Direct Generation of Contour Files from Constructive Solid Geometry Representations

Sashidhar Guduri, Research Assistant
Richard H. Crawford, Assistant Professor
Joseph J. Beaman, Professor
Department of Mechanical Engineering
The University of Texas at Austin
Austin, TX 78712

Abstract

Geometry processing for layer-based Solid Freeform Fabrication consists of at least two steps: slicing the geometry to obtain the part contours for each layer, and scan-converting the layers for laser scanning (or other device-dependent in-layer processing). This paper discusses the generation of contour files directly from Constructive Solid Geometry (CSG) representations for the Selective Laser Sintering process. Previous work at The University of Texas focused on slicing CSG representations composed of quadric primitives. This paper extends previous work at UT to include the torus, a fourth degree surface, as one of the CSG primitives. Slicing a torus results in a fourth degree equation in two variables, which represents a curve in two-dimensional real space. For some special cases, this fourth degree equation may be sub-divided into two second degree equations. For the cases where the fourth degree equation cannot be sub-divided, a method is presented to approximate the fourth degree curve with second degree curve segments.

Introduction

Solid Freeform Fabrication (SFF) techniques manufacture solid objects directly from three-dimensional computer models. Most SFF processes produce parts on a layer-by-layer basis. The process begins by slicing the geometric description of the part into layers. The slicing operation generates the contours of the part for each layer. The contours are then processed in a manner dependent upon the particular SFF technology. For instance, for Selective Laser Sintering (SLS) the contours are discretized into "toggle points" at which the laser beam must be modulated to produce the desired solid.

The geometric description used to represent solid objects significantly affects the accuracy and quality of the final parts produce with SFF. One way to improve the final accuracy and definition of SFF parts is to improve the geometric descriptions that represent three dimensional objects. As described in [1], Constructive Solid Geometry is one geometric description where the accuracy of the final contours can be improved as compared to present geometric representations. A method was presented in [1] for generating contour files from Constructive Solid Geometry representations composed of quadric objects. This paper extends that work to include higher degree surfaces (degree greater than two) in the primitive set. Special attention is given to the torus, a fourth degree surface.

Slicing a higher degree surface results in a boundary curve whose degree may be greater than two. A curve of degree greater than two is parametrizable if the genus (g) of the curve is zero. The genus of a curve is defined by

$$g = \frac{(N-1)(N-2)}{2} - \sum_i \delta(p_i), \tag{1}$$

where N is the degree of the equation and the $\delta(p_i)$ operator appropriately counts the number of times that the curve comes in contact with itself at each singular point p_i. All quadratic curves are genus zero and are therefore parametrizable. A limited set of curves of degree greater than two are parametrizable. Unfortunately, the algebraic equations produced by many geometric design applications are not generally genus zero. It therefore becomes necessary to approximate higher degree curves with lower degree curve segments. These curve segments are then parametrized individually.

The method presented in this paper approximates curves of degree greater than two using second degree curve segments. There are two advantages to using second degree curves. First, all second degree curve segments are parametrizable. Second, the intersection of two second degree curves (required for Boolean set operations) can be computed by well-known, stable algorithms. If curve segments of degree greater than two are used as approximations, calculating the intersection of two curves becomes computationally expensive. The algorithm for generating the approximation is based on degree reduction of the triangular Bernstein form of the curve. The next section of the paper illustrates the representation of the torus as a CSG primitive and gives the equations involved in calculation of the implicit equation of the torus. Following this section, a method for determining the Bernstein coefficients of the implicit curve is described. The remainder of the paper describes the steps in the approximation algorithm: parametrization, approximation and error estimation, subdivision, and resolution of singularities. Examples are presented at the end of the paper.

Torus as a CSG Primitive

The equation of a torus generated by sweeping a circle about the y axis, with the origin at the center, is given by:

$$(x^2 + y^2 + z^2 + R^2 - r^2)^2 - 4r^2(x^2 + z^2) = 0 \tag{2}$$

As a CSG primitive the torus is represented using a base point, an axis vector and two radii, as illustrated in Figure 1.

