can be created in its outside halfspace. Now each nonhorizontal edge of the median face is used to define a partitioning plane which also contains the first/last points of the subcurve. All of the faces corresponding to this subcurves' edges are in the outside halfspace, and so an in cell can be created in its inside halfspace. We have now bisected the subcurve by these planes which contain no faces and can recurse on them. The recursion continues until only a small number of faces/segments remain, say 6, at which point only face planes are used for partitioning, since the cost of the non-face partitioning planes out-weights their contribution to balancing the tree. The result for a path curve of n edges is a nearly balanced tree of size $< 3n$ and height $O(\log n)$.

In some sense, we have constructed a tree that is the cross product of the path curve and the revolved curve: we build a tree of horizontal planes that partitions the revolved curve, and then we form slices of the object by constructing a tree for each segment of the path curve. If the revolved curve has m segments, then the number of faces is nm and the BSP tree is of size $O(nm)$ and height $O(\log nm)$ $= O(\log n + \log m)$.

The object in Figure 5.8 was made by rotating the curve in Figure 5.3 around an ellipse. Its BSP tree is fairly well-balanced. The goblet in Figure 5.9 and 5.10 were made by constructing two objects using the curve in Figure 5.3, and then applying a difference operation to carve a hole in the goblet. The BSP trees in Figure 5.9 were obtained after applying the difference operation, and then a union operation with the ball. It is observed that set operations on well-balanced BSP trees result in well-balanced trees. The set operation and display were done in SCULPT [47], which is an interactive modeling system based on BSP trees.

5.5 Summary

In this chapter, we discussed the problem of piecewise linear approximation of a densely sampled digitized 3D curve. Two algorithms were presented. The algorithm $DYNAMIC$ finds the optimal linear approximation at the high expense of $O(n^3 \log m)$ time and $O(n^2 \log m)$ space. It would be interesting to see if these time and space bounds can be reduced. The algorithm, made of $INIT$ and $ITER$, computes a heuristic linear approximation, based on the fundamental notions of curve length and spherical image of a space curve. This heuristic algorithm finds a good linear approximation quickly. We also showed that our heuristic algorithm can be applied to display of space curves and implicit surfaces, and to adaptive construction of well-balanced binary space partitioning trees of objects created by revolution.

