The Free-form Surface Modelling System

1. Introduction

Smooth curves and surfaces must be generated in many computer graphics applications. Many real-world objects are inherently smooth (fig.1), and much of computer graphics involves modeling the real world.

Fig.1. The sparrow, from Lithuanian villages and fields,
requires a smooth, colorfull modelling.

Computer-aided design (CAD), high-quality character fonts, data plots, and artists' sketches all contain smooth curves and surfaces. The path of a camera or object in an animation sequence is almost always smooth; similarly, a path through intensity or color space must often be smooth.

The need to represent curves and surfaces arises in two cases:

1. in modeling existing objects (a car, a face, a mountain);
2. in modeling "from scratch" (so-called image synthesis), where no preexisting physical object is being represented.

The process of modelling is much easier, if mathematical description of an abject, or at least of part of an object may be applied. In the first case, however, a mathematical description of the object may be unavailable. One way of solution, is to use as a model the coordinates of the infinitely many points of the object, but this is not feasible for a computer with finite storage. More often, the object is merely approximated with pieces (called patches, in the case of surface modelling) of planes, spheres, or other shapes that are easy to describe mathematically, and require that points on the model be close to corresponding points on the object.

In the second case, when there is no preexisting object to model, the user creates the object in the modeling process; hence, the object matches its representation exactly, because its only embodiment is the representation. To create the object, the user may sculpt the object interactively, describe it mathematically, or give an approximate description to be "filled in" by some program. In CAD, the computer representation is used later to generate physical realizations of the abstractly designed object.

The general area of *surface modeling* is quite broad, and only a few most common representations for 3D surfaces are mentioned below:

1. polygon mesh surfaces,
2. parametric surfaces,
3. quadric surfaces,

4. free-form modeling

Our goal is to introduce and give an algorithmic description of the process of free-form modelling of surfaces. Nevertheless, to understand all the concepts needed, we have to introduce polygon mesh surfaces, parametric surfaces, and quadric surfaces.

1.1. Polygon meshes

A polygon mesh is a set of connected polygonally bounded planar surfaces. Open boxes, cabinets, and building exteriors can be easily and naturally represented by polygon meshes, as can volumes bounded by planar surfaces. Polygon meshes can be used, although less easily, to represent objects with curved surfaces, as in fig. 2, however, the representation is only approximate. The obvious errors in the representation can be made arbitrarily small by using more and more polygons to create a better piecewise linear approximation, but this increases space requirements and the execution time of algorithms processing the representation. Furthermore, if the image is enlarged, the straight edges again become obvious.

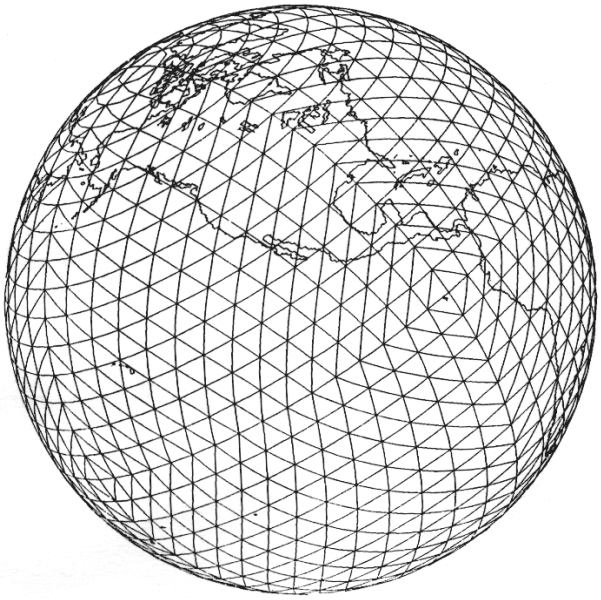

Fig. 2. A 3D object represented by polygon mesh (polygons = triangles)

The polygon meshes have advantages and disadvantages:

- approximation (as precise as possible) – by selecting suitable level of polygon size or representation, the 3D surface may be approximated with a required precision,
- space requirements and execution time – if special data structures are employed, the storage of polygon mesh in computer memory may be minimized, but usually there is trade-off between the amount of memory needed and processing time required,
- algorithm sensible – polygon meshes are sensible to many common transformations of 3D objects, like rotation, translation, scaling, perspective transformation, etc., and special reformulation of an algorithm used to maintain the mesh is needed,
- enlarging picture causes edge problems and other ones – enlargement of a picture makes approximation of an object by polygon mesh more inaccurate, and special efforts to fix the presentation are needed,

- not invariant under basic geometric transforms – requires different representations at internal, external, etc. levels

There are different ways of presentation polygon meshes in computer memory:

- *explicit* (each polygon is represented by a list of vertex coordinates, and the vertices are stored in the order in which they would be encountered travelling around the polygon);
- *pointers to vertex list* (every vertex is stored just once, and polygon is defined by a list of indices (or pointers) into the vertex list);
- *pointers to an edge list* (the vertex list is stored, and the polygon is defined as a list of edges, in which each edge occur just once; in turn each edge in the edge list points to the two vertices in the vertex list defining the edge, and also to the one or two polygons to which the edge belongs).

1.2. Parametric curves

Polylines and polygons are first-degree, piecewise linear approximations to curves and surfaces, respectively. Unless the curves or surfaces being approximated are also piecewise linear, large numbers of endpoint coordinates must be created and stored to achieve reasonable accuracy. Interactive manipulation of the data to approximate a shape is tedious, because many points have to be positioned precisely.

A more compact and more manipulable representation of piecewise smooth curves may be developed. The general approach is to use functions that are of a higher degree than are the linear functions. The functions still generally only approximate the desired shapes, but use less storage and offer easier interactive manipulation than do linear functions.

The higher-degree approximations can be based on one of three methods:

- to express *y* and *z as explicit* functions of *x*, so that $y = f(x)$ and $z = g(x)$;
- to model curves as solutions to *implicit* equations of the form $f(x, y, z) = 0$;
- *to use the parametric representation* for curves, $x = x(t), y = y(t), z = z(t)$.

Parametric curves replace the use of geometric slopes with parametric tangent vectors, and a curve is approximated by a *piecewise polynomial curve*. Each segment *Q* of the overall curve is given by three functions, *x*, *y*, and *z*, which are used to be cubic polynomials in the parameter *t*.

Cubic polynomials are most often used because lower-degree polynomials give too little flexibility in controlling the shape of the curve, and higher-degree polynomials can introduce unwanted wiggles and also require more computation. No lower-degree representation allows a curve segment to interpolate (pass through) two specified endpoints with specified derivatives at each endpoint. Given a cubic polynomial with its four coefficients, four knowns are used to solve for the unknown coefficients. The four knowns might be the two endpoints and the derivatives at the endpoints.