Given the above information, the equation of the torus can be generated by translating the base point of the torus to the origin and rotating about x and z axes such that the resulting axis of the torus is aligned with the y axis. This procedure is summarized below:

1. Normalize the axis vector of the torus.
2. Translate the base of the torus to the origin.
3. Rotate about the x axis by an angle of $-\tan^{-1}\left(\frac{z_a}{y_a}\right)$.
4. Rotate about the z axis by an angle of $\tan^{-1}\left(\frac{x_a}{\sqrt{z_a^2 + y_a^2}}\right)$.

If the above steps are applied the following transformations are obtained

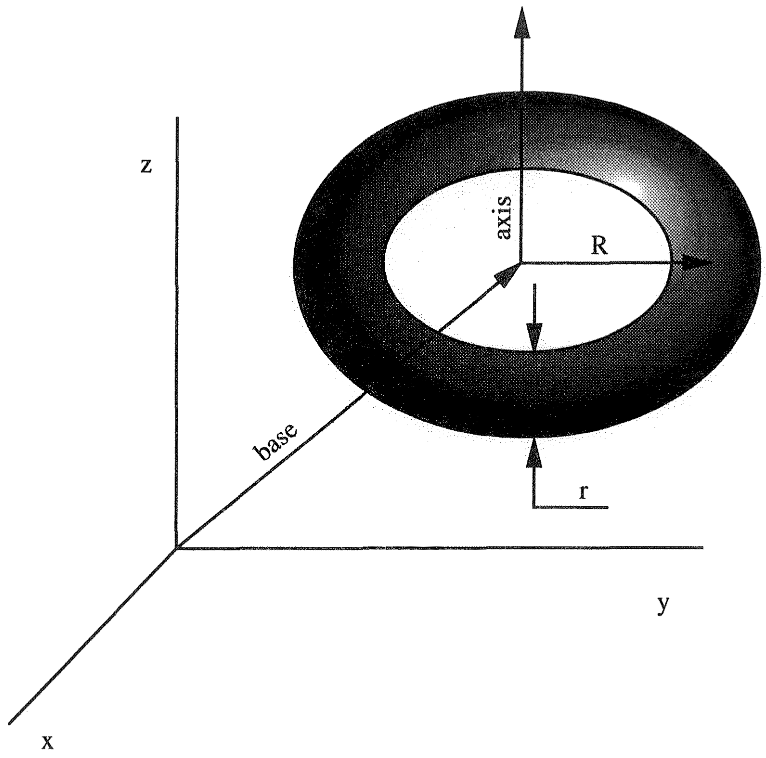

Figure 1. Representation of a torus.

$$x = \sqrt{1-x_a^2}(X-x_b) - \frac{x_a y_a}{\sqrt{1-x_a^2}}(Y-y_b) - \frac{x_a z_a}{\sqrt{1-x_a^2}}(Z-z_b)$$

$$y = x_a(X-x_b) + y_a(Y-y_b) + z_a(Z-z_b)$$

$$z = \frac{-z_a}{\sqrt{1-x_a^2}}(Y-y_b) + \frac{y_a}{\sqrt{1-x_a^2}}(Z-z_b) \quad , \qquad (3)$$

where X,Y and Z define the coordinate system local to the torus. Substituting these equations into equation 2 gives the fourth degree equation of the torus in x, y and z. Substituting the z value of the slicing plane gives a fourth degree equation in x and y that is the implicit equation of the contour for that slicing plane.

