Composing Bézier Simplexes

TONY D. DEROSE
University of Washington

This paper describes two algorithms for solving the following general problem: Given two polynomial maps $\mathbf{f}: \mathbb{R}^n \mapsto \mathbb{R}^N$ and $\mathbf{S}: \mathbb{R}^N \mapsto \mathbb{R}^d$ in Bézier simplex form, find the composition map $\tilde{\mathbf{S}} = \mathbf{S} \circ \mathbf{f}$ in Bézier simplex form (typically, $n \leq N \leq d \leq 3$). One algorithm is more appropriate for machine implementation, while the other provides somewhat more geometric intuition. The composition algorithms can be applied to the following problems: evaluation, subdivision, and polynomial reparameterization of Bézier simplexes; joining Bézier curves with geometric continuity of arbitrary order; and the determination of the control nets of Bézier curves and triangular Bézier surface patches after they have undergone free-form deformations.

Categories and Subject Descriptors: I.3.5 [**Computer Graphics**]: Computational Geometry and Object Modeling—*curve, surface, solid, and object representations*; J.6 [**Computer Applications**]: Computer-Aided Engineering—*computer-aided design (CAD)*

General Terms: Algorithms

Additional Key Words and Phrases: Bézier curves, computer-aided geometric design, free-form deformations, geometric continuity, triangular Bézier surface patches

1. INTRODUCTION

This paper describes two algorithms for solving the following general problem: Given two polynomial maps $\mathbf{f}: \mathbb{R}^n \mapsto \mathbb{R}^N$ and $\mathbf{S}: \mathbb{R}^N \mapsto \mathbb{R}^d$ in Bézier simplex form, find the composition map $\tilde{\mathbf{S}} = \mathbf{S} \circ \mathbf{f}$ also in Bézier simplex form.

A mathematician might find this problem interesting in its own right, especially since the solution will turn out to possess a certain degree of elegance. A practitioner, however, might well question the relevance of the problem to issues in computer-aided geometric design (CAGD). To provide evidence that functional composition is indeed useful in CAGD, we explicitly address several applications of it.

Some simple applications of functional composition are the evaluation, subdivision, and polynomial reparameterization of Bézier simplexes (for now, think of a Bézier simplex as a generalization of a triangular Bézier surface patch [7]). *Evaluation* can be viewed as composition with a constant function;

This work was supported in part by the National Science Foundation under grant DMC-8602141 and in part by the Digital Equipment Corporation under the Faculty Program/Incentives for Excellence.
Author's address: Department of Computer Science, FR-35, University of Washington, Seattle, WA 98195.

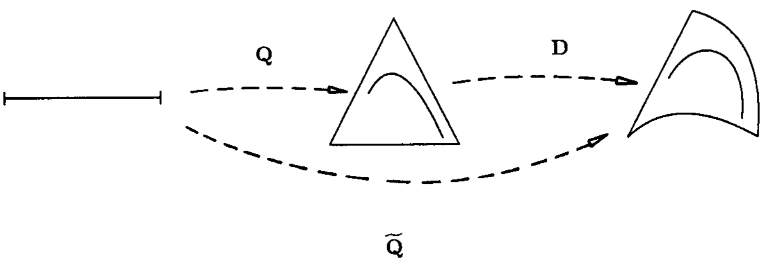


Fig. 1. Free-form deformation.

reparameterization is, by definition, composition with a change of variables; *subdivision* [7, 8] is a special case of reparameterization where the change of variables is a linear function.

