1990

Unifying Parametric and Implicit Surface Representations for Computer Graphics: Parametric Surface Display and Algebraic Surface Fitting

Changrajit L. Bajaj

Report Number: 90-995
UNIFYING PARAMETRIC AND IMPLICIT SURFACE REPRESENTATIONS FOR COMPUTER GRAPHICS: PARAMETRIC SURFACE DISPLAY AND ALGEBRAIC SURFACE FITTING

Chaudrajit L. Bajaj
CSD TR-995
July 1990
Unifying Parametric and Implicit Surface Representations for Computer Graphics: Parametric Surface Display and Algebraic Surface Fitting

Chanderjit Bajaj
Department of Computer Science
Purdue University
West Lafayette, IN 47907

Abstract

These are brief survey notes on recent progress in topics dealing with rational surface display and surface fitting with piecewise algebraic surface patches. It is hoped that the reader shall delve deeper into specific technical results, by tracking the numerous citations to references provided. Several pictures (unfortunately grey scale images) are included to illustrate the results of certain algorithms. These notes are part of a course to be taught at this year's Siggraph 1990. Slides of the course are also included in the appendix.

*Supported in part by NSF grant DMS 85-16286 and ONR contract N00014-88-K-0402
Introduction

Rationality of the algebraic curve or surface is a restriction where advantages are obtained from having both the implicit and rational parametric representations. Numerous facts on rational algebraic curves and surfaces can be gleaned from books and papers on analytic geometry, algebra and algebraic geometry, see for example [61, 70, 69, 78]. For example, a real algebraic surface, in three dimensional space, is represented implicitly by the single polynomial equation \( f(x, y, z) = 0 \) where coefficients of \( f \) are over the real numbers \( \mathbb{R} \), and parameterically by the three equations \( (x = G_1(s, t), y = G_2(s, t), z = G_3(s, t)) \) where the \( G_i \), \( i = 1, \ldots, 3 \) are rational functions, i.e. ratio of polynomials and where coefficients of the \( G_i \) are again over the real numbers \( \mathbb{R} \). Simpler algorithms for geometric modeling and computer graphics are possible when both implicit and parametric representations are available, see for e.g. [11, 36]. For example for shaded displays, the parametric form yields a simple way of polygonalizing the surface, while the implicit form yields an efficient calculation of the exact normals of the surface at each of the vertex endpoints of the constructed polygonal mesh. We utilize both these advantages, and others, to derive

1. efficient methods for displaying rational quadric (degree two) and cubic (degree three) algebraic surfaces (and hypersurfaces in higher dimensions) by constructing adaptive, curvature dependent polygonalizations.

2. algorithms for constructing smooth meshes of algebraic surface patches, by using interpolation and/or least-squares approximation through scattered points and curve data in space, as well as over given triangulations.

Why algebraic surfaces? Manipulating polynomials, as opposed to arbitrary analytic functions, is computationally more efficient. Furthermore algebraic surfaces provide enough generality to accurately model almost all complicated rigid objects. Also as we show here, algebraic curves and surfaces lend themselves very naturally to the problems of display and surface fitting.

Why implicit representations? Most prior approaches to interpolation and surface fitting, have focused on the parametric representation of surfaces [28, 63]. Contrary to major opinion and as we exhibit here, implicitly defined surfaces are also very appropriate. Additionally, while all algebraic surfaces can be represented implicitly, only a subset of them have the alternate parametric representation.
with \( x, y \) and \( z \) given explicitly as rational functions of two parameters. Working with implicit algebraic surfaces of a fixed degree, thus provides a larger number of surfaces to design with. Furthermore, implicit algebraic curves and surfaces have compact storage representations and form a class which is closed under most common operations (boolean set operations, offsets etc.) required by a geometric design system.

When are algebraic curves and surfaces rational? In the plane, all conics (degree two curves) and all singular cubics (degree three curves with a singularity) are rational. Similarly, quartics and higher order curves are rational if and only if they possess the maximum possible number of singularities. The deficiency of singularities in a curve from the maximum possible number is known as the genus of the curve. An algebraic curve is then rational if and only if its genus is zero [see for e.g. [68, 74]].

In three dimensional space, all degree two algebraic surfaces (quadrics or conicoids), are rational. All degree three surfaces (cubic surfaces or cubicoids), except the cylinders of nonsingular cubic curves and the cubic cone, have a rational parameterization, with the exceptions again only having a parameterization of the type which allows a single square root of rational functions. Most algebraic surfaces of degree four and higher are not rational, although parameterizable subclasses can be identified. In general, a necessary and sufficient condition for the rationality of an algebraic surface of arbitrary degree is given by Castelnuovo’s criterion: \( P_a = P_2 = 0 \), where \( P_a \) is the arithmetic genus and \( P_2 \) is the second plurigenus [78].