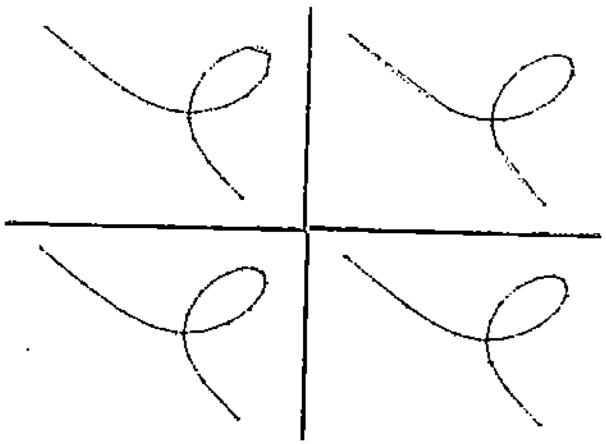

Figure 5.2 Folium of Descartes

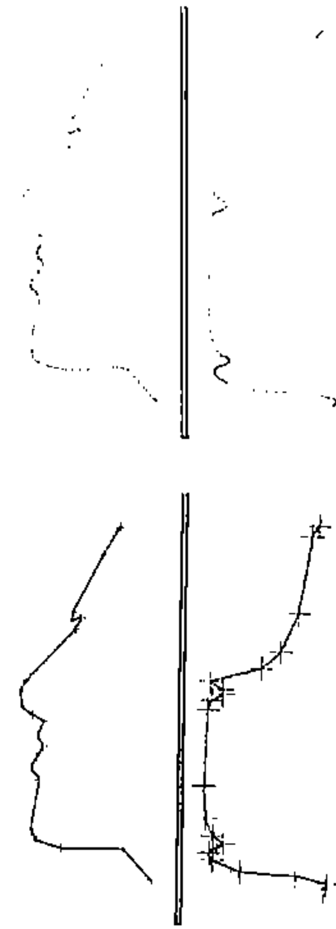

Figure 5.3 A Human Profile and a Goblet

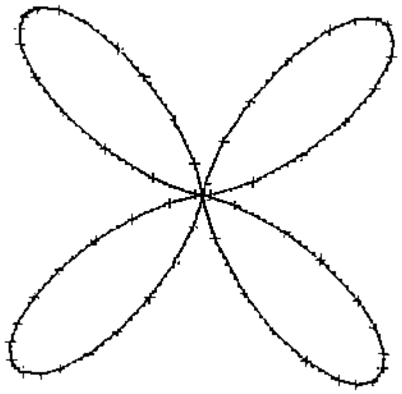

Figure 5.4 A Four Leaved Rose

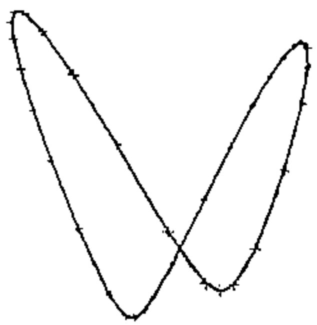

Figure 5.5 A Nonplanar Quartic Curve

Figure 5.6 A Nonplanar Sextic Curve

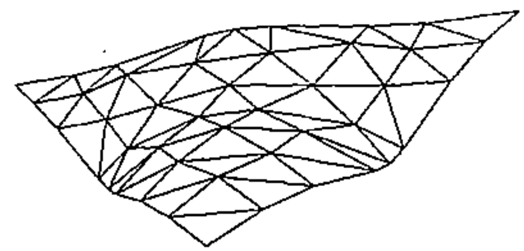

Figure 5.7 A Quartic Surface Patch

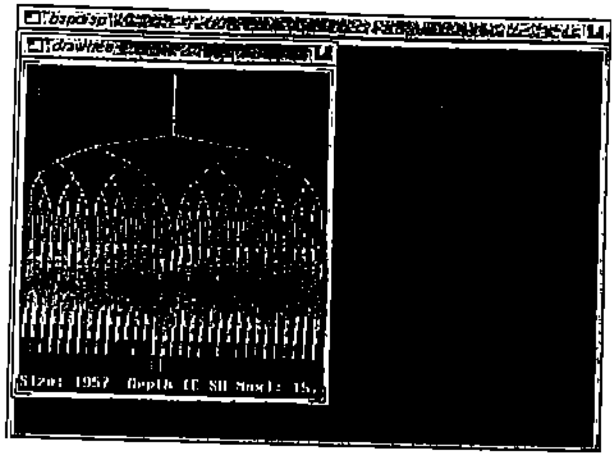

Figure 5.8 A Human Profile Rotated

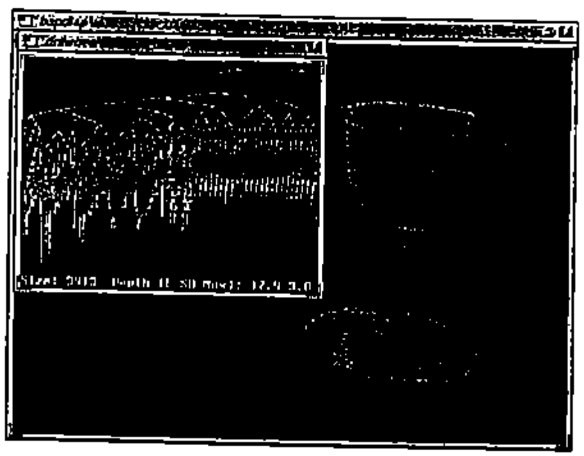

Figure 5.9 A Goblet in BSPT

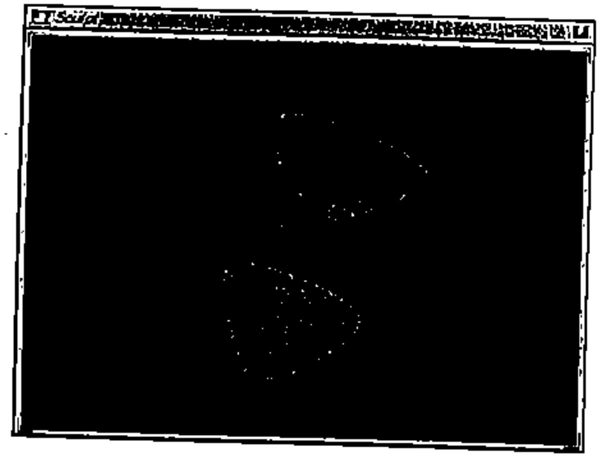

Figure 5.10 Another Goblet in BSPT

Table 5.1 Folium of Descartes

n	109				
m	4	8	16	32	64
LA_0	5.25619e-1	2.21325e-1	8.09768e-2	2.66073e-2	8.98677e-3
LA_k	4.02838e-1	1.12507e-1	3.08774e-2	1.23695e-2	3.76437e-3
(k)	(5)	(9)	(17)	(16)	(14)
LA^*	4.02838e-1	1.12507e-1	3.04525e-2	8.94592e-3	2.76188e-3
LA_k/LA^*	1.000	1.000	1.014	1.393	1.363

Table 5.2 The Goblet Curve

n	237			
m	8	16	32	64
LA_0	1.03375e-1	6.11455e-2	2.95784e-2	8.24535e-3
LA_k	6.07749e-2	2.90067e-2	7.76577e-3	5.63440e-3
(k)	(12)	(17)	(20)	(5)
LA^*	5.87190e-2	2.06813e-2	5.12219e-3	1.75973e-3
LA_k/LA^*	1.035	1.403	1.516	3.202

Table 5.3 The Nonplanar Quartic Curve

n	404				
m	4	8	16	32	64
LA_0	2.01399e-0	3.63921e-1	9.66444e-2	2.67485e-2	8.39998e-3
LA_k	1.86220e-0	2.26435e-1	7.78874e-2	2.24510e-2	6.98709e-3
(k)	(4)	(32)	(16)	(10)	(8)
LA^*	1.85530e-0	2.26435e-1	7.60669e-2	2.05613e-2	5.69636e-3
LA_k/LA^*	1.004	1.000	1.024	1.092	1.227

Table 5.4 The Nonplanar Sextic Curve

n	234				
m	4	8	16	32	64
LA_0	7.24214e-1	1.72242e-1	5.32029e-2	1.64083e-2	1.60168e-2
LA_k	4.81844e-1	1.34728e-1	3.69400e-2	1.53000e-2	3.80315e-3
(k)	(15)	(15)	(13)	(2)	(11)
LA^*	4.81844e-1	1.34728e-1	3.65433e-2	1.05332e-2	3.16273e-3
LA_k/LA^*	1.000	1.000	1.011	1.453	1.202

6. CONCLUSION