Another interesting application arises from a method of geometric modeling that has recently been introduced by Sederberg and Parry [13]. In their scheme, geometric objects such as polyhedrons, spheres, and Bézier curves and surfaces are imagined to be embedded in a deformable medium. The objects can then be manipulated by deforming the medium that surrounds them. Sederberg and Parry note that deformations can be intuitively controlled by designers if they are represented as a polynomial map in Bernstein–Bézier form. For instance, suppose that $\mathbf{Q}$ is a Bézier curve in the plane and that $\mathbf{D}$ is the deformation from the plane onto itself. The deformed curve $\tilde{\mathbf{Q}}$ is defined to be $\mathbf{D} \circ \mathbf{Q}$, as depicted in Figure 1; thus, the deformed curve results from functional composition. (Historical note: Bézier [2] had, much earlier, proposed a very similar idea, but in a slightly different context than Sederberg and Parry. Instead of using composition as a basis for geometric modeling, Bézier's idea was to use composition to locally fine-tune a surface. In the same paper Bézier went some distance toward the problem posed herein in that he provided formulas for the monomial form of the composed object.)

In some applications it may be acceptable to represent $\tilde{\mathbf{Q}}$ by maintaining $\mathbf{Q}$ and $\mathbf{D}$ independently, requiring the software to perform the composition separately for each point of $\tilde{\mathbf{Q}}$ that is required by the application. For instance, if $\tilde{\mathbf{Q}}$ is to be rendered on a computer graphics screen, special-purpose rendering software could be written to compute the points $\mathbf{Q}_t = \mathbf{Q}(t)$ for steadily increasing values of t; for each value of t, the point $\mathbf{D}(\mathbf{Q}_t)$ could then be computed and displayed on the screen. Alternatively, since composition is the mechanism by which deformation is accomplished, the algorithms presented herein can be used to compute directly the control polygon for the deformed curve (assuming the deformation is representable as a Bézier simplex), thereby allowing the use of standard Bézier rendering software. (Aside from computational issues, the second approach might be more desirable from a software-engineering standpoint since it promotes a definite, standard interface between the modeling and rendering phases of the design.)

The last application of composition we examine concerns the problem of joining Bézier curves with geometric continuity [1, 5, 6] of arbitrary order. Geometric continuity is intimately related to reparameterization, and since reparameterization is accomplished via composition, the study of functional composition is a natural outgrowth of the study of geometric continuity.

It was with applications such as these in mind that we were led to the general composition problem posed above. A thorough study of the composition of Bézier techniques would require the examination of the composition of two Bézier simplexes, the composition of two tensor product forms (the case considered in [2]), and the two possible ways of composing a Bézier simplex and a tensor product form. Each of these four possibilities has potential practical application, particularly as a method of free-form deformation. However, in the interests of brevity (and elegance), we have chosen to study only the composition of two Bézier simplexes in this paper; the treatment of the other cases will be left as a topic of a future paper.

The presentation is organized as follows: Section 2 reviews some basic definitions and results concerning Bézier simplexes; in Section 3 the *product algorithm* for the composition of Bézier curves (Bézier simplexes of dimension 1) is developed and studied in some detail; this analysis leads to an alternative algorithm, called the *blossom algorithm*, whose major strength is its geometric interpretation; in Section 4 the product and blossom algorithms are then extended to Bézier simplexes of arbitrary dimension;[1] finally, in Section 5 the application of the composition algorithms to free-form deformations and geometric continuity are presented.

1.1 Notation

Throughout this paper we adhere to the following notational conventions:

—Scalar-valued quantities are set in italics.
—Vector-valued quantities are set in boldface.
—Multiindices are denoted by italic characters ornamented with a diacritical arrow, as in $\vec{i}$. For our purposes, multiindices are tuples of nonnegative integers. We use the notation $\vec{i} \in \mathbf{Z}_+^n$ to mean that $\vec{i}$ is a multiindex containing $n + 1$ components subscripted 0 to n; that is, $\vec{i} = (i_0, \ldots, i_n)$. The norm of $\vec{i}$, denoted by $|\vec{i}|$, is defined to be the sum of the components of $\vec{i}$.

2. BÉZIER SIMPLEXES

This section provides a brief review of the basic definitions and some useful results concerning Bézier simplexes. This is not a complete introduction to the subject; only those aspects of the theory that directly impact composition are discussed. The interested reader is encouraged to consult de Boor [4] for a particularly elegant development of much of the theory of Bézier simplexes (which de Boor calls *B-forms*).