Also, parametric cubics are the lowest-degree curves that are nonplanar in 3D. Higher-degree curves require more conditions to determine the coefficients and can "wiggle" back and forth in ways that are difficult to control. Despite this, higher-degree curves are used in applications-such as the design of cars and planes – in which higher-degree derivatives must be controlled to create surfaces that are aerodynamically efficient. Mathematical development for parametric curves and surfaces is often given in terms of an arbitrary degree *n* (here $n=3$).

The cubic polynomials that define a curve segment $Q(t)=[x(t)\ y(t)\ z(t)]$ are of the form:

$$x(t) = a_x t^3 + b_x t^2 + c_x t + d_x,$$
$$y(t) = a_y t^3 + b_y t^2 + c_y t + d_y,$$
$$z(t) = a_z t^3 + b_z t^2 + c_z t + d_z,$$

To deal with finite segments of the curve, the parameter t may be restricted to the *[0, 1]* interval.

With $T = [t^3, t^2, t, 1]$, and defining the matrix of coefficients of the three polynomials as

$$C = \begin{bmatrix} a_x & a_y & a_z \\ b_x & b_y & b_z \\ c_x & c_y & c_z \\ d_x & d_y & d_z \end{bmatrix}$$

we can rewrite this equation as:

$$Q(t) = [x(t) \quad y(t) \quad z(t)] = T \cdot C$$

Parametric cubic curve segments and their polynomials have the ability of parametrics to represent easily multiple values of y for a single value of x with polynomials that are themselves single valued. Note, the derivative of $Q(t)$ is the parametric *tangent vector* of the curve.

A curve segment $Q(t)$ is defined by constraints on endpoints, tangent vectors, and continuity between curve segments. Each cubic polynomial has four coefficients, so four constraints will be needed, allowing us to formulate four equations in the four unknowns, then solving for the unknowns. The three major types of curves are *Hermite,* defined by two endpoints and two endpoint tangent vectors; *Bezier,* defined by two endpoints and two other points that control the endpoint tangent vectors; and several kinds of *splines,* each defined by four control points. The splines have C^1 and C^0 continuity at the join points and come close to their control points, but generally do not interpolate the points. The types of splines are uniform B-splines, nonuniform B-splines, and 8-splines.

2. Representing Curves

2.1. Hermite Curves

The Hermite form of the cubic polynomial curve segment is determined by constraints on the endpoints P_1 and P_4 and tangent vectors at the endpoints R_1 and R_4.

To find the *Hermite basis matrix* M_H, which relates the *Hermite geometry vector* G_H to the polynomial coefficients, we write four equations, one for each of the constraints, in the four unknown polynomial coefficients, and then solve for the unknowns.

$$G_{Hx} = \begin{bmatrix} P_1 \\ P_4 \\ R_1 \\ R_4 \end{bmatrix}_x \qquad \begin{bmatrix} P_1 \\ P_4 \\ R_1 \\ R_4 \end{bmatrix}_x = G_{Hx} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 3 & 2 & 1 & 0 \end{bmatrix} M_H \cdot G_{Hx}$$

Expanding the product $T \cdot M_H$ in $Q(t) = T \cdot M_H \cdot G_H$ gives the *Hermite blending functions* B_H as the polynomials weighting each element of the geometry vector:

$$Q(t) = (2t^3 - 3t^2 + 1)P_1 + (-2t^3 + 3t^2)P_4 + (t^3 - 2t^2 + t)R_1 + (t^3 - t^2)R_4$$

Figure 3 shows the four blending functions. Notice that, at $t = 0$, only the function labeled P_1 is nonzero: only P_1 affects the curve at $t = 0$. As soon as t becomes greater than zero, R_1, P_4 and R_4, begin to have an influence.

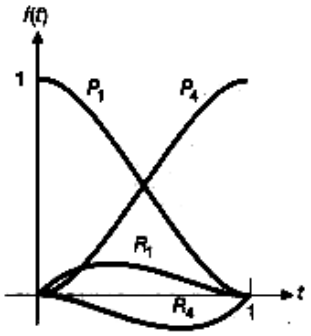

Fig. 3. Blending functions for Hermite curve.

Figure 4 shows a series of Hermite curves. The only difference among them is the length of the tangent vector R_1, the directions of the tangent vectors are fixed. The longer the vectors, the greater their effect on the curve.

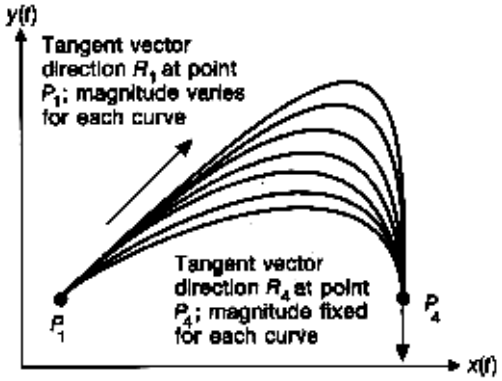

Fig. 4. Family of Hermite parametric cubic curves, where only R_1 is varying.

Figure 5 is another series of Hermite curves, with constant tangent-vector lengths but with different directions. In an interactive graphics system, the endpoints and tangent vectors of a curve are manipulated interactively by the user to shape the curve.

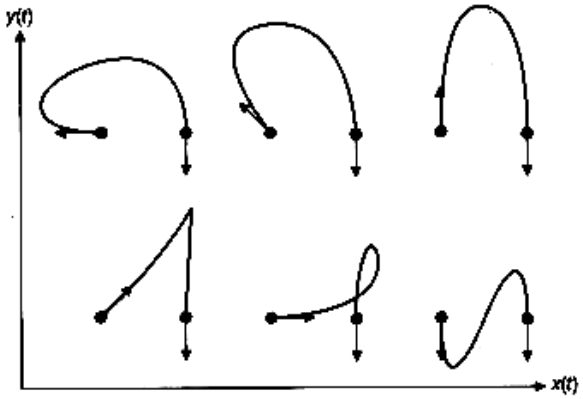

Fig. 5. Family if Hermite curves, where directions of tangent vectors are changing.

Because the cubic curves are linear combinations (weighted sums) of the four elements of the geometry vector, we can transform the curves by transforming the geometry vector and then using it to generate the transformed curve, which is equivalent to saying that the curves are invariant under rotation, scaling, and translation. This strategy is more efficient than is generating the curve as a series of short line segments and then transforming each individual line. The Hermite curves are not invariant under perspective projection.

