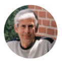
A Tutorial Introduction to Blossoming

Article · January 1991

DOI: 10.1007/978-3-642-76404-2_12

3 authors, including:

T. Derose
Pixar Animation Studios
105 PUBLICATIONS **17,558** CITATIONS
SEE PROFILE



Ronald N. Goldman
Rice University
244 PUBLICATIONS **4,547** CITATIONS
SEE PROFILE

A Tutorial Introduction to Blossoming

Tony deRose
Michael Lounsbery
Department of Computer Science & Engineering FR-35
University of Washington
Seattle, WA 98195

Ronald Goldman
Department of Computer Science
University of Waterloo
Waterloo, Ontario
Canada N2L 3GI

Abstract

A powerful new technique for analyzing Bézier and B-spline curves and surfaces has been developed recently by Ramshaw and de Casteljau. The method, called *blossoming* or *polarization*, is based on an old idea from multilinear algebra where polynomials are studied by replacing them with simple multivariable functions. The purpose of these notes is to provide a brief tutorial introduction to the basic concepts and uses of the technique. Several implementation consequences are also identified.

1 Introduction

The theory of *blossoms*, or *polar forms*, is a highly geometric way to view much of the field of computer aided geometric design (CAGD)[11, 12]. Blossoming provides a simple yet powerful tool for deriving many of the fundamental properties of common curve and surface paradigms. The theory also unifies these paradigms, and provides a solid base upon which to build a geometric modeling system.

Our approach will be fairly informal, stressing intuition rather than rigor; more rigorous treatments of this material can be found elsewhere [5, 6, 11, 12, 14]. In the following, we assume that the reader is familiar with the standard properties of Bézier curves and B-splines. An introduction to this material can be found in [2] and [9]. We begin by reviewing some of the necessary geometric concepts in Section 2. In Section 3, we motivate the definition of blossoms by examining in detail the case of quadratic Bézier curves. The extension of these ideas to higher order curves is presented in Sections 4.1-4.4. B-splines are treated in Sections 4.5-4.7, and a short discussion of surfaces appears in Section 5. We conclude in Section 6 with some remarks on how blossoming can be used as an effective programming tool.

2 Geometric Preliminaries

We outline here the basic geometric facts necessary for manipulating blossoms. For a more complete introduction, consult [7] or [9].

We will be working in the context of *affine spaces.* Intuitively, affine spaces are a slight generalization of the more familiar Euclidean spaces. For our current purposes, it is sufficient to think of an affine space as a collection of points that is closed under *affine combinations.* An affine combination of points $\mathbf{p}_1, \ldots, \mathbf{p}_n$ is an expression of the form

$$\alpha_1\mathbf{p}_1 + \cdots + \alpha_n\mathbf{p}_n, \tag{1}$$

where $\alpha_1, \ldots, \alpha_n$ are real numbers that sum to one.

Affine combinations have nice geometric interpretations. For example, if $\mathbf{p} = \alpha_1\mathbf{p}_1 + \alpha_2\mathbf{p}_2$, then $\mathbf{p}$ lies on the line segment $\mathbf{p}_1\mathbf{p}_2$ so as to break the segment into subsegments of relative lengths $\alpha_2 : \alpha_1$, as shown in Figure 1.

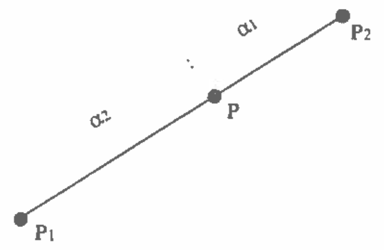


Figure 1: The geometric interpretation of the affine combination $\mathbf{p} = \alpha_1\mathbf{p}_1 + \alpha_2\mathbf{p}_2$

An affine combination of three points has a similar interpretation, except that ratios of areas are used instead of ratios of lengths. Thus, if $\mathbf{p} = \alpha_1\mathbf{p}_1 + \alpha_2\mathbf{p}_2 + \alpha_3\mathbf{p}_3$, then the ratios of the areas of the subtriangles $\Delta_1, \Delta_2, \Delta_3$ in Figure 2 are $\alpha_1 : \alpha_2 : \alpha_3$.

Figure 2 illustrates that if the points $\mathbf{p}_1\mathbf{p}_2\mathbf{p}_3$ form a triangle, then other points $\mathbf{p}$ can be written as affine combinations of $\mathbf{p}_1$, $\mathbf{p}_2$, and $\mathbf{p}_3$. In fact, every other point $\mathbf{p}$ in the plane of $\mathbf{p}_1\mathbf{p}_2\mathbf{p}_3$ can be written *uniquely* as an affine combination of these points. If $\mathbf{p} = \alpha_1\mathbf{p}_1 + \alpha_2\mathbf{p}_2 + \alpha_3\mathbf{p}_3$, the coefficients $\alpha_1, \alpha_2, \alpha_3$ are the *barycentric coordinates* of $\mathbf{p}$ relative to the *domain triangle* $\mathbf{p}_1\mathbf{p}_2\mathbf{p}_3$. The notion of barycentric coordinates extends naturally to affine spaces of arbitrary dimension, with the generalization of domain triangles being domain simplexes.

In addition to affine combinations, *affine maps* play a central role in the theory of blossoms. An affine map is defined in much the same way as a linear map. In particular, a map $T : X \to Y$ between affine spaces X and Y is said to be an affine map if it preserves affine combinations. That is, if $\mathbf{x}_1, \ldots, \mathbf{x}_k$ are points in X, then

$$T(\alpha_1\mathbf{x}_1 + \cdots + \alpha_k\mathbf{x}_k) = \alpha_1 T(\mathbf{x}_1) + \cdots + \alpha_k T(\mathbf{x}_k)$$

must hold for all k, for all choices of $\mathbf{x}_1, \ldots, \mathbf{x}_k$, and for all sets of coefficients $\alpha_1, \ldots, \alpha_k$ that sum to one.

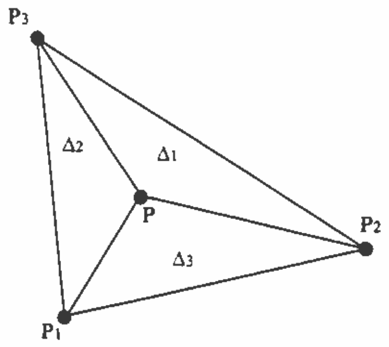


Figure 2: The geometric interpretation of the affine combination $p = \alpha_1 p_1 + \alpha_2 p_2 + \alpha_3 p_3$.