Every polynomial $f(u)$ of degree less than or equal to k can be expressed in the Bernstein–Bézier basis,

$$f(u) = \sum_{p=0}^{k} C_p B_p^k(u), \tag{2.1}$$

[1] Although the proofs of the algorithms for Bézier simplexes of arbitrary dimension follow the same basic lines as the proofs for curves, the notation in the general case is somewhat more cumbersome; it was therefore felt that a more pedagogic presentation would result by treating curves separately.

where $B_p^k(u)$ is the pth Bernstein polynomial of degree k, defined explicitly by

$$B_p^k(u) = \binom{k}{p} u^p (1-u)^{k-p}$$

or recursively by

$$B_p^k(u) = \begin{cases} 1 & \text{if } \ k = p = 0, \\ 0 & \text{if } \ p < 0 \ \text{ or } \ p > k, \\ (1-u)B_p^{k-1}(u) + uB_{p-1}^{k-1}(u) & \text{otherwise.} \end{cases}$$

When a polynomial is expressed as in eq. (2.1), it is said to be given in *Bernstein–Bézier form*, and the scalars $C_0, \ldots, C_k$ are called its *Bernstein coefficients.*

To keep the equations from becoming overly cluttered, we make the notational convention that, when forming a linear combination of Bernstein polynomials, such as in eq. (2.1), the limits of summation will be dropped. This should pose no difficulty since the limits of summation can be inferred by the rule that the summation index is to take on all "sensible" values, that is, all values that keep the summand from vanishing. For example, in eq. (2.1) the index p can be inferred to range over 0 to k since the Bernstein polynomial $B_p^k(u)$ vanishes for values of p outside this range. With this notation eq. (2.1) becomes simply

$$f(u) = \sum_p C_p B_p^k(u).$$

We apply the same convention to summations over expressions containing binomial (or multinomial) coefficients. Thus, looking ahead to eq. (3.2), the summation index j is to take on all values such that neither of the binomial coefficients $\binom{k(s-1)}{j}$ nor $\binom{k}{r-j}$ vanishes.

A *Bézier curve* is simply a parametric polynomial given in Bernstein–Bézier form, for example,

$$\mathbf{S}(u) = \sum_p \mathbf{V}_p B_p^k(u), \qquad u \in [0, 1], \tag{2.2}$$

and the *control points* $\mathbf{V}_p \in \mathbb{R}^d$ are collectively called the Bézier *control polygon* of the curve.

A Bézier curve (or a polynomial given in Bernstein–Bézier form) can be evaluated for any value of its parameter via the algorithm of de Casteljau [3] (see Figure 2). de Casteljau's algorithm can be depicted schematically as a triangle, with the control polygon (or the Bernstein coefficients) appearing along the bottom edge of the triangle and the point of evaluation appearing at the apex, as shown in Figure 3.

Bernstein polynomials of several variables can also be defined. In particular, the n-variate Bernstein polynomials of degree m are explicitly defined by

$$B_{\vec{i}}^m(u_0, u_1, \ldots, u_n) = \binom{m}{\vec{i}} u_0^{i_0} u_1^{i_1} \cdots u_n^{i_n}, \qquad \vec{i} = (i_0, \ldots, i_n), \quad |\vec{i}| = m,$$

where $u_0 + u_1 + \cdots + u_n = 1$, and where

$$\binom{m}{\vec{i}} \equiv \frac{m!}{i_0! i_1! \cdots i_n!}$$

Fig. 2. de Casteljau's algorithm.

