# ROBUST DISPLAY OF ARBITRARY RATIONAL PARAMETRIC SURFACES

**Chandrajit L. Bajaj**
**Andrew V. Royappa**

**CSD-TR-92-054**
**August 1992**

# ROBUST DISPLAY OF ARBITRARY RATIONAL PARAMETRIC SURFACES

Chandrajit L. Bajaj and Andrew V. Royappa
Computer Science Department
Purdue University
West Lafayette, IN 47907
Technical Report CSD-TR-92-054
CAPO Report CER-92-23
August 1992

# Robust display of arbitrary rational parametric surfaces

Chandrajit L. Bajaj
Andrew V. Royappa

Department of Computer Sciences
Purdue University
West Lafayette, IN 47907

## ABSTRACT

A well-known strength of the parametric representation of a curve or surface is the ease by which a piecewise-linear approximation is generated. This is often true in geometric design, where only smooth segments or patches are considered over a well-chosen polynomial basis. Visualizing an *arbitrary* and possibly discontinuous parametric surface is useful but non-trivial for algebraic surfaces defined by rational parameter functions. Such surfaces have *pole curves* in their domain, where the denominators of the parameter functions vanish, domain *base points* that correspond to entire curves on the surface, and other features that cause display algorithms to fail. These are ubiquitous problems occurring even among the natural quadrics. Sophisticated but unsuspecting display techniques (e.g. those implemented in *MapleV*, *Mathematica*) produce completely unintelligible results. We provide a general solution and discuss our implementation. First, projective domain transformations are applied that map the entire surface from a finite domain region. Then, domain pole curves are identified and numerically approximated. Using Delaunay triangulation, a special decomposition of the domain is constructed that avoids discontinuities. This is then mapped onto the surface and clipped against a 3D box. The implementation can display very complicated surfaces, as we shall illustrate. Our techniques generalize in a straightforward way to rational varieties of any dimension.

## 1 INTRODUCTION

The main result of this paper is a practical algorithm that accurately displays the entire real part of an arbitrary rational parametric or surface. Since points on a real curve or surface given in parametric form can be generated easily by sampling across the real parametric domain, the display of continuous surfaces is well-understood [3, 22, 16, 10]. More recent methods address in detail the problem of generating a polygonal mesh on the surface that is sensitive to variations in curve/surface curvature: view-dependent methods [26] as well as view-independent[15, 18, 1], to name a few. All these methods assume continuity of the parametric functions over the domain of interest, since the emphasis is on representing smooth shapes.

We are interested in a rational parametric surface as a mapping from $R^2$ to $R^3$, and we wish to visualize the mapping, even in the presence of discontinuities or other misbehavior. That is, given an arbitrary rational parametric surface, and a convex portion of the space it lies in, produce an accurate picture of all parts of the surface that lie inside that portion. More precisely, given rational parametric functions $(x(s,t), y(s,t), z(s,t))$ defining a real surface, and real numbers $x_{\min} \leq x_{\max}$, $y_{\min} \leq y_{\max}$, $z_{\min} \leq z_{\max}$, compute the set $S$ of all points $(x, y, z)$ on the real surface that satisfy $(x, y, z) \in [x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}] \times [z_{\min}, z_{\max}]$.

The problem can be extended to include rational varieties, but we shall not discuss this here. The general flavor of the methods discussed will still apply, although implementing higher-dimensional methods would require more tools.

In this formulation, the problem is of more interest to mathematicians and free-form geometric designers, rather than to CAD designers working with rational Bezier and B-spline patches with non-negative weights [7]. However,

researchers are increasingly considering generalizations to rational patches (with negative weights) which include base points and poles [5, 7, 27]. Indeed our primary motivation to investigate this problem grew out of unsuccessful attempts (using our own implementation of standard techniques, as well commercial symbolic algebra and graphing programs e.g. *MapleV, Mathematica*), to visualize the output of cubic surface parameterization algorithms. The inadequacy of conventional graphing programs to plot even simple rational parametric surfaces made it imperative that new methods be developed for surfaces.

Since the point sets $C$ and $S$ are portions of continuous objects, in practice we will always compute only a piecewise-linear approximation of these sets. The sets can be abstractly represented in terms of a union of closed regions in the parametric domain, together with the parametric equations. One can approach the problem by either computing the sets directly in the range space, or approximating the closed regions in the parameter (domain) space and then applying the parameter functions to map these onto the curve or surface. There are tradeoffs between the two approaches but in this work we focus on the latter.

The common conception of displaying parametric surfaces is that it is trivial: vary the parameters over some part of the domain, generate points on the surface, and link the points to yield a piecewise-linear approximation of the parametric. This is a misconception because the process works only for "well-behaved" curves and surfaces. Arbitrary rational surfaces can possess unpleasant characteristics that cause naive methods to fail, as we discuss below.

