Parametric Representation of Part Contours in SLS Process

Kenwei Chen, Richard H. Crawford, and Joseph J. Beaman
Department of Mechanical Engineering
The University of Texas at Austin

ABSTRACT

Current layer-based SFF technologies process faceted geometric input data to produce polygonal contours of the part's boundary in each layer. However, for improved part quality, other more accurate representations of part contours are desirable. Likewise, implementation of Wu's minimum time optimal laser tracking control method for selective laser sintering (SLS) requires contour curves that exhibit higher order continuity. In this paper, we first analyze the requirements of optimal laser tracking to develop evaluation criteria for choosing a contour representation. Several possible representation methods are reviewed. We show that the Non-Uniform Rational B-Spline (NURBS) curve meets the criteria. A demonstration program illustrates the advantages of NURBS curves for representing contours with uniform point distributions. The results can be used in other control areas where uniform point distribution or constant velocity is required.

Keywords: computational geometry, NURBS, Bézier curve, tracking control, laser beam control.

1 INTRODUCTION

In layer-based solid freeform fabrication (SFF) processes, a three dimensional geometric description of a solid object is sliced into a stack of two dimensional cross sections whose normals are parallel to the build axis. Current layer data formats, such as CLI from Europe and SLC from 3D Systems, represent the part's boundary in each layer as a set of polygonal contours. SFF technologies then process the layer data differently, depending on the underlying physical process and the sophistication of the control software. For instance, in selective laser sintering (SLS), the laser is raster scanned to fill these cross sections and to build the parts layer by layer. The laser beam is directed by a pair of scanner galvanometers. With this type of process, there is a trade-off between productivity, in terms of build time, and part quality, in terms of surface accuracy, for a given laser power. To increase accuracy, a smaller laser spot is preferred, while productivity demands a larger laser spot to scan more area per unit time. To resolve this dilemma, a strategy based on boundary scanning of the layer contours, followed by a raster fill of the contour interiors, is proposed. Wu and Beaman (Wu and Beaman, 1990; Wu and Beaman, 1991; Wu 1992; Wu and Beaman, 1992) developed an optimal tracking control method for boundary scanning in SLS. The purpose of this paper is to evaluate different parametric representations for layer contours based on their applicability to Wu and Beaman's control method, and for SFF technologies in general.

1.1 Problem Description

Wu's tracking control algorithm is based on the assumption that the tracking path (*i.e.*, contour) is represented parametrically. Wu did not explicitly specify the requirements of the parametric form, but these can be obtained from the derivation of his control algorithm. In general, the parametric form must satisfy the following three requirements: (1) n–1 differentiability, except at corners; (2) parametrization by arc length, so that the first and second derivatives represent tangential velocity and acceleration; and (3) uniform distribution of points. Wu derived the dynamics of scanner galvanometer as a second order system. Therefore, the tracking path must exhibit C^1 continuity. Note that corners have specific meanings. Wu defined a corner as "a point in the prescribed path where tracking velocity must be zero due to the available bounded control torque". To handle corners accurately, the tracking path is divided into several segments such that

corners only exist at the start and end points of the segments. The third requirement allows "near constant tracking speed to gain uniform exposure of laser power".

Three dimensional solid objects are transmitted to SFF processes according to some CAD geometric data format. The current standard for exchanging geometric data form is the so-called STL data file format, which is supported by most commercial CAD system. The STL format represents a 3D part as a triangulated approximation of the part's surfaces. Each facet is defined by its three vertices and its normal vector, which indicates the interior of the part. After slicing the STL data, layer contours are generated and represented by polygonal outlines of the cross-section. However, Wu's method is based on a parametric form meets the criteria described above. Thus, for applying Wu's method to STL data, the polygonal contours must be represented by higher degree parametric curves. On the other hand, other geometric data formats are possible (Darrah, 1990). This paper targets the generation of a parametric representation based on point data. However, we briefly discuss the applicability of parametric representations for contours generated from other 3D geometric data, such as IGES.

2 PARAMETRIC CURVE REPRESENTATIONS

Both parametric and nonparametric curve forms have useful applications. Parametric forms are more dominant in current geometric modeling practice because they are axis-independent, they facilitate redistribution of points along the curve length, they are convenient for generating coordinates of points on the curve, and they represent infinite slopes easily. The well-known and commonly-used parametric curves include the spline, Bézier, B-spline, and Non-Uniform Rational B-Spline (NURBS).

The spline is a set of piecewise polynomials of degree K with continuity of derivatives up to $K-1$ at the common joints between segments. The cubic spline has been found most useful for engineering applications, since it is the lowest order polynomial that allows an inflection point. The cubic spline exhibits second-order or C^2 continuity at the joints. There are three common parametrizations for spline curves (Lee, 1989; Rogers and Adams, 1990): equally spaced, cumulative chord length (which approximates a uniform parametrization), and the centripetal method. However, the spline curve has several disadvantages. In particular, the spline curve can be unstable, especially with higher degree curves (Rogers and Adams, 1990); it does not lend itself well to geometric interpretation, making interactive interfaces difficult; and it does not exactly represent all standard curves, such as conic sections.

