An Intuitive Description of
Parametric Splines
in Computer Graphics

Brian A. Barsky

Computer Science Division
Department of Electrical Engineering and Computer Sciences
University of California
Berkeley, California 94720
U.S.A.

Abstract

This is an intuitive description of some of the parametric spline formulations that are most prevalent in computer graphics. The intent is to provide a "high-level", descriptive presentation rather than a detailed, rigorous one. The specific techniques discussed are piecewise Hermite interpolation, spline interpolation, Bézier curves, B-splines, Beta-splines, and rational splines. The topics of subdivision and geometric continuity are also explained.

1. Introduction

Traditionally, in computer graphics, many objects have been modeled using a polygonal database. Such a model comprises a collection of polygons, which may be planar or non-planar, that are pieced together to form an approximation to the shape. One of the reasons for the ubiquity of this approach is its inherent simplicity. For example, when the polygons are assumed to be planar, the silhouette edges are simply a subset of the polygonal boundaries. However, this simplicity severely limits the utility of this technique. When representing a smooth surface, the edges between polygons are visually objectionable and a faceted appearance results. Furthermore, this discontinuous appearance is exacerbated by the phenomenon of *Mach bands* introduced because of *lateral inhibition* of the human visual system. To alleviate this, a finer set of polygons could be used. However, this causes an increase in storage as well as in the necessary computation. Although this approach may reduce the faceted appearance, it does not get at the heart of the solution. Also, whatever fineness of polygonal approximation is selected, this then becomes a fixed level.

It would be preferable to have the number of polygons be responsive to the particular problem at hand. First, the level of approximation should relate to the size of the object as it is being rendered. The very same object should be approximated with more polygons if it looms large in the foreground of the image and with less polygons if it is small in the distance. Second, even within the object itself, a more advanced approach to the polygonal approximation would allow a non-uniform distribution of polygons where fewer large polygons could be used in regions where the geometry is simple while a denser set of smaller polygons would be indicated in areas of higher geometric complexity.

A more general approach is to use a *quadric* (an implicit quadratic) surface (or its curve analogue, a *conic*).[14,22,34] Examples of quadric surfaces include spheres, ellipsoids, cylinders, cones, paraboloids, hyperboloids, and hyperbolic paraboloids. Although this approach does address some of the shortcomings of the polygonal database, it still does not provide sufficient flexibility to represent a wide class of free-form shapes. The flowing curves of an aerodynamic sports car, for example, cannot be represented as a combination of such regular geometric primitives. It is also not straightforward to represent a finite, bounded portion of the surface.

The motivation for the development of techniques for representing free-form curves and surfaces includes the applications areas of biomedical imaging, the design and construction of automobile bodies, naval architecture, aircraft wing and fuselage design, the development of turbine blades, and the design of bottles. The class of mathematical functions that we study in this undertaking are called *splines*. There are a variety of definitions for a spline, some of which are more exacting and others of which are more relaxed. Nonetheless, the one salient feature that all the definitions share is that the representation be *piecewise* in nature. That is to say that rather than defining

an entire curve or surface by a single mathematical expression, it is instead decomposed into separate pieces each possessing its own mathematical specification. With this decomposition, there must be an associated set of constraints that controls the assembly of the pieces into a coherent form. Such constraints generally address the issue of *smoothness* of the curve or surface. Smoothness is a nebulous notion, and much of our research has investigated such ideas.[4,5]

Let us use the terminology of curve *segment* and surface *patch* for the pieces of a curve and surface, respectively. The points where the curve segments join are called *joints*, and the curves between adjacent patches are called *borders*.

Having established the piecewise nature of splines, the next logical question to ask is what kinds of functions are used for each such piece? In most cases, we restrict our attention to polynomial pieces. Of course, one could imagine pieces of a more general nature, such as trigonometric functions, exponential functions, and rational functions. However, since polynomials are used to approximate more general functions in the computer, it is reasonable to consider polynomials as the class of functions for the pieces.

Thus, we will be considering piecewise polynomials subject to certain continuity constraints. Such a piecewise polynomial is more flexible than a single polynomial since it allows different behaviour on each piece, having only the restriction of smoothness where adjacent pieces meet. Consequently, this means that a wider class of shapes can be represented by such a piecewise polynomial than by a single polynomial. This simple notion corresponds to the mathematical view that the space of polynomials forms a subspace of the space of splines for the same degree. Furthermore, the space of splines itself is a subspace of the space of general piecewise polynomials of the same degree since some degrees of freedom are used to satisfy the constraints of smoothness. Mathematically, the "size" is measured by the *dimension* of the space. There are still some shortcomings that need to be addressed by our piecewise polynomials with continuity constraints. If we use the traditional explicit functional form (such as $y=f(x)$), then the representation of a multiple-valued curve (such as a circle) requires splitting the curve into various segments. And this splitting of the curve might have to be recomputed if the curve were rotated. Furthermore, infinite values would arise in the representation of vertical tangents. Some of these problems could be addressed using implicit functions (having the form $f(x,y)=0$). However, this has shortcomings in the evaluation of a particular point, the calculation of derivatives, and the specification of a portion of a shape (such as a semi-circle).

To address these problems, our piecewise polynomials will be formulated using a *parametric representation*. In this form, each coordinate is represented by its own separate, independent function. Continuing with the example of a circle, this could be easily represented as $(r\cos\theta, r\sin\theta)$. However, the parametric representation is not a panacea; although it addresses the problems outlined above, it does introduce an additional level of complexity. For example, derivatives cease to be *scalar-valued*, but become *vector-valued*. That is to say that the n^{th} derivative is a vector whose components are the n^{th} derivative of each coordinate with respect to the parameter. This means that the derivative information now includes direction in addition to magnitude. This can introduce subtleties even in simple situations. For example, a curve could have a continuous unit tangent vector or slope and yet lack a continuous first derivative due to a jump in the magnitude of the first derivative. This simple idea shows the distinction between *parametric continuity* and *geometric continuity*, which we have been actively investigating,[4,5] and this is covered in more detail in Section 6.

Intuitively, the parametrization can be thought of as a description in terms of time. The vector-valued representation provides the position at a given instant in time. As time passes, the path is traced out. Using this metaphor, it is easy to imagine different parametrizations that trace out the same path. The identical path can be traced out with different velocities. Whether a particle moves with uniform speed or alternatively accelerates and decelerates, the very same path can be traversed. This illustrates that many parametrizations can specify the same curve. Thus, one should distinguish between a *parametrization* and a *curve*. Consequently, the very same curve can be *reparametrized* such that the parametrization changes but the shape of the final curve does not. These ideas are at the heart of our study of geometric continuity. Another complexity introduced by the parametric representation is that there are now two separate spaces with which to deal. The curve or surface itself exists in a *geometric space*. However, there is also a parameter space which is one-dimensional for a curve and two-dimensional for a surface. Another way to interpret this parametric representation is that we are defining a mapping which distorts the parameter space into the corresponding shape in geometric space. Imagine taking an infinitely-stretchable rectangular sheet of rubber (which is the parameter space) and bending and twisting it to form a surface in three-space. Again, from a terminology standpoint,

the values in parameter space that correspond to the joints in geometric space are called *knots*.

There is a vital property that the splines used in computer graphics should possess, even though this may be less important in other problem domains. This property, called *local control*, restricts the effect of a defining point to a small predetermined region of the curve or surface. There are two reasons for the importance of this property. First, it enables more precise control over shape since a user would be able to finely tune a region of the curve or surface without being concerned about altering other regions. Second, the computational requirements for recomputing the curve based on the movement of a defining point would be fixed and independent of the total amount of data. On the other hand, a *global representation*, lacking the local control property would require a recomputation of the entire curve which would have computational requirements dependent on the total amount of data. It should be noted that there are some intermediate levels of control where the effects of moving a point would be felt in remote regions of the curve or surface but to a much lesser degree. It is important to consider, however, that even in this case as long as there is *some* change, albeit minimal, it is still necessary to perform the new computation. Thus, for fast update of a modified curve, local control is an essential property. Note also that with this property it is feasible to have realtime changes to the shape. It is the parametric representation and local control property that distinguish the study of splines in computer graphics from that in other fields.