### 1.1 Poles in the parameter domain

The *poles* of a rational function are the zeroes of the denominator polynoial. Since the denominator is a bivariate polynomial for rational parametric surfaces, the poles form a one-dimensional set, which we call a *pole curve*. Only real poles are considered in this paper. When poles are present, the curve or surface can be discontinuous, e.g. the hyperbola $(x(s), y(s)) = (s, \frac{1}{s})$ has a pole at $s = 0$. The real portion of the curve consists of two branches. A simple stepping algorithm might select a closed interval $[a, b]$ in the parameter domain, generate $n$ equally-spaced parameter values $s_i = a + i(\frac{b-a}{n-1}), i = 0, \ldots, n$, and then connect the points $(x(s_{i-1}), y(s_{i-1})), (x(s_i), y(s_i))$ with a straight-line segment. This method has the disadvantage that a line segment can be drawn between parameter values that lie on opposite sides of a pole, which may result in a line segment being drawn between two branches that are not connected in real affine space. The left picture in Figure 1 shows the output of the program *Mathematica* for plotting the hyperbola above over the domain interval $s \in [-\frac{1}{2}, \frac{1}{2}]$.

A hyperboloid of one sheet, with implicit equation $x^2 + y^2 - z^2 - 1 = 0$, is a surface whose real part is connected. However, if we work from the equivalent parametric representation

$$x(s,t) = \frac{t^2 - s^2 + 1}{s^2 + t^2 - 1}, \quad y(s,t) = \frac{2st}{s^2 + t^2 - 1}, \quad z(s,t) = \frac{2s}{s^2 + t^2 - 1} \tag{1}$$

then problems arise because of the pole curve described by $s^2 + t^2 - 1 = 0$ in the parameter domain. The right picture in Figure 1 shows the output produced by *MapleV* for this surface with $(s,t) \in [-2,2] \times [-2,2]$ (a domain region containing the pole curve).

### 1.2 Domain base points

The rational parameter functions describing curves and surfaces are generally assumed to be reduced to lowest common denominators, i.e., the numerator and denominator of each rational function are relatively prime. Thus for a curve, there is no parameter value that can cause both numerator and denominator of a rational parameter function to vanish. For surfaces, the situation is different. A surface is defined by three bivariate rational functions

$$x(s,t) = \frac{F_1(s,t)}{F_4(s,t)}, \quad y(s,t) = \frac{F_2(s,t)}{F_4(s,t)}, \quad z(s,t) = \frac{F_3(s,t)}{F_4(s,t)} \tag{2}$$

Even if $F_1, F_2, F_3, F_4$ are relatively prime polynomials, it is still possible that there are a finite number of points $(a, b)$ such that $F_1(a,b) = F_2(a,b) = F_3(a,b) = F_4(a,b) = 0$. Each such point is called a *base point* of the parametric

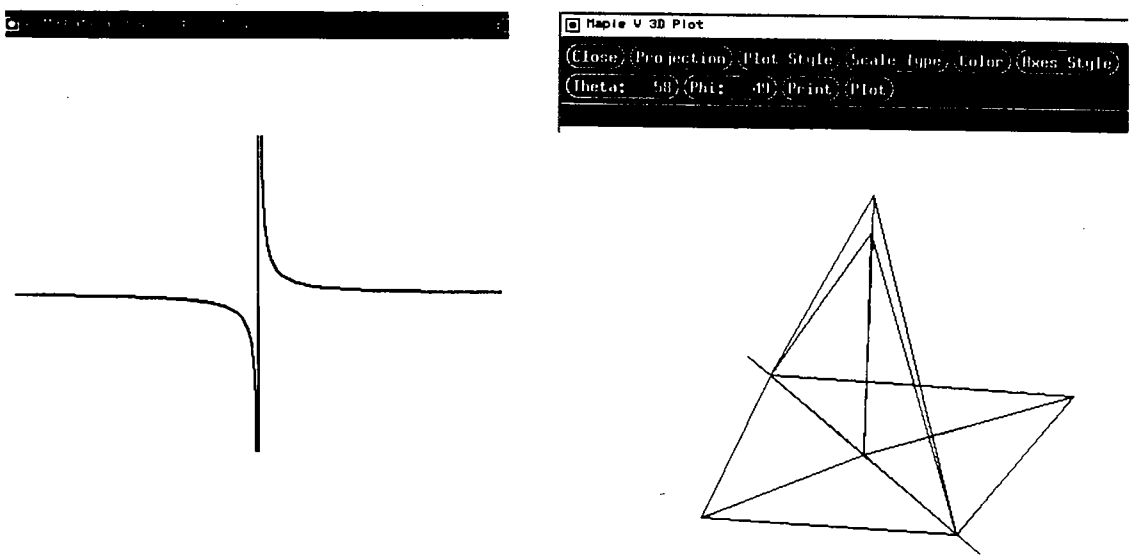


Figure 1: Domain poles: *Mathematica* (left), *MapleV* (right)

surface. There may also be base points at infinity in the parameter domain, and the base points can be complex as well as real-valued. Information about base points can be found in books on algebraic geometry such as [13, 23, 29].