3 Motivation

To motivate some of the ideas underlying blossoming, let us begin by examining quadratic Bézier curves, an example of which is shown in Figure 3. Recall that they are defined by the formula

$$Q(u) = \sum_{i=0}^{2} v_i \cdot B_i^2(u),$$

where $B_i^2(u)$ are the quadratic Bernstein polynomials

$$B_i^2(u) = \binom{2}{i} u^i (1-u)^{2-i}.$$

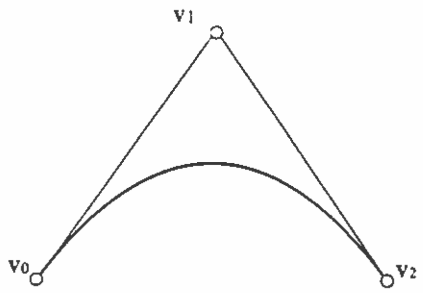


Figure 3: A quadratic Bézier curve

It is well known that for a fixed value of u, the point on the curve $Q(u)$ can be computed using de Casteljau's algorithm (cf. [9]). For quadratics, de Casteljau's algorithm may be stated as:

$$\begin{aligned} v_0^1 &\Leftarrow (1-u)v_0 + uv_1 \\ v_1^1 &\Leftarrow (1-u)v_1 + uv_2 \\ v_0^2 &\Leftarrow (1-u)v_0^1 + uv_1^1 \\ Q(u) &\Leftarrow v_0^2 \end{aligned}$$

The standard proof for de Casteljau's algorithm proceeds by induction by establishing and exploiting a recurrence relation for the Bernstein polynomials. As is often the case with inductive proofs, this derivation of de Casteljau's algorithm suffers from a lack of intuition. Blossoming provides an alternative proof that yields substantially more insight. To see this, let us look at a somewhat nonintuitive bivariate function $q(u_1, u_2)$ defined by

$$q(u_1, u_2) = \mathbf{v}_0(1 - u_1)(1 - u_2) + \mathbf{v}_1\{u_1(1 - u_2) + u_2(1 - u_1)\} + \mathbf{v}_2 u_1 u_2.$$

Notice that this function q has some interesting properties, namely:

1. q is *symmetric* with respect to its 2 arguments: $q(u_1, u_2) = q(u_2, u_1)$.
2. q is *bi-affine* or *2-affine*: When $a + b = 1$,
   (a) $q(at + bv, u_2) = a\,q(t, u_2) + b\,q(v, u_2)$.
   (b) $q(u_1, at + bv) = a\,q(u_1, t) + b\,q(u_1, v)$.
3. q agrees with Q on its *diagonal*: $q(u, u) = Q(u)$.
4. $q(0,0) = \mathbf{v}_0$
5. $q(0,1) = q(1,0) = \mathbf{v}_1$
6. $q(1,1) = \mathbf{v}_2$

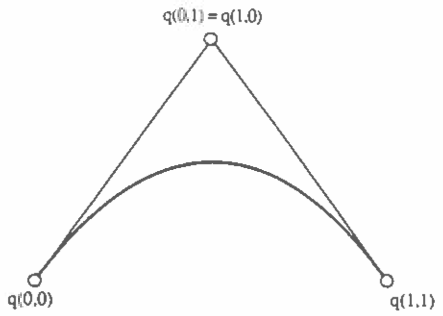


Figure 4: Relabeling of control points.

With observations 4, 5, and 6, we can relabel our diagram of the quadratic Bézier curve as shown in Figure 4. From observations 3-6, we can see that the function q is closely related to the curve Q. q is called a *blossom* (or *polar form*), and is the *polarization* of Q. By working with the blossom of Q instead of with Q itself, we exploit a powerful tool for manipulating Q. The blossom q lets us work singly with small, simple, affine components of Q.

As an example of a simple affine blend, we hold one of the 2 arguments in q constant, and allow the other to vary. Then

$$\begin{aligned} q(u,0) &= q((1-u)\cdot 0 + u \cdot 1, 0) \\ &= (1-u)\,q(0,0) + u\,q(1,0). \end{aligned}$$

showing that the point $q(u,0)$ is an affine blend of the points $q(0,0)$ and $q(1,0)$. As we can see in Figure 5, $q(u,0)$ is the unique point on the line between $q(0,0)$ and $q(1,0)$ that breaks

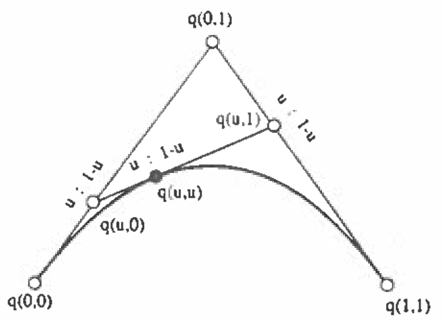


Figure 5: De Casteljau's algorithm for quadratics as blossom evaluation.

the intervals into the ratio $u : 1-u$. Similarly, $q(u,1)$ is on the line between $q(0,1)$ and $q(1,1)$.

Now that we have $q(u,1)$ and $q(u,0)$, we can derive $q(u,u)$ in a similar manner:

$$\begin{aligned} q(u,u) &= q(u,(1-u)\cdot 0 + u\cdot 1) \\ &= (1-u)\,q(u,0) + u\,q(u,1). \end{aligned}$$

This tells us that $q(u,u)$ is at position u on the line between $q(u,0)$ and $q(u,1)$. From property 3, $q(u,u)$ is exactly equivalent to $Q(u)$. Repeated affine combinations of blossom values evaluated with u has led us to the point $Q(u)$ on the original Bézier curve. This is simply a rederivation, using blossom notation, of the de Casteljau algorithm for evaluating quadratic Bézier curves.

Building on simple affine combinations such as these, we will see how to manipulate arbitrarily complex polynomials, both curves and surfaces.

4 Curves

4.1 Blossom Definition

Blossoms for arbitrary degree polynomials are based on the following theorem from multilinear algebra:

Theorem 1 *Let Q be a polynomial of degree d. If $D \geq d$, then there is a unique symmetric D-affine function $q(u_1, \ldots, u_D)$ such that $q(u, \ldots, u) = Q(u)$.*

To be more constructive, let us determine the blossom of the extremely simple polynomial $Q(u) = u^2$, and let us choose D in Theorem 1 to be 3. Thus, we seek a tri-affine function $q(u_1, u_2, u_3)$ such that $q(u,u,u) = u^2$. As an initial attempt, consider the following as a candidate solution:

$$q(u_1, u_2, u_3) = u_1 u_2.$$

This function is affine in each argument and agrees with Q on its diagonal, but is not symmetric. In particular, $q(u_1, u_2, u_3) \neq q(u_3, u_2, u_1)$. We can solve this problem by symmetrizing over the three variables, taken two at a time, to yield

$$q(u_1, u_2, u_3) = \frac{u_1u_2 + u_2u_3 + u_1u_3}{3}.$$

This new function meets all criteria, so by uniqueness it must be the function we seek. Reviewing what we have done, if $Blossom_D()$ represents the mapping from a polynomial to its D-affine blossom, then we have discovered that

$$Blossom_3(u^2) = \frac{u_1u_2 + u_2u_3 + u_1u_3}{3}.$$

Generalizing slightly, it is not difficult to show that

$$Blossom_D(u^k) = \frac{\sum_{i_1,\ldots,i_k} u_{i_1} \cdot u_{i_2} \cdots u_{i_k}}{\binom{D}{k}}, \tag{2}$$

where the summation is taken over all indices $i_1, \ldots, i_k$ such that each index is chosen from the set $\{1, \ldots, D\}$, and such that no two indices are equal.