To overcome these shortcomings, Bézier curves are used as the basis of many modeling systems. Bézier curves are determined by a defining polygon, which is composed of a set of control vertices. Bézier curves have a many attractive advantages, including recursive computational algorithms, the convex hull property, the variation-diminishing property, positivity of the basis functions, and simple degree elevation and reduction algorithms. One disadvantage of Bézier curves is that, except for the start and end points, they do not interpolate the control points. This characteristic makes the Bézier curve inappropriate for direct use for contour descriptions. However, as a special case of the B-spline curve (discussed below), a Bézier curve can be designed to pass through the given data by adding control points between the interpolated control points.

The B-spline curve has many of the advantages of the Bézier curve, while it overcomes some of the disadvantages. In particular, the order of the B-spline curve is not determined by he number of control points, as is the case for Bézier curves. The most general form, the non-uniform rational B-spline (NURBS) curve, is defined as:

$$\mathbf{C}(t) = \frac{\sum_{i=0}^{n} w_i \mathbf{P}_i N_{i,k}(t)}{\sum_{i=0}^{n} w_i N_{i,k}(t)}, \quad (1)$$

where $\mathbf{P}_i$ are the control points, w_i are the weights associated with the control points, and $N_{i,k}(t)$ are the B-spline basis functions of order k. Note that when all weights are 1, the NURBS curve degenerates to the B-spline, and when a special knot vector is chosen, its basis functions are exactly the same as those of the Bézier curve.

NURBS curves have has many desirable properties for both representation and computation (Piegl 1991). Besides the properties mentioned for Bézier curves, NURBS curves: (1) provide a unified approach for modeling curves, including the conic sections; (2) provide greater flexibility; (3) exhibit computational stability and fast evaluation; (4) lend themselves readily to geometric interpretations; (5) are affine invariant; (6) is generally nonglobal with respect to changes to control points; and (7) provide generalizations of B-spline and Bézier curves. Powerful tool kits are available from the literature to provide knot insertion/refinement/removal, degree elevation, splitting, and other desirable operations. The only relative drawbacks to the NURBS representation are its extra storage requirements and computational overhead. However, with current computer technology, these are not the significant problems.

3 EVALUATION CRITERIA FOR PARAMETRIC REPRESENTATION TECHNIQUES

We base our criteria for choosing an appropriate parametric representation for contours on the three requirements for Wu's control method. First, arc length parametrization is has been studied by several researchers (*e.g.*, Guenter, 1990). Other considerations arise when we consider particular SFF technologies, such as SLS. The accuracy of the parts is always a main objective. In SLS, the laser beam follows the tracking paths during boundary scanning. The laser beam is adjusted by scanners that are controlled by galvanometers in x and y directions. Thus, continuity of motion is another important issue. The dynamics of these motion components restrains the maximum acceleration of the scanners. As Wu pointed out, a straight-line vector scanning mode can be slow when curves exist in the contour path, due to repetitive starts and stops required to accurately approximate the curves. Other considerations include computational constraints. Based on these factors, we propose the following criteria for choosing a parametric representation for part contours:

1. **Accuracy, continuity, and uniform point distribution.** In tracking control, position accuracy and tracking speed smoothness are desired. As stated above, Wu's method requires the parametric form to be n–1 continuously differentiable. Thus, the parametric representation methods should have good approximation and continuity . Uniform point distribution requires the parametric form to produce equally spaced points along the contours.

2. **Interpolation of given point data.** In SLS, the point data given or calculated are normally exactly on the actual contours. Therefore, the parametric curves should pass through all data points.

3. **Robustness.** There are many subtleties in parametric representations, such as non-existent loops that often occur in spline representation. Many traditional parametric curve techniques are not developed for representing general paths. Also, there is usually not a unique curve for interpolating given input data. For example, there are infinitely many NURBS curves that interpolate a given set of data. Robustness here means that we will always get physically reasonable contours from sufficient input data, and no abnormalities will exist in the contours. The input data can be points or other description data, such as radii and center data for circles.

4. **Adaptability.** This property indicates the ability to handle special cases in contours, such as corners, straight line segments, and other local geometric features.

5. **Computational constraints.** Computational considerations include ease of implementation of algorithms, storage, and computation overhead. The floating value round-off problems should be avoided if possible.

Based on these considerations, we propose the NURBS curve for representing contours. In the remainder of the paper, we evaluate the performance of NURBS curves based on these criteria.

4 NURBS PARAMETRIC REPRESENTATION TECHNIQUES

Input to a representation problem generally consists of geometric data, such as points and derivatives. The output is a NURBS curve represented by control points, knots, and weights. Furthermore, the degree p must be specified. If C^r continuity is desired for a curve, then the chosen degree p must satisfy $p \geq r + 1$ (assuming no interior knots of multiplicity > 1).

4.1 Quadratic NURBS Representation

NURBS interpolation of a set of given data is introduced by Piegl and Tiller (1995). For uniform parametrization, we have improved his algorithm both in knot vector construction and derivation of continuity at the start and end points of a contour. Additionally, we select rules that are appropriate for physical mechanical parts. We assume $\{\mathbf{Q}_k\}$, $k = 0,1,\ldots,n$, to be a set of contour data from slicing a part described by the STL format (see Figure 1). We also assume no tangent vectors $\mathbf{T}_k$ are provided. The method constructs n rational curve segments, $\mathbf{C}_i(t)$, $i = 0,\ldots,n-1$, such that $\mathbf{Q}_k$ and $\mathbf{Q}_{k+1}$ are the endpoints of $\mathbf{C}_i(t)$. Neighboring segments are joined with some prescribed level of continuity, and the construction proceeds segment-wise, generally from left to right in the knot vector. Any basis equations are local to only a few neighboring control points.