Base points are problematic since there is in general no single surface point for the corresponding domain point. To each base point there may correspond a curve on the surface [23], and since there is no parameter value for surface points on such a curve, the entire curve will be missing from the real parametric surface. Such a curve is called a *seam curve*. See Figure 2 which shows a hyperboloid of one sheet (its equations are in Table 1, (a)). This parameterization has the two base points $(s,t) = (\pm 1, 0)$. The corresponding seam curves can be parameterized in parameters $u, v$ by $(x(u), y(u), z(u)) = (-1, u, u)$ and $(x(v), y(v), z(v)) = (-1, v, -v)$. Thus for a truly accurate display of a parametric surface, one should also display the seam curves, alongside the parametric surface. See Figure 5 of the Examples section, 3.4.

Base points can also cause numerical problems when evaluating the parameter functions. Whereas domain points in the neighborhood of a pole approach large values, domain points in the neighborhood of a base point approach the undefined value $\frac{0}{0}$, causing floating point division to return spurious values. Since every base point is also a point of the pole curve, the methods of dealing with pole curves must necessarily deal with base points.

### 1.3 Summary and organization

We have introduced the difficulties that arise when attempting to display arbitrary rational parametric curves and surfaces, and now go on to propose some means of handling them. A variety of methods are given, in shades ranging from theoretically attractive to feasible for practical implementation. Some of the methods are implemented, and we explore their relative strengths and weaknesses. We also use the advanced graphing facilities of the commercial programs *MapleV* [4] and *Mathematica* [17] as a means of comparison.

This paper is organized as follows. In section 2, some results are shown that handle the problem of infinite parameter values that arise when rational parametric surfaces are displayed. These simplify the display algorithms that we propose. In section 3, a number of solutions for surfaces are sketched, and our experiences with implementing some of our solutions are detailed. We conclude in section 4 by summarizing our research, and indicate future work.

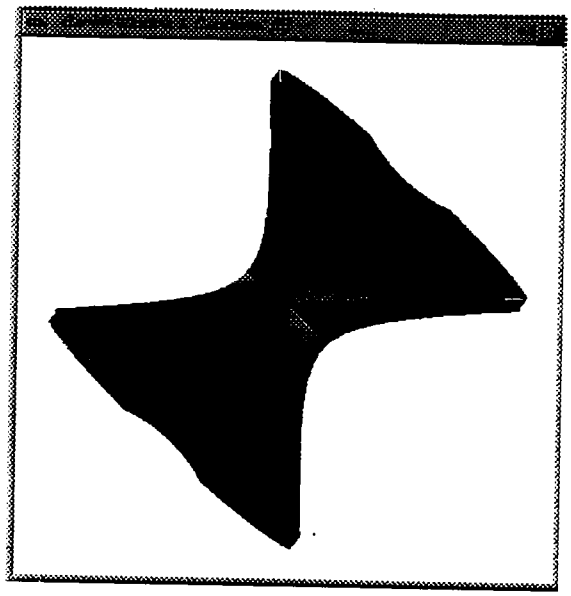

Figure 2: Hyperboloid of one sheet, with seam curve "gaps"

## 2 INFINITE PARAMETER VALUES

Consider the unit circle given by the rational parametric equations $(x(s), y(s)) = (\frac{s^2-1}{s^2+1}, \frac{2s}{s^2+1})$. The point $(1,0)$ corresponds to the parameter value $s = \infty$. Simply using large values of $s$ does not suffice to cover the gap, since as the parameter grows larger the curve points tend to bunch together tightly. In [5], this example is mentioned and a way to graph rational Bezier curves and surfaces is given, using a technique called homogeneous sampling. In [21], the problem is considered for rational curves and surfaces in the monomial basis, and a reason for parameter compression is given. Furthermore, the infinite parameter value problem is solved for rational varieties in the monomial basis using projective linear transformations of the domain. We reproduce without proof the key results, which will be applied in the following sections.

LEMMA. Consider a rational algebraic variety of dimension $n$ in $R^m$, $n < m$, given by parametric equations

$$V(\mathbf{s}) = \begin{pmatrix} x_1(s_1, \ldots, s_n) \\ \vdots \\ x_m(s_1, \ldots, s_n) \end{pmatrix}, \qquad s_i \in [-\infty, +\infty]$$

Let the $2^n$ octant cells in the parameter domain $R^n$ be labelled by the tuples $< \sigma_1, \ldots \sigma_n >$ with $\sigma_i \in \{-1, 1\}$. Then the projective reparameterizations $V(\mathbf{t}_{<\sigma_1,\ldots,\sigma_n>})$ given by

$$s_i = \sigma_i \frac{t_i}{1 - t_1 - t_2 - \ldots - t_n}, \quad i = 1, \ldots, n \tag{3}$$

together map the entire rational variety using only $t_i \geq 0$ such that $0 \leq t_1 + t_2 + \ldots + t_n \leq 1$.

COROLLARY 1. Rational curves $C(s) = (x_1(s), \ldots, x_m(s))^T$, $s \in [-\infty, +\infty]$ are covered by $C(\frac{t}{1-t}), C(\frac{-t}{1-t})$, using only $0 \leq t \leq 1$.