The operator $Blossom_D()$ is *linear*, meaning that if a and b are real numbers, or, more generally, vectors, and if $P(u)$ and $Q(u)$ are two polynomials, then

$$Blossom_D(aP(u) + bQ(u)) = a\ Blossom_D(P(u)) + b\ Blossom_D(Q(u)).$$

The linearity of $Blossom_D()$, together with Equation 2 implies that the D-affine blossom of an arbitrary polynomial $Q(u) = \sum_k \mathrm{c}_k u^k$ can be expressed as

$$Blossom_D(Q(u)) = Blossom_D(\sum_k \mathrm{c}_k u^k) = \sum_k \mathrm{c}_k \sum_{i_1,\ldots,i_k} \frac{u_{i_1} \cdots u_{i_k}}{\binom{D}{k}}. \tag{3}$$

For example, if $Q(u) = 1 + 2u + 4u^2 - u^3$, then

$$Blossom_3(Q(u)) = 1 + 2\frac{u_1 + u_2 + u_3}{3} + 4\frac{u_1u_2 + u_1u_3 + u_2u_3}{3} - u_1u_2u_3.$$

4.2 Blossoms and Bézier Representations

Given a blossom $q(u_1, \ldots, u_d)$, suppose we want to find the Bézier control points $\mathrm{v}_0, \ldots, \mathrm{v}_d$ of the curve $Q(u) = q(u, \ldots, u)$. To solve this problem, we can expand the first argument of q using the identity $u = (1-u) \cdot 0 + u \cdot 1$, then use the fact that q is affine in its first argument. This gives us

$$Q(u) = (1-u)q(0, u, \ldots, u) + uq(1, u, \ldots, u).$$

Continue this process, recursively expanding the terms $q(0, u, \ldots, u)$ and $q(1, u, \ldots, u)$. When this is done and terms are collected, the resulting expression is

$$Q(u) = \sum_{i=0}^{d} q(0, \ldots, 0, \underbrace{1, \ldots, 1}_{i}) \binom{d}{i} u^i (1-u)^{d-i}.$$

Since a polynomial has a unique set of Bézier control points, we deduce that

$$\mathrm{v}_i = q(0, \ldots, 0, \underbrace{1, \ldots, 1}_{i}).$$

To summarize, given a blossom $q(u_1, ..., u_d)$, the i^{th} Bézier control point of its diagonal polynomial can be extracted by evaluating q at 1 a total of i times, and at 0 a total of $d-i$ times.

This observation can be generalized for Bézier curves parametrized on intervals other than $[0,1]$. More specifically, the relationship between Bézier control points parametrized on an arbitrary interval $[s,t]$ to blossom values is $v_i = q(s, ..., s, \underbrace{t, ..., t}_{i})$.

The above discussion shows that if the blossom is known in the sense that an arbitrary value can be computed, then the Bézier control points arise by evaluating the blossom at simply described values. Consider now the converse problem: given the Bézier control points of a polynomial $Q(u)$, compute an arbitrary value of its blossom $q(u_1, ..., u_d)$. In principle, one could write down an expression similar to Equation 3 in terms of the Bézier control points instead of the power basis coefficients c_k. A simpler approach, however, is to provide an algorithm to compute arbitrary values. Such an algorithm is remarkably simple, as shown in Figure 6. De Casteljau's algorithm can be seen to be a special case of the algorithm of Figure 6, occurring when $u_1 = \cdots = u_d = u$; that is, for the computation of a point on the diagonal.

```
EvaluateBlossom( $v_0, ..., v_d, u_1, ..., u_d$)
for $i = 1$ to $d$ do
    for $j = 0$ to $d - i$ do
        $v_j = (1 - u_i)v_j + u_i v_{j+1}$
        { $v_j$ now equals $q(u_1, ..., u_i, \underbrace{1, ..., 1}_{j}, 0, ..., 0)$ }
    end
end
return $v_0$
```

Figure 6: Evaluation algorithm for an arbitrary blossom value $q(u_1, ..., u_d)$ given the Bézier control points $v_0, ..., v_d$.

4.3 Subdivision

Referring to the de Casteljau algorithm depicted in Figure 7, notice that the points

$$q(0,0,\ldots,0,u), q(0,\ldots,0,u,u), \ldots, q(u,u,\ldots,u)$$

are all computed by the algorithm. These are all of the form $q(s,\ldots,s,\underbrace{t,\ldots,t}_{i})$, where $s = 0$ and $t = u$. The results of Section 4.2 therefore imply that these values are exactly the Bézier control points of Q defined on the interval $[0,u]$. Similarly, the points $q(u,\ldots,u,1,\ldots,1)$ were computed as well. These are precisely the control points of Q defined on the interval $[u,1]$. De Casteljau's algorithm therefore effectively subdivides the original Bézier curve into two subcurves by deriving as a side effect the Bézier control points that characterize each subinterval.

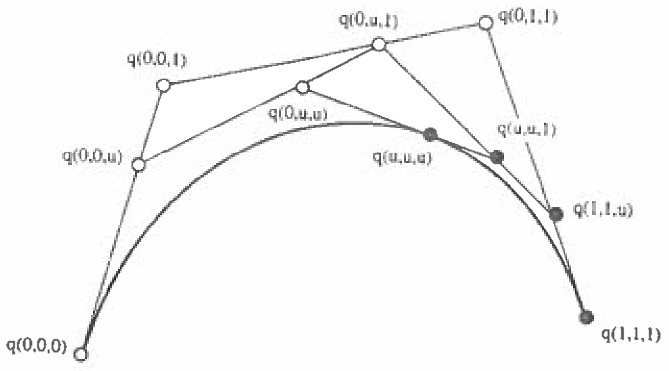


Figure 7: De Casteljau's algorithm for cubics as blossom evaluation.

4.4 Degree Raising

Suppose we have a curve $Q(u) = \sum_{i=0}^{2} \mathbf{v}_i B_i^2(u)$ and we want to find a curve $\tilde{Q}(u) = \sum_{i=0}^{3} \tilde{\mathbf{v}}_i B_i^3(u)$ such that $Q(u) = \tilde{Q}(u)$.

Our method for accomplishing this will be:

1. Find the blossom q of Q.
2. Build the blossom $\tilde{q}$ of Q from q.
3. Evaluate $\tilde{q}$ to extract the $\tilde{\mathbf{v}}_i$.

This process is summarized in Figure 8.

We start with a bi-affine blossom q, fully characterized by $\mathbf{v}_0$, $\mathbf{v}_1$, and $\mathbf{v}_2$. Our goal in degree raising is to find a tri-affine blossom $\tilde{q}$ such that

$$\tilde{Q}(u) = \tilde{q}(u, u, u) = q(u, u) = Q(u).$$

The expression

$$\tilde{q}(u_1, u_2, u_3) = \frac{q(u_1, u_2) + q(u_2, u_3) + q(u_1, u_3)}{3}$$

defines a symmetric tri-affine function that has the proper behavior on the diagonal — by uniqueness, it must therefore be $\tilde{q}$.

We still need to find the control points $\tilde{\mathbf{v}}_i$ that characterize $\tilde{q}$. Now that we have $\tilde{q}$, this is accomplished by simple evaluation:

$$\begin{array}{lllll}
\tilde{\mathbf{v}}_0 & = & \tilde{q}(0,0,0) & = \frac{q(0,0)+q(0,0)+q(0,0)}{3} & = & \mathbf{v}_0 \\
\tilde{\mathbf{v}}_1 & = & \tilde{q}(0,0,1) & = \frac{q(0,0)+q(0,1)+q(0,1)}{3} & = & \frac{\mathbf{v}_0+2\mathbf{v}_1}{3} \\
\tilde{\mathbf{v}}_2 & = & \tilde{q}(0,1,1) & = \frac{q(0,1)+q(1,1)+q(0,1)}{3} & = & \frac{2\mathbf{v}_1+\mathbf{v}_2}{3} \\
\tilde{\mathbf{v}}_3 & = & \tilde{q}(1,1,1) & = \frac{q(1,1)+q(1,1)+q(1,1)}{3} & = & \mathbf{v}_2
\end{array}$$

$$\begin{tikzcd}
\text{Bezier representation} \arrow[r, Rightarrow] \arrow[d, "\text{convert to blossom}"] & \text{Bezier representation} \\
\text{Blossom} \arrow[r, "\text{symmetrize}"'] & \text{Blossom} \arrow[u, "\text{evaluate}"']
\end{tikzcd}$$

Figure 8: Schematic diagram of how blossoms are typically used to solve change of representation problems.