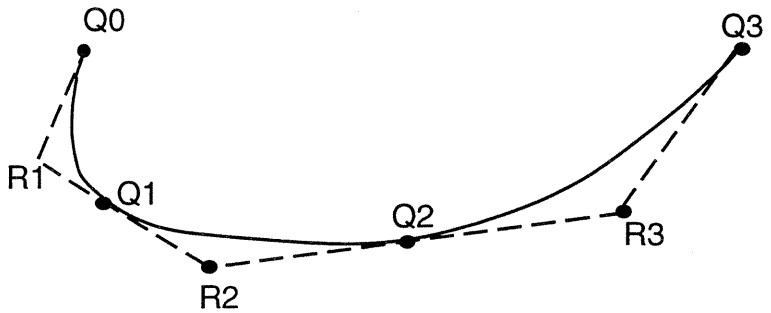


Figure 1. Quadratic B-spline representation.

Now let $\bar{t}_i$ denote the start parameter of $\mathbf{C}_i(t)$ and the end parameter of $\mathbf{C}_{i-1}(t)$, where $\mathbf{C}_i(t)$ and $\mathbf{C}_{i-1}(t)$ meet at $\bar{t}_i$ with G^1 continuity[1] . Note that NURBS curves are continuously differentiable up to $n-1$ except at the knots. G^1 continuity is useful if we regard these knots as corners. C^1 continuity will be obtained later by using cubic Bézier based NURBS representation.

Obtaining the Bézier segments, $\mathbf{C}_i(t)$, requires computation of the inner Bézier control points (one point for quadratics). These control points lie on lines that are tangent to the curve at the $\mathbf{Q}_k$; thus, we require tangent vectors $\mathbf{T}_k$ at each $\mathbf{Q}_k$. In some cases, they can be input along with the $\mathbf{Q}_k$. For instance, they can be obtained when computing intersection points between two surfaces. However, we can compute them approximately from the given data $\mathbf{Q}_k$ (Boehm *et al.*, 1984). The following steps are used to construct the NURBS representation from these data:

Step 1. **Derive tangents at given data points.** We choose the five point scheme to calculate the tangent vectors:

$$\mathbf{T}_k = \frac{\mathbf{V}_k}{|\mathbf{V}_k|} , \tag{2}$$

where

$$\mathbf{V}_k = (1-\alpha_k)\mathbf{q}_k + \alpha_k \mathbf{q}_{k+1} , \tag{3}$$

$$\mathbf{q}_k = \mathbf{Q}_k - \mathbf{Q}_{k-1}, \text{ and} \tag{4}$$

$$\alpha_k = \frac{|\mathbf{q}_{k-1} \times \mathbf{q}_k|}{|\mathbf{q}_{k-1} \times \mathbf{q}_k| + |\mathbf{q}_{k+1} \times \mathbf{q}_{k+2}|}, (k = 2,\ldots n-2). \tag{5}$$

The five-point method has the advantage that three collinear points, $\mathbf{Q}_{k-1}$, $\mathbf{Q}_k$, $\mathbf{Q}_{k+1}$, yield a tangent $\mathbf{T}_k$ that is parallel to the line segment. The denominator of the five-point form vanishes if $\mathbf{Q}_{k-2}$, $\mathbf{Q}_{k-1}$, and $\mathbf{Q}_k$ are collinear and $\mathbf{Q}_k$, $\mathbf{Q}_{k+1}$, and $\mathbf{Q}_{k+2}$ are collinear. This implies either a corner at $\mathbf{Q}_k$, or a straight line segment from $\mathbf{Q}_{k-2}$ to $\mathbf{Q}_{k+2}$. In these cases α_k can be defined in a number of ways; we choose $\alpha_k = 1$, which implies $\mathbf{v}_k = \mathbf{q}_{k+1}$, producing a corner at $\mathbf{Q}_k$. Whether or not to preserve corners depends on the application. The ends of the contour need special treatment:

$$\begin{aligned} \mathbf{q}_0 &= 2\mathbf{q}_1 - \mathbf{q}_2, \\ \mathbf{q}_{-1} &= 2\mathbf{q}_0 - \mathbf{q}_1, \\ \mathbf{q}_{n+1} &= 2\mathbf{q}_n - \mathbf{q}_{n-1}, \text{and} \\ \mathbf{q}_{n+2} &= \mathbf{q}_0. \end{aligned} \tag{6}$$

Note that the last form is used to force a closed contour by setting the end point equal to the start vector. From these forms, we obtain $\mathbf{T}_0$, $\mathbf{T}_1$, $\mathbf{T}_{n-1}$, and $\mathbf{T}_n$.

---

[1]For a discussion of geometric continuity, see Barsky and DeRose, 1989.

