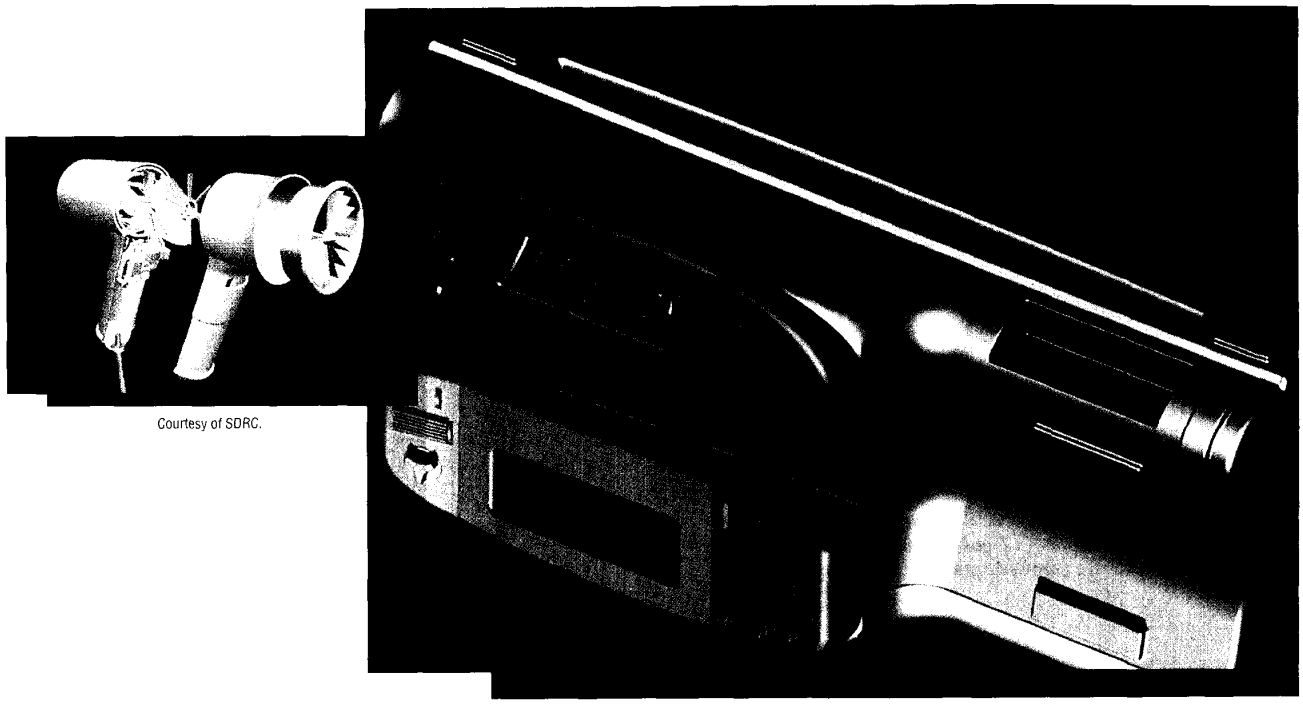

Courtesy of SDRC.

Courtesy of Intergraph.

On NURBS: A Survey

Les Piegl
University of South Florida

This survey summarizes all the important characteristics of NURBS that contributed to their wide acceptance as standard tools for geometry representation and design.

Tracking down the roots of NURBS as used in CAD and graphics is not easy. However, if we consider the two major ingredients of NURBS—rational and B-splines—then we can begin an overview with Coons' "little red book."[1] Coons suggested the use of rational polynomials to represent conic sections precisely. Forrest pursued the ideas further and gave a rigorous treatment of rational conics and cubics.[2] On the basis of theoretical works by Schoenberg, De Boor,[3] Cox, and Mansfield, Riesenfeld introduced B-spline curves and surfaces into CAD/CAM and graphics.[4] Following Riesenfeld, Versprille extended B-splines to rational B-splines. His work in 1975 was the first written account of NURBS.[5] By the late 1970s, the CAD/CAM industry recognized the need for a modeler that had a common internal method of representing and storing different geometric entities. At about the same time, three major groups looked at the possibility of using NURBS.

Boeing began developing the Tiger system in 1979. Integrating B-splines[6-8] with rational Bezier representations[9,10] quickly led to rational B-splines (Figure 1). Boeing felt so strongly about NURBS that they proposed them as part of the standard to the August 1981 International Graphics Exchange

0272-17-16/91/0100-0055$01.00 ©1991 IEEE

Courtesy of R.M. Blomgren.

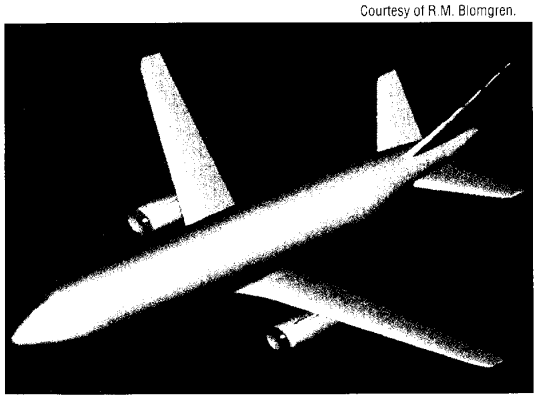

Figure 1. An early NURBS model of an airplane generated by Boeing's Tiger system. Loren Carpenter developed the z-buffer rendering software.

Courtesy of SDRC.

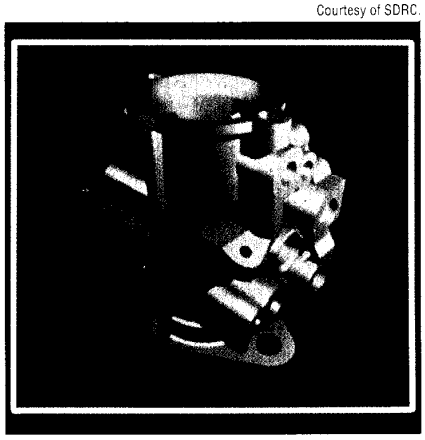

Figure 2. Carburetor housing generated by SDRC's solid modeler I-deas/Geomod.

Courtesy of University of Utah.

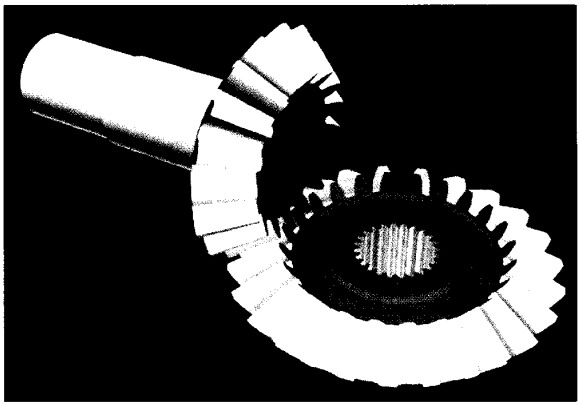

Figure 3. Bevel gears generated by Alpha_1.

Standard meeting.[11] Despite the success of NURBS and the great deal of work put into its development, Boeing abandoned Tiger in 1984.

SDRC (Structural Dynamics Research Corporation) pursued NURBS commercially. In 1978, the company started working on a modeler. Following Versprille, SDRC decided to use NURBS as a single representation form. Progress was announced publicly in 1982,[12-15] and the modeler, called Geomod, was released in 1983 (Figure 2). It was the first commercial modeler based entirely on NURBS.

B-splines have been the subject of much work at the University of Utah. After several years of research, Riesenfeld and his group put their research results into a modeler called Alpha_1[16] (Figure 3). For many years, Alpha_1 has served as a research environment, but recently, Engineering Geometry Systems made a commercial version available.

The above groups tremendously influenced the development of NURBS technology. Many companies followed their paths. Intergraph Corporation started with Bezier in their Bsurf modeler in 1982 and incorporated nonuniform B-splines and NURBS in 1984. In 1985 they started to develop a new system called I/EMS based entirely on NURBS.[17]

The rapid proliferation of NURBS is due partly to their excellent properties and partly to their incorporation in such national and international standards as IGES,[18] PHIGS+,[19] Product Data Exchange Specification, and International Standard Office Standard for the Exchange of Product Model Data.

Why NURBS?

Some reasons for the widespread acceptance and popularity of NURBS in the CAD/CAM and graphics community are as follows:

- They offer a common mathematical form for representing and designing both standard analytic shapes (conics, quadrics, surfaces of revolution, etc.) and free-form curves and surfaces. Therefore, both analytic and free-form shapes are represented precisely, and a unified database can store both.
- By manipulating the control points as well as the weights, NURBS provide the flexibility to design a large variety of shapes.
- Evaluation is reasonably fast and computationally stable.
- NURBS have clear geometric interpretations, making them particularly useful for designers, who have a very good knowledge of geometry—especially descriptive geometry.
- NURBS have a powerful geometric tool kit (knot insertion/refinement/removal, degree elevation, splitting, etc.), which can be used throughout to design, analyze, process, and interrogate objects.
- NURBS are invariant under scaling, rotation, translation, and shear as well as parallel and perspective projection.

- NURBS are genuine generalizations of nonrational B-spline forms as well as rational and nonrational Bezier curves and surfaces.

However, NURBS have several drawbacks:

- Extra storage is needed to define traditional curves and surfaces. For example, to represent the full circle using a circumscribing square requires seven control points and 10 knots. Traditional representation requires the center, the radius, and the normal vector to the plane of the circle. In 3D, this amounts to storing 38 instead of seven numbers.
- Improper application of the weights can result in a very bad parameterization, which can destroy subsequent surface constructions.
- Some interrogation techniques work better with traditional forms than with NURBS. An example is surface/surface intersection, where it is particularly difficult to handle the "just touch" or "overlap" cases.
- Fundamental algorithms, such as inverse point mapping, are subject to numerical instability.

Before you start feeling discouraged about NURBS, let me say that the above problems are *not* peculiar to NURBS. Other free-form schemes, such as those of Bezier, Coons, and Gordon, exhibit the same problems.

What are NURBS?

The mathematical definitions of NURBS curves and surfaces are relatively simple.[5,15,20-22] A NURBS curve is a vector-valued piecewise rational polynomial function of the form

$$\mathbf{C}(u) = \frac{\sum_{i=0}^{n} w_i \mathbf{P}_i N_{i,p}(u)}{\sum_{i=0}^{n} w_i N_{i,p}(u)} \tag{1}$$

where the w_i are the so-called weights, the $\mathbf{P}_i$ are the control points (just as in the case of nonrational curves), and $N_{i,p}(u)$ are the normalized B-spline basis functions of degree p defined recursively as[23,24]

$$N_{i,0}(u) = \begin{cases} 1 & \text{if } u_i \le u < u_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

$$N_{i,p}(u) = \frac{u - u_i}{u_{i+p} - u_i} N_{i,p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(u) \tag{2}$$

where u_i are the so-called knots forming a knot vector

$$\mathbf{U} = \{u_0, u_1, \ldots, u_m\} \tag{3}$$

The degree, number of knots, and number of control points are related by the formula $m = n + p + 1$. For nonuniform and nonperiodic B-splines, the knot vector takes the form

$$\mathbf{U} = \{\alpha, \alpha, \ldots, \alpha, u_{p+1}, \ldots, u_{m-p-1}, \beta, \beta, \ldots, \beta\} \tag{4}$$

where the end knots α and β are repeated with multiplicity $p + 1$. In most practical applications $\alpha = 0$ and $\beta = 1$, as is assumed throughout this article. The basis functions (Equation 2) are defined over the entire line; however, the focus is on the interval [0,1]. The NURBS curve (Equation 1) with the knot vector (Equation 4) is a Bezier-like curve. It interpolates the endpoints and is tangential at the endpoints to the first and last legs of the control polygon. Most properties of nonrational curves apply to NURBS as well. Some details follow later.