It is important to distinguish between the use of splines for *interpolation* versus *approximation*.[15] Interpolation refers to the property that the curve will go between and pass through, or *interpolate*, a set of data points. Approximation means that the curve will pass near, but not necessarily through, the defining points. The idea here is that the shape of the curve mimics the gross shape of the arrangement of the defining data points. To emphasize this distinction, we will refer to the data points in an approximation scheme as *control vertices*. Figure 1 shows a curve that approximates a sequence of control vertices.

The idea of merely approximating, rather than interpolating, may seem to be disconcerting at first glance. However, this seemingly disadvantageous situation is mediated by the fact that there are some additional properties of approximation that are difficult or impossible to attain with interpolation.

For example, the local control property discussed above is easier to achieve with approximation schemes, although it is still possible to obtain with

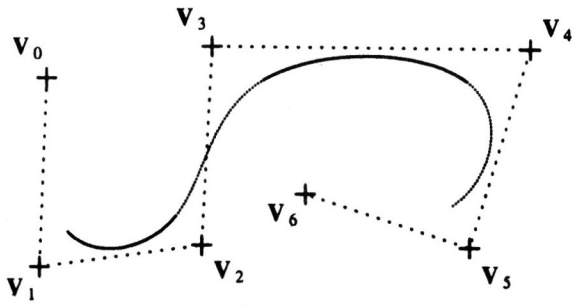

Figure 1:
Curve approximating a sequence of control vertices.

certain interpolation methods. An example of a local interpolating spline in computer graphics is the Catmull-Rom spline. On the other hand, one property which requires an approximation scheme is the *variation-diminishing* property. Intuitively, a curve satisfying this property wiggles less than the underlying data. To be more precise, consider an ordered sequence of straight line segments connecting the data points. A curve is said to be variation diminishing if there does not exist any line that can be drawn that would intersect the curve more often than it would intersect the line segments connecting the data points. Although this property fails for interpolation schemes, it is achieved for approximation schemes such as the Bézier, B-spline, and Beta-spline representations. It is interesting to note that there is no analogous variation-diminishing property for surfaces. Another important property is the *convex hull* property which provides a region in which the curve or surface must lie. Intuitively, the convex hull of a set of points is the region that would be enclosed by an infinitely stretchable rubber material wrapped around the points and pulled taught. In two dimensions, this could be thought of as an elastic rubber band enclosing an area while in three dimensions this could be a rubber sheet enclosing a volume. Generally speaking, most of the approximation schemes used in computer graphics satisfy some kind of convex hull property. In some cases, there is an even tighter convex hull property where the curve or surface lies in the union of convex hulls of a sequence of subsets of points. For example, Figure 2 shows such a region for a cubic B-spline curve.

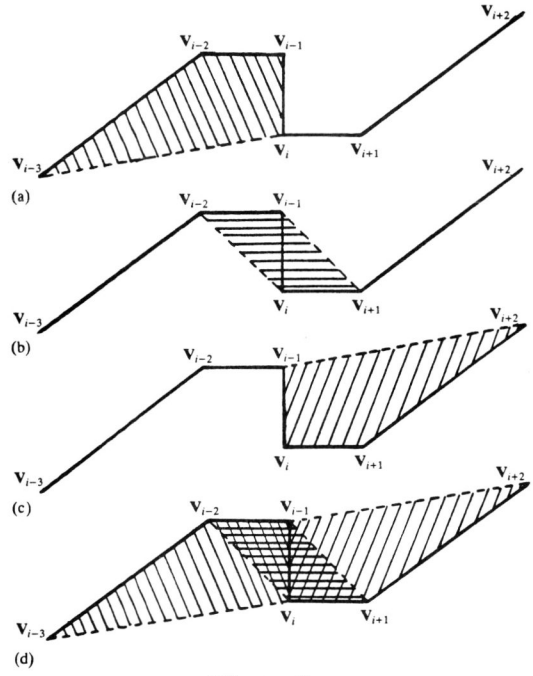

Figure 2:
Union of convex hulls.

2. Piecewise Hermite Interpolation

Piecewise Hermite interpolation is a classical method that strings together a sequence of polynomial pieces between successive data points. In addition to passing through the data points, the curve is constrained to match given derivatives at these points. Each piece is defined separately and must agree in position and derivatives at each of its two endpoints. The most common degree for Hermite interpolation is cubic. In this case, each piece matches the position and first derivative at each endpoint. Using a higher degree, it is also possible to match higher order derivatives. Since each piece has two endpoints, for every additional level of derivative to be matched the degree must be increased by two. Thus, matching position and first and second derivatives requires quintic polynomials, and so on. Since each piece is defined separately, this scheme has local control. Assuming that the derivative is uniquely specified at each point, the level of continuity of the curve will be the same as the level of derivative that was being specified; that is, C^1 for cubics, C^2 for quintics, and so on. It is interesting to note that it is possible to specify different derivatives at the end of one segment than those at the beginning of the succeeding segment for more general shape possibilities at the expense of strictly defined continuity.

The specification of derivatives can either be an advantage or disadvantage depending on the application. It might be useful to have this level of control or it may be considered an encumbrance to require this specification. In the latter case, there are a variety of methods that can be developed to automatically specify a derivative. One of the most interesting methods engenders what has been classically known as the *spline*. That is to say, one can trade off the freedom to specify derivatives for additional continuity. More precisely, it is possible to determine a set of derivatives such that higher order continuity is achieved. In the cubic case, for example, it is possible to determine a set of first derivatives such that second derivative continuity is guaranteed.

3. Spline Interpolation

The most common form of spline, found in many numerical analysis textbooks, is a piecewise cubic polynomial possessing C^2 continuity.[9,35,36] More generally, this can be a piecewise polynomial of degree d with continuity of position and the first $d-1$ derivatives. As mentioned earlier, some splines comprise nonpolynomial pieces, but this is relatively rare. It is also possible to modify this continuity requirement.

At the end of the last section, it was stated that a cubic spline can be constructed by determining a set of first derivatives that ensures second derivative continuity. This can be achieved by establishing a set of simultaneous linear equations whose unknowns are the first derivative vectors at the joints. Each equation enforces the condition of second derivative continuity at a particular joint and involves the unknown first derivative value at three joints. Stepping through the sequence of equations constraining continuity at each successive joint reveals that there is an overlap of two unknown first derivative values between each pair of successive equations. Consequently, these equations are coupled and can be represented in matrix form by a *tridiagonal*† matrix. The sparseness and special structure of a tridiagonal matrix allows for linear time algorithms to solve the equations (instead of the usual $O(n^3)$ algorithms to solve general systems of linear equations). Furthermore, it can be shown that these matrices are *strictly diagonally dominant*‡ which allows for reliable computations. It is important to note that there will be two fewer equations than there are unknowns. This is due to the fact that there is one equation for each *interior* joint whereas there is an

† A *tridiagonal* matrix is a matrix whose nonzero entries are confined to the diagonal, subdiagonal, and superdiagonal.

‡ A matrix is said to be *strictly diagonally dominant* if for every row, the absolute value of the diagonal element exceeds the sum of the absolute values of the off-diagonal elements in that row.

unknown first derivative at every joint including the first and last endpoints. To have the same number of equations as unknowns requires the addition of two end conditions. There are various possibilities in the selection of an end condition. The importance of this choice should not be underestimated because the end condition does affect the shape of the entire curve. One very common choice is to constrain the curve to have its second derivative vanish at each of the two endpoints. This is referred to as the *natural cubic spline.*

The importance of this case is that it models the physical spline. The physical spline is a plastic or wooden lath that is flexible and is used by draftsmen to produce a smooth curve through a set of points. Treating each subsection of the physical device as a beam with small deflection yields the natural cubic spline as its mathematical model. It is interesting to note that the assumption of small deflection is not really valid. The identical mathematical model results by minimizing the energy integral of the physical curve. Interestingly, here the same erroneous assumption appears in the form of assuming that the first derivative is small. Since this integral involves the square of the second derivative, this minimization property is responsible for creating the notion that the natural cubic spline is a "smooth" curve.