In this thesis, we have investigated the possibilities of implicitly defined algebraic surfaces as tools of CAGD. In particular, the work focused to the classes of algebraic surfaces having moderately low degrees. In Chapter 2, we presented an algorithmic characterization, called Hermite interpolation for algebraic surfaces, that finds a class or family of algebraic surfaces of a fixed degree which satisfy given geometric specifications. This algorithm, computing algebraic surfaces meeting with tangent plane continuity, provided a primary tool with the remaining work.

In Chapter 3, the well known least squares approximation method was applied to help select an instance surface from a family computed by our Hermite interpolation algorithm. With Hermite interpolation and least squares approximation, combined with a proper normalization, the geometric problem of finding algebraic surfaces was translated into a constrained minimization problem which can be solved efficiently. We also discussed how geometric information, related to coefficients of a polynomial in barycentric coordinates, can be utilized in interactively controlling the shape of algebraic surfaces in a computed family with geometric intuition.

The class of quintic algebraic surfaces was explored to smooth a convex polyhedron with triangular faces in Chapter 4. In our scheme, the edges of a given polyhedron were replaced by conic curves with associated normal vectors such that the curves and normal vectors agree with the vertex and normal conditions of the polyhedron. Then, the three conics for each face were fleshed by a quintic surface. The degrees of freedom in choosing conics were used to control the shape of the wire frame, and hence the Hermite interpolating algebraic surface patches. Then, we considered the open problem of smoothing an arbitrary polyhedron with quintic algebraic surface

patches. One possibility was to subdivide a face of a polyhedron iteratively until some condition on normal vectors is met.