A NURBS surface is the rational generalization of the tensor-product nonrational B-spline surface and is defined as follows:[15,20-22]

$$\mathbf{S}(u, v) = \frac{\sum_{i=0}^{n} \sum_{j=0}^{m} w_{i,j} \mathbf{P}_{i,j} N_{i,p}(u) N_{j,q}(v)}{\sum_{i=0}^{n} \sum_{j=0}^{m} w_{i,j} N_{i,p}(u) N_{j,q}(v)} \tag{5}$$

where $w_{i,j}$ are the weights, $\mathbf{P}_{i,j}$ form a control net, and $N_{i,p}(u)$ and $N_{j,q}(v)$ are the normalized B-splines of degree p and q in the u and v directions, respectively, defined over the knot vectors

$$\mathbf{U} = \{0, 0, \ldots, 0, u_{p+1}, \ldots, u_{r-p-1}, 1, 1, \ldots, 1\} \tag{6a}$$

$$\mathbf{V} = \{0, 0, \ldots, 0, v_{q+1}, \ldots, u_{s-q-1}, 1, 1, \ldots, 1\} \tag{6b}$$

where the end knots are repeated with multiplicities $p + 1$ and $q + 1$, respectively, and $r = n + p + 1$ and $s = m + q + 1$. Although the surface (Equation 5) was obtained by generalizing the tensor-product surface form, a NURBS surface is, in general, not a tensor-product surface.

Analytic and geometric properties

The curve form (Equation 1) can be rewritten into the following equivalent form:

$$\mathbf{C}(u) = \sum_{i=0}^{n} \mathbf{P}_i R_{i,p}(u), \qquad R_{i,p}(u) = \frac{w_i N_{i,p}(u)}{\sum_{j=0}^{n} w_j N_{j,p}(u)} \tag{7}$$

where $R_{i,p}(u)$ are *rational basis functions*. Their analytic properties determine the geometric behavior of curves. The most significant properties are[20,22]

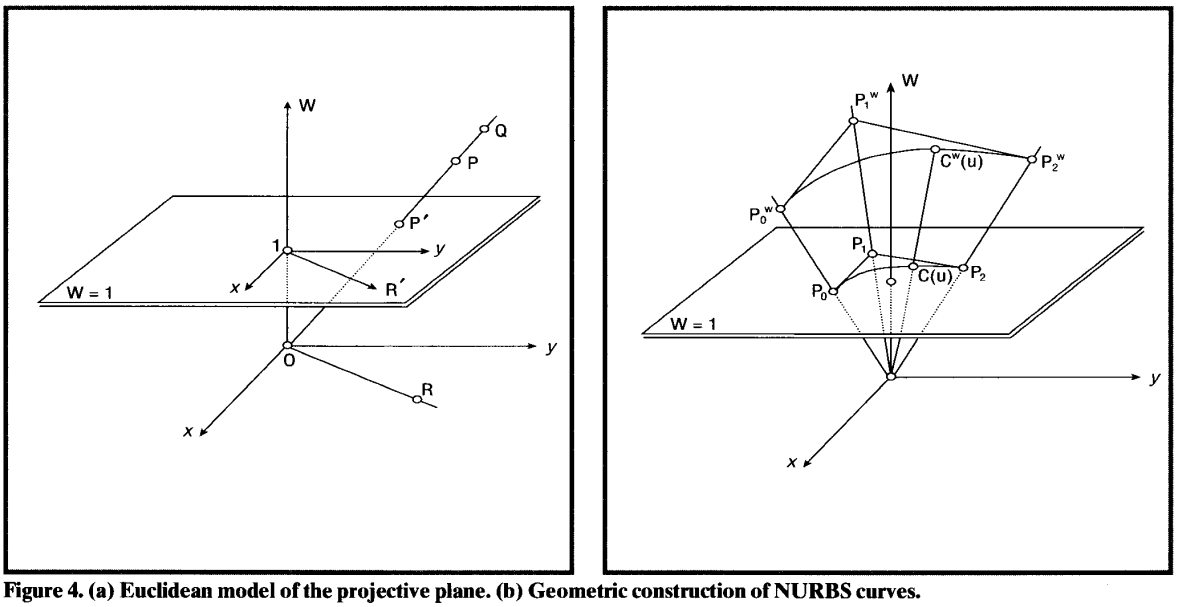

Figure 4. (a) Euclidean model of the projective plane. (b) Geometric construction of NURBS curves.

- Generalization: If all the weights are set to 1, then

$$R_{i,p}(u) = \begin{cases} B_{i,p}(u) & \text{if } \mathbf{U} = \{0, 0, \ldots, 0, 1, 1, \ldots, 1\} \\ N_{i,p}(u) & \text{otherwise} \end{cases}$$

where the 0's and 1's in **U** are repeated with multiplicity $p + 1$, and $B_{i,p}(u)$ denote the Bernstein polynomials of degree p.

- Locality: $R_{i,p}(u) = 0$ if $u \notin [u_i, u_{i+p+1})$

- Partition of unity: $\sum_i R_{i,p}(u) = 1$

- Differentiability: In the interior of a knot span, the rational basis functions are infinitely continuously differentiable if the denominator is bounded away from zero. At a knot they are p– k times continuously differentiable where k is the multiplicity of the knot.

- $R_{i,p}(u;\ w_i = 0) = 0$
- $R_{i,p}(u;\ w_i \to +\infty) = 1$
- $R_{i,p}(u;\ w_j \to +\infty) = 0 \qquad j \neq i$

As a consequence, the NURBS curve will exhibit the following geometric characteristics:

- Bezier and nonrational B-spline curves are special cases.
- Local approximation: If a control point is moved *or* a weight is changed, it will affect the curve only in $p + 1$ knot spans.
- Strong convex hull property: if $u \notin [u_i, u_{i+1})$, then $\mathbf{C}(u)$ lies within the convex hull of $\mathbf{P}_{i-p}, \ldots, \mathbf{P}_i$.
- Invariance under affine and perspective transformations (see the details below).
- The same differentiability property as with the basis functions.
- If a particular weight is set zero, then the corresponding control point has no effect at all on the curve.
- If $w_i \to +\infty$, then $\mathbf{C}(u) = \begin{cases} \mathbf{P}_i & \text{if } u \in (u_i, u_{i+p+1}) \\ \mathbf{C}(u) & \text{otherwise} \end{cases}$

NURBS surfaces can be analyzed similarly using the bivariate rational basis functions

$$R_{i,p;j,q}(u,v) = \frac{w_{i,j} N_{i,p}(u) N_{j,q}(v)}{\sum_{r=0}^{n} \sum_{s=0}^{m} w_{r,s} N_{r,p}(u) N_{s,q}(v)} \qquad \textbf{(8)}$$

Unfortunately, space restrictions do not allow us to pursue a detailed discussion of surfaces.

What, homogeneous coordinates?

In this section I give a geometric definition of NURBS by using a model that embeds the projective n space in Euclidean $(n + 1)$ space. As an example, let us see how the projective plane can be embedded in Euclidean 3D space. Denote the coordinate axes of the Euclidean space by X, Y, and W and let x,y be another coordinate system where x is parallel to X, y is

parallel to Y, and the origin lies at $(X, Y, W) = (0, 0, 1)$ (Figure 4a). Now any point $\mathbf{P}'$ in the projective plane determines a line $\mathbf{OP}'$, and every line passing through $\mathbf{O}$ and not lying on the X,Y plane determines a point in this plane. The line $\mathbf{OP}'$ can be defined by any point $\mathbf{P}$ that lies on this line. The coordinates $(X\mathbf{P}, Y\mathbf{P}, W\mathbf{P})$ of $\mathbf{P}$ are called the *homogeneous coordinates* of $\mathbf{P}'$.[25-28] Obviously, the position of $\mathbf{P}$ along $\mathbf{OP}'$ is completely arbitrary as long as $\mathbf{P}$ differs from $\mathbf{O}$. That is, if $\mathbf{P}$ and $\mathbf{Q}$ are two different points along $\mathbf{OP}'$, then their coordinates are both the homogeneous coordinates of $\mathbf{P}'$ (what mathematicians call an *equivalence class*). In other words, $(X\mathbf{P}, Y\mathbf{P}, W\mathbf{P})$ and $(X\mathbf{Q}, Y\mathbf{Q}, W\mathbf{Q}) = (\mu X\mathbf{P}, \mu Y\mathbf{P}, \mu W\mathbf{P}), \mu \neq 0$, are the homogeneous coordinates of the same point. Since $\mathbf{O}$ does not correspond to any point in the projective plane, the triple $(0, 0, 0)$ does not represent any point.

Two special cases are worth mentioning. The first is when $\mathbf{P} = \mathbf{P}'$. In this case $\mathbf{P}' = (x, y, 1)$; that is, the ordinary coordinates can be obtained by simple divisions: $x = X/W$ and $y = Y/W$. Second, if $\mathbf{R}$ lies on the plane X, Y, then $\mathbf{OR}$ does not intersect the x,y plane. The corresponding projective "point" $\mathbf{R}'$ is represented by a direction and is called a *point at infinity*. Since $\mathbf{R} = (X, Y, 0)$, each triple with a zero third coordinate represents an infinite point.

We can now borrow the above scheme to construct a geometric model of NURBS. For simplicity we consider planar curves only (Figure 4b) (space curves and surfaces are handled analogously). Each point in the X,Y,W coordinate system can be written as (xw, yw, w) if $w \neq 0$ or as $(x, y, 0)$ if $w = 0$ and can be mapped onto the x,y plane by a perspective map

$$\varphi(X,Y,W) = \begin{cases} \left(\frac{X}{W}, \frac{Y}{W}\right) & \text{if } W \neq 0 \\ \text{direction } (X,Y) & \text{if } W = 0 \end{cases} \tag{9}$$

Now, if we are given a set of control points along with the corresponding weights, then we take the following steps:

1. Construct the weighted vertices:

$$\mathbf{P}_i^w = (w_i x_i, w_i y_i, w_i) \quad i = 0, \ldots, n \tag{10}$$

2. Obtain a nonrational B-spline curve in the X,Y,W coordinate system:

$$\mathbf{C}^w(u) = \sum_{i=0}^{n} \mathbf{P}_i^w N_{i,p}(u) \tag{11}$$

3. Map the curve onto the x,y plane:

$$\mathbf{C}(u) = \varphi\left(\mathbf{C}^w(u)\right) = \frac{\sum_{i=0}^{n} w_i \mathbf{P}_i N_{i,p}(u)}{\sum_{i=0}^{n} w_i N_{i,p}(u)} \tag{12}$$

In the above discussion a borrowed model of the projective plane gives a geometric definition of NURBS. Do not confuse the two models: The first gives a 3D Euclidean representation of the projective plane, whereas the second provides a 3D representation of NURBS curves that lie in the 2D Euclidean plane.