Unfortunately, spline interpolation of this form lacks the variation-diminishing and local control properties. To achieve these properties, which are important for computer graphics and geometric modeling, approximation schemes such as the Bézier, B-spline, and Beta-spline representations are often used.

4. Bézier Curves

4.1. Explanation

Bézier curves and surfaces, named for Pierre Bézier, form the nucleus of *Système Unisurf* at Renault.[6,7,8] The Bézier curve is specified by a set of points, called *control vertices*, which are connected in succession to form an open or closed *control polygon.* The resulting curve begins at the first control vertex and ends at the last control vertex but does not necessarily interpolate any of the interior vertices. The curve is tangent to the first and last polygon edge. The shape of the resulting curve mimics that of the control polygon, but in a smoother fashion.

The Bézier curve can be expressed mathematically as a weighted average of these control vertices. A particular point on the curve corresponds to a specific set of weights applied to these control vertices. As the values of these weights are varied, the curve is then traced out. Each weighting factor is a function of a parameter. Thus, the connection between the value of the parameter and a point on a curve is established by evaluating each of these weighting functions at the particular value of the parameter and then computing the corresponding weighted average. Curves of different shapes can be generated by using different positions of the control vertices. This weighted average can also be regarded as a linear combination where the control vertices are the combination coefficients. In this interpretation, the weighting factors play the role of basis vectors. For this reason, these weighting factors are usually referred to as *basis functions.* Like weights, these basis functions are nonnegative and sum to one. The idea, then, is that as the value of the parameter is varied, the basis functions attain various values that alter the weighting of the control vertices thereby producing a set of points to form the final curve.

It is the case that for Bézier curves, these basis functions are the *Bernstein polynomials.* These are the polynomials that are seen in the binomial distribution as well as in the proof of the Weierstrass Theorem. For this reason, Bézier curves are sometimes referred to as *Bernstein-Bézier curves.* Because of the interpretation of these basis functions as the binomial distribution, there are some interesting probabilistic interpretations of Bézier curves that are readily apparent.

The degree of a Bézier curve is equal to the number of edges in the control polygon, that is, one less than the number of control vertices. In this manner, the curve is a single polynomial of this degree. Note that this is not a piecewise representation. Consequently, this approach has global, not local, control. In addition, since this is simply a polynomial, it is C^{∞} continuous.

Although this form is simple and C^{∞} continuous, the lack of local control and the connection of the degree to the number of control vertices are problematic. These impediments can be circumvented through the use of a piecewise version of the Bézier curve, although this is at the expense of a reduction in the level of continuity achieved. The composite (or piecewise) Bézier curve strings together a sequence of Bézier curves, each with its own control polygon, thereby reducing the degree and establishing local control. However, to achieve a given level of continuity requires the application of constraints to the positions of the control vertices. This represents a departure from the idea that the control vertices could be placed in any position desired.

The simplest continuity constraint is positional continuity which would require that the last vertex of one control polygon be in the same position as the first vertex of the succeeding control polygon. In this case, moving any interior control vertex of a control polygon would affect just that one Bézier curve while moving this common control vertex would result in the modification of two Bézier curve segments. If unit tangent vector continuity is required in addition, then the last edge of the previous control polygon must be collinear with the first edge of the next one. This is called *first order geometric continuity* and is denoted by G^1. If *first order parametric continuity* (C^1) is required, then these two edges must also be of equal length (Figure 3). Both curvature vector continuity (G^2) as well as parametric second derivative vector continuity (C^2) at a joint involve constraints on the common control vertex and on the two control vertices on either side of the joint. Consequently, for cubics, maintenance of C^2 continuity when moving one control vertex requires repositioning some control vertices associated with several neighbouring segments. Note that it is sufficient to maintain C^2 continuity by modifying control vertices on only one adjacent segment if higher degree curve segments were used. Since C^2 continuity at a joint affects the common control vertex and the two control vertices on either side of the joint, this continuity could be ensured by defining each curve segment with six control vertices, that is, by using fifth degree curve segments.

More generally, since C^n continuity at a joint affects $n+1$ control vertices per segment including the common control vertex on either side of the joint, this continuity could be ensured by defining each curve segment with $2(n+1)$ control vertices, that is, by using degree $2n+1$ curve segments. However, it is possible to maintain C^n continuity with a composite Bézier curve of degree $n+1$ by adjusting the control vertices of only $n+2$ segments.

To see why this is the case, first recall that degree $n+1$ B-spline curves have this local control property. Then note that it must be possible to represent a composite Bézier curve of degree $n+1$ as a single knot degree $n+1$ B-spline curve since these curves are both C^n piecewise polynomial curves. The effect of moving a control vertex along with the neighbouring vertices that would need to be adjusted in the Bézier representation so as to maintain C^n continuity can be achieved by moving a B-spline control vertex. Finally, the resulting B-spline curve can be converted back to a Bézier representation using knot insertion.[5]

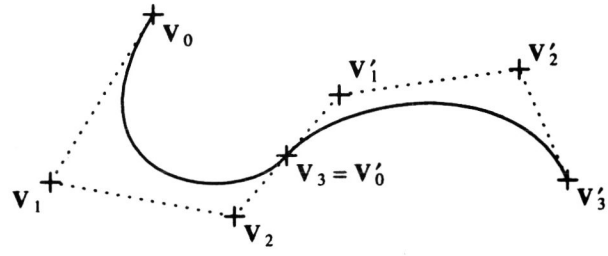

Figure 3:
A composite cubic Bézier curve.

4.2. Subdivision

One of the most important attributes of the Bézier curve is the ease with which it can be *subdivided*. By introducing new control vertices, subdivision splits the curve into two pieces each of which has its own defining control polygon. In specific cases, the positions of the new control vertices can be determined explicitly from the positions of the original control vertices. More generally, a set of intermediate control vertices is introduced. The mathematical formulas for positioning these new control vertices have a very simple and useful geometric interpretation, as illustrated in Figure 4. Simply stated, each edge of the polygon is divided into the ratio u:1–u where u is the parametric value at which the subdivision occurs. For example, for midpoint subdivision each edge is simply divided at its parametric midpoint. Then, these new vertices are connected in succession. This forms a sequence of edges where the number of edges here is one less than in the original polygon. Each of these new edges is then subdivided in the same ratio and these new points are connected again. At each stage of this process, there is one less edge than there was at the preceding stage. Finally, this will result in a single edge which is then subdivided again in the same ratio. This point is then the common vertex of the two new Bézier curves. The remaining vertices of each of the two new Bézier curves are a subset of those found in this development.

Because Bézier curves interpolate their endpoints, this new point just found will lie on the Bézier curve. For this reason, this geometric construction is often given as a method to compute a point on the Bézier curve. This approach is sometimes referred to as the *deCasteljau Algorithm*.[11,12]

Other points on the curve can be found in one of two ways. One possibility is simply to rerun this subdivision at a different value of the parameter u. Thus, subdividing for a sequence of different values of the parameter u will generate a sequence of points on the curve. The other possibility is to then treat each one of the new Bézier curves as a starting point

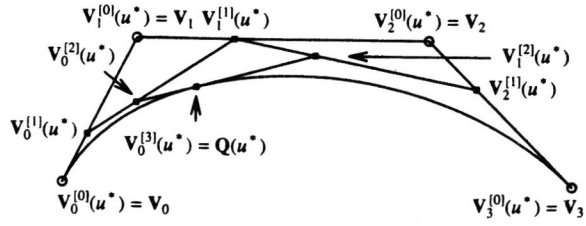

Figure 4:
Geometric construction for point on cubic Bézier curve.

for subdivision and simply recursively subdivide on each side. This latter approach is frequently referred to as *recursive subdivision.* Associated with recursive subdivision is some criterion for termination. One such criterion is *flatness.* In this approach, subdivision of a particular Bézier curve segment stops when that segment is deemed "flat." At that point, the curve segment can be approximated either by its control polygon or by the straight line segment connecting the first and last control vertex for that curve segment.