Conversion of Algebraic Equations to Triangular Bernstein Form

This section describes a method for converting an algebraic equation in two unknowns to Bernstein form over arbitrary triangular regions [2]. Use of the Bernstein simplifies the task of degree reduction and degree elevation of algebraic curves. The triangular form simplifies subdivision of the curve. A method to do this is also presented in this paper.The general form of a bivariate polynomial of degree N is given by:

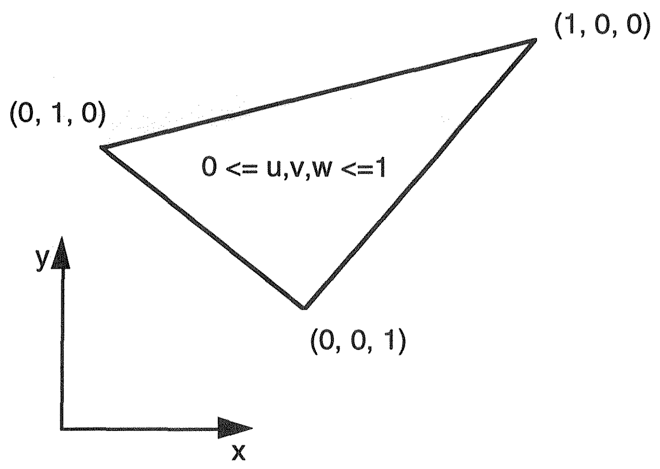

Figure 2. Barycentric coordinate system.

$$P^N = \sum_{i,j=0}^{i+j\leq N} a_{i,j} x^i y^j \tag{4}$$

A triangle can be defined by the three points P_{00}, P_{N0} and P_{0N}. The coordinates of any point inside the triangle can be computed from its barycentric coordinates (also referred to as area or trilinear coordinates) u, v and w, where u+v+w = 1. An arbitrary point P in the triangle is expressed in terms of barycentric coordinates by:

$$P = wP_{00} + uP_{N0} + vP_{0N} \tag{5}$$

The coefficients of the bivariate polynomial are used to compute the elements of a four dimensional array C, as given by the following equations:

$$C_{00}^{r0} = a_{N-r,r} \text{ and } C_{ij}^{rk} \equiv 0, \quad i, j < 0 \text{ and } i+j > k$$

$$C_{ij}^{rk} = C_{i-1,j}^{r,k-1} Y_{N0} + C_{i,j-1}^{r,k-1} Y_{0N} + C_{ij}^{r,k-1} Y_{00} + P(k,i,j) a_{N-r,r-k} \tag{6}$$

where k = 0,...,r and i,j >= 0 and i+j <= k and

$$P(a,b,c) = \frac{a!}{b!\,c!\,(a-b-c)!} \tag{7}$$

The elements of C are used to calculate a tetrahedral array D:

$$D_{ij}^{k} = 0, \tag{8}$$

where i,j < 0 and i+j > k and k = 0,...,N

$$D_{ij}^{k} = D_{i-1,j}^{k-1} X_{N0} + D_{i,j-1}^{k-1} X_{0N} + D_{ij}^{k-1} X_{00} + C_{ij}^{kk} \tag{9}$$

where k = 0,......,N and i,j >= 0 and i+j <= k. Finally the Bernstein coefficients are given by:

$$b_{ij}^{N} = \frac{D_{ij}^{N}}{P(N,i,j)} \tag{10}$$

After the Bernstein coefficients are obtained, the algorithm proceeds with approximation, as described in the next section.

Approximation of Higher Degree Curves

The algorithm for approximating algebraic curves of degree greater than two consists of four stages [3]. The parametrization stage builds a parametric definition. The approximation stage attempts to build a degree N–1 equation that matches the given degree N expression within acceptable error. The subdivision stage partitions the region where the curve is defined into smaller regions, if the region is not already too small. In the fourth stage, the algorithm resolves singularities. The complete algorithm is summarized as follows:

Stage 1: Parametrization

An attempt is made to parametrize the curve f(x, y) = 0 if the degree is less than 3. Should the degree be greater than 2, the algorithm proceeds with the approximation stage, where the degree of the equation is reduced to two.