In Chapter 5, we discussed the problem of piecewise linear approximation of a densely sampled space curve. Two algorithms were presented. One finds an optimal linear approximation at a high expense. The other computes a heuristic linear approximation, based on the fundamental notions of curve length and spherical image of a space curve. The heuristic algorithm turned out to find a good linear approximation quickly. The heuristic algorithm was used in polygonizing the triangular algebraic surface patches computed in Chapter 4.

Our algebraic surface fitting algorithms have been implemented, and included in the geometric modeling system GANITH. GANITH [12] is an X Window System based algebraic geometry toolkit that manipulates algebraic equations. It has been developed to solve systems of algebraic equations and visualize their multiple solutions. Applications of this toolkit include curve and surface display, curve-curve intersections, surface-surface intersections, curve-surface intersections, and etc. For surface fitting, GANITH takes as input the degree of a fitting surface and a collection of data points and space curves with or without associated normal directions. Then, an algebraic surface that fits the given data is computed through the previously described computational model, and when such a surface exists, it is interactively rendered in a display buffer. (See Figure 6.1.) For convex polyhedron smoothing, GANITH takes as input a polyhedron and a ρ value, and computes quadric wires, and then algebraic surfaces that smooth the polyhedron. Then, polygonized triangular patches are rendered interactively in a display buffer. The capabilities of graphics hardwares such as Hewlett-Packard 9000/370 SRX and SGI IRIS workstations can be used through our XS library which interfaces between the X programs and the HP Starbase and SGI GL graphics libraries.

We have seen that implicitly represented algebraic surfaces are very natural for interpolation and approximation. However, there are some difficult problems to be solved before algebraic surfaces can be used effectively for geometric modeling. First, it is not always easy to make sure that input points and curves lie on one real component of an algebraic surface. One heuristic, which can be used, is to include auxiliary points and curves to bridge the gap between separate surface components. Another approach is proposed in [46] where a distance fit is used to guarantee a single sheet of a surface inside a tetrahedron for densely scattered point data. However, the question remains open for producing conditions on the coefficients of the fitting surfaces, which would ensure that all given points and curves lie on the same continuous real surface component.

Another unfavorable fact of algebraic surfaces is singularity. Singularities of algebraic surfaces occur in the forms of sharp points, sharp edges, or self-intersections, and tangent planes to surfaces are not defined at singular points. While, in general, singularities must be avoided in surface modeling, they can be useful in some situation. For instance, the quartic algebraic surface (the dark patch) in Figure 2.7 is singular at the four points where the four cylinders meet pairwise while the surface is regular inside the patch. In fact, singularities are necessary in this case because no regular patch can smoothly join the cylinders. In Section 4.2, singularities were also useful in Hermite interpolating three artificially constructed conic curves whose associated normals do not satisfy the compatibility condition at the three intersection points. However, it is highly desirable to be able to control singularities locally on a specific portion of an algebraic surface, although we do not know how yet.

In this work, we have proposed a direction of exploration of moderately low degree algebraic surfaces as tools in geometric modeling systems, but this is the only first step toward construction of a practical algebraic CAGD system with a complex way of creating and manipulating geometric models of physical objects with various geometries. Much work should now be undertaken.

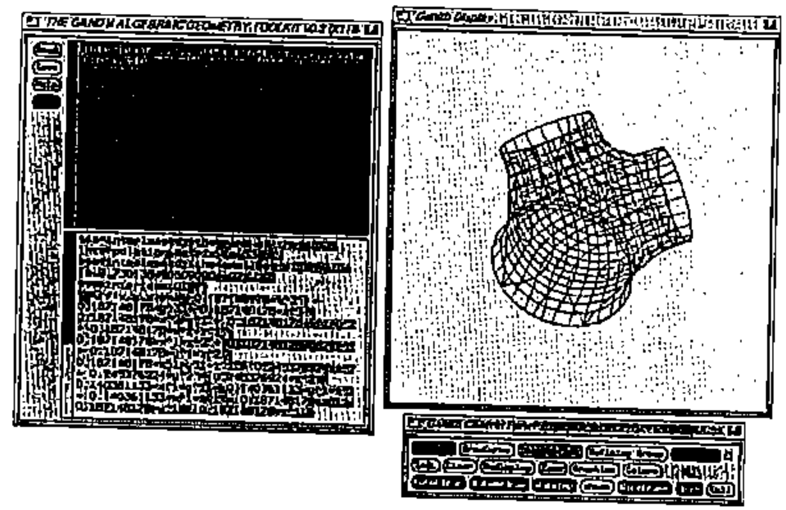

Figure 6.1 The Algebraic Geometry Toolkit GANITH

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] S.S. Abhyankar. *Algebraic Space Curves.* Univ. of Montreal, 1970. Montreal Lecture Notes.