COROLLARY 2. Rational surfaces $S(s_1, s_2) = (x_1(s_1, s_2), \ldots, x_m(s_1, s_2))^T$, $s_1, s_2 \in [-\infty, +\infty]$ are covered by $S(\frac{t_1}{1-t_1-t_2}, \frac{t_2}{1-t_1-t_2}), S(\frac{-t_1}{1-t_1-t_2}, \frac{t_2}{1-t_1-t_2}), S(\frac{-t_1}{1-t_1-t_2}, \frac{-t_2}{1-t_1-t_2}), S(\frac{t_1}{1-t_1-t_2}, \frac{-t_2}{1-t_1-t_2})$, using only $t_i \geq 0 \wedge 0 \leq t_1 + t_2 \leq 1$.

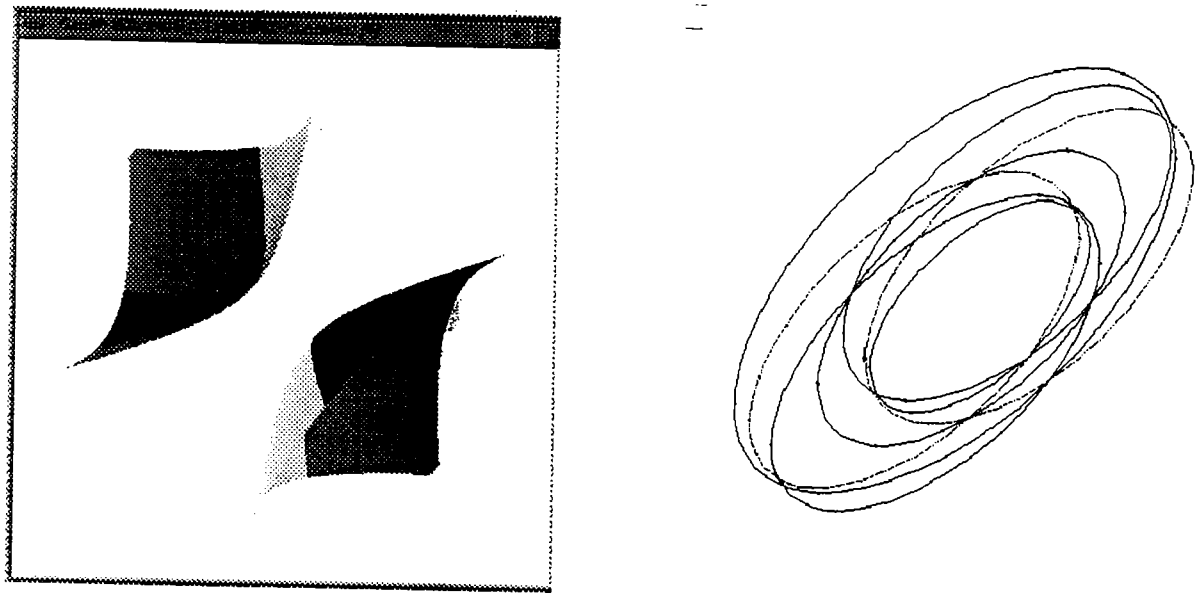

Figure 3: Trimmed hyperboloid of two sheets, and trim curves

The projective reparameterizations are shown here as fractional affine domain transformations for convenience. In practice, the parametric equations of the rational variety would be homogenized using an additional variable and the numerator and common denominator substituted separately as polynomials, thus avoiding rational function manipulation.

We note that the reparameterizations only allow us to reach all the real $s$ parameter values using only real parameter values $t$ in the specified regions. There may yet be real points on the variety corresponding to complex parameter values.

## 3 RATIONAL PARAMETRIC SURFACES

### 3.1 Surface trimming

The parametric surface is intersected with the faces of the bounding region, whose plane equations are $x = x_{\max}, y = y_{\max}$, etc. The intersection is performed by substituting the parameter functions into the equations and equating them to zero, leading to a set of bivariate equations. Each equation defines a *trim curve* in the parameter domain. The trim curves jointly define regions of the surface that pass inside and outside the bounding box. The trimming problem is addressed in [9, 19].

Consider a quadric surface, a hyperboloid of two sheets, whose parameterization is

$$x(s,t) = \frac{4s}{5t^2+6st+5s^2-1}, \quad y(s,t) = \frac{4t}{5t^2+6st+5s^2-1}, \quad z(s,t) = \frac{5t^2+6st-2t+5s^2-2s+1}{5t^2+6st+5s^2-1}$$

The trimmed surface is given in Figure 3, and the trim curves are shown alongside. It is clear that implementing a robust/stable trimming algorithm is difficult. The trim curves in question are all ellipses, but their interrelationship and consequently the trim region is quite complex. The surface in this case is only quadric, so the trim curves are conics, but higher order pole curves could have singularities and isolated points.

The situation is further complicated by the fact that the trim curves are implicitly defined; not only are implicit curves considerably harder to approximate than parametric ones, but trimming algorithms need trim curves to be given by parametric equations, or as a sorted list of points along the curve. Unfortunately, techniques for sorting points on algebraic curves are very expensive [14].

We have sketched this method because it does provide a complete solution to the problem of displaying surfaces with poles. If a robust trimming algorithm is ever developed, it should be applicable with little difficulty.