Input: A control polygon $\mathbf{V}_0, \ldots, \mathbf{V}_m$ defining a Bézier curve $\mathbf{S}$, and a parameter value u.
Output: The point $\mathbf{S}(u)$.

```
for $i \leftarrow 0$ to $m$ do
    $\mathbf{V}_i^{[0]} \leftarrow \mathbf{V}_i$
endfor
for $s \leftarrow 1$ to $m$ do
    for $i \leftarrow 0$ to $m - s$ do
        $\mathbf{V}_i^{[s]} \leftarrow (1-u)\mathbf{V}_i^{[s-1]} + u\mathbf{V}_{i+1}^{[s-1]}$
    endfor
endfor
return $\mathbf{V}_0^{[m]}$
```

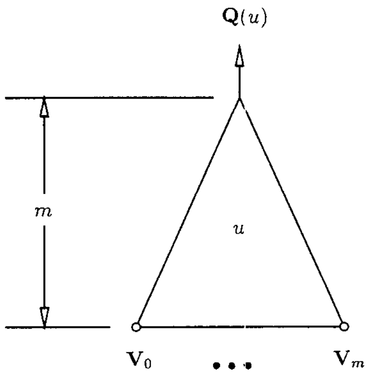


Fig. 3. Schematic of de Casteljau's algorithm.

is the multinomial coefficient. The n-variate Bernstein polynomials can also be defined recursively by

$$B_{\vec{i}}^{m}(u_0, \ldots, u_n) = \begin{cases} 1 & \text{if } m = |\vec{i}| = 0, \\ 0 & \text{if } |\vec{i}| \neq m, \\ \sum_{\alpha=0}^{n} u_\alpha B_{\vec{i}-\vec{e}_\alpha}^{m-1}(u_0, \ldots, u_n) & \text{otherwise,} \end{cases}$$

where $\vec{e}_\alpha \in \mathbf{Z}_+^n$ is the multiindex having 0 in each component, except for the αth component, which is set to 1.

We generally treat the Bernstein polynomials as being defined on *affine spaces.* To do this, we need to introduce the notions of affine combinations, general position, and simplexes.

First, we recall that, loosely speaking, an *affine space* is a collection of elements, called points, that is closed under *affine combinations.* An affine combination of points $\mathbf{P}_0, \cdots, \mathbf{P}_n$ has the form $\alpha_0\mathbf{P}_0 + \alpha_1\mathbf{P}_1 + \cdots + \alpha_n\mathbf{P}_n$, where $\alpha_0 + \alpha_1 + \cdots + \alpha_n = 1$. The points $\mathbf{P}_0, \ldots, \mathbf{P}_n$ are said to be in *general position* if none of them can be written as an affine combination of the others. The dimension of an affine space can be defined to be one less than the largest number of points in general position. For example, in an affine space of dimension 1 (a line), there are at most two points in general position; in an affine space of dimension 2 (a plane), there are at most three points in general position, and so on. Suppose $\mathcal{A}$ is an affine n-space (an affine space of dimension n) and $\mathbf{P}_0, \ldots, \mathbf{P}_k$, $k \leq n$, are points in $\mathcal{A}$ that are in general position. The convex hull of these points is called a *k-simplex,* and the points are called the vertices of the simplex. Although it may not be immediately apparent from their definition, simplexes are rather familiar objects: A 1-simplex is a line segment, a 2-simplex is a triangle, a 3-simplex is a tetrahedron, and so on. Simplexes play the same role in affine geometry that bases play in linear algebra. In particular, if $\mathcal{A}$ is an affine n-space and if $\mathcal{S}$ is an n-simplex with vertices $\mathbf{P}_0, \ldots, \mathbf{P}_n$, then every point $\mathbf{u} \in \mathcal{A}$ can be uniquely represented as an affine combination $\mathbf{u} = u_0\mathbf{P}_0 + u_1\mathbf{P}_1 + \cdots + u_n\mathbf{P}_n$. The numbers $(u_0, \ldots, u_n)$ are called the *barycentric coordinates* of $\mathbf{u}$ relative to $\mathcal{S}$.