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

[3] A. Albano. Representation of digitized contours in terms of conic arcs and straight line segments. *Computer Graphics and Image Processing,* 3:23–33, 1974.

[4] E.L. Allgower and S. Gnutzmann. Polygonal meshes for implicitly defined surfaces. Manuscript, Nov. 1989.

[5] D. Arnon, G. Collins, and S. McCallum. Cylindrical algebraic decomposition 1: The basic algorithm. *Siam J. Computing,* 13(4):865–889, 1984.

[6] C. Bajaj. Geometric modeling with algebraic surfaces. In D. Handscomb, editor, *The Mathematics of Surfaces III,* pages 3–48. Oxford Univ. Press, 1988.

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

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

[9] C. Bajaj and I. Ihm. Algebraic surface design with Hermite interpolation. Technical Report CSD-TR-939, Computer Sci., Purdue Univ., Jan. 1990. Revised in Dec. 1990. (To appear in ACM Transactions on Graphics).

[10] C. Bajaj and I. Ihm. C^1 smoothing of polyhedra with implicit surface patches. Manuscript, 1991.

[11] C. Bajaj, I. Ihm. and J. Warren. Higher order interpolation and least squares approximation using implicit algebraic surfaces. Technical Report 91-035, Computer Sci., Purdue Univ., April 1991.

[12] C. Bajaj and A. Royappa. The GANITH algebraic geometry toolkit. In *DISCO '90,* Capri, Italy, 1990. (Extended Abstract).

[13] R. Barnhill. Surfaces in computer aided geometric design : A survey with new results. *Computer Aided Geometric Design,* 2:1–17, 1985.

[14] W. Böhm, G. Farin, and J. Kahmann. A survey of curve and surface methods in CAGD. *Computer Aided Geometric Design,* 1:1–60, 1984.

[15] R. Biggerstaff. Three variations in dental arch form estimated by a quadratic equation. *Journal of Dental Research,* 51:1509, 1972.

[16] J. Bloomenthal. Polygonization of implicit surfaces. *Computer Aided Geometric Design,* 5:341–355, 1988.

[17] F.L. Bookstein. Fitting conic sections to scattered data. *Computer Graphics and Image Processing,* 9:56–71, 1979.

[18] M. P. Do Carmo. *Differential Geometry of Curves and Surfaces.* Englewood Cliffs, New Jersey, 1976.

[19] S.D. Conte and C. deBoor. *Elementary Numerical Analysis: An Algorithmic Approach.* McGraw-Hill, New York, third edition, 1980.

[20] W. Dahmen. Smooth piecewise quadratic surfaces. In T. Lyche and L. Schumaker, editors, *Mathematical Methods in Computer Aided Geometric Design,* pages 181–193. Academic Press, Boston, 1989.

[21] C. deBoor. *A Practical Guide to Splines.* Springer-Verlag, New York, 1978.

[22] J.G. Dunham. Optimum uniform piecewise linear approximation of planar curves. *IEEE Transanctions on Pattern Analysis and Machine Intelligence,* 8:67–75, Jan. 1986.

[23] S. Mann et al. A survey of parametric scattered data fitting. Manuscript, 1991.

[24] C. Fahn, J. Wang, and J. Lee. An adaptive reduction procedure for the piecewise linear approximation of digitized curves. *IEEE Transanctions on Pattern Analysis and Machine Intelligence,* 11(9):967–973, Sep. 1989.

[25] G. Farin. Triangular Bernstein-Bézier patches. *Computer Aided Geometric Design,* 3:83–127, 1986.

[26] T. Garrity and J. Warren. Geometric continuity. *Computer Aided Geometric Design,* 8:51–65, 1991.

[27] R. Gnanadesikan. *Methods for Statistical Data Analysis of Multivariate Observations.* John Wiley & Sons, 1977.