The 3D model is useful not only for geometric insight into NURBS, but also for obtaining efficient computational algorithms. For example, we can evaluate a NURBS curve in 3D using De Boor's corner-cutting algorithm,[3,29] then locate the result in two dimensions.

Affine and perspective invariance

A general affine transformation is a linear transformation (scaling, rotation, shearing, etc.) followed by a translation. More precisely, $A[\mathbf{P}] = L[\mathbf{P}] + T$, where $\mathbf{P}$ denotes a general point. It is easy to show that NURBS are invariant under affine transformations:

$$A[\mathbf{C}(u)] = L[\mathbf{C}(u)] + T = \sum_i L[\mathbf{P}_i] R_{i,p}(u) + T$$

On the other hand,

$$\begin{aligned} \sum_i A[\mathbf{P}_i] R_{i,p}(u) &= \sum_i (L[\mathbf{P}_i] + T) R_{i,p}(u) \\ &= \sum_i L[\mathbf{P}_i] R_{i,p}(u) + T \sum_i R_{i,p}(u) \\ &= \sum_i L[\mathbf{P}_i] R_{i,p}(u) + T \end{aligned}$$

since the rational basis functions sum to 1. Combining these two equations, we have

$$A[\mathbf{C}(u)] = \sum_i A[\mathbf{P}_i] R_{i,p}(u)$$

That is, obtain the affine image of a NURBS curve by transforming the control points and *leaving the weights unchanged.* For example, parallel projections of NURBS are obtained by projecting the control points.

Now consider perspective projection. If the center of the projection is denoted by $\mathbf{C}$ and the perspective plane is given by the point $\mathbf{Q}$ and by the normal vector $\mathbf{N}$, then, following Lee,[30] the projection of a point $\mathbf{X}$ is

$$\pi(\mathbf{X}) = (1 - \alpha)\mathbf{X} + \alpha\mathbf{C}, \quad \alpha = \frac{(\mathbf{X} - \mathbf{Q}) \cdot \mathbf{N}}{(\mathbf{X} - \mathbf{C}) \cdot \mathbf{N}} \tag{13}$$

The projection of a NURBS curve is obtained as follows:

$$\pi(\mathbf{C}(u)) = \frac{\sum_{i=0}^{n} \overline{w}_i \pi(\mathbf{P}_i) N_{i,p}(u)}{\sum_{i=0}^{n} \overline{w}_i N_{i,p}(u)} \tag{14}$$

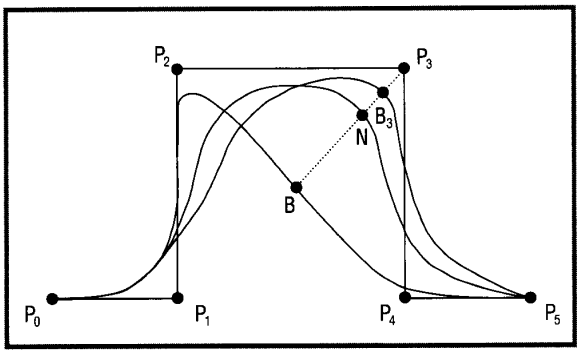

Figure 5. The geometric meaning of w_3.

Here $\pi(\mathbf{P}_i)$ denotes the projection of the control point, and

$$\overline{w}_i = w_i\,(\mathbf{P}_i - \mathbf{C}) \cdot \mathbf{N} \tag{15}$$

Projections of NURBS surfaces are obtained analogously.

More about the weights

At first glance, weights look like pure numbers carrying no geometric meaning whatsoever. Fortunately, we can associate a nice geometric meaning to them. Assume that we are looking for the geometric meaning of w_i. Since it affects the curve only in $[u_i, u_{i+p+1})$, investigations are restricted to this span. Furthermore, we assume for the time being that only w_i changes. Define the following points (see Figure 5):[31-33]

$$\begin{aligned} \mathbf{B} &= \mathbf{C}(u;\, w_i = 0) \\ \mathbf{N} &= \mathbf{C}(u;\, w_i = 1) \\ \mathbf{B}_i &= \mathbf{C}(u;\, w_i \neq \{0,1\}) \end{aligned} \tag{16}$$

Using the parameters

$$\begin{aligned} \alpha &= R_{i,p}(u;\, w_i = 1) \\ \beta &= R_{i,p}(u) \end{aligned} \tag{17}$$

$\mathbf{N}$ and $\mathbf{B}_i$ can be expressed as

$$\begin{aligned} \mathbf{N} &= (1-\alpha)\mathbf{B} + \alpha\mathbf{P}_i \\ \mathbf{B}_i &= (1-\beta)\mathbf{B} + \beta\mathbf{P}_i \end{aligned} \tag{18}$$

Using the expressions of α and β, we obtain the following identity

$$\frac{1-\alpha}{\alpha} : \frac{1-\beta}{\beta} = \frac{\mathbf{P}_i\mathbf{N}}{\mathbf{BN}} : \frac{\mathbf{P}_i\mathbf{B}_i}{\mathbf{BB}_i} = w_i \tag{19}$$

This is called the *cross-ratio* or *double ratio* of the four points $\mathbf{P}_i$, $\mathbf{B}$, $\mathbf{N}$, and $\mathbf{B}_i$. Now, using Equations 17 through 19, we can easily analyze the effects of shape modification:

- If w_i increases/decreases, then β increases/decreases, and so the curve is pulled/pushed toward/away from $\mathbf{P}_i$.
- If w_i increases/decreases, then the curve is pushed/pulled away from/toward $\mathbf{P}_j, j \neq i$.
- As $\mathbf{B}_i$ moves, it sweeps out a straight line segment.
- As $\mathbf{B}_i$ tends to $\mathbf{P}_i$, β approaches 1 and thus w_i tends to infinity.

Surface weights are analyzed quite similarly. Because of space limitations, that discussion—available elsewhere[31,33]—is omitted here.

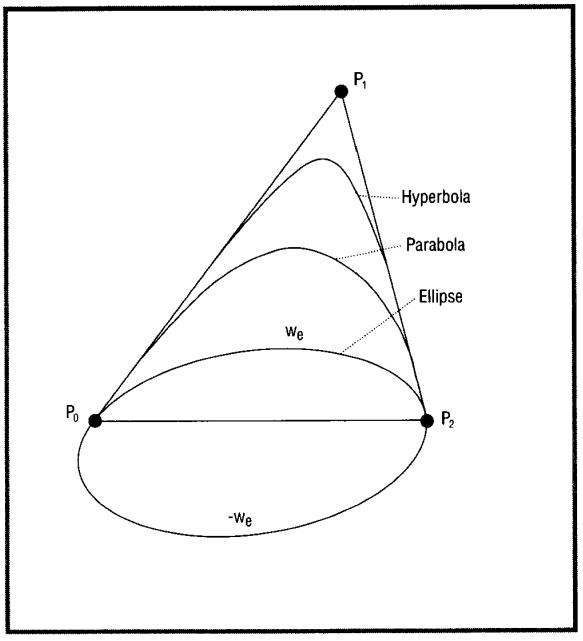

Figure 6. Conic sections defined as single rational Bezier segments.

Conic sections

Conic sections are among the most important curves in CAD/CAM and graphics. A significant reason for using NURBS is their ability to precisely represent conic segments as well as full conics. We start with the representation of one segment whose end tangents are not parallel. Since the conic is a quadratic curve, we try to represent it as a quadratic NURBS:

$$\mathbf{C}(u) = \sum_{i=0}^{2} \mathbf{P}_i\, R_{i,p}(u) \tag{20}$$

where the rational basis functions are defined over the knot vector $\mathbf{U} = \{0, 0, 0, 1, 1, 1\}$. These basis functions are in fact those of rational Bezier curves. Thus the equation of the quadratic NURBS reduces to

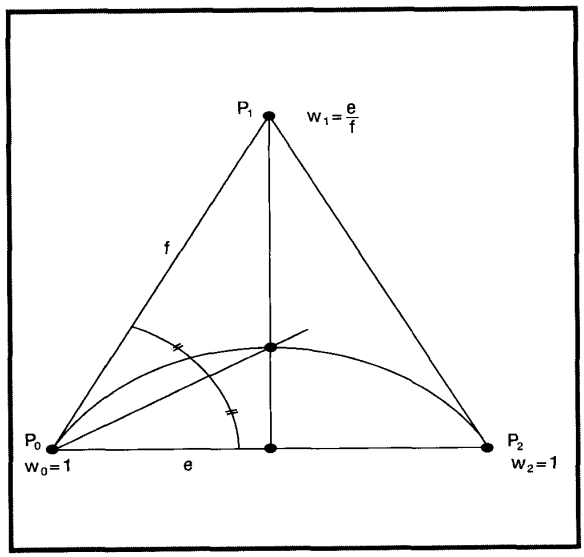

Figure 7. The definition of circular arcs sweeping less than 180 degrees.

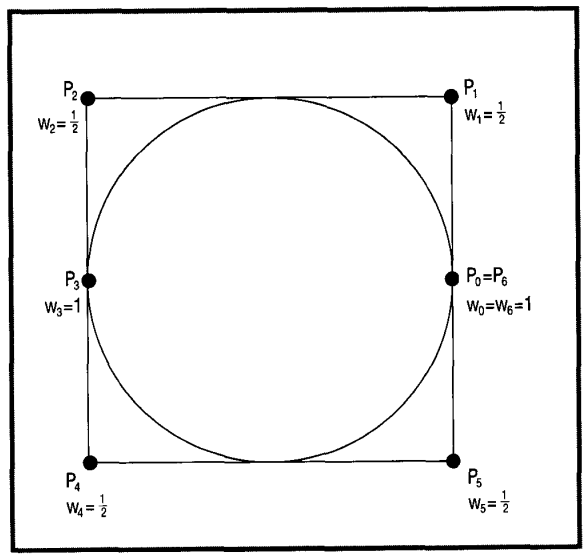

Figure 8. The seven-control-point square-based NURBS circle.

$$\mathbf{C}(u) = \frac{(1-u)^2 w_0 \mathbf{P}_0 + 2u(1-u)w_1\mathbf{P}_1 + u^2 w_2 \mathbf{P}_2}{(1-u)^2 w_0 + 2u(1-u)w_1 + u^2 w_2} \quad (21)$$

We can prove that Equation 21 is the equation of a conic curve and that the ratio

$$\frac{w_1^2}{w_0 w_2} = \text{CSF} \quad (22)$$

is constant for a particular segment.[2,5,30,34] This ratio is called the *conic shape factor*. The value of CSF—not the individual values of the weights—determines a particular conic. More precisely (see Figure 6),

CSF < 1 → ellipse
CSF = 1 → parabola
CSF > 1 → hyperbola

In many applications the end weights are set to 1 and the middle weight is used to describe a family of curves. This choice is particularly useful for obtaining a circular arc. The requirements for the circular arc are (see Figure 7)[2,5,20,30]

- $\mathbf{P}_0\,\mathbf{P}_1\mathbf{P}_2$ must be isosceles.
- If $w_0 = w_2 = 1$,

  then $w_1 = \dfrac{|\mathbf{P}_0 - \mathbf{P}_2|}{2|\mathbf{P}_1 - \mathbf{P}_0|} = \dfrac{e}{f}$,

  else $\text{CSF} = \dfrac{e^2}{f^2}$

During the construction of conic segments, two special cases can occur:

1. The segment is a semiellipse (circle); that is, the end tangent vectors are parallel.
2. The segment lies outside the control triangle.