We turn now to the use of these ideas to define Bernstein polynomials on affine spaces. If $\mathbf{u}$ is a point in an affine n-space whose barycentric coordinates relative to some n-simplex $\mathcal{S}$ are $(u_0, \ldots, u_n)$, then the Bernstein polynomials defined over $\mathcal{S}$ are given by the identification

$$B^m_{\vec{i}}(\mathbf{u}) \equiv B^m_{\vec{i}}(u_0, \ldots, u_n).$$

In preparation for later sections, we state the following lemma showing that products of Bernstein polynomials are simply expressed as a Bernstein polynomial of higher degree:

LEMMA 2.1. *Products of multivariate Bernstein polynomials:*

$$B^m_{\vec{i}}(\mathbf{u})\; B^k_{\vec{j}}(\mathbf{u}) = \frac{\binom{m}{\vec{i}}\binom{k}{\vec{j}}}{\binom{m+k}{\vec{i}+\vec{j}}}\, B^{m+k}_{\vec{i}+\vec{j}}(\mathbf{u}).$$

PROOF. First, note that the Bernstein polynomial $B^m_{\vec{i}}(\mathbf{u})$ can be written succinctly as

$$B^m_{\vec{i}}(\mathbf{u}) = \binom{m}{\vec{i}}\mathbf{u}^{\vec{i}},$$

where

$$\mathbf{u}^{\vec{i}} \equiv u_0^{i_0}u_1^{i_1} \cdots u_n^{i_n}.$$

Armed with this notation, we can proceed with the proof of the lemma by simple manipulation of the explicit definition of the multivariate Bernstein polynomials:

$$\begin{aligned}
B^m_{\vec{i}}(\mathbf{u})B^k_{\vec{j}}(\mathbf{u}) &= \binom{m}{\vec{i}}\mathbf{u}^{\vec{i}}\binom{k}{\vec{j}}\mathbf{u}^{\vec{j}} \\
&= \binom{m}{\vec{i}}\binom{k}{\vec{j}}\mathbf{u}^{\vec{i}+\vec{j}} \\
&= \frac{\binom{m}{\vec{i}}\binom{k}{\vec{j}}}{\binom{m+k}{\vec{i}+\vec{j}}}\binom{m+k}{\vec{i}+\vec{j}}\mathbf{u}^{\vec{i}+\vec{j}} \\
&= \frac{\binom{m}{\vec{i}}\binom{k}{\vec{j}}}{\binom{m+k}{\vec{i}+\vec{j}}}\, B^{m+k}_{\vec{i}+\vec{j}}(\mathbf{u}). \qquad \square
\end{aligned}$$

Finally, by a *Bézier simplex of dimension n*, we mean a map from an n-simplex $\mathcal{S}$ given in Bernstein–Bézier form, as in

$$\mathbf{S}(\mathbf{u}) = \sum_{\vec{i}} \mathbf{V}_{\vec{i}} B^m_{\vec{i}}(\mathbf{u}), \qquad \mathbf{u} \in \mathcal{S},$$

where, following our earlier convention concerning linear combinations of Bernstein polynomials, the summation is to be taken over all multiindices $\vec{i} \in \mathbf{Z}^n_+$ whose norm is m. A Bézier simplex of dimension 1 is called a Bézier curve, a Bézier simplex of dimension 2 is called a Bézier triangle, and a Bézier simplex of dimension 3 is called a Bézier tetrahedron.

Note that we can express a Bézier curve in either of two ways: in "standard form," as in eq. (2.2), or as a Bézier simplex of dimension 1, as in

$$\mathbf{S}(u_0, u_1) = \sum_{(i_0,i_1)} \mathbf{V}_{(i_0,i_1)} B^m_{(i_0,i_1)}(u_0, u_1).$$

3. COMPOSING BÉZIER CURVES

In this section we address the general composition problem for Bézier curves; the problem may be stated as follows:

Given: A Bézier curve $\mathbf{S}(t)$ of degree m whose control points are $\mathbf{V}_0, \ldots, \mathbf{V}_m$, and a polynomial function $f(u)$ of degree k whose Bernstein coefficients are $C_0, \ldots, C_k$.
Find: The Bézier control points $\tilde{\mathbf{V}}_0, \ldots, \tilde{\mathbf{V}}_{mk}$ of the composed (i.e., reparameterized) curve $\tilde{\mathbf{S}}(u) = \mathbf{S}(f(u))$.