Various algorithms have been given for constructing the rational parametric equations of implicitly defined algebraic curves and surfaces, (i.e., hypersurfaces in 2D and 3D). See for instance [2, 3, 42, 51, 67]. The parameterization algorithms presented in [4] and [5] are applicable for irreducible rational plane algebraic curves of arbitrary degree, and irreducible rational space curves arising from the intersection of two algebraic surfaces of arbitrary degree. In [11] we present parametrization algorithms for fixed degree (two and three) hypersurfaces, however in arbitrary dimensional space, \( n \geq 3 \).

2 Parametric Surface Display

Several approaches are known for rendering parametric surfaces. [see for e.g. [25, 41]]. The algorithm in [25] is based on convex hull properties of Bezier surfaces and uses subdivision to polygonalize the surfaces. The polygons are of course then scan converted to produce the displayed image. On the other hand [41] uses scan lining for direct scan conversions of the curved surface. The extension of these basic
techniques for the wire-frame display of hypercubes and simplicies is given in [50], while [22] provide a hidden-line algorithm for such hyperobjects. In [11] algorithms are given for polygonalizing rational quadric and cubic hypersurfaces in \( n \geq 3 \) dimensional space by an adaptive, generalized curvature dependent scheme, and displaying their projections in 3D.

Here we sketch the method for a curvature (and torsion dependent) stepping of the parameters of a parametric surface in 3 space. Consider the rational surface \( S \) defined by the parametric equations

\[
x = X(s, t), \quad y = Y(s, t), \quad z = Z(s, t)
\]

where \( X, Y, Z \) are rational functions. A simple way of displaying \( S \) is to let \( s \) vary from \( s_i \) to \( s_f \) by a constant step of \( \Delta_s \) and let \( t \) vary from \( t_i \) to \( t_f \) by a constant step of \( \Delta_t \). This creates a rectangular grid of \((s, t)\) points. The surface can be directly polygonalized by evaluating it at each grid point and connecting the grid points together to form polygons. A better way of creating the grid is to step adaptively. Let \( X(t) = [X(s, t), Y(s, t), Z(s, t)] \). Then

\[
\Delta_t = \frac{1}{(a\kappa_t + b\tau_t) \| X' \|^2} \Delta\phi
\]

\[
\Delta_s = \frac{1}{(a\kappa_s + b\tau_s) \| X' \|^2} \Delta\phi
\]

Here \( \kappa \) and \( \tau \) are the curvature and torsion of the surface, respectively, and are defined by

\[
\kappa = \frac{\sqrt{(X' \cdot X')(X'' \cdot X'') - (X' \cdot X'')^2}}{(X' \cdot X')^{3/2}}
\]

\[
\tau = \frac{[X', X'', X''']^T}{(X' \cdot X')(X'' \cdot X'') - (X' \cdot X'')^2}
\]