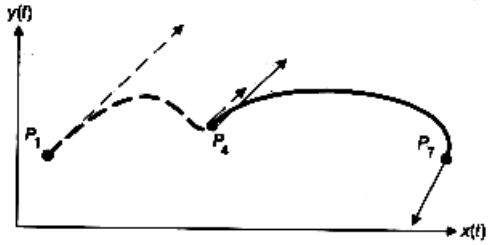

Fig. 6. Joining two Hermite curves together.

2.2. Bezier Curves

The Bezier form of the cubic polynomial curve segment, named after Pierre Bezier, indirectly specifies the endpoint tangent vector by specifying two intermediate points that are not on the curve.

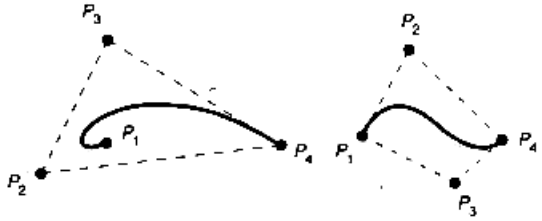

Fig. 7. Bezier curves and their control points.

The starting and ending tangent vectors are determined by the vectors P_1, P_2, and P_3, P_4, and are related to R_1 and R_4 by

$$R_1 = Q'(0) = 3(P_2 - P_1),\ R_4 = Q'(1) = 3(P_4 - P_3)$$

The Bezier curve interpolates the two end control points and approximates the other two. *The Bezier geometry vector GB,* consisting of four points, is

$$G_B = \begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \end{bmatrix}.$$

Then the matrix M_{HB} that defines the relation $G_H = M_{HB} \cdot G_B$ between the Hermite geometry vector G_H and the Bezier geometry vector G_B is just the 4×4 matrix in the following equation, which rewrites Eq. (I 1.24) in matrix form:

$$\vec{r}_H = \begin{bmatrix} P_1 \\ P_4 \\ R_1 \\ R_4 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ -3 & 3 & 0 & 0 \\ 0 & 0 & -3 & 3 \end{bmatrix} \begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \end{bmatrix} = M_{HB} \cdot G_B.$$

To find the *Bezier basis matrix MB,* the Hermite form may be used, by substituting matrices and geometry vectors and carrying out the multiplication:

$$M_B = M_H \cdot M_{HB} = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

and the product is:

$$Q(t) = (1 - t)^3 P_1 + 3t(1 - t)^2 P_2 + 3t^2(1 - i)P_3 + t^3 P_4$$

The four polynomials, which are the weights in equation, are called the *Bernstein polynomials*, and are shown in figure 8:

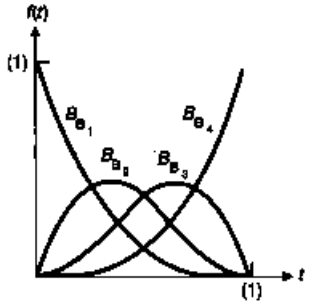

Fig. 8. Blending functions (Bernstein polynomials) for Bezier curves.

Figure 9 shows two Bezier curve segments joined at their common endpoint. This join may follow G^1 continuity, provided at the endpoint when $P_3 - P_4 = k(P_4 - P_5)$, $k > 0$. That is, the three points P_3, P_4, and P_5, must be distinct and collinear. In the more restrictive case when k = 1, there is C^1 continuity in addition to G^1 continuity.

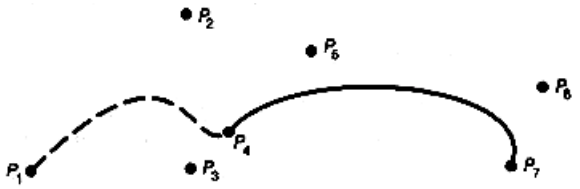

Fig.9. Two Bezier curves joined, when P_3, P_4, and P_5 are collinear.

Examining the four polynomials in fig. 8, we note that their sum is everywhere unity and that each polynomial is everywhere nonnegative for $0 \leq t < 1$. Thus, *Q(t)* is just a weighted average of the four control points. This condition means that each curve segment, which is just the sum of four control points weighted by the polynomials, is completely contained in the *convex hull* of the four control points. The convex hull for 2D curves is the convex polygon formed by the four control points. For 3D curves, the convex hull is the convex polyhedron formed by the control points.

This convex-hull property holds for all cubics defined by weighted sums of control points if the blending functions are nonnegative and sum to one. In general, the weighted average of n points falls within the convex hull of the n points. Another consequence of the fact that the four polynomials sum to unity is that the value of the fourth polynomial for any value of t can be found by subtracting the first three from unity.

The convex-hull property is also useful for clipping curve segments: Rather than clip each short line piece of a curve segment to determine its visibility, we first apply a polygonal clip algorithm to clip the convex hull or its extent against the clip region. If the convex hull (extent) is completely within the clip region, so is the entire curve segment. If the convex hull (extent) is completely outside the clip region, so is the curve segment. Only if the convex hull (extent) intersects the clip region does the curve segment itself need to be examined.

2.3. Uniform Nonrational B-Splines

The term *spline* goes back to the long flexible strips of metal used by draftspersons to lay out the surfaces of airplanes, cars, and ships. "Ducks," weights attached to the splines, were used to pull the spline in various directions. The metal splines, unless severely stressed, had second-order continuity. The mathematical equivalent of these strips, the *natural cubic spline*, is a C^0, C^1, *and* C^2 continuous cubic polynomial that interpolates (passes through) the control points. This is 1 more degree of continuity than is inherent in the Hermite and Bezier forms. Thus, splines are inherently smoother than are the previous forms.

Fig. 10. Natural splines: metal stripes curved by “ducks”.

The polynomial coefficients for natural cubic splines, however, are dependent on all *n* control points; their calculation involves inverting an $(n + 1) \times (n + 1)$ matrix. This has two disadvantages: moving any one control point affects the entire curve, and the computation time needed to invert the matrix can interfere with rapid interactive reshaping of a curve.

B-splines, discussed in this section, consist of curve segments whose polynomial coefficients depend on just a few control points. This is called *local control.* Thus, moving a control point affects only a small part of a curve. In addition, the time needed to compute the coefficients is greatly reduced. B-splines have the same continuity as natural splines, but do not interpolate their control points. A curve segment need not pass through its control points, and the two continuity conditions on a segment come from the adjacent segments. This is achieved by sharing control points between segments, so it is best to describe the process in terms of all the segments at once.