Stage 2: Approximation

If the degree k function f(x, y) monotonically increases or decreases with respect to any one edge of the domain triangle, estimate the error present in the k–1 approximation. If the error is within limits, return the approximation to stage 1 of the algorithm; if not, proceed with the subdivision stage.

Stage 3: Subdivision

If the size of the triangular region is smaller than a preset limit, assume the region contains a singularity and proceed to stage 4. Otherwise, divide the triangle into subregions and return to stage 1.

Stage 4: Resolution of Singularities

Using quadratic transformations for resolving singularities, generate an approximation for the curve segment through the given region.

Parametrization

If the degree of an equation is less than three, it can be parametrized. The procedure consists of two steps. First, the Bernstein equation is transformed into standard polynomial form (eqn. 4). Then the parameter values are calculated at the intersections of the triangle and the curve. The Bernstein equation is a function of parameters u and v. The coordinates x and y vary linearly with respect to u and v according to the following equations:

$$x = wX_{00} + uX_{N0} + vX_{0N}$$

$$y = wY_{00} + uY_{N0} + vY_{0N} \tag{11}$$

Transforming the above equations such that u and v are obtained in terms of x and y and substituting these equations for u and v in f(u,v), we get a bivariate polynomial equation in terms of x and y.

$$u = \frac{x(Y_{01} - Y_{00}) - y(X_{01} - X_{00})}{(X_{10} - X_{00})(Y_{01} - Y_{00}) - (Y_{10} - Y_{00})(X_{01} - X_{00})}$$

$$v = \frac{x(Y_{10} - Y_{00}) - y(X_{10} - X_{00})}{(X_{01} - X_{00})(Y_{10} - Y_{00}) - (Y_{01} - Y_{00})(X_{10} - X_{00})}$$

$$f(x,y) = f(u,v) \tag{12}$$

The parametrization algorithm is described in [4]. The intersection points of the curve and the Bernstein triangle are calculated and the appropriate curve segments are taken by determining if the curve segment lies inside the triangle or not.

Approximation

In this stage of the algorithm approximations of degree N–1 are generated for degree N curves and the characteristics such as convex hull property of the Bernstein polynomial basis are exploited to estimate the maximum error present in the approximation. This section describes the equations involved in degree reducing and degree elevating Bernstein polynomials.

Given a degree N algebraic curve in Bernstein form, an exact representation of this curve can be created using a degree N+1 Bernstein polynomial basis [5]. Mathematically, this means the expression

$$\sum_{i,j=0}^{i+j\leq(N+1)} \frac{(N+1)!}{i!j!(N+1-i-j)!} x^i y^j (1-x-y)^{N+1-i-j} h_{ij} \tag{13}$$

is equivalent to

$$\sum_{i,j=0}^{i+j\leq N} \frac{N!}{i!j!(N-i-j)!} x^i y^j (1-x-y)^{N-i-j} b_{ij} \tag{14}$$

This degree augmentation process, shown in the following equations, defines the coefficients of the degree N+1 expression in terms of those from the degree N equation:

$$h_{N+1,0,0} = b_{N,0,0},$$

$$h_{0,N+1,0} = b_{0,N,0},$$

$$h_{0,0,N+1} = b_{0,0,N}$$

$$h_{i,j,k} = \frac{i * b_{i-1,j,k} + j * b_{i,j-1,k} + k * b_{i,j,k-1}}{N+1}, \ i+j+k = N+1, \tag{15}$$