Step 2. **Determine the control polygon.** We can reconstruct the control points and knot vector to create a NURBS curve. Let $\mathbf{L}_k$ denote the directed line defined by ($\mathbf{Q}_k$, $\mathbf{T}_k$), and $\mathbf{R}_k$ the intersection point of $\mathbf{L}_{k-1}$ and $\mathbf{L}_k$. Then the control points can be constructed by choosing both $\{\mathbf{Q}_k\}$ and $\{\mathbf{R}_k\}$, or only a subset of these points. Also, let $\bar{t}_0 = 0$, $\bar{t}_n = 1$, and $\bar{t}_{i-1} < \bar{t}_i < \bar{t}_{i+1}$. Then we choose a chord length parametrization to approximate a uniform distribution, $\bar{t}_i = \bar{t}_{i-1} + |\mathbf{Q}_i - \mathbf{Q}_{i-1}|$. The control points, together with the knot vector $\mathbf{t} = \left\{0,0,0,\frac{\bar{t}_1}{\bar{t}_n},\frac{\bar{t}_1}{\bar{t}_n},\frac{\bar{t}_2}{\bar{t}_n},\frac{\bar{t}_2}{\bar{t}_n},\ldots,\frac{\bar{t}_{n-1}}{\bar{t}_n},\frac{\bar{t}_{n-1}}{\bar{t}_n},1,1,1\right\}$, defines a rational G^1 continuous, quadratic B-spline curve interpolating the $\{\mathbf{Q}_k\}$. All weights at the $\mathbf{Q}_k$ are set to 1; weights at the $\mathbf{R}_k$ can be freely chosen. In order to maintain circular arc segments, the weights w_k at the $\mathbf{R}_k$ can be set as follows (Piegl and Tiller, 1995):

1. If $\mathbf{Q}_{k-1}$, $\mathbf{R}_k$, and $\mathbf{Q}_k$ are collinear, set $w_k = 1$.

2. If the triangle formed by $\mathbf{Q}_{k-1}$, $\mathbf{R}_k$, and $\mathbf{Q}_k$ is isosceles, set $w_k = \dfrac{|\mathbf{Q}_{k-1} \bullet \mathbf{Q}_k|}{2|\mathbf{Q}_{k-1} \bullet \mathbf{R}_k|} = \cos\theta$, (a precise circle arc).

3. If the triangle is not isosceles, set w_k as follows (see Figure 2):

$$\mathbf{S} = (1 - s)\mathbf{M} + s\mathbf{R}_k, \text{ and}$$

$$w_k = \frac{s}{1 - s},$$

where $\mathbf{M} = \frac{1}{2}(\mathbf{Q}_{k-1} + \mathbf{Q}_k)$ and $\mathbf{S} = \frac{1}{2}(\mathbf{S}_1 + \mathbf{S}_2)$. Note $\mathbf{S}_1$ and $\mathbf{S}_2$ are respectively on the bisectors of $\angle\mathbf{Q}_k\mathbf{Q}_{k-1}\mathbf{R}_k$ and $\angle\mathbf{Q}_{k-1}\mathbf{Q}_k\mathbf{R}_k$. The parameter s determines the particular type of conic section that is generated. For $s = 0$, a line segment is generated, while $0 < s < 0.5$ gives an ellipse, $s = 0.5$ gives a parabola, and $0.5 < s < 1$ gives a hyperbola. For our implementation, described below, we chose $s = 0.5$.

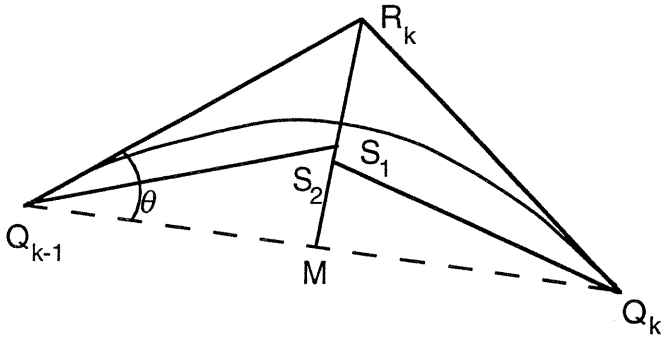


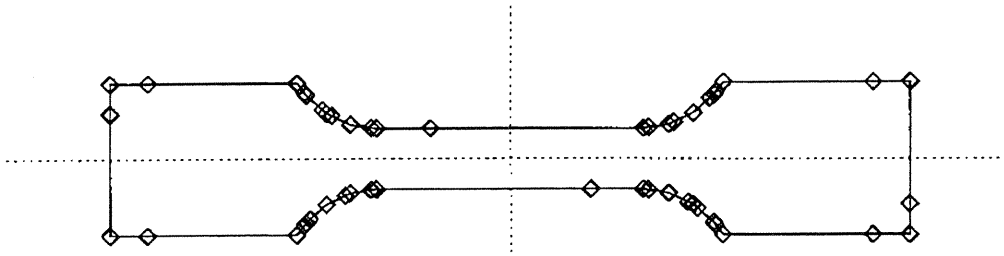
Figure 2. Determining the weight for a curve segment.

Step 3. **Exception handling.** The discussion above assumes that the intersections $\mathbf{R}_k$ exist and that $\mathbf{R}_k = \mathbf{Q}_k + \gamma_{k-1}\mathbf{T}_{k-1}$ and $\mathbf{R}_k = \mathbf{Q}_k + \gamma_k\mathbf{T}_k$, where $\gamma_{k-1} > 0$ and $\gamma_k < 0$. The following special cases occur when $\mathbf{R}_k$ cannot be computed by intersection:

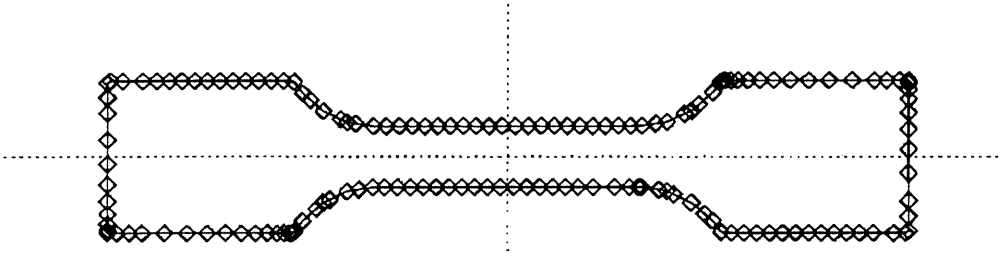
1. $\mathbf{T}_{k-1}$ and $\mathbf{T}_k$ are parallel. This indicates collinear segments, an inflection point, or a 180° turn in the curve; or

2. If $\mathbf{R}_k$ can be computed, but γ_{k-1} and γ_k do not satisfy the above relations. This indicates either an inflection point or a turn of more than 180°.

To handle case 1, set $\mathbf{R}_k = (\mathbf{Q}_{k-1} + \mathbf{Q}_k)$. All other special cases can be handled, for example, by creating two parabolic segments between $\mathbf{Q}_k + \mathbf{Q}_{k-1}$, instead of one. Figure 3(a) shows point data generated from slicing the STL file for a real part. The outline indicates the developed control polygon. Figure 3(b) is the quadratic Bézier based NURBS representation of these data. Note that NURBS curve maintains corners and straight lines.



(3a)



(3b)

Figure 3. Original CAD data and developed control polygon (a), and NURBS representation by quadratic algorithm (b). (◊ indicates points given or calculated.)

4.2 Bézier Based NURBS Representation

Cubics easily handle three-dimensional data and inflection points without special treatment. We begin by recalling a cubic Bézier curve's properties (see Figure 4). Let $\mathbf{P}_0$ and $\mathbf{P}_3$ be two endpoints, and $\mathbf{T}_0$ and $\mathbf{T}_3$ be the corresponding tangent directions with unit length. We want to find two additional control points $\mathbf{P}_1$ and $\mathbf{P}_2$ to construct a cubic Bézier curve, $\mathbf{C}(t)$, $t \in (0,1)$, constrained by $a = |\mathbf{C}'(0)| = |\mathbf{C}'(\frac{1}{2})| = |\mathbf{C}'(1)|$. From Bézier curve properties and the deCasteljau algorithm[2] (Farin, 1990) at $t = \frac{1}{2}$, the following relations hold:

[2]The deCasteljau algorithm is a subdivision algorithm for computing the coordinates of a point on a Bézier curve for a given parameter value.

$$
\begin{aligned}
\mathbf{P}_1 &= \mathbf{P}_0 + \tfrac{1}{3}\alpha\mathbf{T}_0, \\
\mathbf{P}_2 &= \mathbf{P}_3 - \tfrac{1}{3}\alpha\mathbf{T}_3, \\
\mathbf{P}_0^3 &= \mathbf{C}\left(\tfrac{1}{2}\right), \\
\mathbf{P}_1^2 - \mathbf{P}_0^3 &= \tfrac{1}{8}\left(\mathbf{P}_3 + \mathbf{P}_2 - \mathbf{P}_1 - \mathbf{T}_0\right), \text{and} \\
\mathbf{C}'\left(\tfrac{1}{2}\right) &= 6\left(\mathbf{P}_1^2 - \mathbf{P}_0^3\right),
\end{aligned}
\tag{7}
$$

where $\mathbf{P}_1^2$ and $\mathbf{P}_0^3$ are "intermediate deCasteljau points". From these equations, we have

$$
a\alpha^2 + b\alpha + c = 0, \tag{8}
$$

where

$$
\begin{aligned}
a &= 16 - \left|\mathbf{T}_0 + \mathbf{T}_3\right|^2, \\
b &= 12\left(\mathbf{P}_3 - \mathbf{P}_0\right) \bullet \left(\mathbf{T}_3 + \mathbf{T}_0\right), \text{and} \\
c &= -36\left|\mathbf{P}_0 + \mathbf{P}_3\right|^2.
\end{aligned}
$$

Equation 8 has two real solutions for α, only one of which is positive. Substituting this value into equations 7, we can obtain $\mathbf{P}_1$ and $\mathbf{P}_2$.