The crux of the solution to this problem is provided by the following theorem:

THEOREM 3.1. *Let* $f\colon \mathbb{R} \mapsto \mathbb{R}$ *and* $\mathbf{S}\colon \mathbb{R} \mapsto \mathbb{R}^d$ *be given in Bernstein–Bézier form; that is,*

$$f(u) = \sum_p C_p B^k_p(u), \qquad u \in [0, 1], \quad C_p \in \mathbb{R},$$

$$\mathbf{S}(t) = \sum_i \mathbf{V}_i B^m_i(t), \qquad t \in [0, 1], \quad \mathbf{V}_i \in \mathbb{R}^d.$$

Then for any $s \in \{0, \ldots, m\}$,

$$\tilde{\mathbf{S}}(u) = \mathbf{S}(f(u)) = \sum_i B^{m-s}_i(f(u)) \sum_r \mathbf{V}^{[s]}_{i,r} B^{ks}_r(u), \tag{3.1}$$

where the points $\mathbf{V}^{[s]}_{i,r}$, $i = 0, \ldots, m-s$, $r = 0, \ldots, ks$ *are defined recursively by*

$$\mathbf{V}^{[s]}_{i,r} = \begin{cases} \mathbf{V}_i & \text{if } s = 0, \\ \dfrac{1}{\binom{ks}{r}} \sum_j \binom{k(s-1)}{j}\binom{k}{r-j} \mathbf{W}^{[s]}_{i,j,r-j} & \text{otherwise,} \end{cases} \tag{3.2}$$

and where

$$\mathbf{W}^{[s]}_{i,j,r-j} = (1 - C_{r-j})\mathbf{V}^{[s-1]}_{i,j} + C_{r-j}\mathbf{V}^{[s-1]}_{i+1,j}.$$

Discussion. A pictorial interpretation of eq. (3.1) is shown in Figure 4. Notice that the top triangle of Figure 4 is parameterized in terms of $f(u)$, whereas the

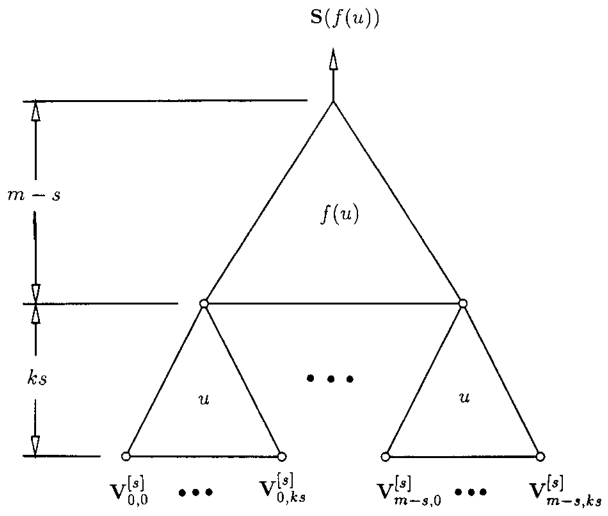

$S(f(u))$
$m-s$
$f(u)$
ks
u
u
$V^{[s]}_{0,0}$ ••• $V^{[s]}_{0,ks}$
$V^{[s]}_{m-s,0}$ ••• $V^{[s]}_{m-s,ks}$

Fig. 4. Pictorial representation of Theorem 3.1.

lower triangles are parameterized in terms of u. The essence of the theorem is that the points $\mathbf{V}^{[s]}_{i,r}$ can be computed if the points with superscript $s - 1$ are known. This relationship will be used to construct an algorithm for determining the control polygon of $\tilde{\mathbf{S}}$.

The proof of the theorem proceeds by induction on s, the parameter that controls how much of the representation is parameterized in u and how much is parameterized in $f(u)$. Although the symbol manipulation becomes rather tedious, the proof relies on only two properties of the Bernstein polynomials: their recursive definition and the ability to easily raise their degree through the product formula given in Lemma 2.1.

