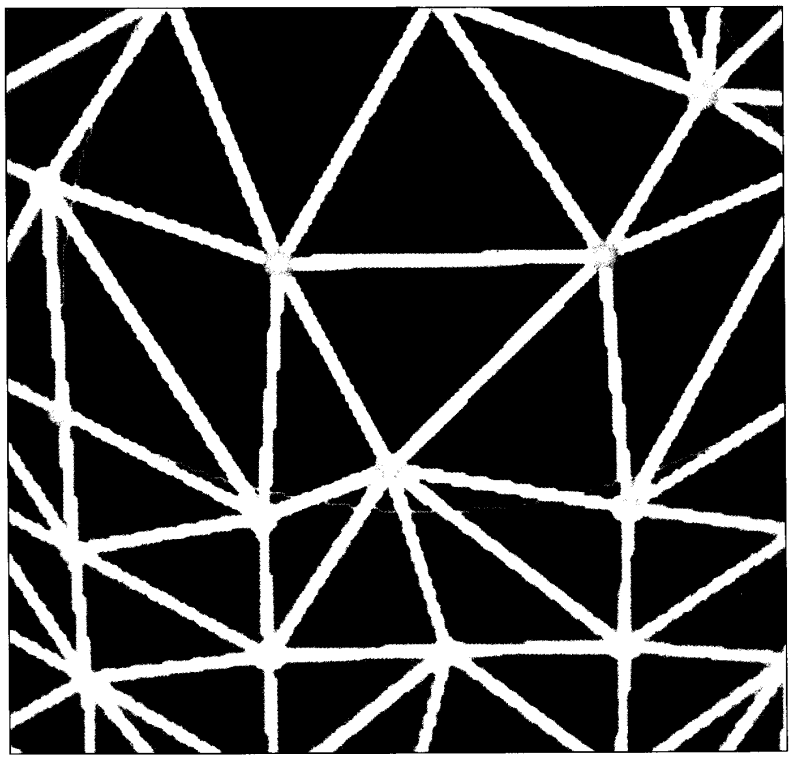
# An Introduction to Polar Forms

**Hans-Peter Seidel**
*University of Erlangen*

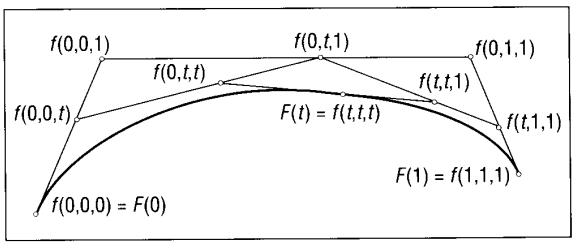


***Polar forms simplify the construction of polynomial and piecewise-polynomial curves and surfaces and lead to new surface representations and algorithms.***

A picture can best explain the main idea behind polar forms. In Figure 1, I show a cubic Bezier curve $F$ over the unit interval [0, 1] with its Bezier points and all the intermediate points that came up while I used the de Casteljau Algorithm to evaluate the curve at a parameter $t$. What's new in this figure is the labeling scheme. While most standard texts use labels like $b_j^l$ for the de Casteljau Algorithm's intermediate points, the labels in Figure 1 are of the form $f(\cdot, \cdot, \cdot)$ where $f$ is the *polar form* of the polynomial $F$. Since $F$ is of degree three, its polar form has three arguments. Furthermore, the polar form is *symmetric*; you can write its three arguments in any order without changing the value of $f$, and $f$ is related to $F$ by the identity $F(u) = f(u, u, u)$. Finally, the labels reflect the incidence structure of the points and lines in Figure 1: All points whose labels share at least two arguments lie on the same line. The exact position of a point on this line is determined by the remaining third label: As $t$ moves with constant speed between 0 and 1, the point $f(0, t, 1)$, for example, moves with constant speed between $f(0, 0, 1)$ and $f(0, 1, 1)$. The point $f(0, t, 1)$ lies $t$ of the way from $f(0, 0, 1)$ to $f(0, 1, 1)$. Moving on a line with constant speed means that the polar form is *affine* in each argument, or simply *multiaffine*. Thus the polar form $f$ of a cubic polynomial curve $F$ is a symmetric triaffine map that satisfies $F(u) = f(u, u, u)$.

The above properties let us reconstruct the point $F(t)$ from the Bezier points $f(0, 0, 0)$, $f(0, 0, 1)$, $f(0, 1, 1)$, and $f(1, 1, 1)$ as follows: First we interpolate linearly along the edges of the control polygon to get the points $f(0, 0, t)$, $f(0, t, 1)$, and $f(t, 1, 1)$. Then we interpolate linearly between these points to get $f(0, 0, t)$ and $f(t, t, 1)$. Finally, the last step of interpolation between these two points yields the point $F(t) = f(t, t, t)$ on the

**Figure 1. A cubic Bezier curve and the de Casteljau Algorithm.**





0272-17-16/93/0100-0038$03.00 © 1993 IEEE

curve. This is exactly what the de Casteljau Algorithm is (we'll return to this algorithm later).

See the sidebar for a brief review of polar forms.

## Polar form of polynomial curves

Let's generalize our introductory discussion from cubics to polynomial curves of arbitrary degree. In particular, I want to establish the so-called *Blossoming Principle*[1-5] that essentially states that every polynomial has a unique polar form.

First we need a little bit of notation. Recall that a map $f: \mathbb{R} \to \mathbb{R}^t$ is affine if it preserves affine combinations, that is, if $f$ satisfies $f(\Sigma_j \alpha_j u_j) = \Sigma_j \alpha_j f(u_j)$ for all scalars $\alpha_1, \ldots, \alpha_m \in \mathbb{R}$ with $\Sigma_j \alpha_j = 1$. A map $f: \mathbb{R}^n \to \mathbb{R}^t$ is *n-affine* (or just multiaffine) if it is an affine map in each argument when the others are held fixed. Thus $f$ is multiaffine if

$$f\left(u_1,\ldots,\sum_j \alpha_j u_{i_j},\ldots,u_n\right) = \sum_j \alpha_j f\left(u_1,\ldots,u_{i_j},\ldots,u_n\right)$$

for all $i = 1, \ldots, n$ and $\alpha_1, \ldots, \alpha_m \in \mathbb{R}$ with $\Sigma_j \alpha_j = 1$. Finally, $f: \mathbb{R}^n \to \mathbb{R}^t$ is called symmetric if it keeps its values under any permutation of its arguments.