Similarly, the Bernstein polynomial formulation can be used to produce a lower degree polynomial approximation [6]. The degree reduction process is summarized below:

$$l_{N-1,0,0} = b_{N,0,0},$$

$$l_{0,N-1,0} = b_{0,N,0},$$

$$l_{0,0,N-1} = b_{0,0,N}$$

$$l^1_{i-1,j,k} = \frac{N * b_{i,j,k} - j * l^1_{i,j-1,k} - k * l^1_{i,j,k-1}}{i}, \ i \neq 1 \text{ and } i \geq j,k$$

$$l^2_{i,j-1,k} = \frac{N * b_{i,j,k} - i * l^2_{i-1,j,k} - k * l^2_{i,j,k-1}}{j}, \ j \neq 1 \text{ and } j \geq i,k$$

$$l^3_{i,j,k-1} = \frac{N * b_{i,j,k} - i * l^3_{i-1,j,k} - j * l^3_{i,j-1,k}}{k}, \ k \neq 1 \text{ and } k \geq i,j \tag{16}$$

Unlike degree elevation, the coefficients in degree reduction are defined only in terms of previously computed coefficients, and all coefficients with negative subscripts are zero. When any two or all three indices are equal, the components given by each of the corresponding equations are averaged to define a single value. Note that if the degree of a polynomial is first elevated and subsequently lowered, the original polynomial coefficients are retrieved.

Once the degree reduction procedure has been applied to the algebraic curve $f(x, y) = 0$ and an approximation $g(x, y)$ is generated, an estimate of the approximation error is required. An upper bound on the error can be derived from a combination of difference values ($\Delta z = f - g$) and directional derivative information taken from the two single valued surface equations $z = f(x,y)$ and $z = g(x,y)$ [7].

To begin, the degree elevation procedure is invoked to ensure f and g have the same polynomial degree, thereby assuring the same control point lattice over the triangle. Applying the convex hull property, the maximum difference between the two surfaces $z = f(x,y)$ and $z = g(x,y)$ is bounded by the largest difference found between the two control nets:

$$|\Delta z_{max}| = \max|f_{ij} - g_{ij}| \tag{17}$$

Tangent plane and normal information can be derived from the surface equations [5, 8]. This information is used to relate the largest surface difference, given in the above equation, to the difference between the two algebraic curves. If point P is located on the curve approximation $g = 0$, the corresponding point on $f = 0$ has to be located. Figure 3 illustrates that there must exist

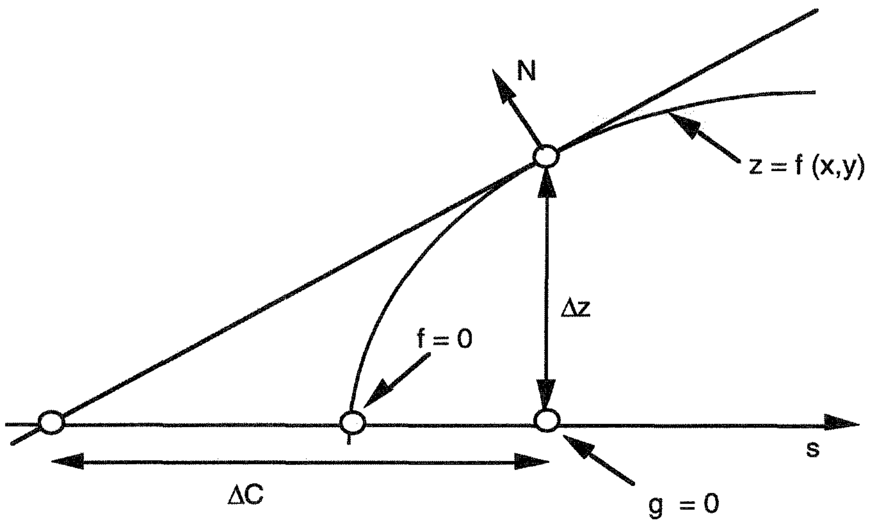

Figure 3. Error estimation.

a point on f = 0 at least within a distance ΔC, where θ is the smallest angle between the tangent plane and a direction s defined in the x-y plane, and ΔC is given by the equation

$$\Delta C = \frac{\Delta z}{\tan\theta} \tag{18}$$

Since the surfaces are single valued, tanθ defines the value of the directional derivative, $\frac{dz}{ds}$, of the function z = f(x,y) with respect to the direction s. If the original Bernstein triangle with its Bernstein coefficients is called the original patch, then the Bernstein coefficients of the directional derivative patch can be calculated from the original patch using the following equation (for direction s shown in Figure 4):

$$b_{i,j} = N * (B_{i,j+1} - B_{i,j}), \tag{19}$$

where i+j <= (N-1) and N is the degree of the original patch.