[28] R. N. Goldman. The role of surfaces in solid modeling. In G. Farin, editor, *Geometric Modeling : Algorithms and New Trends,* pages 69–90. SIAM, Philadelphia, 1987.

[29] G. Golub. Some modified matrix eigenvalue problems. *SIAM Review*, 15(2):318–334, 1973.

[30] G. Golub and R. Underwood. Stationary values of the ratio of quadratic forms subject to linear constraints. *Z. Agnew. Math. Phys.*, 21:318–326, 1970.

[31] G.E. Golub and C.F. Van Loan. *Matrix Computation*. The Johns Hopkins Univ. Press, Baltimore, MD, 1983.

[32] B. Guo. Surface generation using implicit cubics. In N.M. Patrikalakis, editor, *Scientific Visualization of Physical Phenomena*, pages 485–503. Springer-Verlag, Tokyo, 1991.

[33] M. Hall and J. Warren. Adaptive polygonalization of implicitly defined surfaces. *IEEE Computer Graphics and Applications*, pages 33–42, November 1990.

[34] P. Hanrahan. Ray tracing algebraic surfaces. *Computer Graphics*, 17(3):83–90, 1983.

[35] C. Hoffmann and J. Hopcroft. Quadratic blending surfaces. *Computer Aided Design*, 18(6):301–306, 1986.

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

[37] I. Ihm and B. Naylor. Piecewise linear approximations of digitized space curves with applications. In N.M. Patrikalakis, editor, *Scientific Visualization of Physical Phenomena*, pages 545–569. Springer-Verlag, Tokyo, 1991.

[38] H. Imai and M. Iri. Computational geometric methods for polygonal approximations of a cur ve. *Computer Vision, Graphics and Image Processing*, 36:31–41, 1986.

[39] N. Kehtarnavaz and R.J.P. deFigueiredo. A 3-D contour segmentation scheme based on curvature and torsion. *IEEE Transanctions on Pattern Analysis and Machine Intelligence*, 10(5):707–713, Sep. 1988.

[40] M.T. Kosters. High-order implicit blending surfaces of low degree. Technical Report No. CS 9005, Dept. Math. Computer Sci., Groningen Univ., April 1990.

[41] E. Kreyszig. *Differential Geometry*. University of Toronto Press, 1959.

[42] E.T.Y. Lee. The rational Bézier representation for conics. In G. Farin, editor, *Geometric Modeling : Algorithms and New Trends*, pages 3–19. SIAM, Philadelphia, 1987.

[43] R. Lynch. Private communication.

[44] A. A. Melkman and J. O'Rourke. On polygonal chain approximation. In G. T. Toussaint, editor, *Computational Morphology*, pages 87–95. North-Holland, 1988.

[45] A. Middleditch and K. Sears. Blend surfaces for set theoretic volume modeling system. *Computer Graphics*, 19(3):161–170, 1985.

[46] D. Moore and J. Warren. Approximation of dense scattered data using algebraic surfaces. Manuscript, 1991.

[47] B. Naylor. SCULPT : An interactive solid modeling tool. In *Proc. of Graphics Interface*, Halifax, Nova Scotia, May 1990. Graphics Interface '90.

[48] B. Naylor, J. Amanatides, and W. Thibault. Merging bsp trees yields polyhedral set operations. *Computer Graphics*, 24(4), Aug. 1990.

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

[50] J.C. Owen and A.P. Rockwood. Blending surfaces in solid modeling. In G. Farin, editor, *Geometric Modeling : Algorithms and New Trends*, pages 367–383. SIAM, Philadelphia, 1987.

[51] K. Paton. Conic sections in chromosone analysis. *Pattern Recognition*, 2:39–51, 1970.

[52] T. Pavlidis. *Algorithms for Graphics and Image Processing*, pages 281–297. Computer Sciences, New York, 1982.

[53] T. Pavlidis. Curve fitting with conic splines. *ACM Transactions on Graphics*, 2(1):1–31, 1983.

[54] T. Pavlidis and S.L. Horowitz. Segmentation of plane curves. *IEEE Transactions on Computers*, c-23(8):860–870, Aug. 1974.

[55] Jörg Peters. Parametrizing singularly to enclose data points by a smooth parametric surface. To be presented in Graphics Interface '91, Calgary, Canada, June, 1991.