### 3.2 Subtracting pole curve offsets

We now turn to two approximate methods. The first method is to simply remove the offending pole curve. Given a closed region of the parametric domain, we calculate a strip containing the curve as its "spine," and then subtract this strip from the parameter domain. We shall call such strips *offset regions* henceforth, since they can be defined in terms of the curve *offsets*. Simply put, an offset region of width $2r$ around a curve can be defined as the union of circles of radius $r$ that are centered on a point of the curve. The outlines of the offset region are called *offset curves*. Offsets are discussed in [20, 8, 2].

One practical method for accurately displaying a parametric surface consists of the following steps.

METHOD I

1. Apply the projective linear domain transformations of section 2. This immediately restricts the parametric domain of interest to the unit simplex spanned by $(0,0),(0,1),(1,0)$. The following steps are performed four times, once for each reparameterization. Note that other than some slight overhead for reparameterizing, the amount of work is the same: it is proportional to the size of the bounding region and the resolution of the piecewise-linear approximation desired.
2. For the current reparameterization, calculate an offset region of some width $2\epsilon$ of the pole curve, within the unit simplex.
3. Subtract this region from the unit simplex.
4. Map the remaining part of the unit simplex onto the parametric surface.
5. Clip the mapped portion of the surface against the bounding region.

The offset radius $\epsilon$ is user-dependent, and must be chosen small enough so that all parts of the surface inside the bounding region are mapped, but large enough that effort is not wasted in calculating portions of the surface that are outside the bounding region. It is possible to calculate an upper bound on $\epsilon$ based on the trim curves and their relationship to the pole curve, but this investigation is currently incomplete.

The various steps are implemented as follows. First, an approximate offset region is calculated. This entails generating a piecewise-linear approximation to the pole curve using the technique of [12], and then constructing a quadrilateral that forms a strip around each line segment. The curve normal is calculated at each endpoint, a line is extended from each endpoint to a distance of $\epsilon$ in the normal and anti-normal directions. The four extended tips are linked together to form a quadrilateral (shown on the left in Figure 4). Care must be taken to avoid "bow-tie" quadrilaterals being formed this way, if the distance $\epsilon$ is too large with respect to the length of the curve segment. Singular points and isolated points cause problems because of multiple tangents.

The subtraction of the offset region from the domain (in this case a rectangular grid) is shown on the right in Figure 4; offset region polygons are subtracted from the grid polygons. A special data structure was created so that subtraction operations would only occur between polygons that were likely to overlap. Two polygon subtraction algorithms were implemented and experimented with: a Sutherland-Hodgeman algorithm clipper which we modified for subtraction [25], and the Weiler-Atherton algorithm [28]. The first algorithm resulted in considerable fragmentation of polygons (an intrinsic drawback of the algorithm), which led to loss of efficiency, and also minute gaps appeared on the surface due to numerical inaccuracies, when it was shaded. While the second polygon subtraction algorithm we implemented did not suffer the problems of the first, it was not robust, particularly in handling degeneracies.

After the triangles left over from the subtraction are mapped to the surface, a standard Sutherland-Hodgeman clipping algorithm was implemented to clip against the bounding region [25].

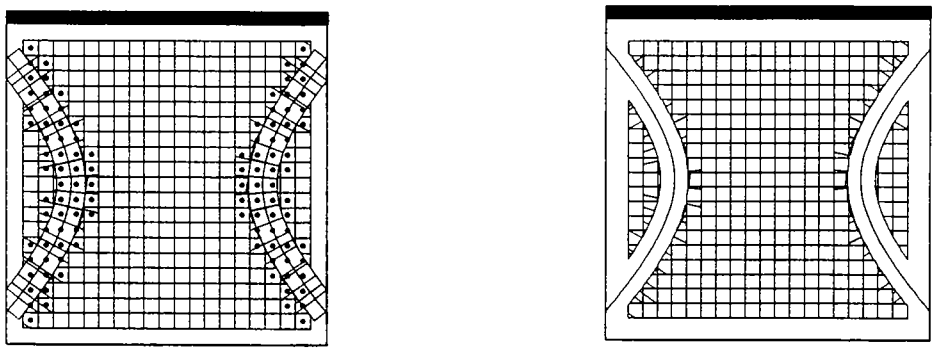

Figure 4: Approximation and subtraction of pole curve offset regions

While this method seems reasonable, our implementation left somewhat to be desired, in terms of accuracy and stability. Better implementation of the sub-algorithms does appear possible although time-consuming. Minor modifications could be tried, such as subtracting the offset polygons directly from the entire unit simplex, and then polygonizing the leftover regions. In general, it appears that implementing robust polygon clipping/subtraction is quite difficult.

Our implementation of method I, however, is a vast improvement over the standard programs; we were able to visualize a number of surfaces clearly. Due to artifacts such as cracks, proliferation of small polygons near the offset regions, etc., we decided on the following approach, detailed in the next sub-section.

### 3.3 Outbound iteration

In this method, an explicit offset region of the pole curve is not calculated, and we try to overcome the difficulty of choosing the correct right offset width by using an iteration. It has more of an ad-hoc flavor, and some steps might fail. Yet, the implementation of this method is more efficient and more robust than the previous one, and we can very accurately display a wide variety of surfaces.