The convex hull property is applied to the directional derivative patches of f(x,y) and g(x,y) to determine a minimum value for tanθ. Combining the maximum surface difference with the minimum value of tanθ, a single error bound is produced:

$$e_{max} \leq \Delta C = \frac{|(\Delta z_{ij})_{max}|}{|(dz_{ij} / ds)_{min}|} \tag{20}$$

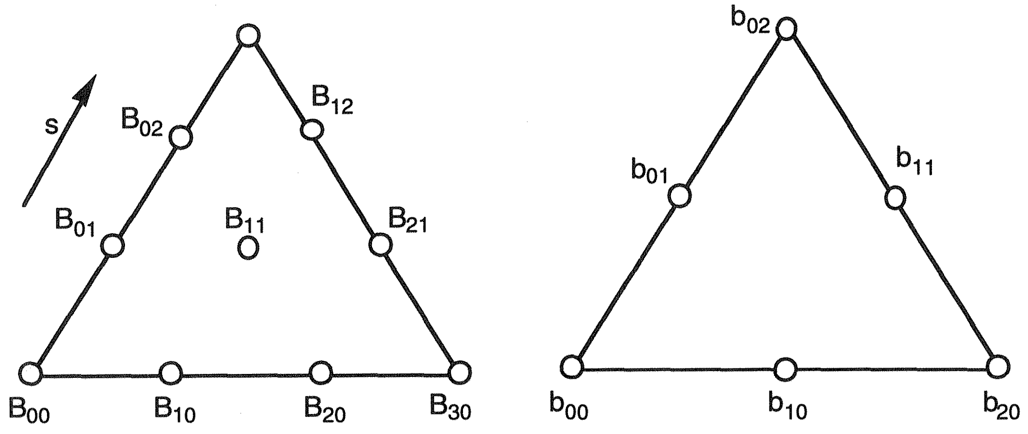

Figure 4. Original patch and its directional derivative patch.

A tighter error bound, but computationally more expensive, can be obtained by calculating the error at all the control points. If the error estimate is not within the specified limits, the triangle must be subdivided. Subdivision techniques are discussed in the next section.

Subdivision

The subdivision stage of the algorithm is reached under two sets of circumstances. When the given function is not monotonic with respect to any edge of the triangle, the triangle is subdivided to generate new sets of edges. The second condition is reached when an unacceptable approximation error is present. The following equations define the procedure for calculating Bernstein coefficients after subdivision [7]:

$$b^0_{i,j,k}(u,v,w) \equiv b_{i,j,k}, \tag{21}$$

where i+j+k = N, and

$$b^m_{i,j,k}(u,v,w) = ub^{m-1}_{i+1,j,k}(u,v,w) + vb^{m-1}_{i,j+1,k}(u,v,w) + wb^{m-1}_{i,j,k+1}(u,v,w) \tag{22}$$

where i+j+k = N–m.

The geometric interpretation of the recursion process is shown in Figure 5. In general, sets of control points from a degree k control net are combined three at a time, weighted by the barycentric coordinates, to generate a degree k–1 control net. This process is repeated N times for a degree N Bernstein surface, ultimately producing a single value which is the desired point on the surface. The three sets of vertices $b^i_{0,j,k}(u,v,w)$, $b^j_{i,0,k}(u,v,w)$ and $b^k_{i,j,0}(u,v,w)$ define the control nets for the new surface patches generated by subdividing the surface at the point with barycentric coordinates (u,v,w).