The above procedure for degree raising extends to curves of any degree. In general, raising a degree d Bézier curve Q whose blossom is q to a degree $d+1$ curve $\tilde{Q}$ with blossom $\tilde{q}$ is based upon the symmetric, multiaffine relation

$$\tilde{q}(u_1, \ldots, u_{d+1}) = \frac{q(u_1, u_2, \ldots, u_d) + \cdots + q(u_1, \ldots, u_{i-1}, u_{i+1}, \ldots u_{d+1}) + \cdots + q(u_2, \ldots u_{d+1})}{d+1}.$$

4.5 B-splines

The previous sections have indicated that blossoms are useful for analyzing individual polynomial curve segments. In this section, we explore the use of blossoms to study piecewise curves, i.e., B-splines. As a prerequisite, we must first understand the relationship between derivatives and blossoms.

4.5.1 Blossoms, Derivatives and C^k Continuity

Two curve segments $F : [r, s] \rightarrow X$ and $G : [s, t] \rightarrow X$, where X is an affine space, are said to meet with C^k continuity at s if they have matching derivatives up to order k at s:

$$F^{(i)}(s) = G^{(i)}(s), \quad i = 0 \ldots k,$$

where superscript (i) refers to the i^{th} derivative.

Suppose two segments F and G meet with C^k continuity at s. What can be said about the relationship between their blossoms? To address this question, let us examine the connection between derivatives and blossoms.

Let f denote F's blossom, and let g denote G's. To simplify the discussion, we will also assume that F and G are of common degree d. The definition of $F^{(1)}(s)$ that is most convenient here is

$$\begin{aligned} F^{(1)}(s) &:= \frac{d}{du} F(s+u)|_{u=0} \\ &= \frac{d}{du} f(s+u, s+u, \ldots, s+u)|_{u=0}. \end{aligned}$$

To continue, we need to invoke the chain rule, which in this instance takes the form

$$\frac{d}{du}f(\underbrace{s+u}_{=u_1},\ldots,\underbrace{s+u}_{=u_d}) = \sum_{i=1}^{d}\frac{\partial f}{\partial u_i}(u_1,\ldots,u_d)\|_{u_1=s,\ldots,u_d=s}\,\frac{du_i}{du}|_{u=0}.$$

The terms $\frac{du_i}{du}|_{u=0}$ are simply disposed of, since $u_i = s+u$ implies that $\frac{du_i}{du}|_{u=0} = 1$. Concentrating now on the terms containing a partial derivative of f, we get:

$$\begin{aligned}\frac{\partial f}{\partial u_i}|_{u_1,\ldots,u_d=s}(u_1,\ldots,u_d) &= \frac{\partial}{\partial u_i}((1-u_i)f(u_1,\ldots,u_{i-1},0,u_{i+1},\ldots,u_n)\\ &\qquad +u_if(u_1,\ldots,u_{i-1},1,u_{i+1},\ldots u_d))|_{u_1,\ldots,u_d=s}\\ &= f(u_1,\ldots,u_{i-1},1,u_{i+1},\ldots,u_n)|_{u_1,\ldots,u_d=s}\\ &\qquad -f(u_1,\ldots,u_{i-1},0,u_{i+1},u_d)|_{u_1,\ldots,u_d=s}\\ &= f(1,s,\ldots,s)-f(0,s,\ldots,s).\end{aligned}$$

A similar result holds for other terms containing partial derivatives of f. Putting this together, and using the symmetry of f, we find that:

$$F^{(1)}(s) = n\left(f(1,s,\ldots,s)-f(0,s,\ldots,s)\right).$$

Applying the process once again reveals that

$$F^{(2)}(s) = n(n-1)\left(f(1,1,s,\ldots,s)-2f(1,0,s,\ldots,s)+f(0,0,s,\ldots,s)\right).$$

By iterating this procedure, the general form for the i^{th} derivative can be shown to be

$$F^{(i)}(s) = \frac{n!}{(n-i)!}\sum_{j=0}^{i}(-1)^j\binom{i}{j}f(\underbrace{1,\ldots,1}_{i-j},\underbrace{0,\ldots,0}_{j},s,\ldots s). \tag{4}$$

Coming back to the question of continuity, if F and G meet with C^1 continuity at s, then we have

$$\begin{aligned}F(s) &= G(s)\\ F^{(1)}(s) &= G^{(1)}(s).\end{aligned}$$

Hence, their blossoms satisfy:

$$\begin{aligned}f(s,\ldots,s) &= g(s,\ldots,s)\\ f(1,s,\ldots,s)-f(0,s,\ldots,s) &= g(1,s,\ldots,s)-g(0,s,\ldots,s).\end{aligned} \tag{5}$$

Adding the first equation in Equation 5 to the second, we find that the left side becomes

$$f(1,s,\ldots,s)-f(0,s,\ldots,s)+f(s,s,\ldots,s),$$

which is an affine combination of three points. Since f and g are multiaffine, the sum of the two equations can be written as

$$f(1-0+s,s,\ldots,s) = g(1-0+s,s,\ldots,s).$$

This shows that if F and G meet with C^1 continuity, then their blossoms satisfy

$$\begin{aligned} f(s,\dots,s) &= g(s,\dots,s) \\ f(s+1,s,\dots,s) &= g(s+1,s,\dots,s). \end{aligned} \tag{6}$$

Moreover, since every value x_1 can be written as an affine combination of s and $s+1$, Equation 6 implies that

$$f(x_1,s,\dots,s) = g(x_1,s,\dots,s), \tag{7}$$

for all x_1. This shows that the conditions of Equation 5 imply Equation 7. Since the converse clearly holds as well, we deduce that Equation 7 is a necessary and sufficient condition for C^1 continuity. The generalization for C^k continuity follows quickly from Equation 4. As a theorem, we have:

Theorem 2 Two polynomial segments $F:[r,s]\to X$ and $G:[r,s]\to X$ of degree n, having blossoms f and g respectively, meet with C^k continuity at s if and only if

$$f(x_1,\dots,x_k,s,\dots,s) = g(x_1,\dots,x_k,s,\dots,s)$$

for all $x_1 \dots x_k$.

4.6 B-splines and Piecewise Blossoms

We saw in Section 4.2 that a blossom $f_i(u_1,u_2,u_3)$ is completely determined by the progressive values $f_i(s,s,s)$, $f_i(s,s,t)$, $f_i(s,t,t)$, $f_i(t,t,t)$, since these are the Bézier control points for the diagonal polynomial $F_i(u) = f_i(u,u,u)$ on the interval $[s,t]$. More generally, if t_i, t_{i+1}, t_{i+2}, t_{i+3}, t_{i+4}, t_{i+5} are distinct non-decreasing values called *knots*, then f_i is determined by the *progressive* blossom values $f_i(t_i,t_{i+1},t_{i+2})$, $f_i(t_{i+1},t_{i+2},t_{i+3})$, $f_i(t_{i+2},t_{i+3},t_{i+4})$, $f_i(t_{i+3},t_{i+4},t_{i+5})$. That is, once these values have been specified, any other value of f_i can be computed. To see that this is so, notice that these values can be used to compute the values $f_i(t_{i+2},t_{i+2},t_{i+2})$, $f_i(t_{i+2},t_{i+2},t_{i+3})$, $f_i(t_{i+2},t_{i+3},t_{i+3})$, $f_i(t_{i+3},t_{i+3},t_{i+3})$, as shown in Figure 9. This latter set of values are the Bézier control points for the segment parametrized on the interval $[t_{i+2},t_{i+3}]$. Since the Bézier control points uniquely characterize a blossom, an arbitrary blossom value can then be computed using the algorithm of Figure 6. A more direct approach proceeds by constructing an arbitrary blossom value for a segment of degree d directly from the progressive values

$$\mathrm{v}_{i+j} = f_i(t_{i+j},t_{i+j+1},\dots,t_{i+j+d-1}), \qquad j=0,\dots,d,$$

using the algorithm of Figure 10.