The idea behind the "outbound iteration" method is as follows: we want all domain triangles to correspond to surface patches that lie entirely inside the bounding region, or straddle it. Those that correspond to surface regions that are outside the region are dropped; those that correspond to straddling patches are mapped and the patch is clipped against the bounding region.

Now, the domain is triangulated in such a way that triangles contain pole points and regular points at their vertices. A domain triangle with a pole at a vertex may map onto an infinite-area surface patch, which may lie partly inside the bounding region. If we determine this to be the case, we iteratively search the edges of the domain triangle for points that lie outside the bounding box, and then attempt to construct a domain polygon whose vertices are all regular points, which does not contain any pole points in its interior, and which straddles the bounding box. This domain polygon is mapped onto the surface and the resulting patch clipped against the bounding box. The algorithm is given in more detail below.

METHOD II.

1. Perform the projective reparameterization of section 2 so the entire surface is mapped in four pieces. Perform the next steps for each piece.

2. Generate points on the pole curve that lie inside the unit simplex.

3. Generate points in the rest of the unit simplex according to some scheme. The two kinds of points are to be distinguished from each other.

4. Compute a triangulation of the points thus generated. If an edge of any such triangle touches the pole curve, insert the intersection point and recompute the triangulation.

5. Every triangle will then have 0,1,2, or 3 pole points. If it has no pole points, it can be mapped immediately to the surface. Suppose it has one pole point and two normal points. Let the pole point be called $\mathbf{p}$ and the normal points $\mathbf{q_1}, \mathbf{q_2}$. We denote the surface point corresponding to a point $\mathbf{x}$ as $S(\mathbf{x})$. We assume that $S(\mathbf{p})$ is a point at infinity, which is likely since $\mathbf{p}$ is a pole. If $S(\mathbf{q_1}), S(\mathbf{q_2})$ both lie outside the bounding region, this triangle will not be mapped. If both lie inside the region, then a bisection-like iteration is performed along the edges from $\mathbf{p}$ to $\mathbf{q_1}$ and $\mathbf{p}$ to $\mathbf{q_2}$. Each iteration stops when a point $\mathbf{x}$ is reached, such that $S(\mathbf{x})$ is outside the bounding box. Then $\mathbf{p}$ is discarded from this triangle and a quadrilateral is constructed using the two new vertices vertices. Now suppose there are two pole points, and one simple point. By a similar process this triangle is either discarded, or the two pole points in the triangle are replaced by normal points whose images lie outside the bounding box. A triangle with three pole points is discarded (this case occured rarely in our experiments).

6. Each triangle is mapped onto the surface, and then clipped against the bounding box.

Once again, pole curve points are generated using the method of [12]. In the the second step, we just generate a constant-size triangular grid on the unit simplex. The grid points are merged with the curve points in a special data structure that allows them to be marked as pole or normal ponits, and the Delaunay triangulation of the entire set of points is constructed [6, 11, 24].

The third step is omitted in our implementation because it has not been frequently needed, and it is inefficient. The Delaunay triangulation is constructed incrementally, and once an intersection point is found, it can be inserted into the existing graph. However, to check each edge for an intersection against the pole curve will be time-consuming if done in a brute-force manner.

This method, while nowhere near as theoretically pleasing as even method I, works generally very well in practice. It is also quite easy to apply to non-faithful parameterizations, where it is required to triangulate a multiply mapped region. Problems that do occur often involve base points, since the outbound iteration does not terminate if one of the pole points is actually a base point; thus we put limits on the iteration count.

### 3.4 Examples

We now show some examples of parametric surfaces, whose equations are given in Table 1. In Figure 5, (a) shows a hyperboloid of one sheet, and with the surface are drawn two lines which are the seam curves corresponding to the two domain base points. Figure 5(b) shows a "elbow" cubic surface of algebraic degree 3 that smoothly blends two cylinders. Part (c) of the same figure shows a singular cubic surface of algebraic degree 3, and (d) a steiner surface of algebraic degree 4. The last does not have any poles or base points, but we show it to illustrate the total surface mappings of section 2. All pictures are generated using an implementation of method II in *Ganith* [1].

## 4 CONCLUSIONS AND FUTURE WORK

We have discussed the problem of displaying an arbitrary parametric curve or surface, and shown that many difficulties arise that have not been considered to date. The most common problem is the vanishing of the denominators of the

| Figure | $(x(s,t), y(s,t), z(s,t))$ |
|---|---|
| (a) | $\left(\frac{t^2-s^2+1}{s^2+t^2-1}, \frac{2st}{s^2+t^2-1}, \frac{2s}{s^2+t^2-1}\right)$ |
| (b) | $\begin{pmatrix} -8t^2+(-2s^2-12s-8)t+8s+16 \\ -8t^2+(2s^2+12s+40)t-4s^2-16s-32 \\ (-4s-12)t^2+(8s+24)t+2s^2+8s \end{pmatrix} /(-4t^2+8t-2s^2-8s-16)$ |
| (c) | $\left(\frac{t^3+2t+s^3-7s^2+1}{t^3+s^3+1}, \frac{2t^3+2t^2-7s^2t+2s^3+2}{t^3+s^3+1}, \frac{2st-7s^3}{t^3+s^3+1}\right)$ |
| (d) | $\left(\frac{2s}{t^2+s^2+1}, \frac{2t}{t^2+s^2+1}, \frac{2st}{t^2+s^2+1}\right)$ |