Resolution of Singularities

A singular point is a point on the curve where the function and its first partial derivatives vanish. One may assume that a region contains a singularity if the side of the triangle becomes smaller than the approximation tolerance. When a region is discovered to be near a singularity, the following quadratic transformation is used:

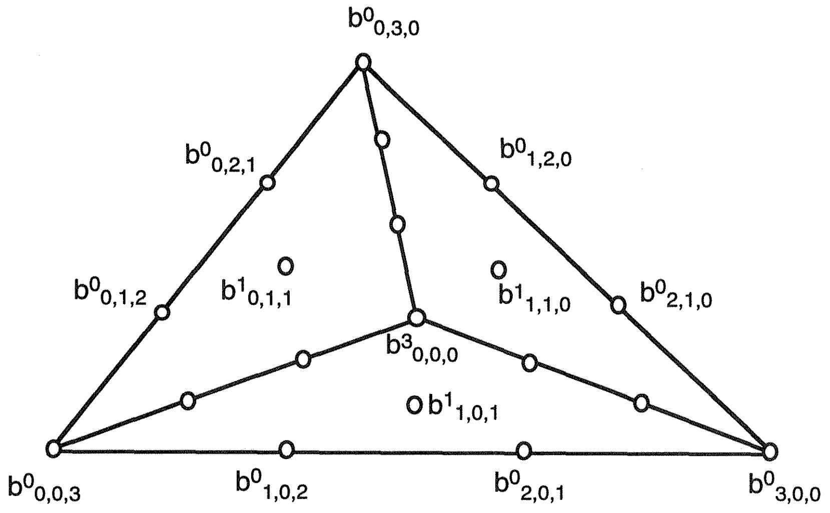

Figure 5. Subdivision of a bivariate Bernstein polynomial.

$$x = x$$

$$y = \frac{y}{x} \tag{23}$$

A piecewise linear approximation is generated near singularities by numerically marching along the various branches of the proper transform. The sequence of points generated are then mapped back to the original coordinate system by reversing the quadratic transformation.

Examples

Examples are presented in this section demonstrating the approximation of a cross-section of a torus using a collection of second degree curves. The curve traces were generated using a collection of procedures written in the C programming language and executed on a Sun^{TM} SparcStation 2. In each of the examples, approximation is developed over a triangular region enclosing the closed curve.

The torus used for slicing is centered at (0, 0, 0) and the axis of the torus is straight line given by the equation y = x. The radius of the torus is 3, and the radius of the disc rotated about the axis to generate the torus is 0.7.

Figure 6 shows the cross-section of the torus at z = 2.2. The number of quadratic curve segments in this approximation is 306.

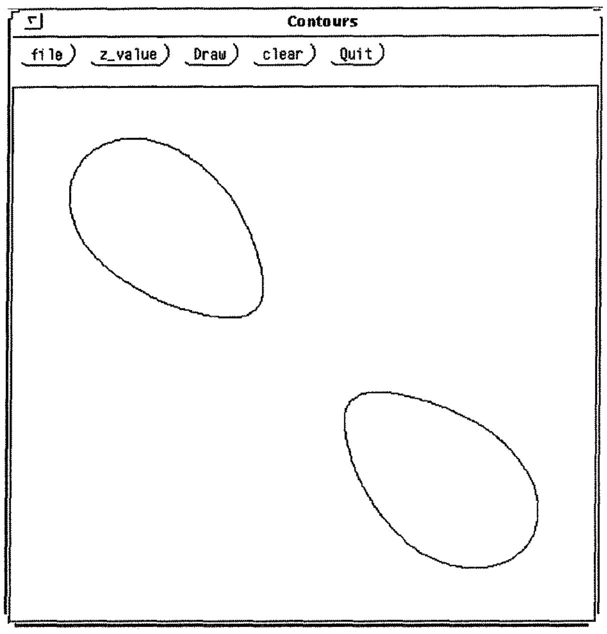

Figure 6. Torus sliced at z = 2.2.