Cubic *B-splines* approximate a series of $m + 1$ control points $P_0, P_1, \ldots P_m$, $m - 3$, with a curve consisting of $m - 2$ cubic polynomial curve segments. Although such cubic curves might be defined each on its own domain, we can adjust the parameter (making a substitution of the form $t = t + k$) so that the parameter domains for the various curve segments are sequential. Thus, we say that the parameter range on which Q_i is defined $t_i \leq t < t_{i+1}$. In the particular case of $m = 3$, there is a single curve segment Q_3 that is defined on the interval $t_3 \leq t < t_4$, by four control points, P_0 *to* P_3.

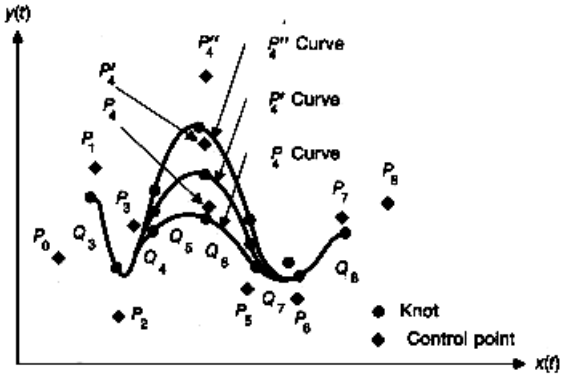

Fig. 11. A B-spline with curve segments Q_3 through Q_9.

The term *uniform* means that the knots are spaced at equal intervals of the parameter t. Without loss of generality, we can assume that $t_3 = 0$ and the interval $t_{i+1} - t_i = 1$. The term *nonrational* is used to distinguish these splines from rational cubic polynomial curves, The

"B" stands for basis, since the splines can be represented as weighted sums of polynomial basis functions, in contrast to the natural splines, for which this is not true. Each of the *m - 2* curve segments of a B-spline curve is defined by four of the *m + 1* control points. In particular, curve segment Q_m is defined by points P_{m-3}, P_{m-2}, P_{m-1} *and* P_m.

Just as each curve segment is defined by four control points, each control point (except for those at the beginning and end of the sequence $P_0, P_1, \ldots P_m$) influences four curve segments. Moving a control point in a given direction moves the four curve segments it affects in the same direction; the other curve segments are totally unaffected (figure 12). This is the local control property of B-splines and of all the other splines discussed here.

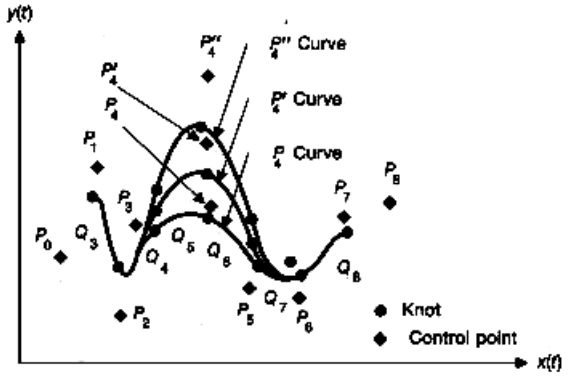

Fig. 12. A B-spline with control point P_4 in different locations.

The B-spline blending functions B_{Bs}, are given by the product $T_i \cdot M_{Bs}$ (figure 13), analogously to the previous Bezier and Hermite formulations. Note that the blending functions for each curve segment are exactly the same, because for each segment *i* the values of $t - t_i$ range from *0* at $t = t_i$ to *1* at $t = t_{i+1}$.

$$M_{Bs} = \frac{1}{6}\begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix}.$$

Fig. 13. The B-spline basic matrix, M_{Bs}.

The blending functions for uniform nonrational B-splines may be given explicitly by formulas in figure 14.

$$\begin{aligned} Q_i(t - t_i) &= T_i \cdot M_{Bs} \cdot G_{Bs_i} = T \cdot M_{Bs} \cdot G_{Bs_i} \\ &= B_{Bs} \cdot G_{Bs_i} = B_{Bs_{i-3}} \cdot P_{i-3} + B_{Bs_{i-2}} \cdot P_{i-2} + B_{Bs_{i-1}} \cdot P_{i-1} + B_{Bs_i} \cdot P_i \\ &= \frac{(1-t)^3}{6} P_{i-3} + \frac{3t^3 - 6t^2 + 4}{6} P_{i-2} + \frac{-3t^3 + 3t^2 + 3t + 1}{6} P_{i-1} \\ &\quad + \frac{t^3}{6} P_i, \; 0 \le t < 1. \end{aligned} \qquad (11.$$

Fig. 14. Polynomials used as blending functions for uniform nonrational B-splines.

The additional continuity afforded by B-splines is attractive, but it comes at the cost of less control of where the curve goes. The curve can be forced to interpolate specific points by replicating control points; this is useful both at endpoints and at intermediate points on the curve.

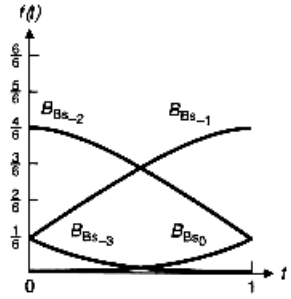

Fig. 15. The B-spline four blending functions, from fig. 14.

If a control point is used several, say three times, then the interpolation of curves can give corners, straight lines, etc.The convex hull in this case is defined by just two distinct points, so the segment has to be a line (figure 16). The price of interpolating the points is loss of G^1 continuity, even though equations show that C^0 continuity is preserved. Another technique for interpolating endpoints, is given by *phantom vertices*. However, with nonuniform B-splines, endpoints and internal points can be interpolated in a more natural way than they can with the uniform B-splines.

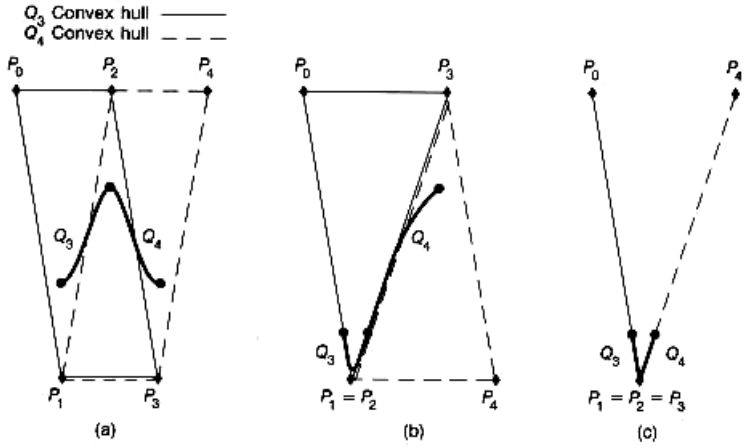

Fig. 16. The effect of multiple control points on a uniform B-spline curve.