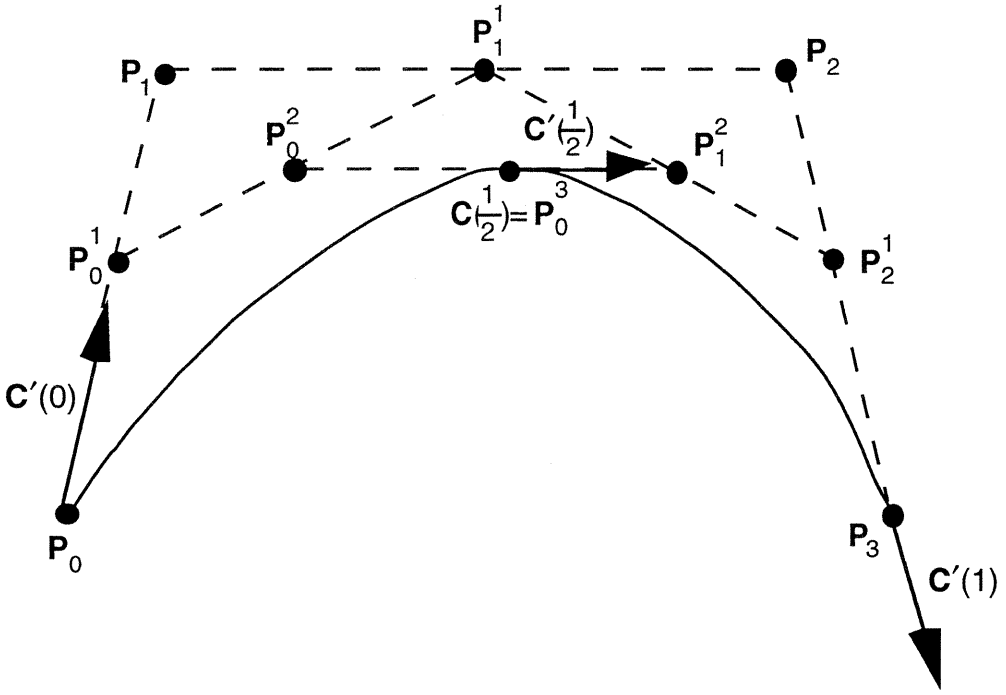


Figure 4. Cubic Bézier control polygon construction.

In general, a cubic curve segment $\mathbf{C}_k(t)$ can be constructed between each pair of data points, $\mathbf{Q}_{k-1}$ and $\mathbf{Q}_k$. The corresponding control polygon consists of the points

$\mathbf{P}_k^0 = \mathbf{Q}_k, \mathbf{P}_k^1, \mathbf{P}_k^2, \mathbf{P}_k^3 = \mathbf{Q}_{k+1}$. The control points $\mathbf{P}_k^1$ and $\mathbf{P}_k^2$ can be calculated just as $\mathbf{P}_1$ and $\mathbf{P}_2$ above. The parametrization is set as $\bar{t}_0 = 0$ and, for a uniform point distribution, $\bar{t}_{k+1} = \bar{t}_k + 3\left|\mathbf{P}_k^1 - \mathbf{P}_k^0\right|$. This yields the knot vector $\mathbf{t} = \left\{0,0,0,0,\frac{\bar{t}_1}{\bar{t}_n},\frac{\bar{t}_1}{\bar{t}_n},\frac{\bar{t}_2}{\bar{t}_n},\frac{\bar{t}_2}{\bar{t}_n},\ldots,\frac{\bar{t}_{n-1}}{\bar{t}_n},\frac{\bar{t}_{n-1}}{\bar{t}_n},1,1,1,1\right\}$. The algorithm produces n Bézier segments, each with speed equal to 1 at the endpoint and midpoints with respect to their parameter ranges. Thus, a C^1 continuous cubic B-spline curve interpolating $\mathbf{Q}_k$ is defined by the control points $\mathbf{Q}_0, \mathbf{P}_0^1, \mathbf{P}_0^2, \mathbf{P}_1^1, \mathbf{P}_1^2, \ldots, \mathbf{P}_{n-2}^2, \mathbf{P}_{n-1}^1, \mathbf{P}_{n-1}^2, \mathbf{Q}_{k+1}$. This scheme is more appropriate for our contour representation because it maintains constant speed and uniform point distribution on part contours. Figure 5(a) shows Wu's example for his optimal control algorithm (Wu, 1992). The outline indicates the developed control polygon. Figures 5(b) and 5(c) are the NURBS representations plotted with 100 and 150 points, respectively, on the contour. Note that NURBS curve maintains straight lines, and the corners maintained by putting close neighboring points at the corner. Figure 6 shows the same contour as Figure 3, but with a cubic NURBS representation. Compared with the results in Figure 3, this algorithm gives a more uniform point distribution.