Such a termination criterion requires some kind of *flatness test.* The flatness test should be some computation performed on the vertices themselves so as to avoid a calculation of a point on the curve. Many different tests are possible. However, it is a challenge to develop a test that is both computationally efficient and provides a correct conclusion over a wide range of cases. Given a particular test, it is interesting to construct counterexamples for which the test would yield a misleading answer. This cannot be solved simply by developing more elaborate tests, since it would defeat the purpose of subdivision if the amount of computation required to perform the flatness test equaled or exceeded that required to perform another level of subdivision.

Although Bézier curves are simple and are easy to subdivide, the trade-offs between the use of a single Bézier curve and a composite Bézier curve provide limitations. It would be desirable to have a curve formulation that had local control and had the selection of the degree dissociated from the number of control vertices without sacrificing the freedom to freely position the control vertices. Such a possibility exists with the more general B-spline which will now be discussed.

5. B-splines

5.1. Explanation

The B-spline is an approximation technique which, in fact, can be viewed as a generalization of the Bernstein-Bézier approach. Like the Bézier technique, a B-spline curve is specified by a set of control vertices connected in succession to form a control polygon. The curve is a piecewise polynomial. The continuity can be controlled without resorting to constraining the positions of the control vertices. The degree of the curve can be set independent of the number of control vertices. One of the most important properties of the B-spline is local control; that is, moving a single control vertex modifies only a limited portion of the curve.

A very common degree for a B-spline is cubic. A cubic B-spline curve comprises a sequence of curve segments each of which is defined by four control vertices. Two successive curve segments have three of their control vertices in common. It is this overlapping of control vertices that builds in the continuity. Moreover, because there are four control vertices specifying each curve segment, it must be the case that each control vertex has influence over four curve segments. This is where the local control property arises. The sequence of curve segments is defined by stepping through the control vertices, dropping the "oldest" vertex from the preceding segment, and adding the "youngest" vertex for the current segment. In this manner, each curve segment is defined by exactly four vertices. The number four is, of course, based on "one more than the degree" of the cubic curve. When working with B-splines, it is often convenient to refer to the quantity that is "one more than the degree"; this is called the *order.* For example, a cubic is of degree three and of order four.

For an arbitrary order k B-spline, each piece is of degree at most $k-1$, the continuity is generally C^{k-2}, each curve segment is controlled by k control vertices, and each control vertex has influence over k curve segments.

Similar to what has been discussed previously, the B-spline curve can be written as a weighted average of its control vertices, where the weights are known as basis functions. In fact, the "B" in B-spline is meant to denote the word "basis." In the mathematical literature, the term "B-spline" is used to refer to these basis functions. This is different than in computer graphics where a B-spline is thought of as a curve. To avoid any confusion, we will use the terms "B-spline basis function" and "B-spline curve," even though there may seem to be some redundancy in one or the other of these depending on one's background.

Due to the weighted average nature of the B-spline, many properties can be seen to be inherited from the basis functions. The basis functions themselves are piecewise polynomials of order k joined together with C^{k-2} continuity. Based on this, the B-spline curve will also generally be of order k with C^{k-2} continuity. It is interesting to note that it is possible to have the B-spline curve have higher order continuity than the underlying basis functions for certain special arrangements of the relative positions of the control vertices. The fact that there are k control vertices specifying each curve segment is equivalent to the fact that there are no more than k basis functions that are nonzero at any parametric value. Equivalently, the local control property limiting the effect of any single control vertex to k curve segments is equivalent to the *local support* property of the basis functions that states that each basis function is nonzero only over k parametric intervals. Recall that the connection between the parametric intervals and the curve segments is that each parametric interval in parameter space is mapped to a curve segment in geometric space. This correspondence can be a little confusing when the introduction of multiple knots is allowed as will be discussed later.

The simplest practical B-spline is that which is piecewise linear, that is, of order two. Here, the B-spline curve coincides with the control polygon. In this case, all the control vertices are interpolated; however, this is not the case for B-splines of order three and greater. The basis functions for the order two, piecewise linear B-spline curve are triangular "hat functions" spanning two parametric intervals each. The curve is formed in the following manner: Each basis function is scaled by its corresponding control vertex in a componentwise manner. At any given parametric value, there are at most two of these scaled basis functions that are nonzero. Adding the values of each of the two scaled basis functions would produce the point on the curve, again done in a componentwise manner. This is illustrated in Figure 5. When the parametric knots are equally spaced, the B-spline is said to be *uniform*; otherwise, the parametric intervals are not all of the same size and the resulting B-spline is *nonuniform*. As might be expected, there is a tradeoff between the simplicity of the uniform B-spline with the generality of the nonuniform B-spline. An advantage of the uniform B-spline is that all the basis functions are simply translates of each other. This fact can be exploited in the construction of efficient evaluation algorithms. On the other hand, the nonuniform B-spline is far more general. The set of parametric values, called the *knot vector*, can take on any set of values provided that these values are nondecreasing and that the same value does not appear more than k times. Having the same parametric value repeated may seem strange but, in fact, this can be used to great advantage.

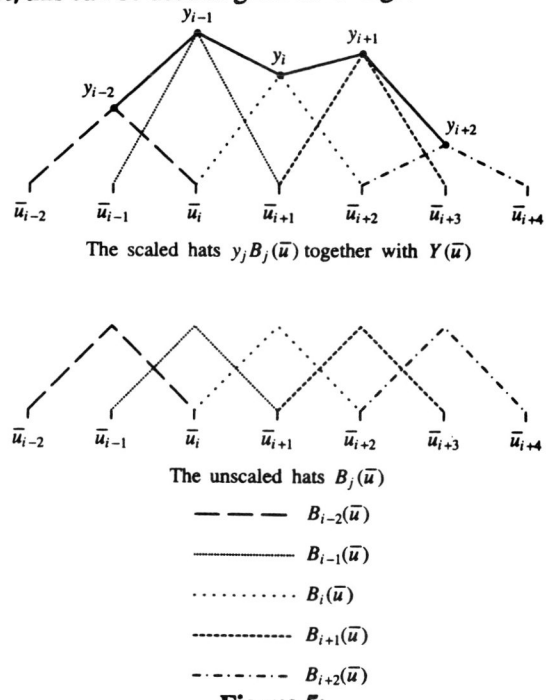

Figure 5:
Linear combination of "hat functions."

When the same value is repeated in the knot vector, this is called a *multiple knot*. The number of times that the same value appears is called the *multiplicity* of the knot. Multiple knots can be used locally to control the continuity at a joint. The level of continuity at a joint is reduced as the multiplicity of the corresponding knot is increased. For an order k curve, the continuity at a single knot would be C^{k-2}. At a double knot, the continuity would be C^{k-3}, at a triple knot the continuity would be C^{k-4}, and so on. Continuing along in this manner, the curve would be only C^0 continuous at a knot of multiplicity $k-1$. Although it may seem surprising, it is even possible to introduce a k-fold knot. In this case, the continuity is denoted C^{-1}, and the specified curve actually splits apart with a gap in it, even though it is still defined by the same mathematical expression. From this, it can be seen that the continuity at a knot can be quantified as $C^{k-1-\mu}$, where μ is the multiplicity of the knot. Note that in the special case of a single knot, μ would have the value 1 and this would reduce to C^{k-2} as would be expected. When dealing with knots having multiplicity greater than 1, it is sometimes convenient to distinguish the entire knot vector from the distinct knot values. The distinct knot values are sometimes referred to as the *breakpoints* of the spline. A break-

point then is simply a value, whereas a knot consists of both a value and an index.