We can extend the domain of a symmetric multiaffine map $f$ to vectors: Let $\hat{\xi}_i = w_i - v_i$ be a vector. We can then define $f(u_1, \ldots, u_{n-q}, \hat{\xi}_1, \ldots, \hat{\xi}_q)$ recursively as

$$f(u_1, \ldots, u_{n-q}, \hat{\xi}_1, \ldots, \hat{\xi}_q) = f(u_1, \ldots, u_{n-q}, w_1, \hat{\xi}_2, \ldots, \hat{\xi}_q) - f(u_1, \ldots, u_{n-q}, v_1, \hat{\xi}_2, \ldots, \hat{\xi}_q)$$

Note that this definition is in fact well-defined. That is, it only depends on the vectors $\hat{\xi}_i$, not on their starting or end points $w_i$ and $v_i$. With this notation in place, we can now state the Blossoming Principle.[1-5]

### Blossoming Principle

Polynomials $F: \mathbb{R} \to \mathbb{R}^t$ of degree $n$ and symmetric multiaffine maps $f: (\mathbb{R})^n \to \mathbb{R}^t$ are equivalent to each other. In particular, given a map of either type, we know there is a unique map of the other type that satisfies the identity $F(u) = f(u, \ldots, u)$. In this situation $f$ is called the *multiaffine polar form* or *blossom* of $F$, while $F$ is called the *diagonal* of $f$. Furthermore, the $q$th derivative of $F$ is given as

$$F^{(q)}(u) = \frac{n!}{(n-q)!} f\left(\underbrace{u,\ldots,u}_{n-q}, \underbrace{\hat{1},\ldots,\hat{1}}_{q}\right) \quad \textbf{(1)}$$

where $\hat{1} = 1 - 0 \in \mathbb{R}$ is the standard unit vector and $f(u, \ldots, u, \hat{1}, \ldots, \hat{1})$ is defined as above.

## A brief history of polar forms

Polar forms are a classical mathematical tool for the study of polynomials. Paul de Faget de Casteljau first considered them in the context of computer graphics and computer-aided geometric design in his work at Citroen,[1,2] and Lyle Ramshaw studied them while at Xerox Palo Alto Research Center.[3-5] In his original work, de Casteljau focused on Bezier curves and triangular Bezier patches, and especially on constructing quasi-interpolants.[6] Ramshaw's treatment of polar forms is much more algebraic and uses techniques like homogenizing and tensoring.

More recently, various researchers have expanded and applied the polar approach. In my own research, I applied polar forms directly to the B-spline blending functions and gave a simple development of B-splines from scratch.[7] I also discussed the relationship between polar forms and knot insertion. P.J. Barry and R.N. Goldman[8,9] related polar forms to other B-spline approaches, and they also used polar forms for a thorough discussion of knot insertion for B-splines. E.T.Y. Lee[10] and K. Strom[11] also contributed to this area. In other research, I used the geometry behind polar forms to extend them to geometrically continuous spline curves,[12] and G. Schmeltz extended this geometric approach to surfaces.[13] T. DeRose and others used polar forms as an abstract data type, the basis of a software library.[14] Among other things, they used these forms for curvature computations and for composing polynomials.

Polar forms have also been helpful in developing new surface schemes. In earlier research, I introduced the B-patch, a surface representation that we can think of as the analog to a B-spline segment for surfaces.[15] More recently, W. Dahmen, C.A. Micchelli, and I[16] combined B-patches with simplex splines and developed a surface scheme that lets us model smooth piecewise polynomial surfaces over arbitrary triangulations. P. Fong[17] has produced a first implementation of this scheme.

Explicit formulas for the polar form $f$ become particularly simple for monomials

$$F(u) = \sum_{i=0}^{n} \mathbf{a}_i u^i$$

In this case the polar form $f$ is given by the formula

$$f(u_1,\ldots,u_n) = \sum_{i=0}^{n} \mathbf{a}_i \binom{n}{i}^{-1} \sum_{\substack{S \subseteq \{1,\ldots,n\} \\ |S|=i}} \prod_{j \in S} u_j$$

and the coefficients $\mathbf{a}_i$ satisfy

$$\mathbf{a}_i = \frac{F^{(q)}(0)}{q!} = \binom{n}{q} f\left(\underbrace{0,\ldots,0}_{n-q}, \underbrace{\hat{1},\ldots,\hat{1}}_{q}\right)$$

For the cubic polynomial

$$F(u) = \mathbf{a}_0 + \mathbf{a}_1 u + \mathbf{a}_2 u^2 + \mathbf{a}_3 u^3$$

we get, for example,

$$f(u_1,u_2,u_3) =$$
$$\mathbf{a}_0 + \frac{\mathbf{a}_1}{3}(u_1+u_2+u_3) + \frac{\mathbf{a}_2}{3}(u_1u_2+u_2u_3+u_3u_1) + \mathbf{a}_3 u_1u_2u_3$$

A coordinate-free formula for the polar form is[3]

$$f(u_1,\ldots,u_n) = \frac{1}{n!}\sum_{\substack{S\subseteq\{1,\ldots,n\}\\ |S|=i}} (-1)^{n-i} i^n F\left(\frac{1}{i}\sum_{j\in S} u_j\right)$$

Now let's consider the continuity conditions between two polynomials in terms of polar forms. Essentially, by rephrasing Equation 1, we get the Theorem for $C^q$ Conditions.

### Theorem for $C^q$ Conditions

Let $F : \mathbb{R} \to \mathbb{R}^t$ and $G : \mathbb{R} \to \mathbb{R}^t$ be two polynomials of degree $n$, and let $u \in \mathbb{R}$. Then the following two statements are equivalent:

- $F$ and $G$ are $C^q$-continuous at $u$.
- $f(u, \ldots, u, u_1, \ldots, u_q) = g(u, \ldots, u, u_1, \ldots, u_q)$ for $u_1, \ldots, u_q \in \mathbb{R}$.

## Bezier curves

Why are polar forms useful? Polar forms simplify the theory of Bezier curves and B-splines to such a state that most results become almost trivial. Let's consider a polynomial curve $F : \mathbb{R} \to \mathbb{R}^t$. Suppose we wish to represent $F$ as a Bezier curve over some given interval $\Delta = [r, s]$. What are the Bezier points? Writing $u$ as an affine combination of $r$ and $s$,

$$u = \frac{s-u}{s-r} r + \frac{u-r}{s-r} s$$

we get

$$\begin{aligned} F(u) = f(u,\ldots,u) &= \frac{s-u}{s-r} f(u,\ldots,u,r) + \frac{u-r}{s-r} f(u,\ldots,u,s) \\ &= \left(\frac{s-u}{s-r}\right)^2 f(u,\ldots,u,r,r) + 2\left(\frac{u-r}{s-r}\right)\left(\frac{s-u}{s-r}\right) f(u,\ldots,u,r,s) \\ &\quad + \left(\frac{u-r}{s-r}\right)^2 f(u,\ldots,u,s,s) \\ &= \sum_{j=0}^{n} B_j^{\Delta,n}(u)\, f\Big(\underbrace{r,\ldots,r}_{n-j},\underbrace{s,\ldots,s}_{j}\Big) \end{aligned}$$

where

$$B_j^{\Delta,n}(u) = \binom{n}{j}\left(\frac{u-r}{s-r}\right)^j\left(\frac{s-u}{s-r}\right)^{n-j},\ j = 0,\ldots,n$$

are the Bernstein polynomials with regard to $\Delta = [r, s]$. Thus we have the Theorem for Bezier Points.[1-5]

### Theorem for Bezier Points

Let $\Delta = [r, s]$ be an arbitrary interval. We can represent every polynomial $F : \mathbb{R} \to \mathbb{R}^t$ as a Bezier polynomial with regard to $\Delta$. The Bezier points are given as

$$\mathbf{b}_j = f\Big(\underbrace{r,\ldots,r}_{n-j},\underbrace{s,\ldots,s}_{j}\Big) \tag{2}$$

where $f$ is the polar form of $F$.