Returning to cubic curves, the values $f_i(t_i,t_{i+1},t_{i+2})$, $f_i(t_{i+1},t_{i+2},t_{i+3})$, $f_i(t_{i+2},t_{i+3},t_{i+4})$, $f_i(t_{i+3},t_{i+4},t_{i+5})$, then, characterize a polynomial $F_i(u) := f_i(u,u,u)$ on the interval $[t_{i+2},t_{i+3}]$. The idea now is to build a piecewise C^2 curve F such that $F(u) := F_i(u) = f_i(u,u,u)$ when $u\in[t_{i+2},t_{i+3}]$. We will do this by building a *piecewise* blossom f. Recall from Theorem 2 that F_{i-1} and F_i meet with C^2 continuity at t_{i+2} if and only if their blossoms satisfy $f_{i-1}(t_{i+2},u,v) = f_i(t_{i+2},u,v)$ for all u,v.

Now we can characterize f_i by the progressive blossom values given above. We can similarly characterize f_{i-1} by the progressive values of three knots, beginning at $t_{i-1}\dots t_{i+4}$.

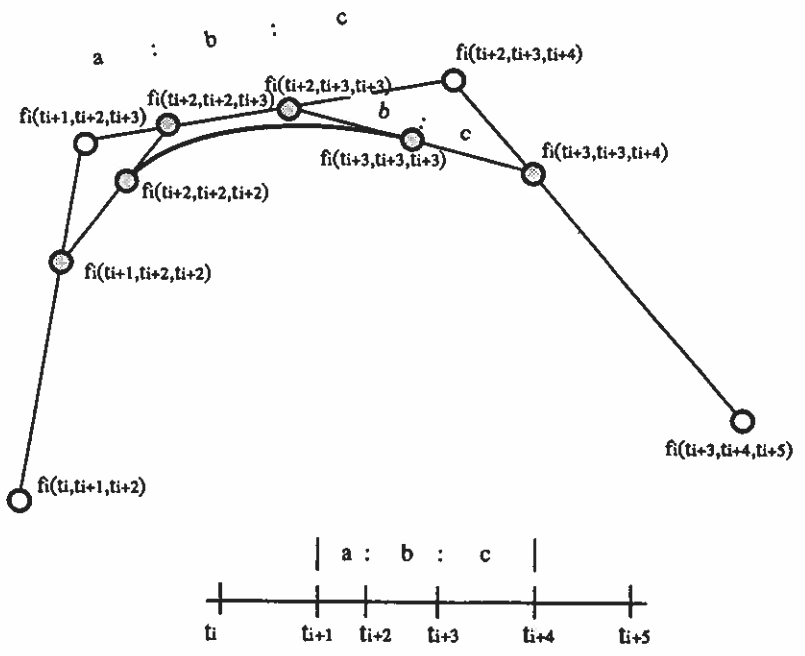


Figure 9: Another way to characterize a curve segment.

We require that f_{i-1} and f_i have values that agree whenever certain knot arguments agree, as given by the following conditions:

$$\begin{aligned}
f_{i-1}(t_i, t_{i+1}, t_{i+2}) &= f_i(t_i, t_{i+1}, t_{i+2}) \\
f_{i-1}(t_{i+1}, t_{i+2}, t_{i+3}) &= f_i(t_{i+1}, t_{i+2}, t_{i+3}) \\
f_{i-1}(t_{i+2}, t_{i+3}, t_{i+4}) &= f_i(t_{i+2}, t_{i+3}, t_{i+4}).
\end{aligned}$$

With these conditions, the C^2 constraints will be satisfied (this is easily verified). Thus, there are only 5 points that determine f_{i-1} and f_i, and hence F_{i-1} and F_i. These points are:

$$\begin{gathered}
f_{i-1}(t_{i-1}, t_i, t_{i+1}) \\
f_{i-1}(t_i, t_{i+1}, t_{i+2}) = f_i(t_i, t_{i+1}, t_{i+2}) \\
f_{i-1}(t_{i+1}, t_{i+2}, t_{i+3}) = f_i(t_{i+1}, t_{i+2}, t_{i+3}) \\
f_{i-1}(t_{i+2}, t_{i+3}, t_{i+4}) = f_i(t_{i+2}, t_{i+3}, t_{i+4}) \\
f_i(t_{i+3}, t_{i+4}, t_{i+5}).
\end{gathered}$$

Moreover, no matter where these 5 points are placed, f_{i-1} and f_i will meet with C^2 continuity at t_i. We may continue by placing another segment f_{i+1}, and another knot t_{i+6}. The result of this construction is seen in Figure 11.

We can improve our notation by dropping the subscripts on the functions and defining

```
EvaluateBlossomProgressive( $v_i, ..., v_{i+d}, t_i, ..., t_{2d-1+i}, u_1, ..., u_d$)
for $k = 0$ to $d-1$ do
    for $\ell = 0$ to $d-k-1$ do
        { $\alpha, \beta$ are barycentric coords of $u_{k+1}$ in $[t_{i+k+\ell}, t_{d+i+k+\ell}]$. }
        $\beta = \frac{u_{k+1} - t_{i+k+\ell}}{t_{d+i+k+\ell} - t_{i+k+\ell}}$
        $\alpha = 1 - \beta$
        $v_{i+\ell} = \alpha\ v_{i+\ell} + \beta\ v_{i+\ell+1}$
        { $v_{i+\ell}$ now equals $q(u_1, ..., u_{k+1}, t_{i+k+l}, \ldots t_{i+d+l-1})$ }
    end
end
return $v_i$
```

Figure 10: Evaluation algorithm for an arbitrary blossom value $q(u_1, ..., u_d)$ given progressive blossom values and knots.

a piecewise blossom $f(u_1, u_2, u_3)$ using the rule:

$$f(u_1, u_2, u_3) = \begin{cases} f_i(u_1, u_2, u_3) & \text{if } u_i\text{'s are not all equal and } u_1, u_2, u_3 \in [t_i, t_{i+5}] \\ f_j(u_1, u_2, u_3) & \text{if } u_i\text{'s are all equal and } u_i \in [t_{j+2}, t_{j+3}]. \end{cases}$$

(Remark: strictly speaking, this isn't strong enough; see Ramshaw[11, 12].) The labeling becomes much simpler now, since we can drop subscripts of f's in Figure 11.