As an example, consider the effect of a triple knot on a cubic B-spline. Here, the curve will be only positionally continuous. This is shown in Figure 6 at the vertex V_4. This C^0 continuity is inherited from the basis functions which also are only C^0 continuous at the corresponding triple knot, as illustrated in Figure 7. Figure 8 shows the various ways in which the multiplicity of a knot can be increased in what would otherwise be a uniform cubic B-spline basis function.

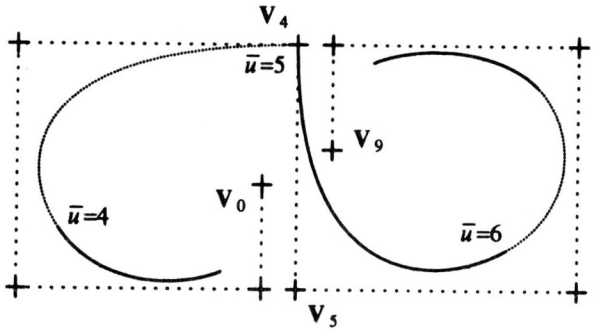

Figure 6:
Positional continuity at a triple knot.

It is worth pointing out that multiple knots are distinct from *multiple vertices*, even though these two phenomena are frequently confused since some of the effects are similar. Increasing the multiplicity of a knot genuinely decreases the corresponding parametric continuity at that knot. On the other hand, increasing the multiplicity of a vertex does not reduce the parametric continuity even though it may engender shape changes such as cusps. An explanation for this is that the first derivative vanishes at the cusp, but this is still without any reduction in continuity.

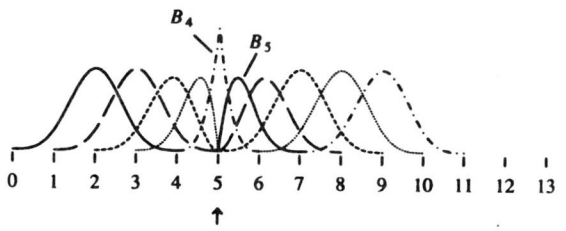

Figure 7:
Basis functions with triple knot for curve in Figure 6.

Allowing nonuniform knot vectors engenders a very general class of B-spline basis functions and curves. Recall from Figure 8 how a wide set of shapes is possible for B-spline basis functions simply by increasing the multiplicity of a single knot. Although each basis function must be piecewise polynomial of order k, a mathematical expression capturing this in its full generality can be somewhat complicated.

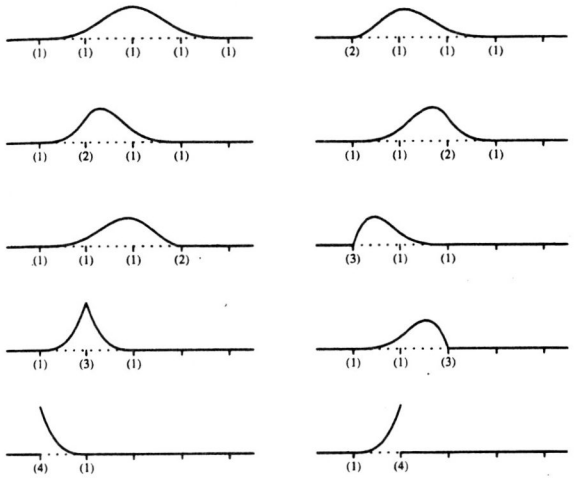

Figure 8:
Various knot multiplicities for a cubic basis function.

The early (1946-1972) formulation of a B-spline basis function involved the *divided difference*† of *one-sided power functions*‡. This definition involves recursively taking the difference between one-sided power functions and dividing by the difference between knot values. Since this process can involve the difference between nearly equal values and division by small numbers, it can be unstable.

Motivated by this instability, Maurice Cox and Carl de Boor independently in 1972 developed a recurrence relation for the evaluation of B-spline basis functions.[9] This has become known as the *Cox / de Boor Algorithm*, and it is a stable and efficient method. This recurrence relation expresses a B-spline basis function in terms of two lower order basis functions. More specifically, consider an r^{th} order B-spline basis function that is nonzero over the parametric range from u_i to u_{i+r}. This is then expressed recursively in terms of two B-spline basis functions of order $r-1$. Since the order of these basis functions is one less than that of the original, the parametric range over which each of these lower order basis functions is nonzero will be exactly one interval less than that for the original basis function. Specifically, one of the lower order basis functions will be nonzero in the parametric range from u_i to u_{i+r-1} while the other basis function will be nonzero in the parametric range from u_{i+1} to u_{i+r}. Each of these lower order basis functions is multiplied by a

† A *divided difference* is an operator applied to a function that can be viewed as a discrete analogue of a derivative.

‡ A *one-sided power function* is a function whose value is set to zero when its argument is negative.

factor that is linear in the parameter u. Thus, this expression is a sum of two terms, each of which is a product of a linear factor times a piecewise polynomial of order $r-1$; this combines to yield a piecewise polynomial of order r which will be the original basis function. The two terms of the recurrence are presented pictorially in Figures 9 and 10. In each figure, the dashed line represents the linear factor that is applied and the solid curve represents an $r-1^{st}$ order B-spline basis function. The recurrence relation takes the product of the dashed line times the solid curve for each of these cases and then adds the two results to yield the B-spline basis function of order r.

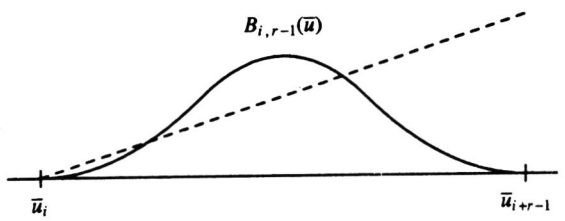

Figure 9:
The lower-indexed term of the recurrence.

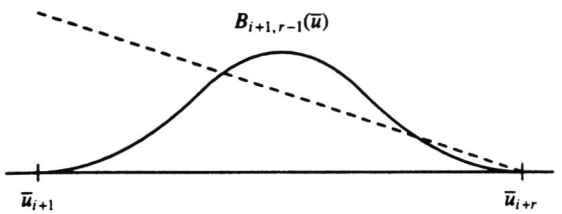

Figure 10:
The higher-indexed term of the recurrence.

This recurrence relation can handle the evaluation of a B-spline basis function for any legal knot vector. Recall that the only restrictions imposed on a knot vector are that it be nondecreasing and that no knot have multiplicity greater than the order of the B-spline. The only case that is not straightforward occurs when a knot has multiplicity equal to the order. In that case, either one or the other of the two $r-1^{st}$ order basis functions is defined over a set of knots all of which have the same value. Hence, this basis function is vacuous, and consequently that term in the recurrence is dropped. Because the multiplicity cannot exceed the order, at most one of the two terms can be dropped in this manner; it is impossible for both terms to be vacuous.

The derivation of the recurrence relation is based on factoring out a linear term from the divided difference formulation of the B-spline basis function and taking the divided difference of this product by invoking the Leibinz rule for divided differences.

5.2. Subdivision

Similar to Bézier subdivision, it is possible to subdivide a B-spline to express the same curve or surface in terms of a larger number of control vertices. As the number of control vertices is increased, so are the number of joints, knots, and basis functions. In the process, some of the existing vertices and basis functions will be modified.

Knot insertion refers to the process of adding a single knot. More generally, *refinement* refers to adding an arbitrary number of new knots to an existing knot vector such that the result is still a valid knot vector (nondecreasing and no knot having multiplicity exceeding the order). *Splitting* occurs when a k-fold knot is added, thereby breaking the curve or surface into two pieces.