2.4. Nonuniform, Nonrational B-Splines

Nonuniform, nonrational B-splines differ from the uniform, nonrational B-splines in that the parameter interval between successive knot values need not be uniform. The nonuniform knot-value sequence means that the blending functions are no longer the same for each interval, but rather vary from curve segment to curve segment.

These curves have several advantages over uniform B-splines. First, continuity at selected join points can be reduced from C^2 to C^1 to C^0 to none. If the continuity is reduced to C^0, then the curve interpolates a control point, but without the undesirable effect of uniform B-splines, where the curve segments on either side of the interpolated control point are straight lines. Also, starting and ending points can be easily interpolated exactly, without at the same time introducing linear segments. It is possible to add an additional knot and control point to nonuniform B-splines, so the resulting curve can be easily reshaped, whereas this cannot be done with uniform B-splines.

The increased generality of nonuniform B-splines requires a slightly different notation than that used for uniform B-splines. As before, the spline is a piecewise continuous curve made up of cubic polynomials, approximating the control points P_0 through P_m. The *knot-value sequence* is a nondecreasing sequence of knot values t_0 through t_{m+4} (that is, there are four

more knots than there are control points). Because the smallest number of control points is four, the smallest knot sequence has eight knot values and the curve is defined over the parameter interval from t_3 to t_4.

The only restriction on the knot sequence is that it be nondecreasing, which allows successive knot values to be equal. When this occurs, the parameter value is called a *multiple knot* and the number of identical parameter values is called the *multiplicity* of the knot (a single unique knot has multiplicity of 1). For instance, in the knot sequence (0, 0, 0, 0, 1, 1, 2, 3, 4, 4, *5*, *5*, 5, 5), the knot value 0 has multiplicity four; value 1 has multiplicity 2; values 2 and 3 have multiplicity 1; value 4 has multiplicity 2; and value 5 has multiplicity 4.

Curve segment Q_i is defined by control points P_{i-3}, P_{i-2}, P_{i-1}, P_i and by blending functions $B_{i-3,4}(t)$, $B_{i-2,4}(t)$, $B_{i-1,4}(t)$, $B_{i,4}(t)$, as the weighted sum.

There is no single set of blending functions, as there was for other types of splines. The functions depend on the intervals between knot values and are defined recursively in terms of lower-order blending functions. $B_{i,j}(t)$ is the j^{th}-order blending function for weighting control point P_i. Because we are working with fourth-order (that is, third-degree, or cubic) B-splines, the recursive definition ends with $B_{i,4}(t)$ and can be easily presented in its "unwound" form. The recurrence for cubic B-splines is:

$$B_{i,1}(t) = \begin{cases} 1, & t_i \le t < t_{i+1} \\ 0, & \text{otherwise,} \end{cases}$$

$$B_{i,2}(t) = \frac{t - t_i}{t_{i+1} - t_i} B_{i,1}(t) + \frac{t_{i+2} - t}{t_{i+2} - t_{i+1}} B_{i+1,1}(t),$$

$$B_{i,3}(t) = \frac{t - t_i}{t_{i+2} - t_i} B_{i,2}(t) + \frac{t_{i+3} - t}{t_{i+3} - t_{i+1}} B_{i+1,2}(t),$$

$$B_{i,4}(t) = \frac{t - t_i}{t_{i+3} - t_i} B_{i,3}(t) + \frac{t_{i+4} - t}{t_{i+4} - t_{i+1}} B_{i+1,3}(t).$$

Fig. 17. Recursive definition of blending functions.

Figure 18 shows how equations in figure 17 can be used to find the blending functions, using the knot vector (0, 0, 0, 0, 1, 1, 1, 1) as an example. The figure also makes clear why eight knot vectors are needed to compute four blending functions.

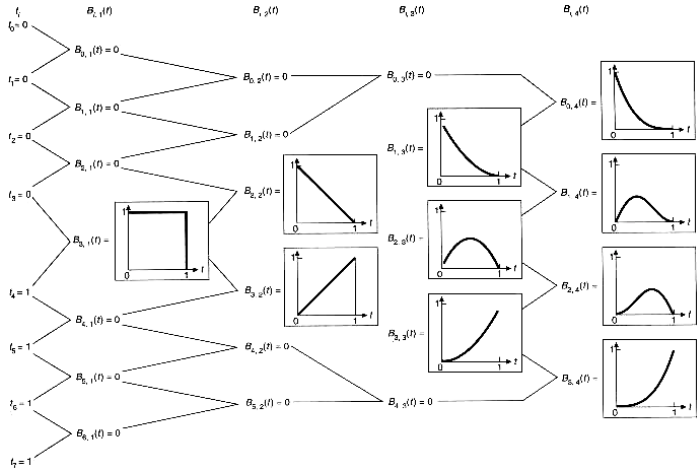

Fig. 18. The relationships defined between the knot vector and blending functions.

Computing the blending functions takes time. By restricting B-spline knot sequences to have intervals that are either 0 or 1, it is possible to store just a small number of matrices corresponding to equations in figure 17, which covers all such possible knot configurations. This eliminates the need to reevaluate equations for each curve segment.

It can be shown that the blending functions are nonnegative and sum to one, so nonuniform B-spline curve segments lie within the convex hulls of their four control points. For knots of multiplicity greater than one, the denominators can be zero because successive knot values can be equal: division by zero is defined to yield zero.

2.5. Nonuniform Rational B-splines (NURBS)

General rational cubic curve segments are ratios of polynomials. A rational curve is defined by a ratio of polynomials:

$$\mathbf{b}^n(t) = \mathbf{b}_0^n(t) = \frac{\sum_{j=0}^{n} w_j \mathbf{b}_j B_j^n(t)}{\sum_{j=0}^{n} w_j B_j^n(t)}$$

where w_j are weights and $B_j^n(t)$ are all cubic polynomial curves whose control points are defined in homogeneous coordinates. We can also think of the curve as existing in homogeneous space as $Q(t) = [X(t), Y(t), Z(t), W(t)]$. As always, moving from homogeneous space to 3-space involves dividing by $W(t)$. Any nonrational curve can be transformed to a rational curve by adding $W(t) = 1$ as a fourth element.

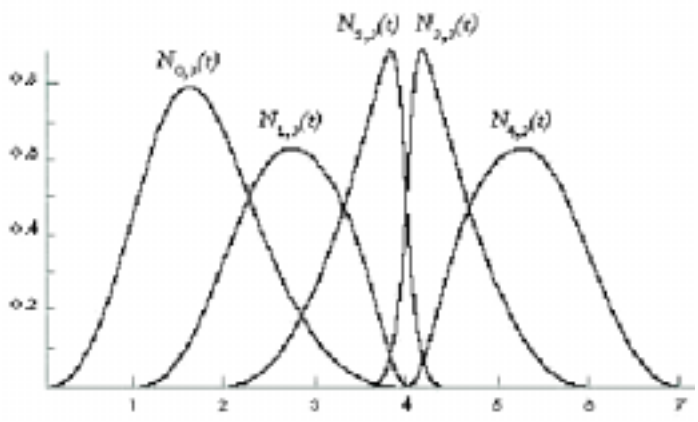

Fig. 19. Basic functions for nonuniform B-spline.