Semiellipses are conveniently defined by infinite control points.[35] The curve form is

$$\mathbf{C}(u) = \frac{(1-u)^2\mathbf{P}_0 + u^2\mathbf{P}_2}{(1-u)^2 + u^2} + \frac{2u(1-u)\mathbf{V}}{(1-u)^2 + u^2} \quad (23)$$

where $\mathbf{V}$ is a direction vector parallel to the end tangent vectors. To avoid the use of a direction vector, we can insert knots to obtain "regular" control points.[36]

If the arc is elliptical and lies outside the control triangle, then we can represent it as a complementary arc, using a negative weight (see Figure 6). Inserting a knot at $u = \frac{1}{2}$ removes the negative weight and creates a new control polygon that contains the arc in its convex hull.[36]

We can piece segments together to obtain full conic curves. For example, the full circle is composed of four segments, each sweeping 90 degrees (see Figure 8).[37] The circle has the representation

$$\mathbf{C}(u) = \sum_{i=0}^{6} \mathbf{P}_i R_{i,p}(u) \quad (24)$$

where the control points form a square, and

$$\mathbf{U} = \{0, 0, 0, \tfrac{1}{4}, \tfrac{1}{2}, \tfrac{1}{2}, \tfrac{3}{4}, 1, 1, 1\}$$
$$w_i|_{i=0}^{6} = \{1, \tfrac{1}{2}, \tfrac{1}{2}, 1, \tfrac{1}{2}, \tfrac{1}{2}, 1\} \quad (25)$$

The full ellipse is obtained from the circle by an affine transformation. Since the affine map of a NURBS curve is obtained by transforming the control points and leaving the weights unchanged, the ellipse is described by a circumscribing rectangle using the above knot vector and weights.[36]

The shape invariance factor of one Bezier segment can be generalized to quadratic NURBS. The segment contained in the convex hull of $\mathbf{P}_{i-1}$, $\mathbf{P}_i$, and $\mathbf{P}_{i+1}$ is tangential to the legs at

$$\mathbf{Q}_0 = \frac{(u_{i+2} - u_{i+1})w_{i-1}\mathbf{P}_{i-1} + (u_{i+1} - u_i)w_i \mathbf{P}_i}{(u_{i+2} - u_{i+1})w_{i-1} + (u_{i+1} - u_i)w_i} \tag{26}$$

$$\mathbf{Q}_1 = \frac{(u_{i+3} - u_{i+2})w_i \mathbf{P}_i + (u_{i+2} - u_{i+1})w_{i+1}\mathbf{P}_{i+1}}{(u_{i+3} - u_{i+2})w_i + (u_{i+2} - u_{i+1})w_{i+1}}$$

and therefore it is represented by a Bezier curve segment with control points $\mathbf{Q}_0$, $\mathbf{P}_i$, and $\mathbf{Q}_1$ and with weights w_0^Q, w_i, and w_1^Q, where

$$w_0^Q = \frac{(u_{i+2} - u_{i+1})w_{i-1} + (u_{i+1} - u_i)w_i}{(u_{i+2} - u_i)}$$

$$w_1^Q = \frac{(u_{i+3} - u_{i+2})w_i + (u_{i+2} - u_{i+1})w_{i+1}}{(u_{i+3} - u_{i+1})} \tag{27}$$

The Bezier curve is defined over $[u_{i+1}, u_{i+2}]$. Since the shape of this curve is independent of the parameterization, we use the conic shape invariance formula to yield Equation 28, shown in Figure 9.

Here Equation 30 is defined over the knot vectors **U** and **V**, where **U** is the knot vector of Equation 29, **V** = {0, 0, 1, 1}, and

$$\mathbf{P}_{i,0} = \mathbf{Q}_i$$
$$\mathbf{P}_{i,1} = \mathbf{Q}_i + d \cdot \mathbf{W} \tag{31}$$

Natural quadrics

The natural quadrics are the plane, cylinder, cone, and sphere. The planar surface patch is described by a bilinear NURBS surface whose control points are the corners of the planar patch. A cylindrical patch or the full trimmed cylinder is obtained by extruding a circular arc or a full circle. The cone is a special case of the cylinder, achieved by degenerating one of the boundary curves in the u direction. The sphere, being a surface of revolution, is discussed below.

General quadrics

Quadrics of revolution are the most commonly used quadrics. For example, the hyperboloid of one sheet is obtained by rotating a hyperbolic arc around an axis. The NURBS representation of these surfaces is obtained by using the technique for surfaces of revolution discussed below. Nonrotationally-symmetric quadrics can be obtained by applying affine transformations to the rotationally symmetric ones. For example, applying shear transformations to the control points of the sphere results in a general ellipsoid. For a detailed discussion of quadric patches see Hildebrand.[38,39]

$$\text{CSF} = \frac{(u_{i+3} - u_{i+1})(u_{i+2} - u_i)w_i^2}{[(u_{i+2} - u_{i+1})w_{i-1} + (u_{i+1} - u_i)w_i]\,[(u_{i+3} - u_{i+2})w_i + (u_{i+2} - u_{i+1})w_{i+1}]}$$

Figure 9. Equation 28.

Commonly used surfaces

The most commonly used surfaces are extruded surfaces, natural quadrics, general quadrics, ruled surfaces, and surfaces of revolution. Other types of surfaces such as swept surfaces are covered in the "Surface design" section below.

Extruded surfaces

Extruded surfaces are obtained by creating a profile curve and by extruding it in a certain direction **W** for a given distance d. If the profile curve is given as

$$\mathbf{C}(u) = \sum_{i=0}^{n} \mathbf{Q}_i R_{i,p}(u) \tag{29}$$

then the extruded surface's equation takes the form

$$\mathbf{S}(u, v) = \sum_{i=0}^{n} \sum_{j=0}^{1} \mathbf{P}_{i,j} R_{i,p;j,1}(u, v) \tag{30}$$

Ruled surfaces

Given two general NURBS curves,

$$\mathbf{C}_i(u) = \sum_{j=0}^{n_i} \mathbf{P}_j^i R_{j,p_i}(u), \quad i = 1, 2 \tag{32}$$

defined over the knot vectors $\mathbf{U}_1$ and $\mathbf{U}_2$, we want a ruled surface

$$\mathbf{S}(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{1} \mathbf{P}_{i,j} R_{i,p;j,1}(u, v) \tag{33}$$

such that $\mathbf{S}(u, 0) = \mathbf{C}_1(u)$ and $\mathbf{S}(u, 1) = \mathbf{C}_2(u)$. This representation assumes that the two curves have the same degree and are defined over the same knot vector. Therefore, we need to do the following:

- If the degrees differ, elevate the degree of the lower order curve.[40,41]

Courtesy of SDRC.

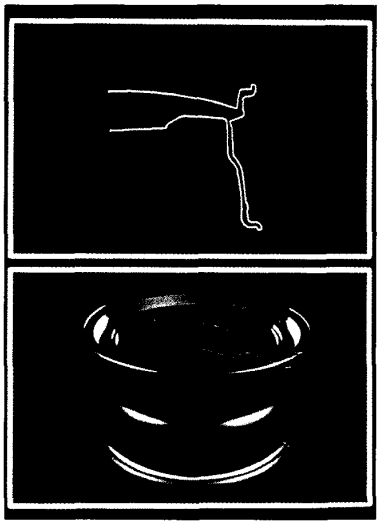

Figure 10. Surface of revolution.

• If the knot vectors differ, then merge the two knot vectors.[42,43]

Surfaces of revolution

Surfaces of revolution are probably the most frequently used surfaces in engineering design and graphics. The most convenient way to define such surfaces is to define a profile curve in the x,z plane, say, and rotate it around the z axis. Assume that the profile curve has the form

$$\mathbf{C}(u) = \sum_{j=0}^{m} \mathbf{Q}_j R_{j,q}(v) \tag{34}$$

Then the surface of revolution (see Figure 10) is obtained by combining Equation 34 with one of the circle definitions. For example, a full surface of revolution is given by

$$\mathbf{S}(u, v) = \sum_{i=0}^{6} \sum_{j=0}^{m} \mathbf{P}_{i,j} R_{i,2;\, j,q}(u, v) \tag{35}$$

where for fixed j, $\mathbf{P}_{i,j}$ lie in a plane perpendicular to the z axis and form a square with center on the z axis. For fixed j the weights are

$$w_{i,j} = \left\{w_j, \frac{1}{2} w_j, \frac{1}{2} w_j, w_j, \frac{1}{2} w_j, \frac{1}{2} w_j, w_j\right\}, \quad i = 0, ..., 6 \tag{36}$$

where w_j are the weights for the profile curve (Equation 34). Common surfaces such as the sphere or torus are obtained by rotating a full circle or a semicircle around an axis. This sphere representation results in two "poles" where the partial derivatives vanish. Using triangular patches[44] or a tiling method[45] results in a non-tensor-product representation of the sphere without degeneracy.

Curve and surface design

The most commonly used curve and surface design techniques are interpolation and data filtering. The sections below summarize some techniques used frequently in practical applications.

Curve design

There are three basic ways to design NURBS curves: sketch a control polygon, interpolate through a set of points, and fit a curve passing near a set of points. Here we discuss the last two methods in some detail.

Interpolation

We distinguish between two kinds of interpolations. In the first, we have pure data points unrelated to any other entities in the system; in the second, we have data points from another process that are related to other entities such as section curves. In the first case, I recommend using nonrational curves (except when specific local interpolants seem more suitable than a general method). In the second case, "true" rational curves have to be computed if the existing entities are rational curves as well.