To study the subdivision process, consider a knot vector that is refined to form a new knot vector. Since this refinement process is effected solely by sprinkling new knots amongst the original knots, it must be the case that all the original knots are still present in the refined knot vector, although they will be indexed differently. Since the basis functions are constructed over a given knot vector, there must be a different set of basis functions for the original knot vector than there is for the refined knot vector. These two sets of basis functions will be referred to as the "original" basis functions and "new" basis functions, respectively. Since the refined knot vector contains additional knots, there will be more new basis functions than there were original basis functions. More precisely, the difference in the number of new basis functions and old basis functions is equal to the number of additional knots. However, this is not to say that there is simply this number of new basis functions introduced because some of the original basis functions will also be modified. Now, it is the case that any of the old basis functions can be expressed as a weighted average of new basis functions. These weights turn out to be a very special set of numbers known as *discrete B-splines* and are sometimes referred to as the α's. In this notation, $\alpha_{i,k}(j)$ is the weight applied to the j^{th} new basis function in computing the i^{th} old basis function, where k is the order. Although the number of basis functions has increased, the order has not been affected. Note that this result expresses an *old* basis function in terms of *new* basis functions, not the other way around. However, this result can be used to derive a similar expression, again using the α's as weights, that will yield the new control vertices in terms of the original control vertices. This is accomplished by substituting the weighted average of new basis functions in for the old basis functions in the weighted average for the

curve and then setting the result equal to a different weighted average for the curve involving new control vertices and new basis functions. Equating coefficients of the new basis functions in both expressions results in an expression for each new control vertex expressed as a weighted average of the original control vertices. The weights are again the α's where $\alpha_{i,k}(j)$ is the weight applied to the i^{th} old control vertex in the weighted average for the j^{th} new control vertex.

The consequence of this is that if the α's were known, then it would be feasible to compute new control vertices for a refined knot vector that would define the same curve as the original control vertices with the original knot vector. This naturally leads to the question of how the α's can be computed.

It is possible to derive a recurrence relation for the α's that bears a striking resemblance to the Cox / de Boor Algorithm for the evaluation of B-spline basis functions. There are a variety of proofs for this result, some of which are more complicated than others. But regardless of the proof, the result is quite elegant. Similar to the Cox / de Boor recurrence, it is possible to express a discrete B-spline in terms of two lower order discrete B-splines. More precisely, $\alpha_{i,r}(j)$ can be written in terms of $\alpha_{i,r-1}(j)$ and $\alpha_{i+1,r-1}(j)$. It is important to remember that this is a discrete equation, not a continuous equation as was the case for the Cox / de Boor recurrence. The weighting factors are not linear but are simply numbers formed as ratios involving particular knot values from both the original and refined knot vectors. Similar to the Cox / de Boor Algorithm, the only subtlety occurs when there is an r-fold knot in which case the term involving the discrete B-spline that is defined over a vacuous interval would be dropped from the recurrence.

The discrete B-splines have properties analogous to the continuous B-spline basis functions. They are all nonnegative. For a given j, there are at most k $\alpha_{i,k}(j)$'s that are nonzero, and these sum to unity. Because of the role of the $\alpha_{i,k}(j)$ as weights in the subdivision process, the above properties have interesting interpretations for subdivision. It must be the case that each new vertex is the weighted average of at most k of the original control vertices and, furthermore, this new vertex will be in the convex hull of these same k original control vertices.

From the point of view of the new control vertices, it is of interest to substitute the recurrence for the α's into the expression for a new vertex in terms of the original vertices which consequently yields a recurrence for the control vertices themselves. Using this recurrence, it is possible to directly compute the new control vertices given the original control vertices, the original knot vector, and the refined knot vector. These two recurrence relations, one for the α's and one for the control vertices, are both often referred to as the "Oslo Algorithm."

6. Geometric Continuity

As was mentioned in the introduction to this paper, there are many subtleties involved in establishing continuity constraints for parametric splines. Traditionally, the continuity constraints that have been used have been the same as for nonparametric splines, except simply transposed into a parametric form. In other words, it has been the derivatives that have been constrained to be continuous. For the parametric representation, the derivatives in question are parametric derivative vectors, where each component is the derivative of a coordinate with respect to the parameter. Because this form of derivative is fundamentally different than the scalar-valued functional derivatives to which we are accustomed, our intuition regarding the associated geometry often fails. As an example, it is possible to parametrize a piecewise representation of a circle to have a discontinuity in the second derivative vector, although the curvature (and curvature vector) would be continuous. This leads to the idea that it would be of interest to constrain the unit tangent and curvature vectors to be continuous rather than the first and second derivative vectors. We have named this form of continuity as *geometric continuity*. In the case of constraining the unit tangent and curvature vectors to be continuous, we refer to this as G^2 continuity. We call the traditional method of maintaining continuous parametric derivative vectors as *parametric continuity* to distinguish this from the geometric continuity approach.[4,5]

In addition to being a more appropriate measure of continuity for parametric curves, geometric continuity has the advantage that it is a more relaxed form of continuity. The continuity constraints that it generates are generalizations of the continuity constraints for parametric continuity. These more general geometric continuity constraints liberate some degrees of freedom that can be captured to provide further control of shape. These are the *shape parameters* that will be described in the next section.

Having defined geometric continuity of order two (G^2), it is of interest to investigate generalizing to higher order. How can we define geometric continuity of order n, for an arbitrary n? The key to our answer to this is the observation, which was made in the introduction to this paper, that many different parametrizations can describe the same curve.[5] Two regular† C^n parametrizations are said to be *equivalent*

† A parametrization is *regular* if its first derivative vector never vanishes.

if there exists a regular C^n function that is regular and onto and that when composed with one of the parametrizations yields the other one. Intuitively, equivalent parametrizations trace out the same set of points in the same order. Thus, it is possible to alter the parametrization of a curve without changing its shape; this is referred to as *reparametrization.*

From this observation, we are now ready to define geometric continuity for arbitrary order n. Two regular C^n parametrizations meet with n^{th} order geometric continuity, denoted G^n, if it is possible to reparametrize one of the parametrizations such that it would meet the other parametrization with C^n continuity. Although this does provide a definition of geometric continuity of arbitrary order, it is not very practical because it still leaves open the question of how to determine whether or not such a reparametrization exists. Based on this definition of geometric continuity of arbitrary order n and using the idea of composition for the equivalent parametrizations yields equations for the first n derivatives in terms of two different parameters. Performing the prescribed differentiation requires invoking the chain rule at each level of differentiation. The resulting equations involve the first n derivatives of one parameter with respect to the other. Denoting the j^{th} such derivative at the joint by β_j yields the so-called *Beta-constraints.* These β's are the same β's that appear as shape parameters in the Beta-spline, as we will see in the next section. The Beta-constraints provide a set of necessary and sufficient conditions that two parametrizations meet with G^n continuity; specifically, two parametrizations meet with G^n continuity if and only if there exist numbers $\beta_1, \cdots, \beta_n$ that satisfy the Beta-constraints‡.

From this, it can be seen that one use of the Beta-constraints is to determine if two curve segments meet with G^n continuity. Another and more important use of the Beta-constraints is to provide a foundation upon which to construct a family of basis functions. Because geometric continuity requires that there exist some set of β's for which the Beta-constraints are satisfied, this allows a wide range of shapes that will all be geometrically continuous as the values of the β's are changed. This is one of the underlying ideas of the Beta-spline approach; this representation is discussed in the following section.

‡ To maintain the orientation preserving property, β_1 is constrained to be positive.

7. Beta-splines

Similar to the B-spline, the Beta-spline is an approximation technique that produces a piecewise curve defined by a control polygon. However, its curve segments meet with geometric continuity rather than parametric continuity. Recall that the geometric continuity constraints are more general than those for parametric continuity and that this generality appears in the form of extra degrees of freedom that are captured in the shape parameters or β's. These shape parameters can then be made available to a designer in a geometric modeling system environment. By adjusting their values, the shape parameters can be used as additional means for shape control, producing predictable and intuitive changes in shape.