The values $f(t_{i-1}, t_i, t_{i+1}), \ldots, f(t_{i+2}, t_{i+3}, t_{i+4}), \ldots, f(t_{i+4}, t_{i+5}, t_{i+6})$ are the cubic B-spline control points for the *knot vector* $t_{i-1}, \ldots, t_{i+6}$. In fact, when $u_1 = u_2 = \cdots = u_d = u$ the algorithm of Figure 10 is exactly the de Boor algorithm for computing a point on a B-spline (cf. Bartels et al [2] or Farin [9]).

These ideas can be used to define the cubic B-spline design scheme:

Input: A sequence of control points $v_0 \ldots v_m$, a sequence of knot values $t_0 \ldots t_{m+2}$, and a point $u \in [t_2, t_m]$.

Output: A point $F(u)$.

Such that: The curve $F(u)$, $u \in [t_2, t_m]$ is C^2.

Method: Let i be such that $u \in [t_{i+2}, t_{i+3}]$, and set $v_{i+j} = f(t_{i+j}, t_{i+j+1}, t_{i+j+2})$, $j = 0, 1, 2, 3$, then use the de Boor algorithm of Figure 10 to compute the point $f_i(u, u, u) = F(u)$.

We now generalize this construction to an arbitrary degree d: For a general degree d, C^{d-1} continuity at t_{i+d-1} requires that adjacent blossoms must satisfy

$$f_{i-1}(t_{i+d-1}, u_2, \ldots, u_d) = f_i(t_{i+d-1}, u_2, \ldots, u_d)$$

for all $u_2, \ldots, u_d$. This is guaranteed if f_{i-1} and f_i have blossom values that agree at their

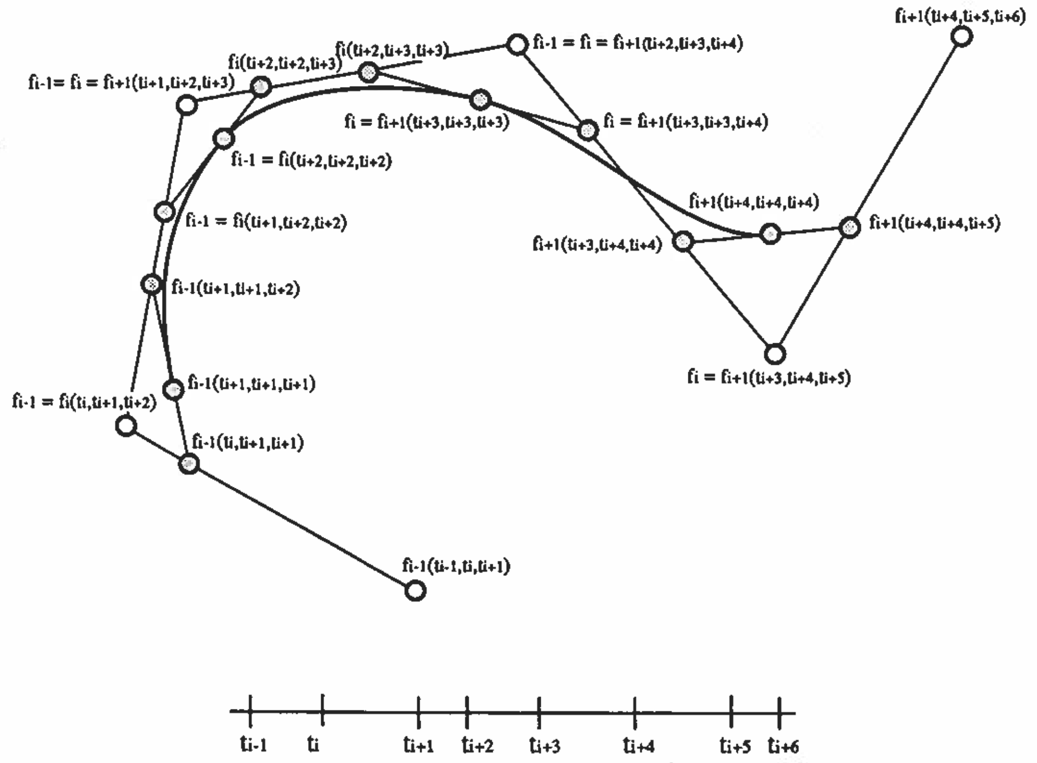


Figure 11: Three segments joining with C^2 continuity.

overlapping knots:

$$\begin{aligned}
f_{i-1}(t_i,\ldots,t_{i+d-1}) &= f_i(t_i,\ldots,t_{i+d-1}) \\
&\vdots \\
f_{i-1}(t_{i+d-1},\ldots,t_{i+2d-2}) &= f_i(t_{i+d-1},\ldots,t_{i+2d-2})
\end{aligned}$$

We can therefore define a piecewise blossom $f(u_1,\ldots,u_d)$ according to:

$$f(u_1,\ldots,u_d) = \begin{cases} f_i(u_1,\ldots,u_d) & \text{if } u\text{'s are not all equal and } u_1,\ldots,u_d\text{'s} \in [t_i, t_{i+2d-1}] \\ f_j(u_1,\ldots,u_d) & \text{if } u\text{'s are all equal and } u \in [t_{j+d-1}, t_{j+d}]. \end{cases}$$

4.7 Knot Insertion

Knot insertion refers to the problem of finding control points for a curve over a knot vector $\hat{T}$, given the control points for the curve over a knot vector T, where $\hat{T}$ is a *refinement* of T; that is, where $T \subset \hat{T}$.

The CAGD literature describes two major knot insertion algorithms: Boehm's algorithm [3], and the Oslo algorithm [4]. The Oslo algorithm is capable of solving the problem no matter how many knots are added to T to obtain $\hat{T}$, whereas Boehm's algorithm is restricted to the case where T and $\hat{T}$ differ by a single knot. (Boehm's algorithm can be used to insert any number of knots by successively inserting one knot at a time.) Boehm's algorithm was originally proved using properties of B-spline basis functions; here we show how Boehm's algorithm is interpreted from the perspective of blossoming. For concreteness we shall restrict the discussion to cubic curves. The general case presents no conceptual difficulties, but the additional notation is sufficiently cumbersome that we omit it here.

Let $F(u)$ be the cubic B-spline curve in question, let $T = \{t_i\}$ be the original knot vector, let $\{\mathbf{v}_i\}$ be the control points of Q over T, and let $\hat{t} \in [t_i, t_{i+1}]$ be the knot to be inserted. From the blossoming point of view, we know that the B-spline control points are obtained by evaluating the blossom f of F at consecutive triples of knots. After $\hat{t}$ is included, the control points

$$\begin{aligned} \mathbf{v}_{i-1} &= f(t_{i-1}, t_i, t_{i+1}) \\ \mathbf{v}_i &= f(t_i, t_i, t_{i+1}) \end{aligned}$$

are no longer valid for $\hat{T}$. Instead, these must be replaced with the new control points

$$\begin{aligned} \hat{\mathbf{v}}_{i-1} &= f(t_{i-1}, t_i, \hat{t}) \\ \hat{\mathbf{v}} &= f(t_i, \hat{t}, t_{i+1}) \\ \hat{\mathbf{v}}_{i+1} &= f(\hat{t}, t_{i+1}, t_{i+2}). \end{aligned}$$

This procedure is summarized in Figure 12.

The Oslo algorithm, while somewhat more complicated due to its generality, can be shown to be equivalent to the evaluation of off-diagonal blossom values using the algorithm of Figure 10.

5 Surfaces

We now turn to the theory of blossoms for surfaces. The generalization to surfaces can be done in two different ways, yielding either tensor product or non-tensor product descriptions. We first examine the tensor product construction.