[56] Jörg Peters. Smooth interpolation of a mesh of curves. *Constructive Approximation*, 7:221–246, 1991.

[57] L. Piegl. Interactive data interpolation by rational Bézier curves. *IEEE Computer Graphics and Applications*, pages 45–58, July 1987.

[58] V. Pratt. Techniques for conic splines. *Computer Graphics*, 19(3):151–159, 1985.

[59] V. Pratt. Direct least squares fitting of algebraic surfaces. *Computer Graphics*, 21(3):145–152, 1987.

[60] U. Ramer. An iterative procedure for the polygonal approximation of plane curves. *Computer Graphics and Image Processing*, 1:244–256, 1972.

[61] K. Reumann and A.P.M. Witkam. Optimizing curve segmentation in computer graphics. In A. Gunther, B. Levrat, and H. Lipps, editors, *International Computer Symposium*, pages 467–472. American Elsevier, New York, 1974.

[62] J. Roberge. A data reduction algorithm for planar curves. *Computer Vision, Graphics and Image Processing*, 29:168–195, 1985.

[63] A.D. De Rose. *Geometric Continuity : A Parametrization Independent Measure of Continuity for Computer Aided Geometric Design*. PhD thesis, Computer Science, Univ. of California, Berkeley, 1985.

[64] G. Salmon. *A Treatise on Conic Sections*. Longmans, Green, and Co., London, third edition, 1896.

[65] P.D. Sampson. Fitting conic sections to very scattered data : An iterative refinement of the bookstein algorithm. *Computer Graphics and Image Processing*, 18:97–108, 1982.

[66] T.W. Sederberg. Piecewise algebraic surface patches. *Computer Aided Geometric Design*, 2(1-3):53–59, 1985.

[67] T.W. Sederberg. Techniques for cubic algebraic surfaces. *IEEE Computer Graphics and Applications*, 10(5):12–21, Sept. 1990.

[68] T.W. Sederberg. Techniques for cubic algebraic surfaces. *IEEE Computer Graphics and Applications*, 10(4):14–25, July 1990.

[69] T.W. Sederberg and A.K. Zundel. Scan line display of algebraic surfaces. *Computer Graphics*, 23(3):147–156, 1989.

[70] J. Sklansky and V. Gonzalez. Fast polygonal approximation of digitized curves. *Pattern Recognition*, 12:327–331, 1980.

[71] G. Strang. *Linear Algebra and its Applications*. Harcourt Brace Jovanovich, San Diego, third edition, 1988.

[72] G. T. Toussaint. On the complexity of approximating polygonal curves in the plane. In *Proceedings of the IASTED International Symposium on Robotics and Automation*, pages 59–62, Lugano, Switzerland, 1985.

[73] R. Walker. *Algebraic Curves*. Springer Verlag, New York, 1978.

[74] K. Wall and P.E. Danielsson. A fast sequential method for polygonal approximation of digitized curves. *Computer Vision, Graphics and Image Processing*, 28:220–227, 1984.

[75] J. Warren. *On Algebraic Surfaces Meeting With Geometric Continuity*. PhD thesis, Cornell University, 1986.

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

[77] C.M. Williams. An efficient algorithm for the piecewise linear approximation of planar curves. *Computer Graphics and Image Processing*, 8:286–293, 1978.

[78] O. C. Zienkiewicz. *The Finite Element Method*. McGraw-Hill, New York, 1978.

VITA

Born in Seoul, Korea, on May 3, 1962, Insung Ihm entered Seoul National University, Korea, in 1981. In 1985, he graduated *Magna Cum Laude* with a Bachelor of Science Degree in both Computer Science and Statistics. He studied Computer Science at Rutgers University in New Brunswick, New Jersey, from 1985 to 1987, and received his Master's Degree in Computer Science. In the fall of 1987, he became a Ph.D. student in the Department of Computer Sciences at Purdue University in West Lafayette, Indiana, to pursue his doctoral study. During the summer of 1990, he worked for AT&T Bell Laboratories in Murray Hill, New Jersey, as a summer intern. He is a recipient of the Bong Shin Fellowship from Seoul National University (1983-1985), and the David Ross Fellowship from Purdue University (1988-1989). His research interests include Computer Graphics and Computer Aided Geometric Design.