To develop a spline formulation based on geometric continuity requires the construction of a set of basis functions that satisfy the Beta-constraints. The subtlety here is that these basis functions will be functions not only in terms of the parameter u, but also in terms of the shape parameters or β's. Consequently, the derivation of these basis functions requires symbolic, not numeric, computation and consequently a computer algebra system was employed for this purpose.

It has been shown that G^n basis functions exist for any n. However, like all the other spline formulations, the degree usually used is three. A G^n Beta-spline involves shape parameters $\beta_1, \cdots, \beta_n$ and is of degree $n+1$. Let us restrict our attention to cubic Beta-splines. In the case of cubics, the Beta-spline is G^2 and has the two shape parameters β_1 and β_2.

The simplest form of a cubic Beta-spline is one in which each of the two shape parameters has a single value across the entire curve or surface. This is known as a *uniformly-shaped* Beta-spline. This simple case has the advantage that for a given value of each shape parameter, all the basis functions are translates of each other. Also, it is possible to factor the expressions for this canonical basis function to allow for more efficient evaluation.

Consider the effect on the shape of the curve as each of the two shape parameters is varied. The β_1 shape parameter produces asymmetric results and is called the *bias.* It controls the relative influence of the unit tangent vector on the two segments meeting at a joint. Depending on the value of β_1, the unit tangent vector has greater influence on one segment or the other. In particular, a value greater than one means more influence on the segment having higher parametric values (what is usually referred to as the "rightmost" segment). That is, the curve will "continue in the direction of the tangent" farther in this segment. A value between zero and one has the

reciprocal effect, causing the curve to lie closer to the tangent on the other side of the joint. Figure 11 shows a sequence of curves with increasing values of β_1. Note also that increasing values of β_1 cause the curve to shift towards lower parametric values, that is towards the "left," and vice versa for decreasing values of β_1.

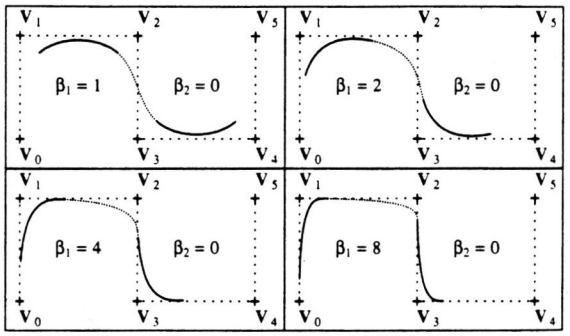

Figure 11:
Increasing β_1 on a uniformly-shaped curve.

The β_2 shape parameter is called *tension* and is symmetric in nature. Increasing the value of β_2 causes each curve segment to become flatter and the shape of the curve around each joint to become sharper. In addition, each joint moves towards a corresponding control vertex. In fact, it can be shown that the locus of a successive set of joints for increasing β_2 is a straight line segment towards the corresponding control vertex. In the limit of infinite tension, the curve approaches the control polygon. One of the interpretations of this shape parameter is that it provides some control over how close the approximating curve will be to the control polygon. Figure 12 shows a sequence of curves for increasing β_2. Unlike β_1, there is no restriction that β_2 be positive; indeed, negative values of β_2 can produce some very interesting effects.[4]

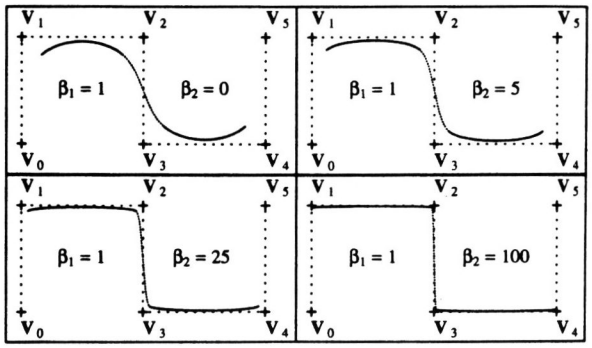

Figure 12:
Increasing β_2 on a uniformly-shaped curve.

Since the Beta-spline has local control with respect to vertex movement, it is of interest to generalize the uniformly-shaped Beta-spline to allow for local control with respect to the shape parameters. There are two different ways to accomplish this, each with its advantages and disadvantages. These two techniques are called the *continuously-shaped* Beta-spline and the *discretely-shaped* Beta-spline.

In the continuously-shaped case, each shape parameter is replaced by a corresponding function that attains a given set of values at each joint and is sufficiently smooth so as not to destroy the geometric continuity of the curve. In this manner, there is a value of each shape parameter at *any* parametric value along the curve.

In the discretely-shaped approach, each shape parameter can take on a given value at each joint, but there is no notion of a value of the shape parameter anywhere else along the curve. This technique results from deriving the basis functions with different values of the shape parameters at each knot.

It is of interest to note that Beta-splines can be represented by B-splines with more vertices. The B-spline representation of cubic Beta-splines is discussed in Chapter 19 of.[5] Cubic Beta-splines are represented by triple-knot B-splines with three times as many control vertices as in the original cubic Beta-spline.

Because of the generality of Beta-splines, some results that would be analogous to those for B-splines are difficult to determine. These are providing some challenging problems for researchers in this field.

8. Rational Splines

Recently, there has been increasing interest in the *rational* variety of polynomials and splines, especially that of the nonuniform B-spline. The term "rational" refers to the ratio which characterizes this approach; for example, a nonuniform rational B-spline curve (sometimes dubbed a "NURB") is the ratio of two nonuniform B-spline curves.

The key advantage of the rational form is its ability to represent conic[14,22,34] curves and quadric surfaces; moreover, this form can also represent free-form curves and surfaces. The latter fact can be seen by noting that the *integral* or non-rational variant is always available as a special case by easily arranging that the denominator be unity. A further advantage is that a rational formulation is invariant under projective transformation (in addition to affine transformation, as is the case for the integral counterpart). Since perspective is a projection, this property can be exploited to generate a perspective projection of a rational curve or surface without resorting to applying

this projection for every point to be displayed, as is the case for the integral version. Additionally, there are weights which can be used to control shape in a manner similar to shape parameters.

A rational spline can be viewed as an integral spline in a vector space whose dimension is one higher than that of the space of the rational Beta-spline. This "next higher" dimensional space is referred to as the *homogeneous coordinate space*. For details on homogeneous coordinates, the reader is referred to.[19,26,27,28,31]

Denoting the dimension of the space of the rational spline by N, then in this scheme, the "extra" coordinate is used as a denominator for the first N coordinates. When each coordinate is a polynomial, the $N+1$ coordinates taken together can be interpreted as N rational polynomial coordinates each sharing the same denominator. A rational spline in $\mathbf{R}^N$ is then the projection of an integral spline in the corresponding homogeneous coordinate space, $\mathbf{R}^{N+1}$.

For illustrative purposes, consider a rational curve in the plane, that is, $N=2$. This rational curve in two dimensions is defined in a three-dimensional space represented by homogeneous coordinates. The third coordinate is the *weight* and is assumed to be positive. There is a distinct weight associated with each vertex. The three-dimensional curve can be projected to two dimensions yielding a *rational* curve.

An illustration of how a rational curve is the projection of an integral curve from a vector space of one higher dimension is provided in Figure 13. The curve drawn dashed is an integral Beta-spline in $\mathbf{R}^3$, while the curve drawn solid is a rational Beta-spline in $\mathbf{R}^2$. The solid curve is the projection of the dashed curve.

The rational form provides a unified representation for free-form curves and surfaces along with conic sections and quadric surfaces, is invariant under projective transformation, and possesses weights which can be used to control shape in a manner similar to shape parameters.

9. Conclusion

Parametric splines provide a flexible mechanism for representing complicated curved shapes. Such a capability is becoming increasingly important with the advances in image synthesis. Proposals for graphics standards such as PHIGS+, PEX, etc., provide an impetus for the inclusion of more sophisticated mathematical tools for curve and surface modelling.