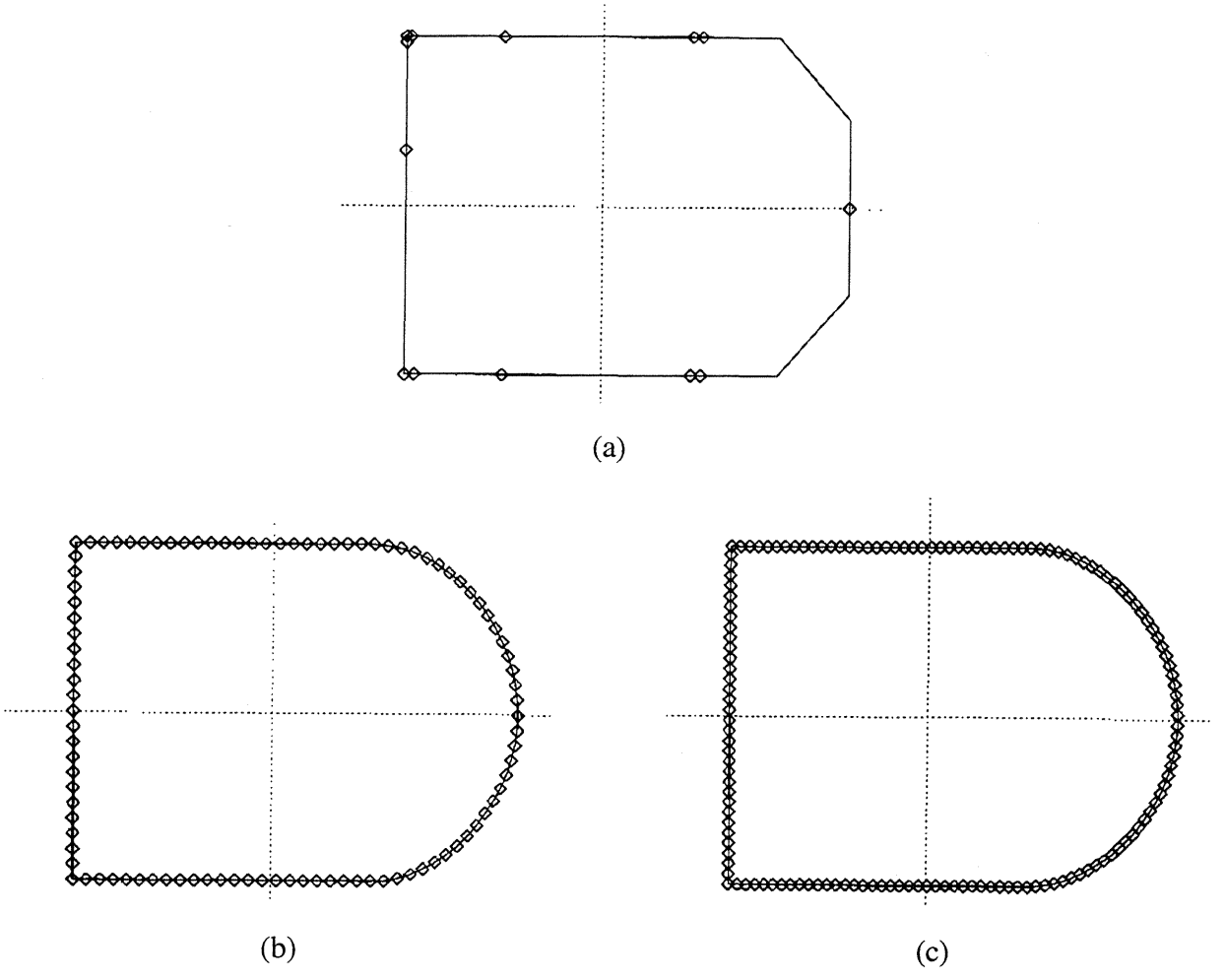
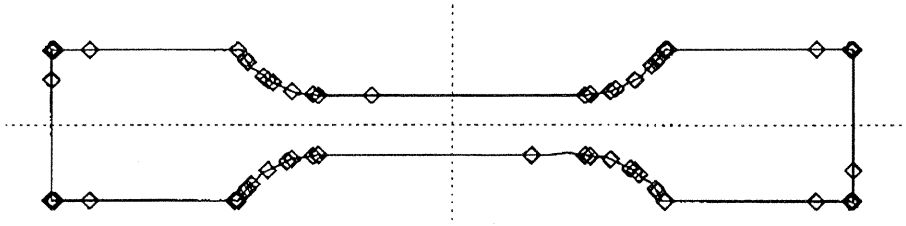
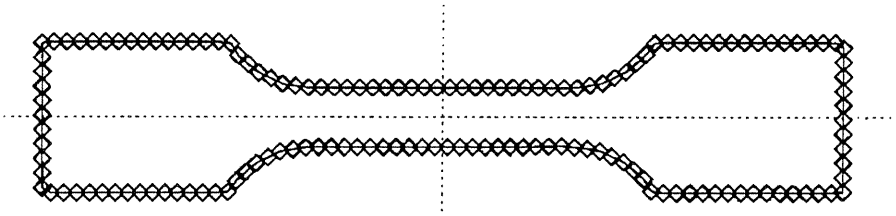


Figure 5. Given data and developed control polygon (a), and NURBS representation by Bézier algorithm with 100 points (b) and 150 points (c).



(a)



(b)

Figure 6. Cubic NURBS representation of the contour from Figure 3.

5 PARAMETRIC REPRESENTATIONS FROM OTHER GEOMETRIC DATA

Guduri *et al.* (Guduri *et al.*, 1992; Guduri, 1993; Guduri *et al.*, 1993) developed an approach to obtain an exact implicit form of part contours as piecewise curve segments with degree less than 4, such as conics, or an approximate implicit form for higher degree curves, such as the intersection curves generated by slicing a torus. The algorithm is based on constructive solid geometry input models. Using the CSG representation, an object is represented as a binary tree with geometric primitives (*e.g.*, spheres, cones, cylinders, parallelepipeds, torii, etc.) at the leaf nodes and regularized Boolean operators at the interior nodes. These primitives are first sliced to get the boundaries curves. These curves can be exactly or approximately represented by implicit second degree curves. Based on this result, the parametric form can be obtained by Abhyankar's method (Abhyankar and Bajaj, 1987). The Boolean operations are then applied to the primitive slices to obtain the part contour.

NURBS provide a single curve form that can exactly represent all Guduri's contours parametrically. NURBS can exactly represent all common curves, including circles, ellipses, parabolas, and straight lines. Additionally, NURBS are not limited in degree. Generally, intersection curves between higher order surfaces can not be analytically obtained. Parametric curve representation techniques can be used to represent higher order intersections and contours to any desired degree of accuracy. Many researchers have focused on the development of accurate and efficient surface-to-surface (or plane) intersection algorithms (Patrikalakis, 1993; Hanna, 1983; Lee, 1984). Generally, such intersection curves cannot be represented by a single analytical equation. Current approaches are based on numerical solutions. The NURBS representation can directly use numerical data to provide one form for modeling contours that consist of several connected intersection curves. Note that the NURBS basis functions are effective for only a few neighboring spans between control points. Also, the amount of data and shape of contours has little effect on the computing performance of NURBS curves.