Equation 2 immediately leads to an evaluation algorithm that recursively computes the value

$$\begin{aligned} \mathbf{b}_j^l(u) &= f\Big(\underbrace{r,\ldots,r}_{n-l-j},\underbrace{u,\ldots,u}_{l},\underbrace{s,\ldots,s}_{j}\Big) \\ &= \frac{s-u}{s-r} f\Big(\underbrace{r,\ldots,r}_{n-l-j+1},\underbrace{u,\ldots,u}_{l-1},\underbrace{s,\ldots,s}_{j}\Big) + \frac{u-r}{s-r} f\Big(\underbrace{r,\ldots,r}_{n-l-j},\underbrace{u,\ldots,u}_{l-1},\underbrace{s,\ldots,s}_{j+1}\Big) \\ &= \frac{s-u}{s-r}\mathbf{b}_j^{l-1}(u) + \frac{u-r}{s-r}\mathbf{b}_{j+1}^{l-1}(u) \end{aligned}$$

from the given control points. For $l = n$, we finally compute $\mathbf{b}_0^n(u) = f(u, \ldots, u) = F(u)$, which is the desired point on the curve. Figures 2 and 3 illustrate the resulting computational scheme. Paul de Faget de Casteljau first studied this algorithm, and thus it's called the de Casteljau Algorithm.[1,2]

Equation 2 also shows that the de Casteljau Algorithm offers much more than just evaluation. Suppose that we want to subdivide a Bezier curve $F$ over a given interval $\Delta = [s, t]$ at an arbitrary parameter $u \in \Delta$. What are the new Bezier points of the left and right segments $\mathbf{F}_l$ and $\mathbf{F}_r$ with respect to the subin-

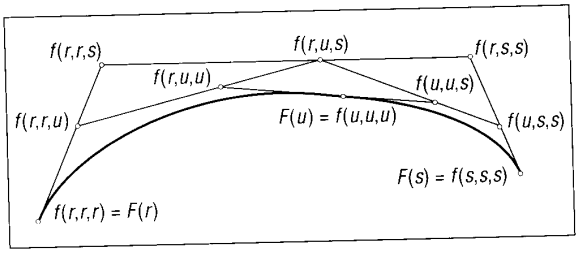



**Figure 2. The de Casteljau Algorithm for the case $n = 3$.**

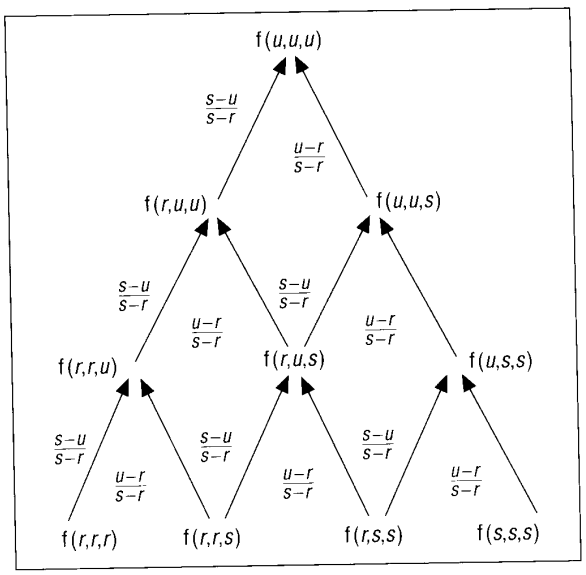



**Figure 3. The de Casteljau Algorithm for the case $n = 3$.**

tervals $\Delta_l = [r, u]$ and $\Delta_r = [u, s]$? Equation 2 tells us that the new Bezier points after subdivision are given as

$$\mathbf{b}_0^l = f(r, \ldots, r), \mathbf{b}_1^l = f(r, \ldots, r, u), \ldots, \mathbf{b}_n^l = f(u, \ldots, u)$$

and

$$\mathbf{b}_0^r = f(u, \ldots, u), \mathbf{b}_1^r = f(u, \ldots, u, s), \ldots, \mathbf{b}_n^r = f(s, \ldots, s) \quad \textbf{(3)}$$

Inspection shows that these points are automatically computed during the de Casteljau Algorithm and are stored along the left and right diagonals.

Finally, we can use a slight modification of the de Casteljau Algorithm to compute arbitrary polar values $f(u_1, \ldots, u_n)$ by recursively computing the values

$$\begin{aligned}\mathbf{b}_j^l(u_1,\ldots,u_l) &= f\Big(\underbrace{r,\ldots,r}_{n-l-j},\underbrace{u_1,\ldots,u_l}_{l},\underbrace{s,\ldots,s}_{j}\Big)\\ &= \frac{s-u_l}{s-r} f\Big(\underbrace{r,\ldots,r}_{n-l-j+1},\underbrace{u_1,\ldots,u_{l-1}}_{l-1},\underbrace{s,\ldots,s}_{j}\Big)\\ &\quad + \frac{u_l-r}{s-r} f\Big(\underbrace{r,\ldots,r}_{n-l-j},\underbrace{u_1,\ldots,u_{l-1}}_{l-1},\underbrace{s,\ldots,s}_{j+1}\Big)\\ &= \frac{s-u_l}{s-r}\mathbf{b}_j^{l-1}(u_1,\ldots,u_{l-1}) + \frac{u_l-r}{s-r}\mathbf{b}_{j+1}^{l-1}(u_1,\ldots,u_{l-1})\end{aligned}$$

For $l = n$, we finally compute $\mathbf{b}_0^n(u_1, \ldots, u_n) = f(u_1, \ldots, u_n)$. This algorithm is called the *Multiaffine de Casteljau Algorithm*.

How about derivatives? After writing

$$\hat{1} = \frac{1}{s-r}(s-r)$$

we find that Equation 1 implies

$$F'(r) = \frac{n}{s-r}\big(f(r,\ldots,r,s) - f(r,\ldots,r)\big) = \frac{n}{s-r}\big(\mathbf{b}_1 - \mathbf{b}_0\big)$$

and similarly

$$F''(r) = \frac{n(n-1)}{(s-r)^2}\big(\mathbf{b}_2 - 2\mathbf{b}_1 + \mathbf{b}_0\big)$$

and so forth. These are the well-known derivative formulas for Bezier curves.

Let me remark briefly on affine invariance. Let $\tilde{F} = \phi \circ F$ be the image of $F$ under an affine map (for example, translation, scaling, rotation) $\phi$. The uniqueness part of the Blossoming Principle implies that the polar form $\tilde{f}$ of $\tilde{F}$ is given as $\tilde{f} = \phi \circ f$. It follows that the Bezier points $\tilde{b}_j$ of $\tilde{F}$ satisfy

$$\tilde{\mathbf{b}}_j = \tilde{f}(r, \ldots, r, s, \ldots, s) = \phi(f(r, \ldots, r, s, \ldots, s)) = \phi(\mathbf{b}_j)$$

This means that the relationship between the curve $F$ and its Bezier control polygon is invariant under affine maps.