A bi-d-c tensor product surface $F(u, v)$ takes the form

$$F(u,v) = \sum_{i,j} \mathbf{w}_{i,j} b_i(u) b_j(v),$$

where w's denote the (rectangular array of) control points, and where b's denote a collection of univariate blending functions of degree d. Linearity of the operator $Blossom_D()$ implies that blossoming can occur independently in u and v, creating a function $f(u_1, ..., u_D; v_1, ..., v_D)$ with the properties

1. f is affine in each of its $2D$ argument;
2. f is symmetric with respect to interchange of u's;
3. f is symmetric with respect to interchange of v's;

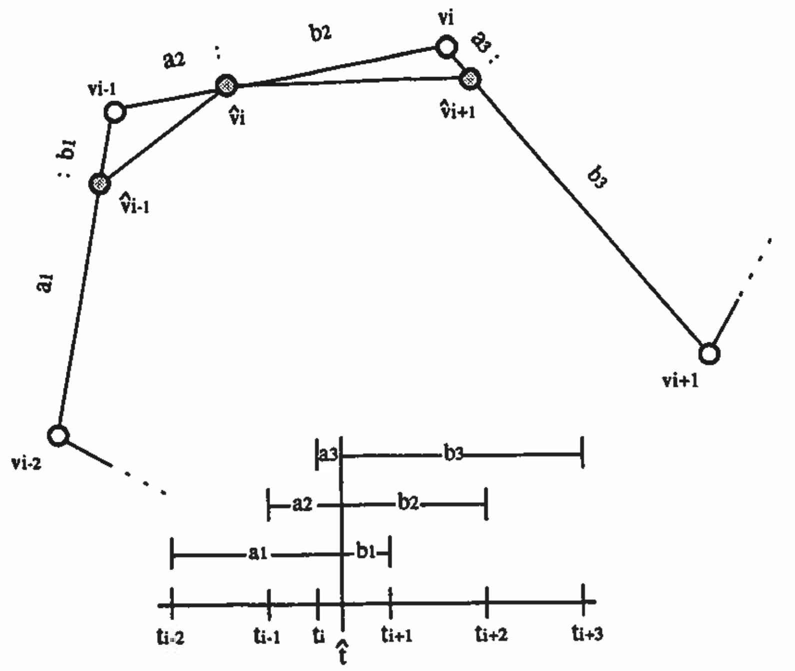


Figure 12: Boehm's knot insertion algorithm.

4. $F(u,v) = f(u,...,u;v,...,v)$.

Note that f is not guaranteed to be symmetric if one of the u arguments is interchanged with one of the v's. For instance, in general

$$f(u_1, u_2, ..., u_D; v_1, v_2, ..., v_D) \neq f(v_1, u_2, ..., u_D; u_1, v_2, ..., v_D).$$

All results of the blossoming theory for curves can be applied to $f(u_1, ..., u_d; v_1, ..., v_D)$, including extraction of Bézier and B-spline points by evaluation, degree raising in u or v, etc.

Non-tensor product surface forms arise when the domain parameter is allowed to range over $\Re^2$ instead of $\Re$. (The following readily generalizes to higher dimensions.) We will map u into a point $Q(u)$ on a surface using the following uniqueness theorem for surfaces:

Theorem 3 Let Q be a bivariate polynomial of degree d. If $D \geq d$, there is a unique symmetric D-affine function q such that $Q(u) = q(\underbrace{u,\ldots,u}_{D})$.

Similar to our treatment of curves, q is the blossom of Q. The Bézier points of Q can be extracted directly by evaluating q at simply described values. Figure 13 shows what these values are for quadratic surfaces. In general, if q is a blossom of Q, then Q's Bézier control

points rela

5.1 Ev

Given a po
for curves,

Also,

and simila
simple, as s

5.2 Sul

Subdivision
algorithm.

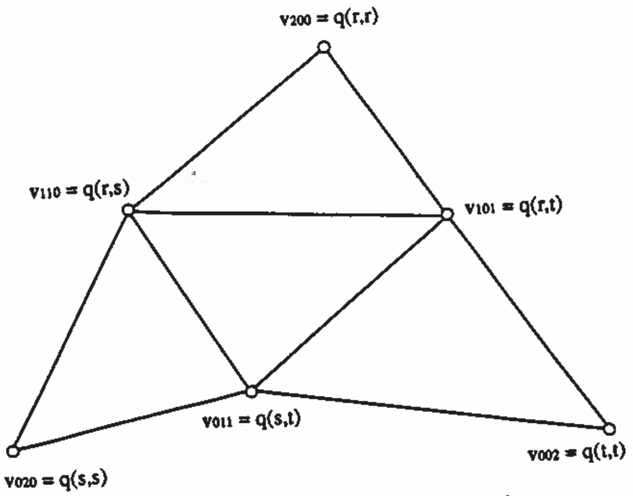


Figure 13: Blossom labels for quadratic surfaces.

points relative to a domain triangle rst are given by

$$\mathrm{v}_{i_1,i_2,i_3} = q(\underbrace{r,\ldots,r}_{i_1},\underbrace{s,\ldots,s}_{i_2},\underbrace{t,\ldots,t}_{i_3}).$$

5.1 Evaluation

Given a point $u = \alpha_1 r + \alpha_2 s + \alpha_3 t$, we would like to find the point $Q(u) = q(u,u)$. Just as for curves, we can find it by a series of affine combinations. First, we have

$$\begin{aligned} q(u,u) &= q(\alpha_1 r + \alpha_2 s + \alpha_3 t, u) \\ &= \alpha_1 q(r,u) + \alpha_2 q(s,u) + \alpha_3 q(t,u). \end{aligned}$$

Also,

$$\begin{aligned} q(r,u) &= q(r, \alpha_1 r + \alpha_2 s + \alpha_3 t) \\ &= \alpha_1 q(r,r) + \alpha_2 q(r,s) + \alpha_3 q(r,t), \end{aligned}$$

and similar expressions can be found for $q(s,u)$ and $q(t,u)$. Geometrically, this is very simple, as shown in Figure 14.

5.2 Subdivision

Subdivision for Bézier surfaces comes directly from the points evaluated in the de Casteljau algorithm. The subdivision given by the algorithm breaks the surface into 3 pieces about the

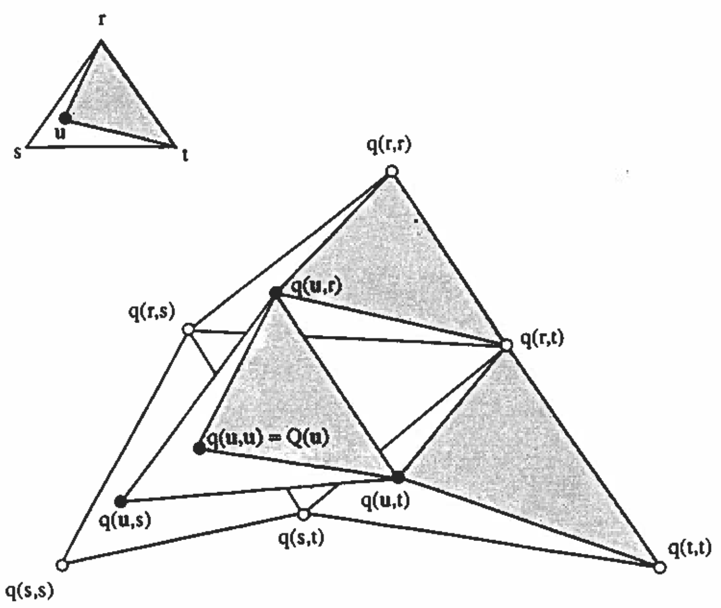


Figure 14: De Casteljau's algorithm for surfaces using blossom labeling.

point of evaluation. Using blossoming, the proof is again fairly simple. We need only observe that the parameters of the intermediate points found by the algorithm have the proper form to be control points for a degree d polynomial. By uniqueness, they must be correct. For example, the subtriangle shown in gray in Figure 14 has as its control points $q(r,r)$, $q(u,r)$, $q(r,t)$, $q(u,u)$, $q(u,t)$, and $q(t,t)$. These blossom values are sufficient to describe Q when its domain is restricted to the triangle urt.