The global curve-interpolation problem can be solved relatively easily.[3,21,46,47] Given a set of data points $\mathbf{Q}_k$, $k = 0, ..., n$, we seek a B-spline curve that for certain parameter values u_k, $k = 0, ..., n$ agrees with $\mathbf{Q}_k$; that is,

$$\mathbf{Q}_k = \mathbf{C}(u_k) = \sum_{i=0}^{n} \mathbf{P}_i N_{i,p}(u_k) \tag{37}$$

To solve this equation, we need the parameter values at which the data points are assumed, the knot vector, and the degree of the curve. The degree in practical applications is generally 2 or 3; however, the method we are about to consider works with arbitrary degree. One of several methods to compute the parameter values is the centripetal method:[48]

$$u_0 = 0, \quad u_i = u_{i-1} + \frac{\left|\mathbf{Q}_i - \mathbf{Q}_{i-1}\right|^{\frac{1}{2}}}{\sum_{j=1}^{n-1} \left|\mathbf{Q}_j - \mathbf{Q}_{j-1}\right|^{\frac{1}{2}}}, \quad u_n = 1 \tag{38}$$

Given the parameter values, we need a knot vector that reflects the distribution of these parameters. The following averaging method worked well in practice:[21]

$$\mathbf{U} = \{0, 0, ..., 0, v_1, ..., v_{n-p}, 1, 1, ..., 1\} \tag{39}$$

where the end points are repeated with multiplicity $p + 1$, and

$$v_j = \frac{1}{p} \sum_{i=j}^{j+p-1} u_i, \quad j = 1, ..., n - p \tag{40}$$

It can be proved[3] that the coefficient matrix

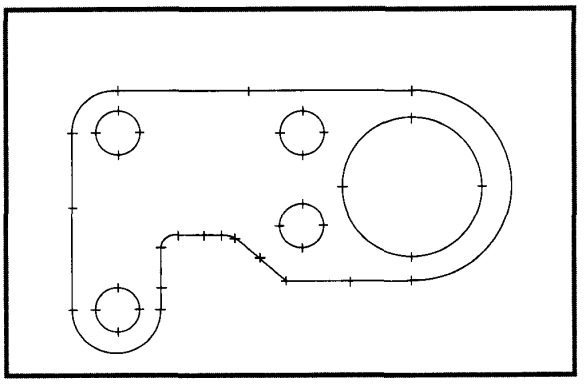

Figure 11. Local quadratic NURBS interpolants. Note that circles are reproduced if the data points are symmetrically placed.

$$N_{i,p}(u_k)\,|_{\,i,\,k=0,\ldots,n}$$

is totally positive and banded with bandwidth less than p. Therefore the linear system (Equation 37) can be solved safely by Gauss elimination without pivoting. Now, if in addition to the data points, derivatives (tangent vectors) are also given, then we need $2(n+1)$ control points to find an interpolatory spline. The systems to be solved are

$$\mathbf{Q}_k = \mathbf{C}(u_k) = \sum_{i=0}^{2n+1} \mathbf{P}_i\, N_{i,p}(u_k) \tag{41a}$$

$$\mathbf{D}_k = \mathbf{C}(u_k) = p \sum_{i=1}^{2n+1} \frac{\mathbf{P}_i - \mathbf{P}_{i-1}}{v_{i+p} - v_i} N_{i,p-1}(u_k) \tag{41b}$$

where the $\mathbf{D}_k$ are the derivative vectors and the v_i are the knots. The parameter values can be obtained as above, or we can take advantage of the tangents and pick another parameterization that is closer to the arc length (fit a parabola between two neighboring data points and approximate its arc length). The knot vector is obtained as follows: We need $2(n+1)+p+1$ knots because there are $2(n+1)$ data points. Because quadratic and cubic curves are the most frequently used curves, we consider the cases $p=2$ and $p=3$. If $p=2$, then choose

$$\mathbf{U} = \left\{0,0,0,\frac{u_1}{2}, u_1, \frac{u_1+u_2}{2}, u_2, \ldots, \frac{u_{n-1}+1}{2}, 1,1,1\right\} \tag{42}$$

If $p=3$, then

$$\mathbf{U} = \left\{0,0,0,0,\frac{u_1}{2}, \frac{2u_1+u_2}{3}, \frac{u_1+2u_2}{3}, \ldots, \frac{u_{n-2}+2u_{n-1}}{3}, \frac{u_{n-1}+1}{2}, 1,1,1,1\right\} \tag{43}$$

If we merge Equations 41a and 41b in an alternating fashion, then the resulting $2(n+1)\times 2(n+1)$ system is banded and can be solved without pivoting.

Now we turn our attention to rational interpolation. If we are given data points and associated weights, then we simply interpolate the weighted data points. To improve the interpolation, tangent information may also be used. Because the tangents are in three dimensions, we need to find derivative information in four dimensions. That is, from given 3D derivatives

$$(\dot{x}_i, \dot{y}_i, \dot{z}_i)$$

we need to find 4D derivatives

$$[(w_i \dot{x}_i), (w_i \dot{y}_i), (w_i \dot{z}_i), \dot{w}_i]$$

Since

$$(w_i \dot{x}_i) = \dot{w}_i x_i + w_i \dot{x}_i$$

we need to compute $\dot{w}_i$ only. We interpolate a 1D spline $w(u)$ through the data points w_i, $i=0,\ldots,n$ so that $w(u_i)=w_i$. From this we have

$$\dot{w}_i = \dot{w}(u)\,|_{\,u=u_i}$$

The above rational and nonrational interpolation techniques are *global* methods; that is, the curve is generated by using all the data points. We can define *local* interpolants by considering two consecutive data points at a time.[49,50] These methods are rather heuristic, but they do provide surprisingly good-looking results (see Figure 11). Quadratic NURBS (conics) are particularly useful in engineering design. The basic steps to obtain a local quadratic interpolant can be sketched as follows:

- Compute tangent directions using a local method such as that of Akima[50] or Renner.[51]
- Use Bezier segments to interpolate between neighboring data points.
- Compute the weights to generate a circular arc if the control triangle happens to be isosceles.
- Use double knots to represent the piecewise Bezier curve as one quadratic NURBS curve.

In general, this method provides a $G1$ continuous curve with a very good parameterization. It is possible to obtain a $C1$ parameterization without double knots either by recomputing the weights using the conic shape invariance or by repositioning the knots. Unfortunately, both methods result in a very bad parameterization and are not applicable to closed curves (for example, the circle[37]).

Data fitting

In many applications we receive a large amount of data. Interpolating a lot of points, some of which are subject to error, results in a wiggly interpolant and a huge database. A better solution is to approximate, that is, to generate a curve that goes near the data points and passes through only a few of them. The most popular approximation is least-squares fitting. Writing

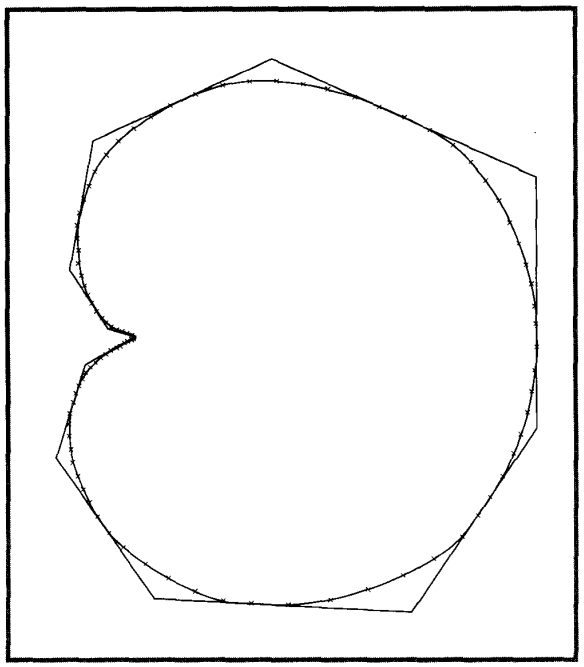

Figure 12. Curve fitting with quadratic NURBS.

for $s = 0$ to m do
  $u_{0,s} = 0$; $u_{n,s} = 1$;
  for $r = 1$ to n do

$$u_{r,s} = u_{r-1,s} + \frac{\left|Q_{r,s} - Q_{r-1,s}\right|^{\frac{1}{2}}}{\sum_{t=1}^{n} \left|Q_{t,s} - Q_{t-1,s}\right|^{\frac{1}{2}}}$$

  end
end
$u_0 = 0$; $u_n = 1$;
for $r = 1$ to $n - 1$ do

$$u_r = \frac{1}{m+1} \sum_{s=0}^{m} u_{r,s}$$

end

Figure 13. The averaging technique needed to compute the parameter values in surface interpolation.

Equation 37 in a matrix form yields $\mathbf{Q} = \mathbf{NP}$, where $\mathbf{Q}$ and $\mathbf{P}$ are $(n + 1) \times 1$ matrices and $\mathbf{N}$ is an $(n + 1) \times (n + 1)$ matrix. Now, if we have more data points than control points, then the equation $\mathbf{Q} = \mathbf{NP}$ is overdetermined and can be solved approximately as follows:[46,52,53]

$$\mathbf{Q}^* = (\mathbf{N}^T\mathbf{N})^{-1}\mathbf{N}^T\mathbf{P} \quad \textbf{(44)}$$

where $\mathbf{N}^T$ is the transpose of $\mathbf{N}$. Based on this, the following algorithm creates a least-squares fit that approximates the data up to a given tolerance:

1. Assign initial parameters to the data points.
2. Generate a least-squares fit by using Equation 44.
3. If the fit is not acceptable, then compute new parameter values and go to Step 2.

This algorithm, combined with a segmentation algorithm, usually gives reasonable curve fits. One snag is an occasional failure to converge; that is, the computed new parameter values do not improve the fit or do not improve it up to a given tolerance. Hence the program does not leave the loop after a reasonable number of iterations.

Geometric considerations may help solve the curve-fitting problem locally.[54] The fitting part of a quadratic NURBS fitting program works as follows: Consider a set of data points $\mathbf{Q}_0, ..., \mathbf{Q}_n$ lying within a control triangle defined by the first and last data points and by the tangents there. Fit conics through $\mathbf{Q}_1, ..., \mathbf{Q}_{n-1}$. If these points lie on a single arc, then all the conic-fitting curves result in the same curve. However, if they are scattered, then fitting conics through these points results in $(n-1)$ different arcs. To measure the scatter, use the distance between the shoulder points (points computed at $u = \frac{1}{2}$) of two external conics. If this distance is less than a user-specified tolerance, then a resultant curve is fitted that passes through the mean of the shoulder points involved. Otherwise the point set is subdivided and the process is repeated. This technique is very reliable and, despite the use of conics, it can be used to fit very complicated data points (see Figure 12).

Surface design

The most frequently used surface design methods are interpolation, fitting, and cross-sectional design (surface definition based on curves).

Surface interpolation

The curve-interpolation technique above can be extended to surfaces relatively easily. Assume that we are given a set of $(n + 1) \times (m + 1)$ data points $\mathbf{Q}_{r,s}$, $r = 0, ..., n$, $s = 0, ..., m$. The objective is to find a degree (p,q) surface that agrees with $\mathbf{Q}_{r,s}$ at certain parameter values, that is

$$\mathbf{Q}_{r,s} = \mathbf{S}(u_r, v_s) = \sum_{i=0}^{n} \sum_{j=0}^{m} \mathbf{P}_{i,j} N_{i,p}(u_r) N_{j,q}(v_s) \quad \textbf{(45)}$$

To solve Equation 45, we need the parameter values and the two knot vectors. To compute the parameter values, we can use the averaging technique shown in Figure 13.[21]

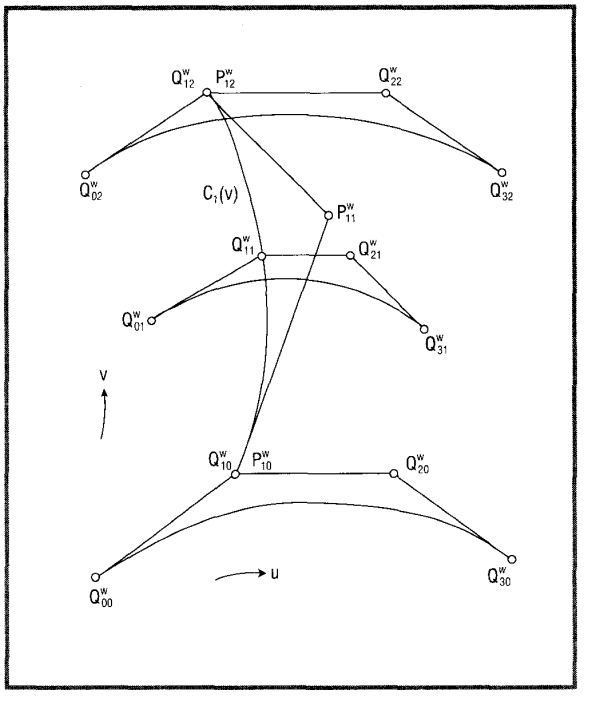

Figure 14. NURBS skinning: surface interpolation through cross-sectional curves.