## B-spline curves

How can we extend the results of the preceding section from Bezier curves to B-splines? We have seen that a polynomial curve $F$ and its polar form $f$ is completely defined by its Bezier points $\mathbf{b}_j = f(r, \ldots, r, s, \ldots, s)$. Let

$$r_n \le \ldots \le r_1 < s_1 \le \ldots \le s_n$$

be a nondecreasing sequence of real numbers. We wish to show that $F$ can be equally well defined by its *de Boor points* $\mathbf{d}_j = f(r_1, \ldots, r_{n-j}, s_1, \ldots, s_j)$. Since $r_i \neq s_j$, we can express $u$ as an affine combination with regard to $r_i$ and $s_j$,

$$u = \frac{s_j - u}{s_j - r_i} r_i + \frac{u - r_i}{s_j - r_i} s_j$$

and by successively expanding

$$\begin{aligned}\mathbf{d}_j^l(u) &= f(r_1,\ldots,r_{n-l-j},u,\ldots,u,s_1,\ldots,s_j)\\ &= \frac{s_{j+1}-u}{s_{j+1}-r_{n-l-j+1}} f(r_1,\ldots,r_{n-l-j+1},u,\ldots,u,s_1,\ldots,s_j)\\ &\quad + \frac{u-r_{n-l-j+1}}{s_{j+1}-r_{n-l-j+1}} f(r_1,\ldots,r_{n-l-j},u,\ldots,u,s_1,\ldots,s_{j+1})\\ &= \frac{s_{j+1}-u}{s_{j+1}-r_{n-l-j+1}}\mathbf{d}_j^{l-1}(u) + \frac{u-r_{n-l-j+1}}{s_{j+1}-r_{n-l-j+1}}\mathbf{d}_{j+1}^{l-1}(u)\end{aligned}$$

we see that $F(u) = \mathbf{d}_0^n(u)$ is in fact completely determined by the points $\mathbf{d}_j = f(r_1, \ldots, s_j)$.

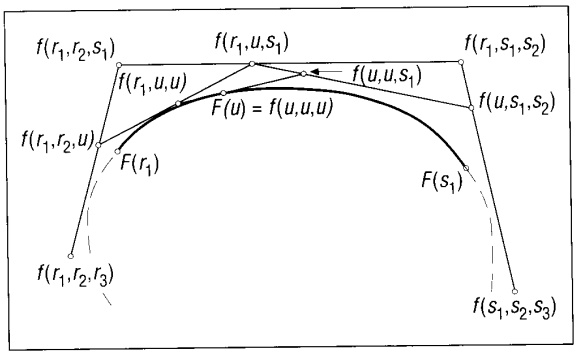



**Figure 4. The de Boor Algorithm for the case $n = 3$.**

Conversely, suppose that the points $\mathbf{d}_j = f(r_1, \ldots, r_{n-j}, s_1, \ldots, s_j)$ are given. We can then use the above recurrence to evaluate the curve $F$ at an arbitrary parameter value $u$. Inspection shows that the resulting algorithm is identical to the de Boor Algorithm for the evaluation of a B-spline segment from its endpoints. We thus have the following theorem.[4,5]

### Theorem for de Boor Points

We can represent every polynomial $F : \mathbb{R} \to \mathbb{R}^t$ as a B-spline segment over a nondecreasing knot sequence $r_n \le \ldots \le r_1 < s_1 \le \ldots \le s_n$. The de Boor points are given as

$$\mathbf{d}_j = f(r_1, \ldots, r_{n-j}, s_1, \ldots, s_j) \qquad \textbf{(4)}$$

where $f$ is the polar form of $F$.

I illustrate this theorem and the de Boor Algorithm itself in Figures 4 and 5. Again, we can use the multiaffine version of the algorithm to compute arbitrary polar values $f(u_1, \ldots, u_n)$ from the given control points.

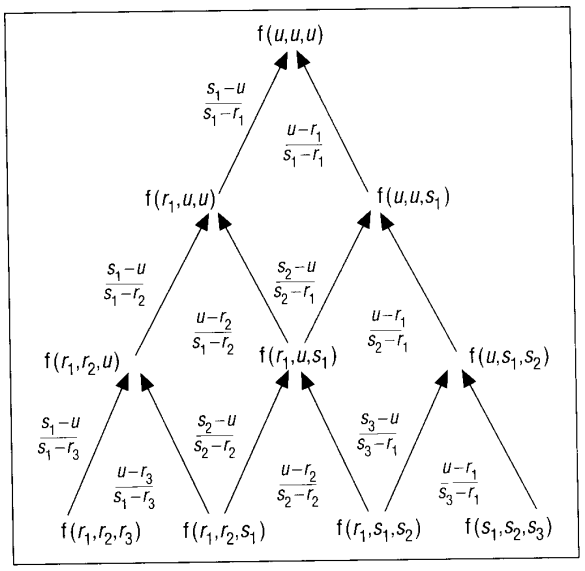



**Figure 5. The de Boor Algorithm for a cubic B-spline segment.**

Even more important than evaluation is *knot insertion*. Suppose that the knot sequence $r_n \le \ldots \le r_1 < s_1 \le \ldots \le s_n$ is given and that we want to insert a new knot $t$ with $r_1 \le t < s_1$. Equation 4 tells us that the new control points $\mathbf{d}_j^*$ after knot insertion are given as

$$\begin{aligned}\mathbf{d}_j^* &= f(t, r_1, \ldots, r_{n-j-1}, s_1, \ldots, s_j) \\ &= \frac{s_{j+1} - t}{s_{j+1} - r_{n-j}} f(r_1, \ldots, r_{n-j}, s_1, \ldots, s_j) \\ &\quad + \frac{t - r_{n-j}}{s_{j+1} - r_{n-j}} f(r_1, \ldots, r_{n-j-1}, s_1, \ldots, s_{j+1}) \\ &= \frac{s_{j+1} - t}{s_{j+1} - r_{n-j}} \mathbf{d}_j + \frac{t - r_{n-j}}{s_{j+1} - r_{n-j}} \mathbf{d}_{j+1}\end{aligned}$$

This is exactly what the *Boehm Algorithm* is. The Boehm Algorithm is identical to the first step of the de Boor Algorithm. Barry and Goldman have studied other knot insertion algorithms (such as the Oslo Algorithm) in great detail.[8,9,18]

Let's briefly consider the miracle that B-spline curves are $C^{n-q}$-continuous at a knot of multiplicity $q$. To keep our discussion as simple as possible, let's look at only the case where $\{t_i\}_i$ is a sequence of simple knots. Let $F_i : [t_i, t_{i+1}] \to \mathbb{R}^t$ and $F_{i+1} : [t_{i+1}, t_{i+2}] \to \mathbb{R}^t$ be two adjacent B-spline segments that join at the knot $t_{i+1}$. Since $f_i(t_{i-n+j+1}, \ldots, t_{i+j}) = f_{i+1}(t_{i-n+j+1}, \ldots, t_{i+j})$, for $j = 1, \ldots, n$, successive expansion shows that

$$f_i(t_{i+1}, u, \ldots, u) = f_{i+1}(t_{i+1}, u, \ldots, u)$$

Then Equation 1 implies that $F_i$ and $F_{i+1}$ are in fact $C^{n-1}$-continuous at $t_{i+1}$. In Figure 6, I show the overlapping de Boor schemes of two adjacent cubic B-spline segments.