If all weights are equal, the definition fits to the definition of regular Bezier curves, because of the sum of all Bernstein polynomials is = 0. Polynomials in a rational curve can be Bezier, Hermite, or any other type. When they are B-splines, we have nonuniform rational B-splines, called *NURBS*.

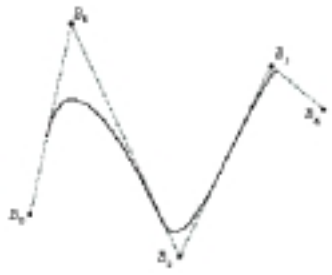

Fig. 20. Nonuniform B-spline curve.

Rational curves are useful for two reasons. The first and most important reason is that they are invariant under rotation, scaling, translation and perspective transformations of the control points (nonrational curves are invariant under only rotation, scaling, and translation).

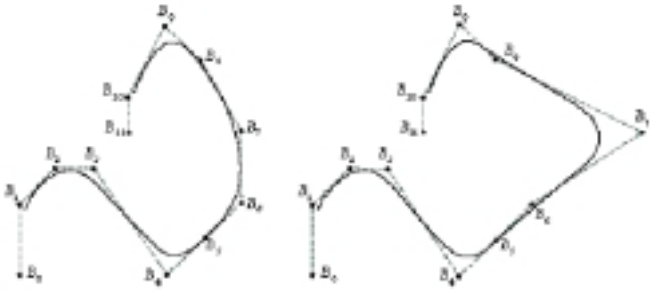

Fig. 21. Potentials of NURBS.

This means that the perspective transformation needs to be applied to only the control points, which can then be used to generate the perspective transformation of the original curve. The alternative to converting a nonrational curve to a rational curve prior to a perspective transformation is first to generate points on the curve itself and then to apply the perspective transformation to *each* point, a far less efficient process. This is analogous to the observation that the perspective transformation of a sphere is not the same as a sphere whose center and radius are the transformed center and radius of the original sphere.

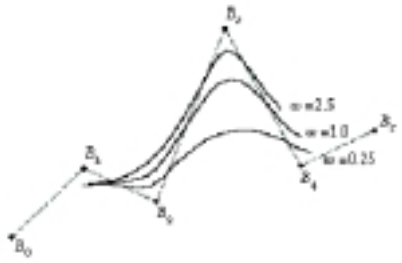

Fig. 22. Different weights of a control point.

A second advantage of rational splines is that, unlike nonrationals, they can define precisely any of the conic sections. There is full analogy of conic sections to the definition of rational Bezier curves. A conic can be only approximated with nonrationals, by using many control points close to the conic. This second property is useful in those applications, particularly CAD, where general curves and surfaces as well as conics are needed. Both types of entities can be defined with NURBS.

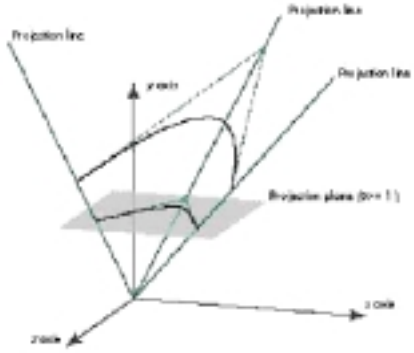

Fig. 23. NURBS as a conic section.

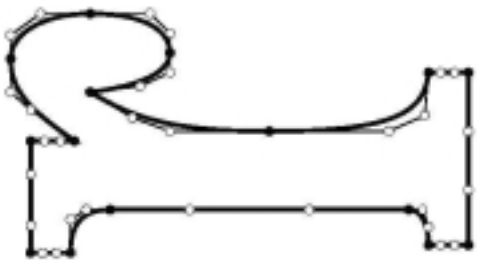

Fig. 24. True type font characters as Bezier spline application.

3. Representing Surfaces

Parametric polynomial curves define points on a 3D curve by using three polynomials in a parameter *t*, one for each of *x*, *y*, and *z*. The coefficients of the polynomials are selected such that the curve follows the desired path. Although various degrees of polynomials can be used, the most common case, cubic polynomials (that have powers of the parameter up through the third) may be considered. The term cubic curve, will often be used for such curves.

Parametric bivariate (two-variable) polynomial surface patches define the coordinates of points on a curved surface by using three bivariate polynomials, one for each of *x*, *y*, and *z*. The boundaries of the patches are parametric polynomial curves.

$$Q(s,t) = S \cdot M \cdot G(t) = S \cdot M \cdot \begin{bmatrix} G_1(t) \\ G_2(t) \\ G_3(t) \\ G_4(t) \end{bmatrix}, \qquad Q(s,t) = S \cdot M \cdot \begin{bmatrix} g_{11} & g_{12} & g_{13} & g_{14} \\ g_{21} & g_{22} & g_{23} & g_{24} \\ g_{31} & g_{32} & g_{33} & g_{34} \\ g_{41} & g_{42} & g_{43} & g_{44} \end{bmatrix} \cdot M^T \cdot T^T$$

$$Q(s,t) = S \cdot M \cdot \mathbf{G} \cdot M^T \cdot T^T, \; 0 \le s, t \le 1.$$

Fig. 25. Equations for parametric bicubic surfaces.

$$x(s,t) = S \cdot M \cdot \mathbf{G}_x \cdot M^T \cdot T^T,$$
$$y(s,t) = S \cdot M \cdot \mathbf{G}_y \cdot M^T \cdot T^T,$$
$$z(s,t) = S \cdot M \cdot \mathbf{G}_z \cdot M^T \cdot T^T.$$

Fig. 26. Separation of coordinates for bicubic surface.

Many fewer bivariate polynomial surface patches than polygonal patches are needed to approximate a curved surface to a given accuracy. The algorithms for working with bivariate polynomials, however, are more complex than are those for polygons. As with curves, polynomials of various degrees can be used, but we discuss here only the common case of polynomials that are cubic in both parameters. The surfaces are accordingly called bicubic surfaces.