Courtesy of Intergraph Corp.

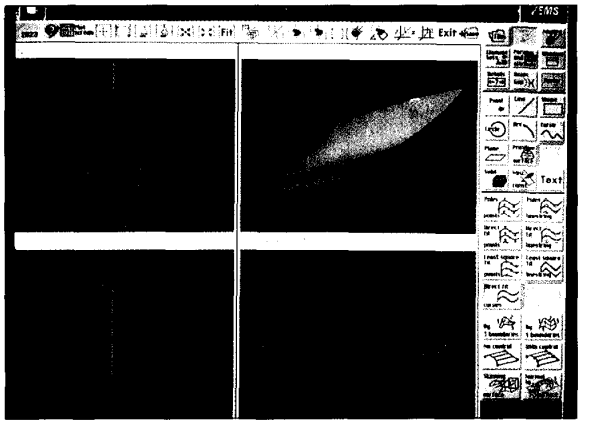

Figure 15. Ship hull surface from surface skinning.

The computation of v_s is analogous. Using the parameters just computed, we get the **U** and **V** knot vectors in exactly the same way as with curves. There are two basic ways to solve for the unknown control points in Equation 45. The first is to solve directly the matrix equation $\mathbf{Q} = \mathbf{UPV}$,[55] where

$$\mathbf{Q} = \left[\mathbf{Q}_{r,s}\right]$$
$$\mathbf{U} = \left[N_{i,p}(u_r)\right]$$
$$\mathbf{P} = \left[\mathbf{P}_{i,j}\right] \quad (46)$$
$$\mathbf{V} = \left[N_{j,q}(v_s)\right]$$

Since **U** and **V** are positive definite (and banded), they are invertible. Thus we have

$$\mathbf{P} = \mathbf{U}^{-1}\mathbf{Q}\mathbf{V}^{-1} \quad (47)$$

The second method is to interpolate each row (column) of data points and to fit a surface through these sectional curves. This method is actually a surface lofting,[56] which will be discussed later.

Surface fitting

With a large amount of data, surface fitting is used instead of interpolation. The curve-fitting technique can be generalized easily to surfaces to yield

$$\mathbf{P}^* = (\mathbf{UV}^T\mathbf{UV})^{-1}\mathbf{UV}^T\mathbf{P}, \quad (48)$$
$$\mathbf{UV} = \left[N_{i,p}(u_r)N_{j,q}(v_s)\right]$$

Cross-sectional design

Cross-sectional design[57-60] is concerned with surface construction based on curves to generate B-spline surfaces. The most frequently used techniques are skinning, sweeping, and swinging.

Briefly, NURBS skinning[15,59,60] works as follows: We are given a set of NURBS curves through which a NURBS surface is to pass. In practice these curves are usually planar curves positioned in 3D space with a so-called spine curve. The skinned surface is obtained in three steps (see Figure 14):

1. Make all cross-sectional curves compatible. That is, all the curves should have the same degree and number of control points and be defined over the same knot vector. Assume this has been done; then

$$\mathbf{C}^w_k(u) = \sum_{i=0}^{n} \mathbf{Q}^w_{i,k} N_{i,p}(u), \quad k = 0,..., K \quad (49)$$

are u-directional curves lying on the surface (isoparametric lines in the u direction) and defined over the same knot vector **U**.

2. Compute v values and a knot vector **V** for interpolation with degree-q NURBS curves (use the technique outlined for curves). The v values are needed as the curves (Equation 49) are assumed to be at a certain fixed v.

3. Using the knot vector **V** and the v values computed in Step 2, interpolate curves through the control points of Equation 49. More precisely, for each i, $i = 0, ..., n$, obtain

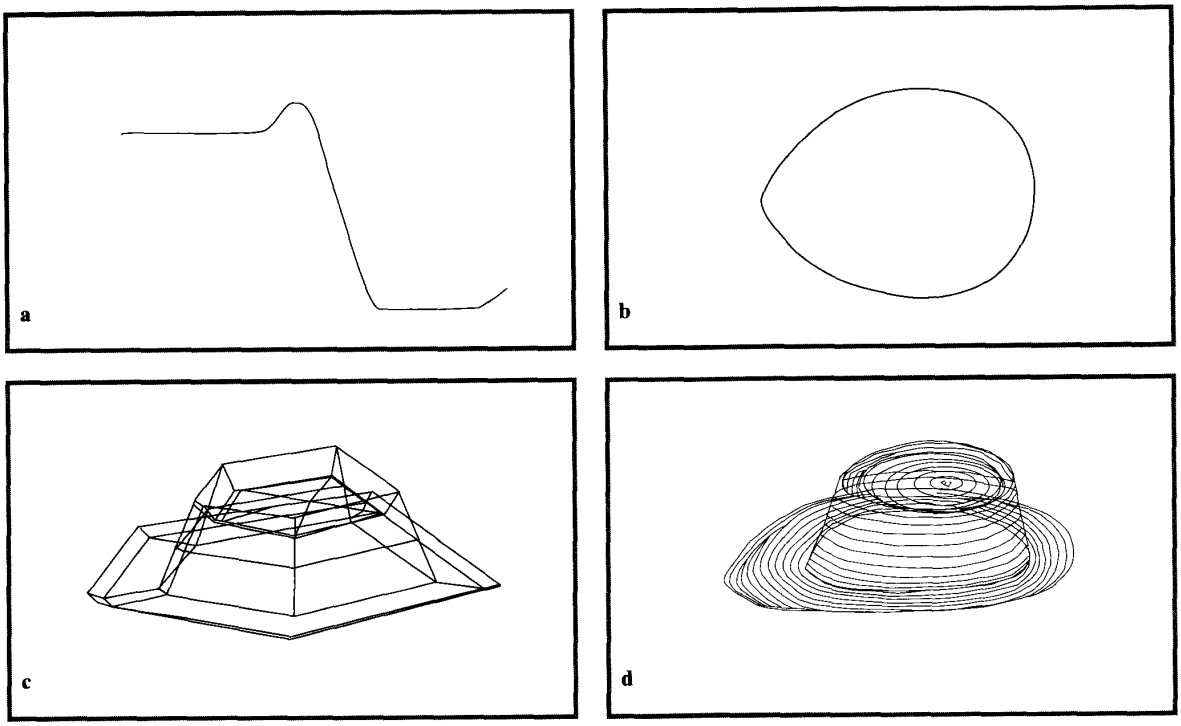

Figure 16. NURBS swinging: (a) profile curve, (b) trajectory curve, (c) control points of the swung surface, and (d) swung surface.

$$\mathbf{C}_i^w(v) = \sum_{j=0}^{m} \mathbf{P}_{i,j}^w N_{j,q}(v) \tag{50}$$

so that Equation 50 interpolates $\mathbf{Q}_{i,k}^w$ at certain v values (note that if the u-directional curves of Equation 49 are rational, then rational interpolations are to be used). The control points of Equation 50 are then the control points of the skinned surface (Figure 15)

$$\mathbf{S}^w(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{m} \mathbf{P}_{i,j}^w N_{i,p}(u) N_{j,q}(v) \tag{51}$$

defined over the knot vectors **U** and **V**.

Although the surface interpolates through the cross-sectional curves, its shape is determined by the following additional factors:

- The position of the cross-sectional curves. If they are very unevenly positioned (not the case in most practical applications), then the surface can behave badly.
- The choice of the v values and the **V** knot vector in Step 2.
- The continuity of the cross-sectional curves. If they are rational, then the algorithm works in four dimensions. Now, even if the rational curve is C^1 in three dimensions, its 4D correspondent may be only C^0 if multiple knots are used (think of the circle discussed above). That is, the skinning algorithm may produce C^0 surfaces in four dimensions whose map in three dimensions will exhibit discontinuities. An obvious fix is not to use multiple knots (whenever possible).

To improve the shape of the surface—say, to remove unwanted oscillations—we can impose additional constraints. For example, if the cross-sectional curves are planar, then at each control point of Equation 49 the normal vector of its plane can be used as a tangent constraint to improve curve interpolations in Step 3.

NURBS sweeping is a special case of skinning that uses a constant section curve. If the spine curve is a v curve, then the constant section curve is positioned along the spine at the same u value.

NURBS swinging is a generalization of rotational sweeping. Assume that we have a profile curve $\mathbf{P}(u)$ in the x,z plane and a trajectory curve $\mathbf{T}(v)$ in the x,y plane (Figure 16):

$$\begin{aligned} \mathbf{P}(u) &= \sum_{i=0}^{n} \mathbf{P}_i R_{i,p}(u) \\ \mathbf{T}(v) &= \sum_{j=0}^{m} \mathbf{T}_j R_{j,q}(v) \end{aligned} \tag{52}$$

Swinging the profile around the z axes yields the following surface[59]:

Courtesy of University of Utah.

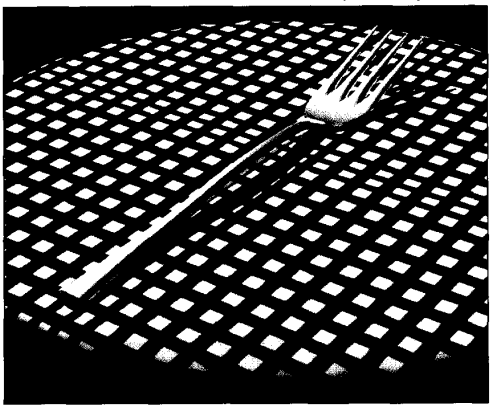

Figure 17. Fork designed with Alpha_1 using a variety of shaping operations: warping, bending, and tapering.