## Tensor product surfaces

By far the most popular surfaces in computer-aided geometric design are tensor product surfaces. Given a curve scheme $F(u) = \sum_{i=0}^{n} B_i(u)\mathbf{b}_i$, $\mathbf{b}_i \in \mathbb{R}^t$, the corresponding tensor product scheme is defined as

$$F(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{m} B_i(u)\ B_j(v)\ \mathbf{b}_{ij}, \quad \mathbf{b}_{ij} \in \mathbb{R}^t$$

which we can also write as

$$F(u,v) = \sum_{i=0}^{n} B_i(u)\ \mathbf{b}_{i_v} \quad \text{with } \mathbf{b}_{i_v} = \mathbf{b}_i(v) = \sum_{j=0}^{m} B_j(v)\ \mathbf{b}_{ij}$$

This equation demonstrates that tensor product surfaces may be considered as curves of curves. This equation also shows that we first have to understand curves to understand tensor product surfaces.

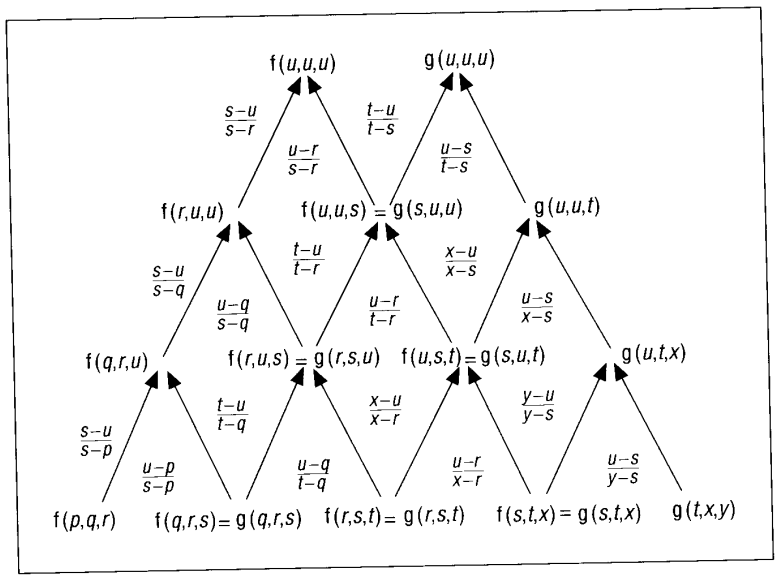



**Figure 6. Overlapping de Boor schemes for two adjacent cubic B-spline segments**
**$F : [r, s] \to \mathbb{R}^t$ and $G : [s, t] \to \mathbb{R}^t$ over the knot sequence $\ldots, p, q, r, s, t, x, y, \ldots$**

Now let's consider the polar form of a polynomial tensor product surface. Let

$$F : \mathbb{R} \times \mathbb{R} \to \mathbb{R} : (u, v) \mapsto F(u, v)$$

be a polynomial tensor product surface of degree $n$ in $u$ and of degree $m$ in $v$. To compute the corresponding polar form $f_{TP}$ of $F$ we simply polarize both independent variables $u$ and $v$ separately. The resulting map

$$f_{TP} : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^t : (u_1, \ldots, u_n; v_1, \ldots, v_m) \mapsto f_{TP}(u_1, \ldots, u_n; v_1, \ldots, v_m)$$

is then characterized by the following properties:

- Symmetry: $f_{TP}$ is symmetric in the variables $u_i$ and $v_j$ separately. That is, we get

$$f_{TP}(u_1, \ldots, u_n; v_1, \ldots, v_m) = f_{TP}(u_{\pi(1)}, \ldots, u_{\pi(n)}; v_{\sigma(1)}, \ldots, v_{\sigma(m)})$$

  for all permutations $\pi \in \Sigma_n$ and $\sigma \in \Sigma_m$.
- Multiaffine property: $f_{TP}$ is affine in each of the variables $u_i$ and $v_j$ separately.
- Diagonal property: $f_{TP}(u, \ldots, u; v, \ldots, v) = F(u, v)$

In generalizing the curve case, we find that the Bezier points $\mathbf{b}_{ij}$ of $F$ in the representation $F(u, v) = \Sigma_{i=0}^{n} B_i^n(u) B_j^m(v) \mathbf{b}_{ij}$ as a tensor product Bezier surface over $[p, q] \times [r, s]$ are given as

$$\mathbf{b}_{ij} = f_{TP}\Big(\underbrace{p, \ldots, p}_{n-i}, \underbrace{q, \ldots, q}_{i}, \underbrace{r, \ldots, r}_{m-j}, \underbrace{s, \ldots, s}_{j}\Big)$$

while the de Boor points $\mathbf{d}_{ij}$ of $F$ in the representation $F(u, v) = \Sigma_i \Sigma_j N_i^n(u) N_j^m(v) \mathbf{d}_{ij}$ as segment of a tensor product B-spline surface over the knot vectors $S = \{s_i\}$ and $T = \{t_j\}$ are given as

$$\mathbf{d}_{ij} = f_{TP}(s_{i+1}, \ldots, s_{i+n}; t_{j+1}, \ldots, t_{j+m})$$

You can then generalize many of the algorithms I discussed earlier from Bezier and B-spline curves to Bezier and B-spline tensor product surfaces.

## True surfaces

From now on, let's discuss "true" surfaces. We start with the Blossoming Principle, which generalizes almost word-by-word from curves to surfaces.

### Blossoming Principle for surfaces

Polynomials $F : \mathbb{R}^2 \to \mathbb{R}^t$ of degree $n$ and symmetric multiaffine maps $f : (\mathbb{R}^2)^n \to \mathbb{R}^t$ are equivalent to each other. In particular, given a map of either type, we know that there is a unique map of the other type that satisfies the identity $F(\mathbf{u}) = f(\mathbf{u}, \ldots, \mathbf{u})$. In this situation $f$ is called the *multiaffine polar form* or *blossom* of $F$, while $F$ is called the *diagonal* of $f$. Furthermore, the $q$th directional derivative of $F$ with respect to vectors $\hat{\xi}_1, \ldots, \hat{\xi}_q \in \mathbb{R}^2$ is given as

$$D_{\hat{\xi}_1, \ldots, \hat{\xi}_q} F(\mathbf{u}) = \frac{n!}{(n-q)!} f\left(\mathbf{u}, \ldots, \mathbf{u}, \hat{\xi}_1, \ldots, \hat{\xi}_q\right) \qquad (5)$$

where $f(\mathbf{u}, \ldots, \mathbf{u}, \hat{\xi}_1, \ldots, \hat{\xi}_q)$ is defined in a way similar to the definition in Equation 1.