Table 1: Rational parametric equations of surfaces in Figure 5

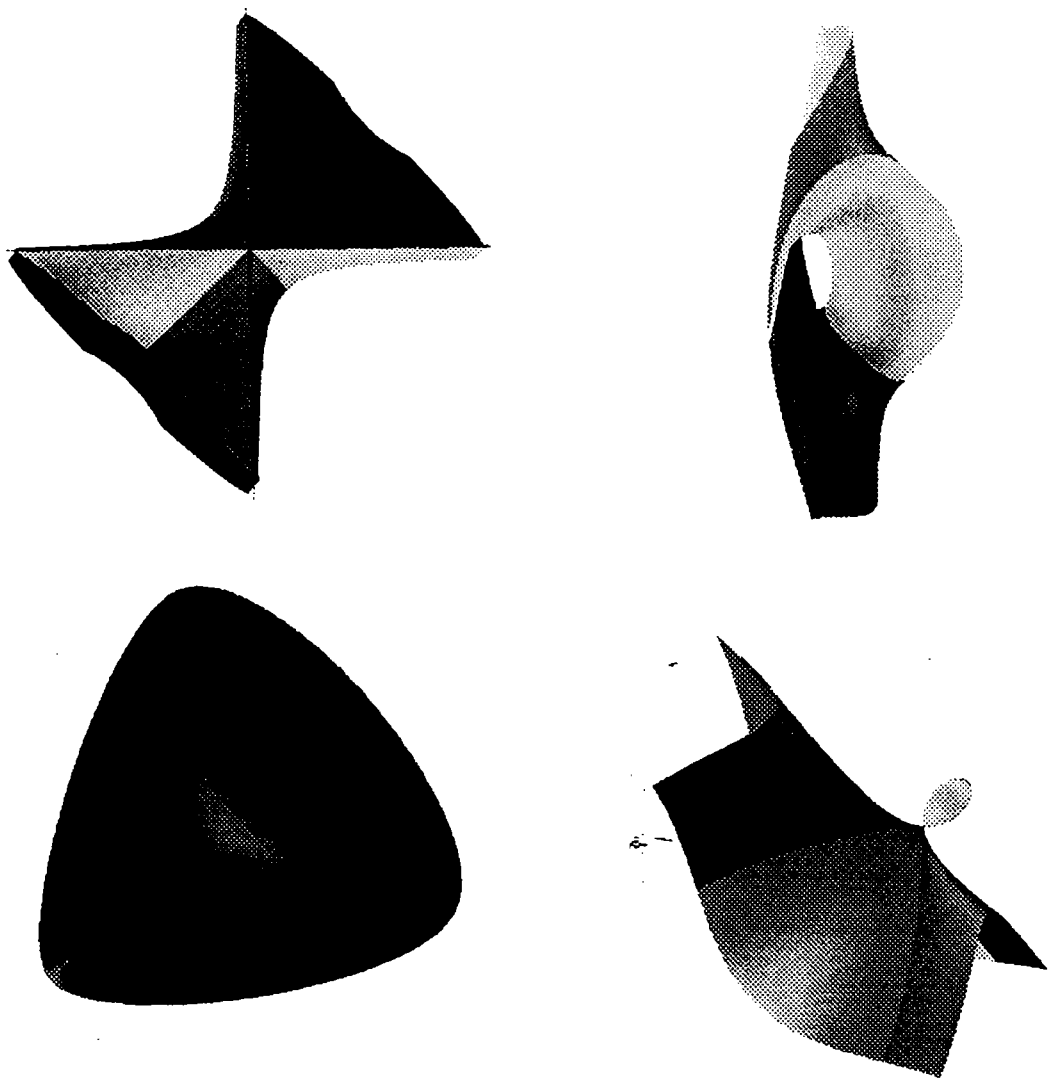

Figure 5: (a)-(d) clockwise from upper left

rational parametric functions, which we showed causes chaos in ordinary display algorithms. Several ways of handling this problem were outlined. Our implementation required several moderately powerful subprograms: implicit curve approximation, polygon subtraction, polygon clipping, and Delaunay triangulation. More work is needed on several fronts, before the robust display problem can be considered fully solved. Nevertheless, we feel that we have made considerable headway, and while future methods will be more robust, they will not lead to our being able to visualize any significant class of rational parametric surfaces that is impossible now. For almost all surfaces, the current implementation generates a good enough approximation to at least get a good feel for the shape of the surface, even if minute gaps are present.

The implementation has proved time-consuming and often frustrating, because of the many numerically delicate problems that arise. However, we feel amply rewarded by being able to visualize the extremely complex and often aesthetically pleasing geometry of rational parametric surfaces.

## 5 ACKNOWLEDGMENTS

This research was supported in part by NSF grants CCR 90-02228, DMS 91-01424 and AFOSR contract 91-0276.