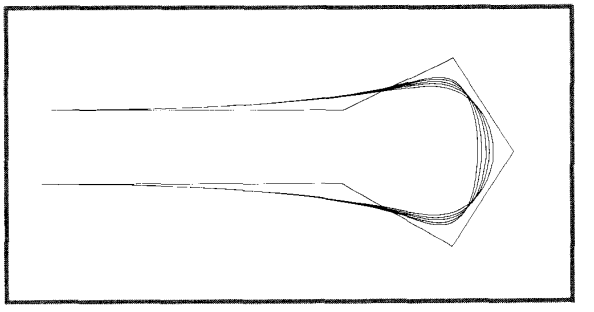

Figure 18. Pushing a NURBS curve with equal increments.

$$\mathbf{S}(u, v) = (\alpha \mathbf{P}_x(u)\mathbf{T}_x(v), \alpha \mathbf{P}_x(u)\mathbf{T}_y(v), \mathbf{P}_z(u)) \tag{53}$$

where α is an arbitrary scaling factor. Multiplying the x and y components of Equation 52 yields the control points and the weights of the swung surface:

$$\mathbf{Q}_{i,j} = (\alpha \mathbf{P}_{i,x}\mathbf{T}_{j,x}, \alpha \mathbf{P}_{i,x}\mathbf{T}_{j,y}, \mathbf{P}_{i,z})$$
$$w_{i,j} = w_i \cdot w_j \tag{54}$$

Shape modifications

There are several ways to alter a shape with the definition of NURBS:

- Reposition control points (including the application of multiple control points to generate sharp edges).
- Change the weights.
- Modify the knot vector.
- Move data points and reinterpolate.

I discuss the first two in some detail.

Repositioning control points

The simplest technique is based on repositioning one control point. Considering curves for simplicity, we reposition a control point as follows: We are given the parameter u/point on the curve, a direction vector $\mathbf{W}$, and a distance d. The curve is pulled/pushed a distance d in the direction $\mathbf{W}$ by recomputing the position of $\mathbf{P}_i$ as $\mathbf{P}_i^* = \mathbf{P}_i + \alpha\mathbf{W}$,[32,33] where

$$\alpha = \frac{d}{|W| R_{i,p}(u)}, \quad u \in [u_i, u_{i+p+1}) \tag{55}$$

Many applications require shape modifications more specific than simple push or pull. The most commonly used methods are warping, flattening, bending, stretching, and twisting. To achieve these operations on NURBS, you use conventional methods[61-63] applied to the control points.[64,65]

Warping introduces bumps into flat surfaces. Picking a warp center on the control net, we reposition control points in a certain direction using the distance function

$$\bar{d} = \begin{cases} \left(1 - \frac{d}{D}\right)^{2^{-w}} & w < 0 \\ \left(1 - \frac{d}{D}\right) & w = 0 \\ \left(1 - \frac{d}{D}\right)^{2^{w}} & w > 0 \end{cases} \tag{56}$$

where w is the warp factor, D is the maximum distance a point can be moved, and d is the distance between a control point and the warp center (Figure 17).

The idea behind flattening is to use an infinite plane and to replace certain control points by their projection onto this plane.

Bending is performed by bending the control net using a polyhedral bending formulation. Too coarse a control mesh or low order can result in such unexpected shapes as a self-intersecting surface. Knot insertion can improve the smoothness of the bent surface.

Stretching is achieved by applying the functional generalization of scaling to the control points. For example, using $g(z) = (x' - x)/x$, a radially symmetric transformation[66] is defined by $(x', y', z') = (x(g(z) + 1), y(g(z) + 1), z)$. Applying this transformation to the control points stretches a NURBS surface.

Twisting is accomplished by applying a functional rotation to the control points. Rotation around the z axis is obtained by $(x', y', z') = ((x \cos\beta - y \sin\beta), (x \sin\beta + y \cos\beta), z)$. If we set $\beta = \beta(z)$, then the above rotation will produce a twist.

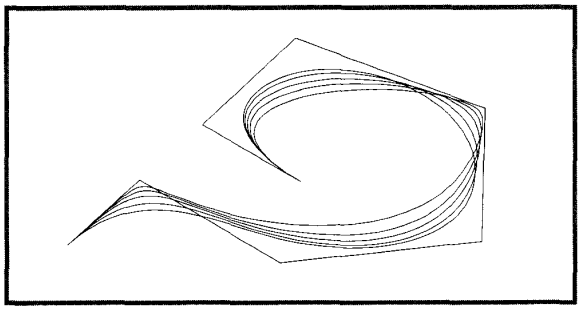

Figure 19. Simultaneous pulls toward nonneighboring control points.

Changing the weights

The geometric meaning of the weights can be used to increase and decrease the fullness of NURBS. Here we consider only the curve case, because surfaces can be handled analogously. Assume that the NURBS curve, at a certain parameter value u, is to be pulled/pushed toward/away from P_i a distance d. This is achieved by recomputing the corresponding weight as follows[32]:

$$w_i^* = w_i \left[1 \pm \frac{d}{R_{i,p}(u)(D-d)}\right], \qquad D = \left|\mathbf{P}_i - \mathbf{S}_i\right|, \quad \mathbf{S}_i = \mathbf{C}(u) \tag{57}$$

We implement this formula as follows:

- Pick a point **S** on the curve.
- Pick a point **P** on the control polygon. The system prompts the designer by drawing the line **SP**.
- Pick a new position of **S** along the line **SP**.

In this process no weights are used explicitly. Recomputing one weight constrains the curve to pass through a point. The picked point **P** must not be an existing control point. If it lies between two existing control points, a knot is inserted so that **P** will be a new control point. To help the designer modify the shape, the above process is automated as follows (Figure 18):

- Pick one point anywhere on the control polygon. The system automatically inserts a knot (if necessary), computes a parameter value (the node[4]), and sets a default increment d.
- When a key is hit, the curve is pulled or pushed with the default increment (which can be overridden). If the designer keeps hitting the key, the curve gets blown up or down until the desired shape is reached.

In many cases, it is desirable to manipulate weights simultaneously. Let $\mathbf{M} = \mathbf{C}(u; w_i = w_{i+1} = 0)$ and $\mathbf{S} = \mathbf{C}(u)$. Then

$$\mathbf{S} = (1 - a_i - a_{i+1})\mathbf{M} + a_i \mathbf{P}_i + a_{i+1}\mathbf{P}_{i+1}, \qquad a_k = R_{k,p}(u), \quad k = i, i+1 \tag{58}$$

Now we would like to pull the curve toward $\mathbf{P}_i$ and $\mathbf{P}_{i+1}$ at the same time by recomputing the corresponding weights simultaneously. We do that by repositioning **S** within $\mathbf{MP}_i\mathbf{P}_{i+1}$:

$$\mathbf{S}^* = (1 - a_i^* - a_{i+1}^*)\mathbf{M} + a_i^*\mathbf{P}_i + a_{i+1}^*\mathbf{P}_{i+1}, \qquad a_k^* = R_{k,p}(u; w_i^*, w_{i+1}^*), \quad k = i, i+1 \tag{59}$$

If we write $w_i^* = \beta_i w_i$ and $w_{i+1}^* = \beta_{i+1} w_{i+1}$, then

$$\beta_i = \frac{1 - a_i - a_{i+1}}{a_i} : \frac{1 - a_i^* - a_{i+1}^*}{a_i^*}$$
$$\beta_{i+1} = \frac{1 - a_i - a_{i+1}}{a_{i+1}} : \frac{1 - a_i^* - a_{i+1}^*}{a_{i+1}^*} \tag{60}$$

The constants $a_i, \ldots, a_{i+1}^*$ are computed geometrically without evaluating the rational basis functions.[32] Here we used them purely for convenience. The derivation above shows that the control points must not be neighboring. Based on the locality properties of NURBS curves, any two points $\mathbf{P}_r$ and $\mathbf{P}_s$ can be picked, as long as $|s - r| \geq p$ (Figure 19).

Conclusions

While NURBS have been used in the graphics and CAD industry for about a decade, publications and basic research have fallen behind technical development. I hope the pointers given in this survey will provide sufficient information until the gap is filled. Several research problems currently being considered are

- Trimmed NURBS surfaces and their visualization.
- Skinning revisited. The technique as used today is not invariant under linear transformation of the cross-sectional curves. That is, if you rotate the set of curves in 3-space, then the skinning operator produces different surfaces (because it works in 4D). In addition to this, multiple knots, needed to define the circle, destroy the continuity of the surface. The application of DeBoor-Fix functionals seems to be promising to remedy some problems.
- Bidirectional skinning (NURBS representation/approximation of Gordon-Coons type surface construction).
- Rational curve and surface interpolation and data fitting.
- Geometry processing of NURBS with particular emphasis on blending, filleting, and offsetting, and the associated utilities such as surface-surface intersection.
- Shaping operators (sculpting tools) and the utilization of the extra degrees of freedom (the *weights*).
- Approximation of NURBS with nonrational splines.
- Visualization techniques based on precise geometry as opposed to polygonal approximation.

In my opinion, there is a great deal of potential inherent in the rational form. Current success with NURBS-based modelers has proved that NURBS are excellent candidates for geometry representation with a unified database. However, the best modeler, capable of coping with all the problems listed in this survey, is yet to come. ❑

Acknowledgments

A great many people have contributed to this article. I owe special thanks to Robert Blomgren, Eugene Lee, Al Klosterman, Wayne Tiller, Elaine Cohen, Rich Riesenfeld, and Dieter Breden for providing valuable information about the history of NURBS and for the numerous unpublished technical reports and memos that gave me invaluable insight into the evolution and development of NURBS technology. Discussions with Dave Rogers, Robert Blomgren, and Wayne Tiller improved substantially the clarity and presentation of this article. This research was supported in part by the Florida High Technology and Industry Council.