Again, things are particularly simple for monomials. For example, for the quadratic polynomial

$$F(\mathbf{u}) = \mathbf{a}_{00} + \mathbf{a}_{10} u + \mathbf{a}_{01} v + + \mathbf{a}_{20} u^2 + \mathbf{a}_{11} uv + \mathbf{a}_{02} v^2$$

we get

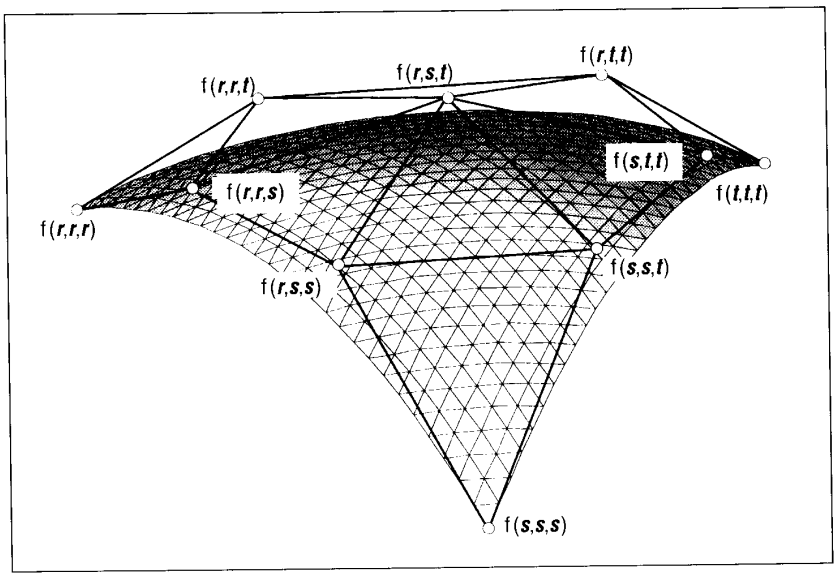
$$f(\mathbf{u}_1, \mathbf{u}_2) = \mathbf{a}_{00} + \frac{\mathbf{a}_{10}}{2}(u_1 + u_2) + \frac{\mathbf{a}_{01}}{2}(v_1 + v_2) + \mathbf{a}_{20}\, u_1 u_2 + \frac{\mathbf{a}_{11}}{2}(u_1 v_2 + u_2 v_1) + \mathbf{a}_{02}\, v_1 v_2$$

The coordinate-free formula for the polar form becomes[3]

$$f(\mathbf{u}_1, \ldots, \mathbf{u}_n) = \frac{1}{n!} \sum_{\substack{S \subseteq \{1, \ldots, n\} \\ |S| = i}} (-1)^{n-i} i^n F\left(\frac{1}{i} \sum_{j \in S} \mathbf{u}_j\right)$$

and the continuity conditions translate into the $C^q$-Conditions Theorem for Surfaces.

**Figure 7. A cubic Bezier patch.**




### $C^q$-Conditions Theorem for Surfaces

Let $F : \mathbb{R}^2 \to \mathbb{R}^t$ and $G : \mathbb{R}^2 \to \mathbb{R}^t$ be two polynomials of degree $n$, and let $\mathbf{u} \in \mathbb{R}^2$. Then the following statements are equivalent:

- $F$ and $G$ are $C^q$-continuous at $\mathbf{u}$.
- $f(\mathbf{u}, \ldots, \mathbf{u}, \mathbf{u}_1, \ldots, \mathbf{u}_q) = g(\mathbf{u}, \ldots, \mathbf{u}, \mathbf{u}_1, \ldots, \mathbf{u}_q)$ for $\mathbf{u}_1, \ldots, \mathbf{u}_q \in \mathbb{R}$.

## Bezier triangles

The straightforward analog to Bezier curves are triangular Bezier patches (see Figure 7). Consider a polynomial surface $F : \mathbb{R}^2 \to \mathbb{R}^t$. Suppose we want to represent $F$ as a triangular Bezier patch over some given domain triangle $\Delta = \Delta(\mathbf{r}, \mathbf{s}, \mathbf{t})$. Representing $\mathbf{u} \in \mathbb{R}^2$ in barycentric coordinates with regard to $\Delta$,

$$\mathbf{u} = r(\mathbf{u})\mathbf{r} + s(\mathbf{u})\mathbf{s} + t(\mathbf{u})\mathbf{t}, \qquad r + s + t = 1$$

we get

$$\begin{aligned} F(\mathrm{u}) &= f(\mathrm{u},\ldots,\mathrm{u}) \\ &= r(\mathrm{u})\, f(\mathrm{u},\ldots,\mathrm{u},\mathrm{r}) + s(\mathrm{u})\, f(\mathrm{u},\ldots,\mathrm{u},\mathrm{s}) \\ &\quad + t(\mathrm{u})\, f(\mathrm{u},\ldots,\mathrm{u},\mathrm{t}) \\ &= \sum_{i+j+k=n} B_{ijk}^{\Delta,n}(\mathrm{u})\, f(\underbrace{\mathrm{r},\ldots,\mathrm{r}}_{i},\underbrace{\mathrm{s},\ldots,\mathrm{s}}_{j},\underbrace{\mathrm{t},\ldots,\mathrm{t}}_{k}) \end{aligned}$$

where

$$B_{ijk}^{\Delta,n}(\mathbf{u}) = \binom{n}{ijk} r(\mathbf{u})^i\, s(\mathbf{u})^j\, t(\mathbf{u})^k$$

are the Bernstein polynomials with regard to $\Delta = \Delta(\mathbf{r}, \mathbf{s}, \mathbf{t})$. We have thus shown the Theorem for Triangular Bezier Points.

### Theorem for Triangular Bezier Points

Let $\Delta = \Delta(\mathbf{r}, \mathbf{s}, \mathbf{t})$ be an arbitrary triangle. We can represent every polynomial $F : \mathbb{R}^2 \to \mathbb{R}^t$ as a Bezier triangle with regard to $\Delta$. The Bezier points are given as

$$\mathbf{b}_{ijk} = f(\underbrace{\mathbf{r},\ldots,\mathbf{r}}_{i},\underbrace{\mathbf{s},\ldots,\mathbf{s}}_{j},\underbrace{\mathbf{t},\ldots,\mathbf{t}}_{k}) \tag{6}$$

where $f$ is the polar form of $F$.

Similar to the curve case, this theorem leads directly to the de Casteljau Algorithm for evaluating, subdividing, and computing the polar form. I illustrate the resulting computational scheme in Figure 8. The well-known derivative formulas and continuity conditions for Bezier triangles follow directly from Equations 5 and 6. For $\hat{\xi} = \mathbf{s} - \mathbf{r}$, for example, we get

$$D_{\hat{\xi}}F(\mathbf{r}) = n(f(\mathbf{r},\ldots,\mathbf{r},\mathbf{s}) - \mathbf{f}(\mathbf{r},\ldots,\mathbf{r},\mathbf{r}) = n(\mathbf{b}_{n-1,1,0} - \mathbf{b}_{n,0,0})$$

Finally, affine invariance also follows in exactly the same way as in the curve case.