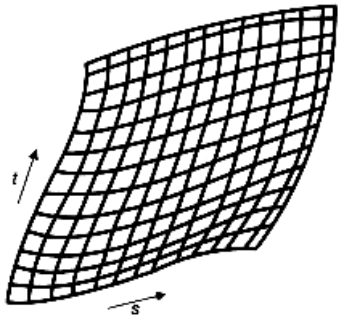

Fig. 27. Bicubic patch.

3.1. Hermite Surfaces

Hermite surfaces are completely defined by a 4 x 4 geometry matrix G_H. Derivation of G_H follows the same approach used to find equation for *Q (s, t)*. The derivation here may be elaborated, applying it just to *x(s, t).* Replacing *t by s*, to get $x(s) = S \cdot M_H \cdot G_H$, and rewriting this further so that the Hermite geometry vector G_H, is not constant, but is rather a function of *t*, we obtain

$$x(s, t) = S \cdot M_H \cdot G_{H_x}(t) = S \cdot M_H \begin{bmatrix} P_1(t) \\ P_4(t) \\ R_1(t) \\ R_4(t) \end{bmatrix}_x .$$

The functions $P_1(t)$ and $P_4(t)$ define the components of the starting and ending points for the curve in parameter s. Similarly, $R_l(t)$ and $R_4(t)$ are the tangent vectors at these points. For any specific value of t, there are two specific endpoints and tangent vectors. Figure 28 shows $P_1(t)$, $P_4(t)$, and the cubic in *s* that is defined when *t = 0.0, 0.2, 0.4, 0.6, 0.8,* and *1.0*. The surface patch is essentially a cubic interpolation between $P_1(t) = Q\ (0, t)$ and $P_4(t) = Q\ (1, t)$ or, alternatively, between *Q(s, 0)* and *Q(s, 1).*

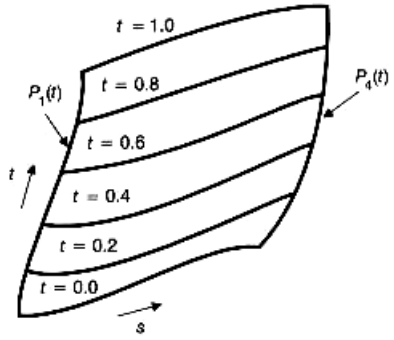

Fig. 28. Bicubic Hermite surface with lines of constant parameter values.

In the special case that the four interpolants *Q(0, t), Q(1, t), Q(s, 0),* and *Q(s, 1)* are straight lines, the result is a *ruled surface* (figure 29). If the interpolants are also coplanar, then the surface is a four-sided planar polygon.

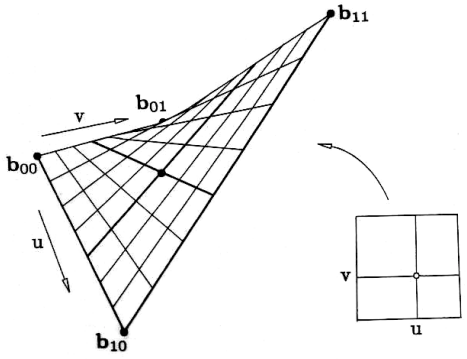

Fig. 29. Hyperbolic paraboloid as an example of ruled surface.

The three 4×4 matrixes G_{Hx}, G_{Hy}, and G_{Hz}, play the same role for Hermite surfaces as did the single matrix G_H for curves. The meanings of the 16 elements of G_H can be understood by relating them back to equations for *x (s, t),* taking into account starting points, ending points, starting tangent vectors, starting slopes:

$$\mathbf{G}_{H_x} = \begin{bmatrix} x(0,0) & x(0,1) & \frac{\partial}{\partial t}x(0,0) & \frac{\partial}{\partial t}x(0,1) \\ x(1,0) & x(1,1) & \frac{\partial}{\partial t}x(1,0) & \frac{\partial}{\partial t}x(1,1) \\ \frac{\partial}{\partial s}x(0,0) & \frac{\partial}{\partial s}x(0,1) & \frac{\partial^2}{\partial s \partial t}x(0,0) & \frac{\partial^2}{\partial s \partial t}x(0,1) \\ \frac{\partial}{\partial s}x(1,0) & \frac{\partial}{\partial s}x(1,1) & \frac{\partial^2}{\partial s \partial t}x(1,0) & \frac{\partial^2}{\partial s \partial t}x(1,1) \end{bmatrix}.$$

The upper-left 2 x 2 portion of G_{Hx}, contains the x coordinates of the four corners of the patch. The upper-right and lower-left 2 x 2 areas give the x coordinates of the tangent vectors along each parametric direction of the patch. The lower-right 2 x 2 portion has at its comers the partial derivatives with respect to both s *and* t. These partials are often called *the twists,* because the greater they are, the greater the corkscrewlike twist at the corners. Figure 30 shows a patch whose corners are labeled to indicate these parameters.

This Hermite form of bicubic patches is an alternative way to express a restricted form of the *Coons patch*. These more general patches permit boundary curves and slopes to be any curves. (The Coons patch was developed by the late Steven A. Coons, an early pioneer in CAD and computer graphics after whom SIGGRAPH's prestigious *Steven A. Coons Award for Outstanding Contributions to Computer Graphics is* named.) When the four twist vectors in the Hermite form are all zero, the patches are also called *Ferguson* surfaces after another early developer of surface representations.

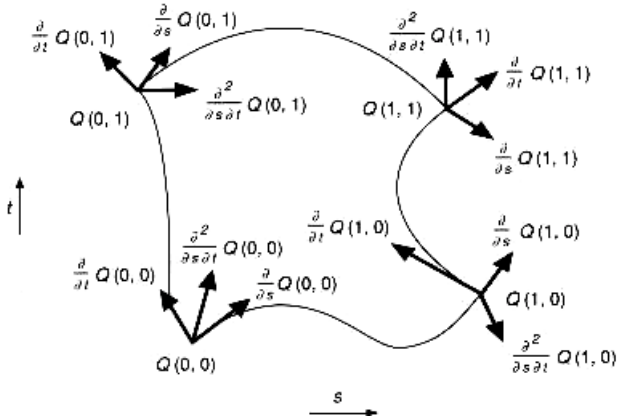

Fig. 30. Components of the geometry matrix for a Hermite surface.

Just as the Hermite cubic permits C^1 *and* G^1 continuity from one curve segment to the next, so too the Hermite bicubic permits C^1 and G^1 continuity from one patch to the next. First, to have C^1 continuity at an edge, the matching curves of the two patches must be identical, which means the control points for the two surfaces must be identical along the edge. The necessary conditions for C^1 continuity are that the control points along the edge and the tangent and twist vectors across the edge be equal. For G^1 continuity, the tangent vector requirement is relaxed so that the vectors must be in the same direction, but do not need to have the same magnitude.