References

1. S.A. Coons, "Surfaces for Computer-Aided Design of Space Forms," Tech. Report MAC-TR-41, MIT, Cambridge, Mass., 1967.
2. R.A. Forrest, *Curves and Surfaces for Computer-Aided Design*, doctoral dissertation, Cambridge Univ., Cambridge, UK, 1968.
3. C. de Boor, *A Practical Guide to Splines*, Springer-Verlag, New York, 1978.
4. R.F. Riesenfeld, *Applications of B-Spline Approximation to Geometric Problems of Computer-Aided Design*, doctoral dissertation, Syracuse Univ., Syracuse, N.Y., 1973.
5. K.J. Versprille, *Computer-Aided Design Applications of the Rational B-Spline Approximation Form*, doctoral dissertation, Syracuse Univ., Syracuse, N.Y., 1975.
6. R.M. Blomgren, "B-Spline Curves," Class notes, Boeing document B-7150-BB-WP-2811D-4412, 1981.
7. R.M. Blomgren, "Mathematical Methods for Advanced Geometric Modeling. 3-D Curves and Surfaces," CASE/SME Seminar, SME, Dearborn, Mich., Mar. 1982.
8. E.T.Y. Lee, "A B-Spline Primer," Boeing document, Boeing, Seattle, Wash., 1983.
9. E.T.Y. Lee, "A Treatment of Conics in Parametric Rational Bezier Form," Boeing document, Boeing, Seattle, Wash., Feb. 1981.
10. E.T.Y. Lee, "The Rational Bezier Representation for Conics," Boeing document, Boeing, Seattle, Wash., 1983.
11. R.D. Fuhr, "A Theoretical Introduction to the Rational B-Spline Representation for Curves and Surfaces," Boeing document D6-48379-100, Boeing, Seattle, Wash., 1981.
12. A.L. Klosterman, R.H. Ard, and J.W. Klahs, "Geometric Modeling Speeds Design of Mechanical Assemblies," *Computer Technology Review*, Winter 1982, pp. 103-111.
13. A.L. Klosterman, R.H. Ard, and J.W. Klahs, "A Geometric Modeling Program for the System Designer," *Proc. Conf. CAD/CAM Technology in Mechanical Eng.*, MIT, Cambridge, Mass., 1982.
14. A.L. Klosterman, "A Geometric Modeler Based on a Dual Geometry Representation—Polyhedra and Rational B-Splines," *NASA Symp. Computer-Aided Geometric Modeling*, NASA, 1983, pp. 7-13.
15. W. Tiller, "Rational B-Splines for Curve and Surface Representation," *CG&A*, Vol. 3, No. 10, Sept. 1983, pp. 61-69.
16. R.F. Riesenfeld et al., "Using the Oslo Algorithm as a Basis for CAD/CAM Geometric Modeling," *Proc. NCGA National Conf.*, NCGA, Fairfax, Va., 1981, pp. 345-356.
17. G. Maiorino, "Intergraph Approach in Computer Aided Geometric Design Using Free-Form Surfaces," *Proc. Symp. Automotive Technology and Automation*, Graz, Austria, Vol. 2, 1985, pp. 401-421.
18. "Initial Graphics Exchange Specification, Version 3.0," Doc. No. NBSIR 86-3359, NIST, Gaithersburg, Md., 1986.
19. A. van Dam, "PHIGS+, Functional Description Revision 3.0," *Computer Graphics*, Vol. 22, No. 3, July 1988, pp. 125-218.
20. L. Piegl and W. Tiller, "Curve and Surface Constructions Using Rational B-Splines," *Computer-Aided Design*, Vol. 19, No. 9, Nov. 1987, pp. 485-498.
21. W. Tiller, "Geometric Modeling Using Non-Uniform Rational B-Splines: Mathematical Techniques," Siggraph tutorial notes, ACM, New York, 1986.
22. D.F. Rogers and J.A. Adams, *Mathematical Elements for Computer Graphics*, 2nd Edition, McGraw-Hill, New York, 1990.
23. C. de Boor, "On Calculating with B-Splines," *J. Approximation Theory*, Vol. 6, No. 1, July 1972, pp. 50-62.
24. M.G. Cox, "The Numerical Evaluation of B-Splines," *J. Inst. Mathematics and Applications*, Vol. 10, 1972, pp. 134-149.
25. H.S.M. Coxeter, *Projective Geometry*, Univ. of Toronto Press, Toronto, 1974.
26. M.A. Penna and R.R. Patterson, *Projective Geometry and Its Applications to Computer Graphics*, Prentice-Hall, Englewood Cliffs, N.J., 1986.
27. R.F. Riesenfeld, "Homogeneous Coordinates and Projective Planes in Computer Graphics," *CG&A*, Vol. 1, No. 1, Jan. 1981, pp. 50-55.
28. L.G. Roberts, "Homogeneous Matrix Representation and Manipulation of *N*-Dimensional Constructs," Tech. Report MS-1405, Lincoln Lab, MIT, Cambridge, Mass., 1965.
29. C. de Boor, "B(asic)-Spline Basics," in *Fundamental Developments in CADCAM Geometric Modeling*, L. Piegl, ed., Butterworth-Heinemann, Guildford, UK, 1991.
30. E.T.Y. Lee, "Rational Quadratic Bezier Representation for Conics," in *Geometric Modeling: Algorithms and New Trends*, G. Farin, ed., SIAM, Philadelphia, 1987, pp. 3-19.
31. L. Piegl, "A Geometric Investigation of the Rational Bezier Scheme of Computer-Aided Design," *Computers in Industry*, Vol. 7, No. 5, Oct. 1986, pp. 401-410.

32. L. Piegl, "Modifying the Shape of Rational B-Splines. Part 1: Curves," *Computer-Aided Design*, Vol. 21, No. 8, Oct. 1989, pp. 509-518.

33. L. Piegl, "Modifying the Shape of Rational B-Splines. Part 2: Surfaces," *Computer-Aided Design*, Vol. 21, No. 9, Nov. 1989, pp. 538-546.

34. A.R. Forrest, "The Twisted Cubic Curve: A Computer-Aided Geometric Design Approach," *Computer-Aided Design*, Vol. 12, No. 4, July 1980, pp. 165-172.

35. L. Piegl, "Infinite Control Points—A Method for Representing Surfaces of Revolution Using Boundary Data," *CG&A*, Vol. 7, No. 3, Mar. 1987, pp. 45-55.

36. L. Piegl, "Algorithms for Computing Conic Splines," *ASCE J. Computing in Civil Engineering*, Vol. 4, No. 3, July 1990, pp. 180-198.

37. L. Piegl and W. Tiller, "A Menagerie of Rational B-Spline Circles," CG&A, Vol. 9, No. 5, Sept. 1989, pp. 48-56.

38. D. Hildebrand, *Rationale Polynomialkurven und -flaechen 2. Grades fur CAD [Rational Polynomial Curves and Surfaces of Degree Two for CAD]*, Diplomarbeit, Darmstadt Technical Univ., Darmstadt, West Germany, 1986.

39. D. Hildebrand, "Darstellung von Flaechen 2. Ordnung als rationale Bezierflechen 2. Grades" *[Description of Surfaces of Second Degree as Rational Bezier Surfaces]*, tech. report, Darmstadt Technical Univ., 1986.

40. H. Prautzsch, "Degree Elevation of B-Spline Curves," *Computer Aided Geometric Design*, Vol. 1, No. 2, Nov. 1984, pp. 193-198.

41. E. Cohen, T. Lyche, and L.L. Schumaker, "Algorithms for Degree-Raising of Splines," *ACM Trans. Graphics*, Vol. 4, No. 3, July 1985, pp. 171-181.

42. W. Boehm, "Inserting New Knots into B-Spline Curves," *Computer-Aided Design*, Vol. 12, No. 4, July 1980, pp. 199-201.

43. E. Cohen, T. Lyche, and R.F. Riesenfeld, "Discrete B-Splines and Subdivision Techniques in Computer-Aided Geometric Design and Computer Graphics," *Computer Graphics and Image Processing*, Vol. 14, No. 2, Oct. 1980, pp. 87-111.

44. G. Farin, B. Piper, and A.J. Worsey, "The Octant of a Sphere as a Non-Degenerate Triangular Bezier Patch," *Computer Aided Geometric Design*, Vol. 4, No. 4, Dec. 1987, pp. 329-332.

45. J.E. Cobb, "Tiling the Sphere with Rational Bezier Patches," Tech. Report TR UUCS-88-009, Computer Science Dept., Univ. of Utah, Salt Lake City, 1988.

46. M.G. Cox, "Algorithms for Spline Curves and Surfaces," in *Fundamental Developments in CADCAM Geometric Modeling*, L. Piegl, ed., Butterworth-Heinemann, Guildford, UK, 1991.

47. D.F. Rogers, "B-Spline Curves and Surfaces for Ship Hull Definition," Soc. of Naval Architects and Marine Engineers, New York, 1977, pp. 79-96.

48. E.T.Y. Lee, "Choosing the Nodes in Parametric Curve Interpolation," *Computer-Aided Design*, Vol. 21, No. 6, July/Aug. 1989, pp. 363-370.

49. L. Piegl, "Interactive Data Interpolation by Rational Bezier Curves," *CG&A*, Vol. 7, No. 4, Apr. 1987, pp. 45-58.

50. H. Akima, "A New Method of Interpolation and Smooth Curve Fitting Based on Local Procedures," *J. ACM*, Vol. 17, No. 4, Oct. 1970, pp. 589-602.

51. G. Renner, "A Method of Shape Description for Mechanical Engineering Practice," *Computers in Industry*, Vol. 3, 1982, pp. 137-142.

52. D.F. Rogers, "Constrained B-Spline Curve and Surface Fitting," *Computer-Aided Design*, Vol. 21, No. 10, Dec. 1989, pp. 641-648.

53. D.F. Rogers and S.G. Satterfield, "Dynamic B-Spline Surfaces," *Proc. 4th Conf. Computer Applications in the Automation of Shipyard Operation and Ship Design*, North-Holland, Amsterdam, 1982, pp. 1-8.

54. L. Piegl, "A Technique for Smoothing Scattered Data with Conic Sections," *Computers in Industry*, Vol. 9, No. 3, Nov. 1987, pp. 223-237.

55. B.A. Barsky and D.P. Greenberg, "Determining a Set of B-Spline Vertices to Generate an Interpolating Surface," *Computer Graphics and Image Processing*, Vol. 14, No. 3, Nov. 1980, pp. 203-226.

56. I.D. Faux and M.J. Pratt, *Computational Geometry for Design and Manufacture*, Ellis Horwood, Chichester, UK, 1979.

57. J.K. Hinds and L.P. Kuan, "Surfaces Defined by Curve Transformation," *Proc. 15th Numerical Control Soc. Ann. Meeting and Technical Conf.*, 1978, pp. 325-340.

58. H.P. Moreton and R.D. Bergeron, "SUDS—Surface Description System," *Proc. Eurographics 86*, North-Holland, Amsterdam, 1986, pp. 221-236.

59. C.D. Woodward, "Cross-Sectional Design of B-Spline Surfaces," *Computers and Graphics*, Vol. 11, No. 2, 1987, pp. 193-201.

60. C.D. Woodward, "Skinning Techniques for Interactive B-Spline Surface Interpolation," *Computer-Aided Design*, Vol. 20, No. 8, Oct. 1988, pp. 441-451.

61. R.E. Parent, "A System for Sculpting 3-D Data," *Computer Graphics* (Proc. Siggraph), Vol. 11, No. 2, July 1977, pp. 138-147.

62. W.E. Carlson, *Techniques for the Generation of Three Dimensional Data for Use in Complex Image Synthesis*, doctoral dissertation, Ohio State Univ., Columbus, Ohio, 1982.

63. J.P. Duncan and G.W. Vickers, "Simplified Method for Interactive Adjustment of Surfaces," *Computer-Aided Design*, Vol. 12, No. 6, Nov. 1980, pp. 305-308.

64. R.F. Riesenfeld, "Aspects of Modeling in Computer-Aided Geometric Design," *Proc. Nat'l Computer Conf.*, AFIPS, Reston, Va., 1975, pp. 597-602.

65. E.S. Cobb, *Design of Sculptured Surfaces Using the B-Spline Representation*, doctoral dissertation, Univ. of Utah, Salt Lake City, 1984.

66. A.H. Barr, "Global and Local Deformations of Solid Primitives," Siggraph seminar notes, *State of the Art of Image Synthesis*, ACM, New York, 1983.

Les Piegl is assistant professor of computer science and engineering at the University of South Florida. His research interests are in geometric and solid modeling, engineering computer graphics, computer-aided geometric design, and applied computing. He is editor for Butterworth-Heinemann's *Computer-Aided Design and Manufacturing* series and an advisory editor for the journal *Computer-Aided Design*.

Piegl holds an MS in mathematics and a PhD in computational geometry. He is a member of ACM Siggraph, NCGA, IEEE Computer Society, BCS, Institute of Mathematics and its Applications, and SIAM.

Piegl can be reached at the Department of Computer Science and Engineering, ENG 118, University of South Florida, 4202 Fowler Ave., Tampa, FL 33620, piegl@sol.cmd.usf.edu.