## B-patches

In the previous section, we saw that a polar form $f$ is uniquely defined by its values $f(\mathbf{r}, \ldots, \mathbf{r}, \mathbf{s}, \ldots, \mathbf{s}, \mathbf{t}, \ldots, \mathbf{t})$ on the vertices of a triangle $\Delta = \Delta(\mathbf{r}, \mathbf{s}, \mathbf{t})$. We can generalize this definition by assigning a family of—usually different—knots to each vertex of the triangle $\Delta$. The resulting surface representation is a *B-patch*.[12,15]

We say that $A = \{\mathbf{r}_1, \ldots, \mathbf{r}_n, \mathbf{s}_1, \ldots, \mathbf{s}_n, \mathbf{t}_1, \ldots, \mathbf{t}_n\}$ is a *knot arrangement* if all triangles $\Delta_{ijk} = \Delta(\mathbf{r}_i, \mathbf{s}_j, \mathbf{t}_k)$ are nondegenerate. In this situation, we can represent $\mathbf{u}$ in barycentric coordinates with regard to $\Delta(\mathbf{r}_i, \mathbf{s}_j, \mathbf{t}_k)$,

$$\mathbf{u} = r_{ijk}(\mathbf{u})\mathbf{r}_{i+1} + s_{ijk}(\mathbf{u})\mathbf{s}_j + t_{ijk}(\mathbf{u})\mathbf{t}_k, \qquad r_{ijk} + s_{ijk} + t_{ijk} = 1$$

and by successively expanding

$$\begin{aligned} \mathbf{d}_{ijk}^{l}(\mathbf{u}) &= f(r_1,\ldots,\mathbf{r}_i,\mathbf{s}_1,\ldots,\mathbf{s}_j,\mathbf{t}_1,\ldots,\mathbf{t}_k,\mathbf{u},\ldots,\mathbf{u}) \\ &= r_{i+1,j+1,k+1}(\mathbf{u})\, f(\mathbf{r}_1,\ldots,\mathbf{r}_{i+1},\mathbf{s}_1,\ldots,\mathbf{s}_j,\mathbf{t}_1,\ldots,\mathbf{t}_k,\mathbf{u},\ldots,\mathbf{u}) \\ &\quad + s_{i+1,j+1,k+1}(\mathbf{u})\, f(\mathbf{r}_1,\ldots,\mathbf{r}_i,\mathbf{s}_1,\ldots,\mathbf{s}_{j+1},\mathbf{t}_1,\ldots,\mathbf{t}_k,\mathbf{u},\ldots,\mathbf{u}) \\ &\quad + t_{i+1,j+1,k+1}(\mathbf{u})\, f(\mathbf{r}_1,\ldots,\mathbf{r}_i,\mathbf{s}_1,\ldots,\mathbf{s}_j,\mathbf{t}_1,\ldots,\mathbf{t}_{k+1},\mathbf{u},\ldots,\mathbf{u}) \end{aligned}$$

we see that $F(\mathbf{u}) = \mathbf{d}_{000}^{n}(\mathbf{u})$ is in fact completely determined by

the points $\mathbf{d}_{ijk} = f(\mathbf{r}_1, \ldots, \mathbf{t}_k)$. We thus obtain the Theorem for B-patch Control Points.

### Theorem for B-patch Control Points

Let a knot arrangement $A = \{\mathbf{r}_1, \ldots, \mathbf{t}_n\}$ be given as above. We can represent every polynomial $F : \mathbb{R}^2 \to \mathbb{R}^l$ as a B-patch over $A$ with control points

$$\mathbf{d}_{ijk} = f(\mathbf{r}_1, \ldots, \mathbf{r}_i, \mathbf{s}_1, \ldots, \mathbf{s}_j, \mathbf{t}_1, \ldots, \mathbf{t}_k) \quad (7)$$

where $f$ is the polar form of $F$.

This theorem shows that, for surfaces, B-patches are the analog to B-spline curve segments. In particular, B-patches have a de Boor-like evaluation algorithm that computes a point $F(\mathbf{u})$ on the surface from the given control points through successive linear interpolation. Again, we can use the multiaffine version of this algorithm to compute an arbitrary polar value $f(\mathbf{u}_1, \ldots, \mathbf{u}_n)$. Figure 9 illustrates the resulting computational scheme.

## Triangular B-spline scheme

By combining B-patches and simplex splines, Dahmen, Micchelli, and I recently developed a new multivariate B-spline scheme.[12] We based this surface scheme on blending functions and control points. It lets you construct smooth piecewise polynomial surfaces over arbitrary triangulations of the parameter plane. I summarize some of the surface scheme's main features in the following theorem.

### Theorem for Triangular B-splines

Let $F(\mathbf{u}) = \Sigma_I N^I_{ijk}(\mathbf{u})\mathbf{c}^I_{ijk}$ be a triangular B-spline surface. Then this surface has the following properties:

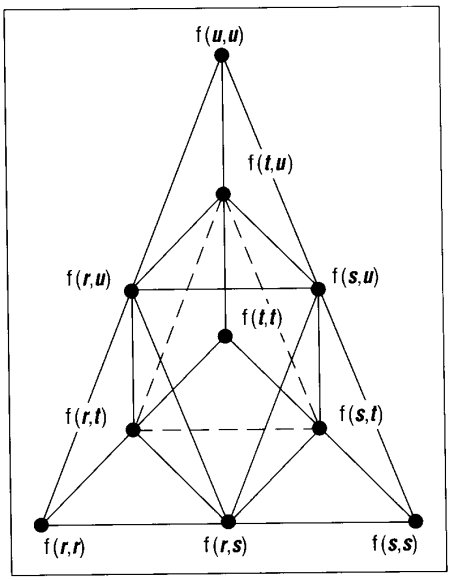
- Piecewise polynomial: $F(\mathbf{u})$ is a piecewise polynomial of degree $n$.
- Locality: Movement of a single control point $\mathbf{c}^I_{ijk}$ influences only the surface on the triangle $\Delta(I)$ and on the triangles directly surrounding $\Delta(I)$.
- Convex hull property: $F(\mathbf{u})$ lies inside the convex hull of its control net.
- Smoothness: The surface $F(\mathbf{u})$ is generically $C^{n-1}$-continuous everywhere.
- Affine invariance: The relationship between the surface $F$ and its control net is affinely invariant.

In a first implementation of this new surface scheme, we successfully demonstrated the practical feasibility of the fundamental algorithms underlying the new scheme.[12,17] With this implementation, you can edit and manipulate quadratic and cubic surfaces over arbitrary triangulations in real time.

For a surface scheme to be useful in practice, it must let you represent as many surfaces as possible. The following result[12] is remarkable for just that reason.