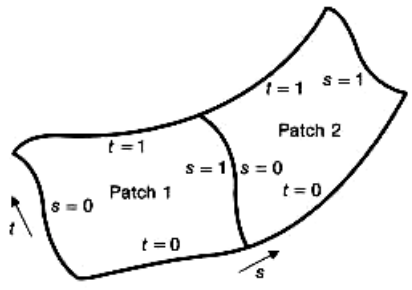

Fig. 31. Two joined Hermite surface patches.

3.2 Bezier Surfaces

The Bezier bicubic formulation can be derived in exactly the same way as the Hermite cubic. The results are:

$$x(s, t) = S \cdot M_B \cdot G_{B_x} \cdot M_B^T \cdot T^T,$$
$$y(s, t) = S \cdot M_B \cdot G_{B_y} \cdot M_B^T \cdot T^T,$$
$$z(s, t) = S \cdot M_B \cdot G_{B_z} \cdot M_B^T \cdot T^T.$$

The Bezier geometry matrix G consists of 16 control points, as shown in figure 31. Bezier surfaces are attractive in interactive design for the same reason as Bezier curves are:

- some of the control points interpolate the surface, giving convenient precise control, whereas tangent vectors also can be controlled explicitly.

When B6zier surfaces are used as an internal representation, their convex-hull property and easy subdivision are attractive.

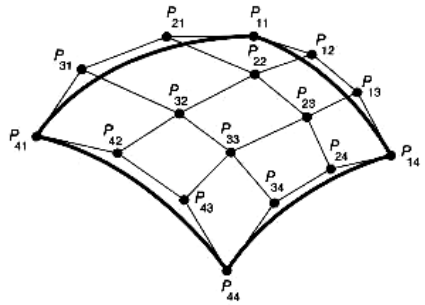

Fig. 32. Control points for a Bezier bicubic patch.

C^1 and G^1 continuity across patch edges is created by making the four common control points equal. *G^1* continuity occurs when the two sets of four control points on either side of the edge are collinear with the points on the edge. In figure 33, the following sets of control points are collinear and define four line segments whose lengths all have the same ratio *k.*

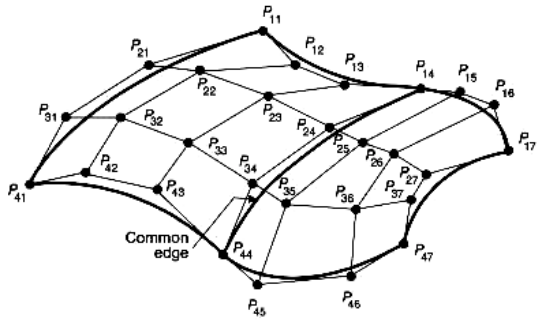

Fig. 33. Two Bezier patches joined along the edge.

3.3. B-Spline Surfaces

B-spline patches are represented as:

$$x(s, t) = S \cdot M_{Bs} \cdot G_{Bs_x} \cdot M_{Bs}^T \cdot T^T,$$
$$y(s, t) = S \cdot M_{Bs} \cdot G_{Bs_y} \cdot M_{Bs}^T \cdot T^T,$$
$$z(s, t) = S \cdot M_{Bs} \cdot G_{Bs_z} \cdot M_{Bs}^T \cdot T^T.$$

C^1 continuity across boundaries is automatic with B-splines; no special arrangements of control points are needed except to avoid duplicate control points, which create discontinuities.

Bicubic nonuniform and rational B-spline surfaces and other rational surfaces are similarly analogous to their cubic counterparts. All the techniques for subdivision and display carry over directly to the bicubic case.

Like curves, surfaces can be displayed either by iterative evaluation of the bicubic polynomials or by subdivision, which is essentially an adaptive evaluation of the bicubic polynomials.

Iterative evaluation is best suited for displaying bicubic patches in the style of figure 27. Each of the curves of constant s and constant t on the surface is itself a cubic, so display of each of the curves is straightforward.

Surface display using recursive subdivision is also a simple extension of subdividing technique. Nearly planar quadrilaterals are created, then are displayed as shaded flat quadrilaterals using linear interpolation methods. The process is again simplest if the Bezier form is used.

Flatness is tested by computing the plane passing through three of the four corner control points and finding the distance from each of the other 13 control points to the plane and is just a generalization of the flatness test used for curves.

Surface subdivision is done by splitting the surface along one parameter, say s, and then splitting each of the two resulting surfaces along t (figure 34).

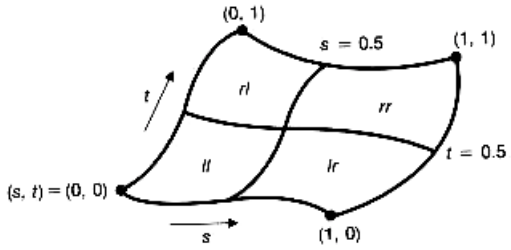

Fig. 34. A surface subdivided into four surfaces.

It is sometimes desirable to display only a portion of a bicubic surface. For instance, there may be a hole in a patch caused by a pipe going through the surface. Displaying only a portion can be done with *trimming curves,* which are just spline curves defined in *(s, t)* parameter space instead of in *(x, y, z)* space on the bicubic surface.

3.4. Quadric Surfaces

The implicit surface equation of the form

$$f(x, y, z) = ax^2 + by^2 + cz^2 + 2dxy + 2eyz + 2fxz + 2gx + 2hy + 2jz + k = 0$$

defines the family of quadric surfaces. For example, if $a = b = c = -k = 1$ and the remaining coefficients are zero, a unit sphere is defined at the origin. If a through f are zero, a plane is defined. Quadric surfaces are particularly useful in specialized applications such as molecular modeling, and have also been integrated into solid modeling systems. Recall, too, that rational cubic curves can represent conic sections; similarly, rational bicubic surfaces can represent quadrics. Hence, the implicit quadratic equation is an alternative to rational surfaces, if only quadric surfaces are being represented. Other reasons for using quadrics include case of

- computing the surface normal;

- testing whether a point is on the surface;
- computing z given x and y (important in hidden-surface algorithms);
- calculating intersections of one surface with another.

An alternative representation of quadric surfaces:

$$P^{T} \cdot Q \cdot P = 0,$$

$$Q = \begin{bmatrix} a & d & f & g \\ d & b & e & h \\ f & e & c & j \\ g & h & j & k \end{bmatrix} \quad \text{and} \quad P = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}.$$

The surface represented by Q can be easily translated and scaled. Given a 4×4 transformation matrix M of the form developed, the transformed quadric surface Q' is given by:

$$Q' = (M^{-1})^{T} \cdot Q \cdot M^{-1}.$$