Figure 7 shows the cross-section of the torus at z = 2.3 (the singularity case). The number of quadratic curve segments in this approximation is 318.

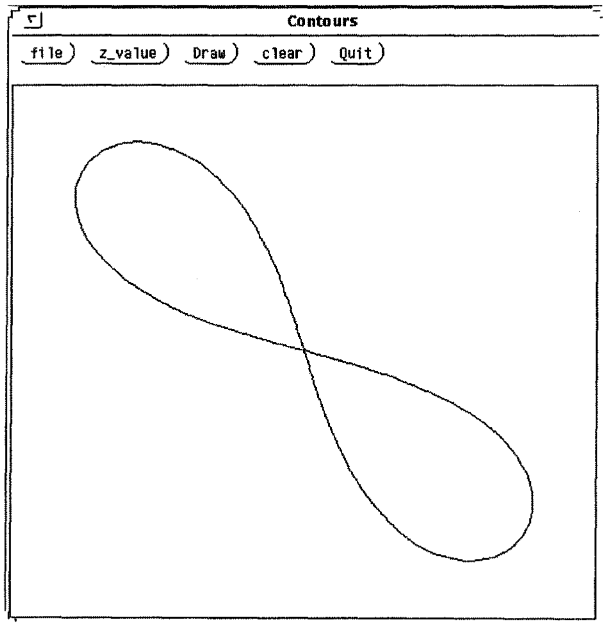

Figure 7. Torus sliced at z = 2.3.

Conclusions

This paper discusses a method to directly process Constructive Solid Geometry representations and obtain contour files. Aspects involved in slicing higher degree surfaces, in particular the torus, are highlighted. The algorithm is applicable to other higher degree surfaces as well, including rational bicubic parametric surfaces. The algorithm provides a rational basis for approximating geometry for SFF applications.

References

1. Sashidhar, Guduri., Crawford, Richard H. and Beaman, Joseph J., "A Method to Generate Exact Contour Files for Solid Freeform Fabrication", *Proceedings of Solid Freeform Fabrication Symposium*, 1992, Marcus, Harris L., Beaman, Joseph J., Barlow, Joel W., Bourell, David L., Crawford, Richard H., eds., Austin, TX, August 3-5. pp. 95-101.

2. Waggenspack, Warren N. Jr., Anderson, David C., "Converting Standard Bivariate Polynomials to Bernstein Form Over Arbitrary Triangular Regions", *Computer Aided Design*, Vol. 18, No 10, 1986. pp 529-532.

3. Waggenspack, Warren N. Jr., Anderson, David C., "Piecewise Parametric Approximations for Algebraic Curves", *Computer Aided Geometric Design*, 1989. pp 33-53.

4. Abhayankar, Shreeram S. and Bajaj, Chanderjit, "Automatic Rational Parametrization of Curves and Surfaces I: Conics and Conicoids", *Computer Aided Design*, Vol. 19, No 1, 1987. pp 11-14.

5. Farin, Gerald, "Triangular Bernstein-Bezier Patches", Department of Mathematics, University of Utah, Salt Lake City, Utah, USA, 1986.

6. Peterson, Carl S., "Adaptive Contouring of Three-Dimensional Surfaces", *Computer Aided Geometric Design*, Vol. 1, 1984. pp. 61-74.

7. Sederberg, Thomas W., "Planar Piecewise Algebraic Curves", *Computer Aided Geometric Design*, Vol. 1, No 4, December 1984. pp 241-255.

8. Farin, Gerald, "Bezier Polynomials Over Triangles and Construction of Piecewise C^r Polynomials", Tech Report TR/91, Department of Mathematics, Brunel University, UxBridge, Middlesex, UK, 1980.