### Polynomial and piecewise polynomial surfaces

We can represent any polynomial or piecewise polynomial surface $F$ with the new B-spline scheme. In this situation we get the control points as



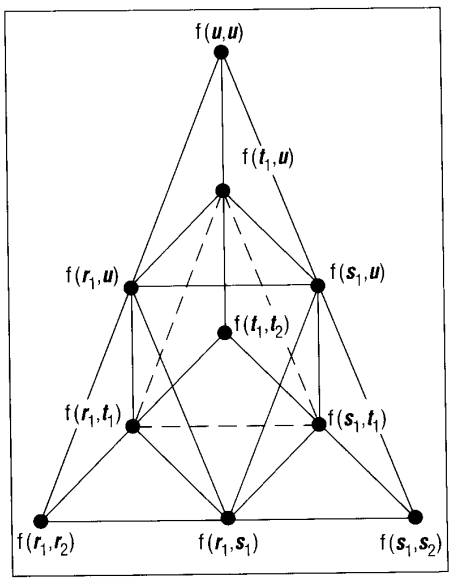




**Figure 8. (Left) The de Casteljau Algorithm for a quadratic Bezier patch. Figure 9. (Right) The de Boor Algorithm for a quadratic B-patch.**

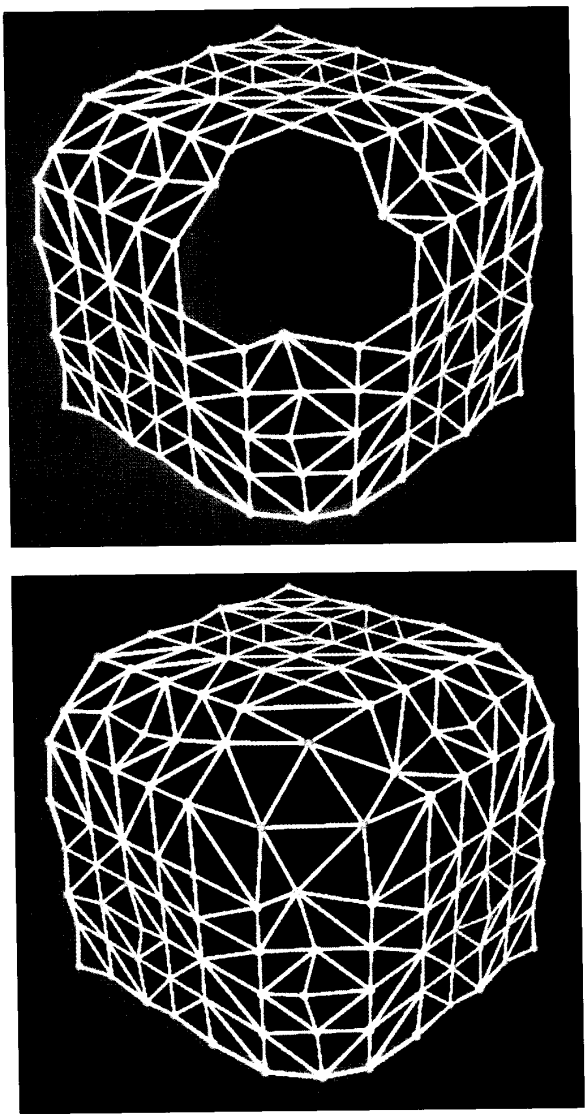


**Figure 10. Solving the polygonal hole problem using triangular B-splines. First, represent the piecewise polynomial surface around the hole as a linear combination of B-splines (a). Then extend this B-spline surface to produce an overall $C^{n-1}$-continuous fill of the hole (b). Note that this method can achieve $C^1$-continuity with piecewise quadratics, and $C^2$-continuity with piecewise cubics.**

$$\mathbf{c}_{ijk}^{l} = f_l(\mathbf{r}_1^l, \ldots, \mathbf{r}_i^l, \mathbf{s}_1^l, \ldots, \mathbf{s}_j^l, \mathbf{t}_1^l, \ldots, \mathbf{t}_k^l) \quad \textbf{(8)}$$

where $f_l$ is the polar form of the restriction of $F$ to the triangle $\Delta_l$.

Among other applications, we can use this theorem to solve the polygonal hole problem, as shown in Figure 10. ❑


## Acknowledgments

This work has been partly supported by the Natural Sciences and Engineering Research Council of Canada through Personal Operating Grant OGP0105573 and Strategic Operating Grant STR0040527.


## References


1. P. de Casteljau. "Outillages Methodes Calculations," tech. report, Andre Citroen, Paris, 1959.
2. P. de Casteljau. *Formes à Pôles,* Hermes, Paris, 1985.
3. L. Ramshaw, "Blossoming: A Connect-the-Dots Approach to Splines," tech. report, Digital Systems Research Center, Palo Alto, Calif., 1987.
4. L. Ramshaw, "Bezier and B-splines as Multiaffine Maps," *Theoretical Foundations of Computer Graphics and CAD,* R. Earnshaw, ed., 1988, pp. 757-776.
5. L. Ramshaw. "Blossoms are Polar Forms," *Computer Aided Geometric Design,* Vol. 6, 1989, pp. 323-358.
6. P. de Casteljau, *Le Lissage*, Hermes, Paris, 1990.
7. H.P. Seidel, "A New Multiaffine Approach to B-splines," *Computer Aided Geometric Design,* Vol. 6, 1989, pp. 23-32.
8. P.J. Barry and R.N. Goldman, "Algorithms for Progressive Curves: Extending B-spline and Blossoming Techniques to the Monomial, Power, and Newton Dual Bases," in *Knot Insertion and Deletion Algorithms for B-Spline Modeling,* R.N. Goldman and T. Lyche, eds., SIAM, Bellingham, Wash., 1992.
9. R.N. Goldman, "Blossoming and Knot Insertion Algorithms for B-spline Curves," *Computer Aided Geomtric Design,* Vol. 7, 1990, pp. 69-81.
10. E.T.Y. Lee, "A Note on Blossoming," *Computer-Aided Geometric Design*, Vol. 6, 1989, pp. 359-362.
11. K. Strom, *Splines, Polynomials, and Polar Forms*, Phd thesis, University, of Oslo, Oslo, Norway, 1992.
12. H.P. Seidel, "Polar Forms and Triangular B-Spline Surfaces," in *Euclidean Geometry and Computers*, A. Du and F. Hwang, eds., World Scientific Publishing, River Edge, N.J., 1992.
13. G. Schmeltz. *Variation Diminishing Curve Representations and Curvature Criteria for Bezier Surfaces*, PhD thesis, Technical University of Darmstadt, Germany, 1992.
14. T. DeRose, R.N. Goldman, and M. Lounsberry, "A Tutorial Introduction to Blossoming," in *Geometric Modelling, Methods and Applications,* H. Hagen and D. Roller, eds., Springer Verlag, New York, 1991, pp. 267-286.
15. H.P. Seidel, "Symmetric Recursive Algorithms for Surfaces: B-patches and the de Boor Algorithm for Polynomials over Triangles," *Constructive Approximations,* Vol. 7, 1991, pp. 257-279.
16. W. Dahmen, C.A. Micchelli, and H.P. Seidel, "Blossoming Begets B-splines Built Better by B-patches," *Mathematics of Computation*, Vol. 59, No. 199, July 1992, pp. 97-115.
17. P. Fong, *Shape Control for B-splines over Arbitrary Triangulations,* master's thesis, University of Waterloo, Waterloo, Canada, 1992.
18. P.J. Barry et al., *Blossoming: The New Polar-Form Approach to Spline Curves and Surfaces*, Siggraph 91 Course Notes, No. 26. Siggraph, Chicago, 1991.



**Hans-Peter Seidel** is a professor of computer science at the University of Erlangen, Germany, and an adjunct professor of computer science at the University of Waterloo, Canada. His research interests include computer graphics, computer-aided geometric design, free-form curves and surfaces, and geometric modeling. He is editor in chief of *Computer Graphics Forum.*

Readers may reach Seidel at Universität Erlangen, IMMD IX - Graphische Datenverarbeitung, Am Weichselgarten 9, W-8520 Erlangen, Germany, e-mail seidel@informatik.uni-erlangen.de.