## 6 REFERENCES

[1] Bajaj, C., and Royappa, A., (1990) "The Ganith Algebraic Geometry Toolkit", in *DISCO '90*, Capri, Italy. Also Computer Science Technical Report No. 91-065, Purdue University.

[2] Bajaj, C., and Kim, M., (1989), "Generation of Configuration Space Obstacles: The Case of Moving Algebraic Curves", *Algorithmica*, 4, 2, 157-172.

[3] Catmull, E., (1975) "Computer Display of Curved Surfaces", in *IEEE Conference Proceedings on Computer Graphics, Pattern Recognition and Data Structures*, May 1975, reprinted in *Tutorial and Selected Readings in Interactive Computer Graphics*, H. Freeman (ed.), IEEE, 1980, 309-315.

[4] Char, B. W., Geddes, K. O., Gonnet, G. H., Leong, B. L., Monagan, M. B., and Watt, S. M., (1991), *First Leaves: A Tutorial Introduction to Maple V*, Springer-Verlag.

[5] DeRose, T., (1991), "Rational Bezier Curves and Surfaces on Projective Domains", in *NURBS for Curve and Surface Design*, G. Farin, Editor, SIAM Press, 1-14.

[6] Edelsbrunner, H., (1987), *Algorithms in Combinatorial Geometry*, Springer-Verlag.

[7] Farin, G., (1991) *NURBS for Curve and Surface Design*, (Editor), SIAM Press.

[8] Farouki, R.T., and Neff, C.A., (1990), "Analytic Properties of Plane Offset Curves", *Computer Aided Geometric Design*, Vol. 7, 83-99.

[9] Farouki, R.T., (1986), "Trimmed Surface Algorithms for the Evaluation and Interrogation of Solid Boundary Representations", *IBM Journal of Research and Development*, No. 31, 314-344.

[10] Foley, J., Van Dam, A., Feiner, S.K., and Hughes, J.F., (1990), *Fundamentals of Interactive Computer Graphics*, 523-529, Addison Wesley Publishing Co.

[11] Fortune, S., (1989), "Numerical Stability of Algorithms for 2D Delaunay Triangulations", in *Proceedings of the 8th ACM Symposium on Computational Geometry*, 83-92.

[12] Geisow, A., (1983), "Surface Interrogations", Ph.D. Dissertation, School of Computing and Accountancy, University of East Anglia.

[13] Hartshorne, R., (1977), *Algebraic Geometry*, Springer-Verlag.

[14] Johnstone, J. and Bajaj, C., (1990), "On the Sorting of Points Along an Algebraic Curve", *SIAM Journal on Computing*, 19, 5, 925-967.

[15] Kosters, M., (1991), "Curvature Dependent parameterization of Curves and Surfaces", *Computer Aided Design.*

[16] Lane, J. and Carpenter, C., (1979), "A Generalized Scan Line Algorithm for the Computer Display of Parametrically Defined Surfaces", in *Computer Graphics and Image Processing*, 11, 3, 290-297.

[17] Wolfram, S., (1991), *Mathematica: A System for Doing Mathematics by Computer*, Addison-Wesley.

[18] Patterson, R.R., and Bajaj, C., (1989) Curvature Adjusted Parameterizations of Curves, Computer Science Technical Report CSD-TR-907 and CAPO Report CER-89-18.

[19] Rockwood, A., Heaton, K., and Davis, T., (1989), "Real-Time Rending of Trimmed Surfaces", in *Proceedings of ACM SIGGRAPH '89*, Boston, 107-116.

[20] Rossignac, J., and Requicha, A., (1986), Offsetting Operations in Solid Modeling, *Computer Aided Geometric Design*, 3, 2, 129-148.

[21] Royappa, A., (1992), *Symbolic Methods in Computer Graphics and Geometric Modeling*, Ph.D. Thesis, Purdue University.

[22] Schweitzer, D. and Cobb, E.S., (1982), "Scanline Rendering of Parametric Surfaces", *Computer Graphics*, 16, 3, 265-271.

[23] Semple, J.G., and Roth, L. (1985), *Introduction to Algebraic Geometry*, Oxford University Press, 1985.

[24] Sugihara, K., and Iri, M., (1989), "Construction of the Voronoi Diagram for One Million Generators in Single Precision Arithmetic", First Canadian Conference on Computational Geometry, Montreal, Canada.

[25] Sutherland, I.E., and Hodgman, G.W., (1974), "Reentrant Polygon Clipping", *Communications of the ACM*, 17, 1, 32-42.

[26] Von Herzen, B., (1989), *Applications of Surface Networks to Sampling Problems in Computer Graphics*, Ph.D. Thesis, California Institute of Technology.

[27] Warren, J., (1992), "Creating Multisided Rational Bezier Surfaces Using Base Points", in *ACM Transactions on Graphics*, 11, 2, 127-139.

[28] Weiler, K. and Atherton, P., (1977), "Hidden Surface Removal Using Polygon Area Sorting", SIGGRAPH '77 Proceedings, published as *Computer Graphics* 11, 2, 214-222.

[29] Zariski, O., (1971), *Algebraic Surfaces*, Springer-Verlag.