This paper has provided a descriptive explanation of some of the more common spline formulations used in computer graphics, without attempting to be rigorous. The topics of subdivision and geometric continuity were also explained. The particular techniques discussed were piecewise Hermite interpolation, spline interpolation, Bézier curves, B-splines, Beta-splines, and rational splines. It should be noted that there are still many other spline representations that are not included among these. Since there are literally thousands of reference papers in this eclectic field, the references provided are restricted only to books on the subject. The reader is referred to these books for further details on many of these topics.

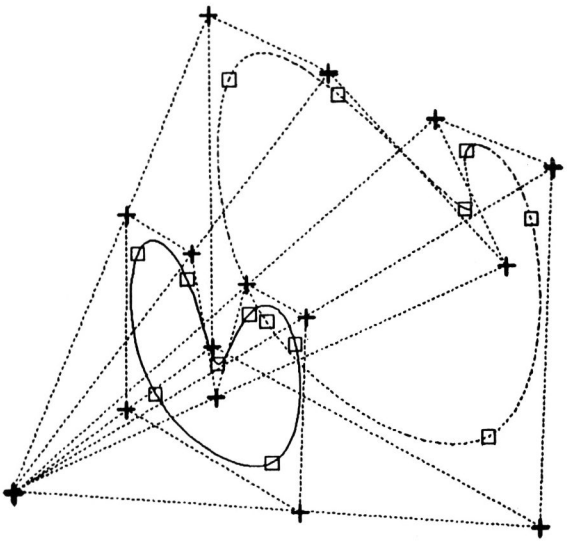

Figure 13:
Rational curve is the projection of an integral curve.

Acknowledgements

This work was supported in part by a National Science Foundation Presidential Young Investigator Award (number CCR-8451997). Figures 1 to 12 in this paper are reprinted from .[4,5]

Reference List of Relevant Books

[1] Robert E. Barnhill and Wolfgang Boehm, *Surfaces in Computer Aided Geometric Design*, North-Holland, Amsterdam, 1983. Conference held 25-30 April 1982 in Oberwolfach, F.R.G.

[2] Robert E. Barnhill and Wolfgang Boehm, *Surfaces in CAGD '84*, North-Holland, Amsterdam, 1985.

[3] Robert E. Barnhill and Richard F. Riesenfeld, *Computer Aided Geometric Design*, Academic Press, New York, 1974.

[4] Brian A. Barsky, *Computer Graphics and Geometric Modeling Using Beta-splines*, Springer-Verlag, Heidelberg, 1988.

[5] Richard H. Bartels, John C. Beatty, and Brian A. Barsky, *An Introduction to Splines for Use in Computer Graphics and Geometric Modeling*, Morgan Kaufmann Publishers, Inc., San Mateo, California, 1987.

[6] Pierre E. Bézier, *Emploi des machines à commande numérique*, Masson et Cie., Paris, 1970. Translated by Forrest, A. Robin and Pankhurst, Anne F. as *Numerical Control -- Mathematics and Applications*, John Wiley and Sons, Ltd., London, 1972.

[7] Pierre E. Bézier, *The Mathematical Basis of the UNISURF CAD System*, Butterworths Scientific Ltd., Guildford, Surrey, England, May 1986.

[8] Pierre E. Bézier, *Mathématiques et CAO 4: Courbes et surfaces*, Editions Hermès, Paris, 1987.

[9] Carl de Boor, *A Practical Guide to Splines*, Springer-Verlag, 1978.

[10] Kenneth W. Brodlie, *Mathematical Methods in Computer Graphics and Design*, Academic Press, New York, 1980.

[11] Paul de Faget de Casteljau, *Mathématiques et CAO 2: Formes à poles*, Editions Hermès, Paris, 1985.

[12] Paul de Faget de Casteljau, *Mathematics and CAD Volume 2: Shape Mathematics and CAD*, Kogan Page Ltd., London, 1986.

[13] Charles K. Chui, *Multivariate Splines*, American Mathematical Society, Providence, Rhode Island, 1989.

[14] J. L. Coolidge, *A History of the Conic Sections and Quadric Surfaces*, Oxford University Press, 1945.

[15] Philip J. Davis, *Interpolation and Approximation*, Ginn-Blaisdell, New York, 1963. Republished by Dover Publ., 1975.

[16] Ding Qiulin and B. J. Davies, *Surface Engineering Geometry for Computer-Aided Design and Manufacture*, Ellis Horwood Ltd., Chichester, 1987.

[17] Gerald Farin, *Geometric Modeling: Algorithms and New Trends*, SIAM, 1987.

[18] Gerald Farin, *Curves and Surfaces for Computer Aided Geometric Design*, Academic Press, Boston, 1988.

[19] Ivor D. Faux and Michael J. Pratt, *Computational Geometry for Design and Manufacture*, Ellis Horwood Ltd., Chichester, 1979.

[20] Peter C. Gasson, *Geometry of Spatial Forms*, Ellis Horwood Ltd., Chichester, 1983.

[21] John A. Gregory, *The Mathematics of Surfaces*, Clarendon Press, Oxford, 1986. Conference held 17-19 September 1984 at the University of Manchester.

[22] Christoph M. Hoffmann, *Geometric and Solid Modeling*, Morgan-Kaufmann, San Mateo, California, 1989.

[23] Peter Lancaster and Kestutis Salkauskas, *Curve and Surafce Fitting: An Introduction*, Academic Press Ltd., London, 1986.

[24] Eric A. Lord and C. B. Wilson, *Geometry of Spatial Forms*, Ellis Horwood Ltd. , Chichester, 1984.

[25] Tom Lyche and Larry L. Schumaker, *Mathematical Methods in Computer Aided Geometric Design*, Academic Press, Inc., Boston, 1989. Conference held 16-22 June 1988 in Oslo.

[26] E. A. Maxwell, *Methods of Plane Projective Geometry Based on the Use of General Homogeneous Coordinates*, Cambridge University Press, Cambridge, England, 1946.

[27] E. A. Maxwell, *Co-ordinate Geometry with Vectors and Tensors*, Oxford University Press, 1958.

[28] E. A. Maxwell, *General Homogeneous Coordinate in the Space of Three Dimensions*, Cambridge University Press, London, 1951, 1959.

[29] Michael E. Mortenson, *Geometric Modeling*, John Wiley and Sons , New York, 1985.

[30] Anthony W. Nutbourne and Ralph R. Martin, *Differential Geometry Applied to Curve and Surface Design Volume 1: Foundations*, Ellis Horwood Ltd., Chichester, 1988.

[31] Michael A. Penna and Richard R. Patterson, *Projective Geometry and its Applications to Computer Graphics*, Prentice-Hall, Englewood Cliffs, 1986.

[32] Paddy M. Prenter, *Splines and Variational Methods*, John Wiley and Sons, Inc., New York, 1975.

[33] David F. Rogers and J. Alan Adams, *Mathematical Elements for Computer Graphics*, McGraw-Hill, New York, 1990. Second edition. (First edition published in 1976.)

[34] G. Salmon, *A Treatise on Conic Sections*, Longmans, Green, & Co., London, 1879. Sixth edition. Reprinted by Dover Publications Inc., New York.

[35] Martin H. Schultz, *Spline Analysis*, Prentice-Hall, Inc., Englewood Cliffs, N.J., 1973.

[36] Larry L. Schumaker, *Spline Functions: Basic Theory*, John Wiley and Sons, Inc., New York, 1981.

[37] Su Bu-quing and Liu Ding-yuan, *Computational Geometry -- Curve and Surface Modeling*, Academic Press, Inc., San Diego, 1989. Translated by Chang Geng-zhe.

[38] Fujio Yamaguchi, *Curves and Surfaces in Computer Aided Geometric Design*, Springer-Verlag, Berlin, 1988.