6 DISCUSSION AND CONCLUSION

As we have been seen, NURBS are generally differentiable up to degree $n-1$ between adjacent knots, and are $n-p$ times continuously differentiable for knots of multiplicity p. Uniform

point distributions are generated by appropriately setting the knot vector. Since arc length parametrization has been developed by Guenter (1990), the NURBS representation satisfies Wu's requirements. Additionally, the NURBS curve has many advantages for curve and surface manipulation, and it provides a flexible and adaptable representation scheme. The NURBS curve form satisfies all the evaluation criteria outlined in the section 3. NURBS can approximate given data points to any degree of accuracy and with various continuity requirements. The NURBS form not only represents the conic sections exactly, but it also has handles the special curve features, such as corners, and straight lines. NURBS algorithms are generally stable and robust. In geometric modeling, the NURBS form provides a common representation for a large variety of curves, with the resulting unified database for its implementation. The only drawbacks of the NURBS form are its extra storage and overhead compared to other parametric forms.

In this paper, we have shown that the NURBS parametric form is appropriate for Wu's control method. NURBS curves also provide a consistent representation for SFF that can handle both point data or higher order CAD data. Successful utilization of different geometric representations by SFF technologies requires the development of algorithms to extract and convert the geometric information correctly. NURBS can be used to represent these geometric features unless they are known or can be detected. Additionally, NURBS provide an approach that can represent higher degree contours to support, for example, the control models based on higher order dynamics.

REFERENCES

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

2. Barsky, B. A., and DeRose, T. D., 1989, "Geometric Continuity of Parametric Curves: Three Equivalent Characterizations", *IEEE Computer Graphics & Applications*, Vol. 9, No. 6, pp. 60-68.

3. Darrah, J., and Wielgus, M., "A New CAD Model Format for SFF Machines?", *Solid Freeform Fabrication Proceedings 1990*, Austin, TX, August 6-8, 1990, pp. 121-125.

4. Farin, G., 1990, *Curves and Surfaces for Computer Aided Geometric Design*, 2nd edition, Academic Press.

5. Guduri S., 1993, *A Method to Generate Contour Files Directly from Higher Level Geometry Representations For SFF*, M.S. thesis, The University of Texas at Austin, Austin, TX.

6. Guduri, S., Crawford, R. H., and Beaman, J. J., 1992, "A Method to Generate Exact Contour Files for Solid Freeform Fabrication," *Proceedings of the 1992 Solid Freeform Fabrication Symposium*, Austin, TX, pp. 95-101, August 3-5, 1992.

7. Guduri, S., Crawford, R. H., and Beaman, J. J., 1993, "Boundary Evaluation for Solid Freeform Fabrication," *Towards World Class Manufacturing 1993*, proceedings of the IFIP TC5 WG5.2/WG5.3 Conference, M. J. Wozny and G. Olling, eds., Litchfield Park, AZ, pp. 301-312, September 12-16, 1993.

8. Guenter B., 1990, "Computing the Arc Length of Parametric Curves", IEEE *Computer Graphics & Applications*, Vol. 10, No. 3, pp. 72-78.

9. Hanna, S. L., *et al.*, 1983, "Intersection of Parametric Surfaces by Means of Look-Up Tables", *IEEE Computer Graphics and Applications*, October 1983, pp. 39-48.

10. Lee, E. T. Y., 1989, "Choosing Nodes in Parametric Curve Interpolation", *Computer-Aided Design*, Vol. 21, pp. 363-370.

11. Lee, R. B., 1984, "Intersection of Parametric Surfaces and a Plane", *IEEE Computer Graphics and Applications*, August 1984, pp. 48-51.

12. Patrikalakis, N. M., 1993, "Surface-to-Surface Intersections", *IEEE Computer Graphics and Applications*, January 1993, pp. 89-94.

13. Piegl, L., 1991 "On NURBS: A Survey", *IEEE Computer Graphics and Applications*, Vol. 10, No. 1, pp. 55-71.

14. Piegl, L., and Tiller, W., 1995, *A NURBS Book*, Springer-Verlag, Heidelberg, Germany.

15. Rogers, D. F., and Adams, J. A., 1990, *Mathematical Elements for Computer Graphics*, 2nd ed., McGraw-Hill, New York.

16. Wu, Y-J. E., and Beaman, J. J., "Solid Freeform Fabrication Laser Tracking Control", *Solid Freeform Fabrication Proceedings 1991*, Austin, TX, August 12-14, 1991, pp. 37-45.

17. Wu, Y-J. E., and Beaman, J. J., 1990, "Contour Following for Scanning Control in SFF Application: Control Trajectory Planning", *Solid Freeform Fabrication Proceedings 1990*, Austin, TX, August 6-8, 1990, pp. 126-134.

18. Wu, Y-J. E., and Beaman, J. J., 1992, "Laser Tracking Control Implementation for SFF Applications", *Solid Freeform Fabrication Proceedings 1992*, Austin, TX, August 3-5, 1992, pp. 161-173.

19. Wu, Y.-J., 1992, *A Minimum Time Laser Tracking Control Technique For Selective Laser Sintering*, Ph.D. Thesis, The University of Texas at Austin, Austin, TX.