To get \( \Delta_t \), all derivatives are performed with respect to \( t \), and for \( \Delta_s \), with respect to \( s \). Given some pair \((s_0, t_0)\), to step along \( t \), we compute \( \Delta_t \) by evaluating the formula above at \( s = s_0, t = t_0 \), and likewise to step along \( s \). One can use constant-stepping in one variable and adaptive stepping in the other, or adaptive stepping in both. The latter approach is more expensive, but we find it produces smoother-looking surfaces. We used the following stepping process. The algorithm below fills the given grid with \( n^2 \) \((s, t)\) pairs. Stepping along \( s \) and \( t \) starts at \( s_0 \) and \( t_0 \) respectively.

```makegrid(grid, s_0, t_0, n)```
local s, t, i, k;

/* initialize row and column 1 */
grid(1,1) ← (s_0, t_0);
for i := 2 to n do {
    s ← grid(1, i-1).s;  t ← grid(1, i-1).t;
    grid(1, i) ← (s, t + Δ_s(s, t));
    s ← grid(i-1, 1).s;  t ← grid(i-1, 1).t
    grid(i, 1) ← (s + Δ_s(s, t), t)
}

/* initialize rows and columns, diagonally*/
for k := 2 to n do {
    /* row k */
    for i := k to n do {
        s ← grid(k, i-1).s;  t ← grid(k, i-1).t
        grid(k, i) ← (s, t + Δ_s(s, t))
    }
    /* col k */
    for i := k + 1 to n do {
        s ← grid(i-1, k).s;  t ← grid(i-1, k).t;
        grid(i, k) ← (s + Δ_s(s, t), t)
    }
}

Example figures are given at the end, Figs. 1-4. using GANITH: a package for visualizing and displaying algebraic equations [16].

3 Algebraic Surface Fitting

There has been extensive prior work in surface fitting. Much of it has concentrated on polynomial parametric (and occasionally rational parametric) surface fitting through scattered point data in 3D,
see for example the surveys by Alfeld, Bohm et. al. and Pratt [6, 19, 58]. Exact and approximate fitting
of curves (primarily conics) has been considered by several authors, see for eg [20, 29, 53, 62]. Paper
[63] presents techniques for constructing a $C^1$ continuous surface of rectangular Bézier (parametric)
surface patches, interpolating a net of cubic Bézier curves. Other approaches to parametric surface
fitting and transfinite interpolation are also mentioned in that paper, as well as in [55, 56]. Pratt [58]
and some others [40] consider the least-squares fitting problem, however only for scattered point data.
The results of [13] generalizes the results of [58] in two ways. One, it considers exact fits of algebraic
surfaces through given space curves as well as data points. Second, it also considers similar surface
fits when derivative information ("normals") are also provided at the given data points and along
the given data curves. Meshing of given algebraic surface patches using control techniques of joining
Bézier polyhedrons is shown in [66]. Some of the results in [66] are extended in [13] for purposes
of interactive shape control of a family of solution surfaces. Paper [14] considers higher order surface
fitting as well as least-squares approximations. Surface blending consisting of "ranging" and "filleting"
surfaces (smoothing the intersection of two primary surfaces), a special case of surface fitting, has been
considered for polyhedral models in [33] and for algebraic surface models in [12, 13, 14, 76, 55]. The
generalized techniques for $C^1$ continuous surface meshes, presented in [14] also provide algorithms to
generate such blending and joining surfaces.

The generation of a mesh of smooth surface patches or splines that interpolate or approximate
\textit{triangulated space data} is also one of the primary topics of CAGD. Bohm et. al [19] and Chui [23]
summarize much of the history of previous work. These splines are traditionally defined over a given
planar triangulation with a polynomial function or parametric surface for each triangular face [7, 8, 17,
18, 37, 38, 49, 77].

Interpolatory spline problems can be classified by the following factors:

- What kind of surface patches do they generate: parametric, functional, or implicit?
- What kind of triangulation of data do they assume: functional values over 2D triangulation or
  arbitrarily spaced 3D triangulation?
- What kinds of information do they need as input data: $C^0$ data, $C^1$ data, $C^2$ data, and so on?
- How smoothly do the patches meet along the boundary curves?
- Are they local, that is, each patch is constructed only from nearby data, or global?
• Do they split one macro triangle into many micro triangles or not?

• Do they handle general data or just special data?

• How efficient are they?

• And many more ...

Dahmen [26] presents the construction of tangent plane continuous piecewise triangular quadratic surfaces. In his construction, in order to model complex shapes keeping the degree of surfaces low, a macro patch is split into 6 micro quadratic patches. Bajaj and Im [15] discuss some initial results in which a mesh of quintic implicit algebraic surface patches is built, however using only 1 surface patch per face. The input to the algorithm is a polyhedra with triangular facets. From the input, a quadratic wireframe is built which smoothly interpolates the vertices of the polyhedra with unique prespecified normals. For now, the normal at a vertex is chosen as the average of the normals of the incident facets at a vertex. Following the curvilinear wireframe construction, each face is interpolated with a quintic algebraic surface.

Example figures are given at the end, Figs. 5-12.

References


Fig. 1: Nodal Cubic Cylinder – $3 \leq s$, $t \leq 3$, 0.1 equal parameter stepping, 3600 polygons.
Fig. 2: Nodal Cubic Cylinder – $3 \leq s$, $T \leq 3$, adaptive parameter stepping, 640 polygons.
Fig 3: Example parametric display of a hyperboloid in GANITH, adaptive parameter stepping.
DrawSurface([S*T,S^2*T,-3.0,3.0,-3.0,3.0,0.2,1])

Fig. 4: Example parametric display of the CARTAN umbrella in GANITIl, adaptive parameter stepping.
Fig. 5: Corner blending with a quartic surface.

Fig. 6: $G^1$-join of three cylinders with a quartic surface.
Fig. 7: Two different least-squares approximation surfaces with $G^1$-join to the cylinders. The point data for approximation are given in Fig. 9.
(a) Six Points in $S_1$

(b) Six Points in $S_2$

Fig. 8: Data points which are least-squares approximated by surfaces $S_1$ and $S_2$ of Fig. 7.
Fig. 9: Input polyhedral and constructed wireframe mesh of conics.

Fig. 10: Polyhedra is smoothed with $G^1$-continuous quintic algebraic surface patches.
Fig. 11: Shaded display, smoothed polyhedra of Fig. 10.
Fig. 12: Quintic algebraic surfaces meshing with $G^1$-continuity, 1 quintic patch per facet.