5.3 Degree Raising

Given a quadratic Q defined by its control points $\mathbf{v}_{i_1,i_2,i_3}$, suppose we want to raise its degree to a cubic $\tilde{Q}$ defined by the control points $\tilde{\mathbf{v}}_{i_1,i_2,i_3}$ such that $Q(u) = \tilde{Q}(u)$. The blossom $\tilde{q}$ of $\tilde{Q}$ in terms of q is

$$\tilde{q}(u_1, u_2, u_3) = \frac{q(u_1,u_2) + q(u_2,u_3) + q(u_1,u_3)}{3}.$$

Note that this is the same as the equation used in degree raising a Bézier curve! The domain is in a higher dimension, but the resulting equations are exactly the same.

As with degree raising of curves, the control points $\tilde{\mathbf{v}}_{i_1,i_2,i_3}$ can now be determined easily by evaluation. For example,

$$\begin{aligned}
\tilde{\mathbf{v}}_{300} &= \tilde{q}(r,r,r) &= \tfrac{q(r,r)+q(r,r)+q(r,r)}{3} &= \mathbf{v}_{200} \\
\tilde{\mathbf{v}}_{111} &= \tilde{q}(r,s,t) &= \tfrac{q(r,s)+q(s,t)+q(r,t)}{3} &= \tfrac{\mathbf{v}_{110}+\mathbf{v}_{011}+\mathbf{v}_{101}}{3}
\end{aligned}$$

Unlike the close relation between Bézier curves and Bézier surfaces, a similarly nice generalization of B-spline curves to surfaces is yet to be found.

6 Blossoms as Abstract Data Types

We have thus far been treating blossoming primarily as a theoretical tool for developing algorithms for the manipulation of Bézier and B-splines curves and Bézier surfaces. In this section, we briefly describe how blossoming can also be used as an effective tool for computer programming.

The key to exploiting blossoming for programming is to create a software library that supports blossoms as an *abstract data type* [1]. An abstract data type (ADT) is simply a collection of data types together with a collection of operations for manipulating them. An ADT for performing affine and Euclidean geometric programming has previously been detailed [7, 8]. Here we outline how the geometric ADT can be expanded to embrace blossoming, and hence Bézier and B-spline curves and surfaces.

The fundamental data types supported in the geometric ADT include `Space`, `Point`, `Vector`, and `AffineMap`. The `Space` data type is a model for arbitrary (finite) dimensional affine spaces. `AffineMaps` between spaces can be created by specifying how the map transforms the points of a domain simplex. For instance, if the points rst form a triangle in a two dimensional affine space X, and if RST are points in a space Y (Y need not be two dimensional), then an affine map $M : X \rightarrow Y$ can be created by pseudo-code similar to

```
M := AffineMapCreate( r, s, t, R, S, T)
```

Once `M` is created, executing a statement like

```
AffineMapEvaluate(M,p)
```

returns the point $M(p)$ in Y.

Blossoms can be introduced into the ADT by providing a `Blossom` data type. Since blossoms are nothing more than symmetric multiaffine maps, support for them is most easily achieved as a generalization of the `AffineMap` data type; that is, the `AffineMap` type can be considered as a univariate blossom. One method of creating blossoms would be to generalize the pseudo-code above by specifying an appropriate number of argument lists, together with their image under the blossom. This is not terribly convenient in practice, as there are a number of restrictions on the argument lists used. A somewhat more convenient way is to specify a blossom by specifying its diagonal polynomial in Bézier form. For instance, if `V` denotes the control net (a sequence of points in some space Y) for a degree d triangular Bézier surface Q, defined on a domain triangle rst in a two dimensional space X, then Q's blossom can be created using pseudo-code such as

```
q := BlossomCreateFromBezierNet( r, s, t, V).
```

Once created, an arbitrary value of `q` can be evaluated using pseudo-code such as

```
BlossomEvaluate( q, u1, ...  , ud)
```

where `u1, ..., ud` are arbitrary points in X. For curves, occurring when the domain space X is one dimensional, it is also convenient to create blossoms from the B-spline representation of their diagonal.

The advantage to this approach is that the fundamental operations of many algorithms are encapsulated in the blossom evaluation routine `BlossomEvaluate`. The implementation of this algorithm is essentially given in Figure 10. Indeed, as we've already seen, the following algorithms are all based on blossom evaluation:

- The de Casteljau and de Boor algorithms.
- Sablonniere's algorithm [13] for the conversion between Bézier and B-spline curves, illustrated in Figure 9.
- Boehm's knot insertion algorithm.
- The Oslo algorithm.
- Bézier subdivision.

References

[1] A. Aho, J. Hopcroft, and J. Ullman. *Data Structures and Algorithms.* Addison-Wesley, 1983.

[2] Richard H. Bartels, John C. Beatty, and Brian A. Barsky. *An introduction to splines for use in computer graphics & Geometric Modeling.* Morgan Kaufmann, Los Altos, CA, 1987.

[3] W. Boehm. Inserting new knots into B-spline curves. *CAD*, 12(4):199–201, 1980.

[4] E. Cohen, T. Lyche, and R. Riesenfeld. Discrete B-splines and subdivision techniques in computer aided geometric design and computer graphics. *Computer Graphics and Image Processing*, 14(2):87–111, 1980.

[5] Pierre de Casteljau. Formes à pôles. Hermes, Paris, 1985.

[6] Pierre de Casteljau. *Shape Mathematics and CAD.* Kogan Page, Ltd., London, 1986.

[7] Tony D. DeRose. A coordinate-free aproach to geometric programming. In W. Strasser and H.-P. Seidel, editors, *Theory and Practice of Geometric Modeling*, pages 291–306. Springer-Verlag, Berlin, 1989.

[8] Tony D. DeRose. Coordinate-free geometric programming. Technical Report 89-09-16, University of Washington, Seattle, WA 98195, September 1989.

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

[10] Gary Herron. Techniques for visual continuity. In Gerald E. Farin, editor, *Geometric Modeling: Algorithms and New Trends*, pages 163–174. SIAM, 1987.

[11] Lyle Ramshaw. Blossoming: A connect-the-dots approach to splines. Technical Report 19, Digital Systems Research Center, Palo Alto, CA, 1987.

[12] Lyle Ramshaw. Béziers and B-splines as multiaffine maps. In *Theoretical Foundations of Computer Graphics and CAD*, pages 757–776. Springer, New York, 1988.

[13] P. Sablonniere. Spline and Bézier polygons associated with a polynomial spline curve. *CAD*, 10(4):257–261, 1978.

[14] Hans-Peter Seidel. Computing B-spline control points. In *Theory and Practice of Geometric Modeling*, pages 17–32. Springer-Verlag, Berlin, 1989.