PROOF. By induction on s; the basis, $s = 0$, is trivially true.

Inductive hypothesis

$$\tilde{\mathbf{S}}(u) = \sum_i B_i^{m-s+1}(f(u)) \sum_j \mathbf{V}^{[s-1]}_{i,j} B_j^{k(s-1)}(u), \tag{3.3}$$

where the points $\mathbf{V}^{[s-1]}_{i,j}$ are defined recursively as in the statement of the theorem. For convenience, we define

$$\mathbf{T}_i^{[s-1]}(u) = \sum_j \mathbf{V}^{[s-1]}_{i,j} B_j^{k(s-1)}(u). \tag{3.4}$$

Using eq. (3.4) together with the recursive definition of the Bernstein polynomials

$$B_i^{m-s+1}(f(u)) = (1 - f(u))B_i^{m-s}(f(u)) + f(u)B_{i-1}^{m-s}(f(u)), \tag{3.5}$$

eq. (3.3) can be rewritten as

$$\tilde{\mathbf{S}}(u) = \sum_i B_i^{m-s}(f(u))\{(1 - f(u))\mathbf{T}_i^{[s-1]}(u) + f(u)\mathbf{T}_{i+1}^{[s-1]}(u)\}. \tag{3.6}$$

By using the Bernstein representation of $f(u)$ and the fact that the Bernstein polynomials form a partition of unity, the term in curly braces can be expressed as

$$\sum_{p}\left((1-C_p)\mathbf{T}_i^{[s-1]}(u)+C_p\mathbf{T}_{i+1}^{[s-1]}(u)\right)B_p^k(u). \tag{3.7}$$

Substituting eq. (3.4) into expression (3.7) yields

$$\sum_{j,p}\left((1-C_p)\mathbf{V}_{i,j}^{[s-1]}+C_p\mathbf{V}_{i+1,j}^{[s-1]}\right)B_j^{k(s-1)}(u)B_p^k(u). \tag{3.8}$$

Substituting expression (3.8) in place of the term in curly braces in eq. (3.6) and using the definition of the points $\mathbf{W}_{i,j,p}^{[s]}$ result in

$$\tilde{\mathbf{S}}(u)=\sum_{i}B_i^{m-s}(f(u))\sum_{j,p}\mathbf{W}_{i,j,p}^{[s]}B_j^{k(s-1)}(u)B_p^k(u). \tag{3.9}$$

Lemma 2.1, specialized to the case of univariate Bernstein polynomials, can now be used to rewrite eq. (3.9) as

$$\tilde{\mathbf{S}}(u)=\sum_{i}B_i^{m-s}(f(u))\sum_{j,p}\frac{\binom{k(s-1)}{j}\binom{k}{p}}{\binom{ks}{j+p}}\mathbf{W}_{i,j,p}^{[s]}B_{j+p}^{ks}(u). \tag{3.10}$$

The proof is completed by regrouping the terms in the inner two summations by choosing summation indices j and $r \equiv j+p$. The resulting sequence of expressions is

$$\begin{aligned}\tilde{\mathbf{S}}(u)&=\sum_{i}B_i^{m-s}(f(u))\sum_{r}\left\{\frac{1}{\binom{ks}{r}}\sum_{j}\binom{k(s-1)}{j}\binom{k}{r-j}\mathbf{W}_{i,j,r-j}^{[s]}\right\}B_r^{ks}(u)\\&=\sum_{i}B_i^{m-s}(f(u))\sum_{r}\mathbf{V}_{i,r}^{[s]}B_r^{ks}(u).\end{aligned} \tag{3.11}$$

□

COROLLARY 3.2. *Let f and $\mathbf{S}$ be as in Theorem 3.1. The control polygon $\tilde{\mathbf{V}}_0, \ldots, \tilde{\mathbf{V}}_{mk}$ for the composed (reparameterized) curve $\tilde{\mathbf{S}}$ is given by*

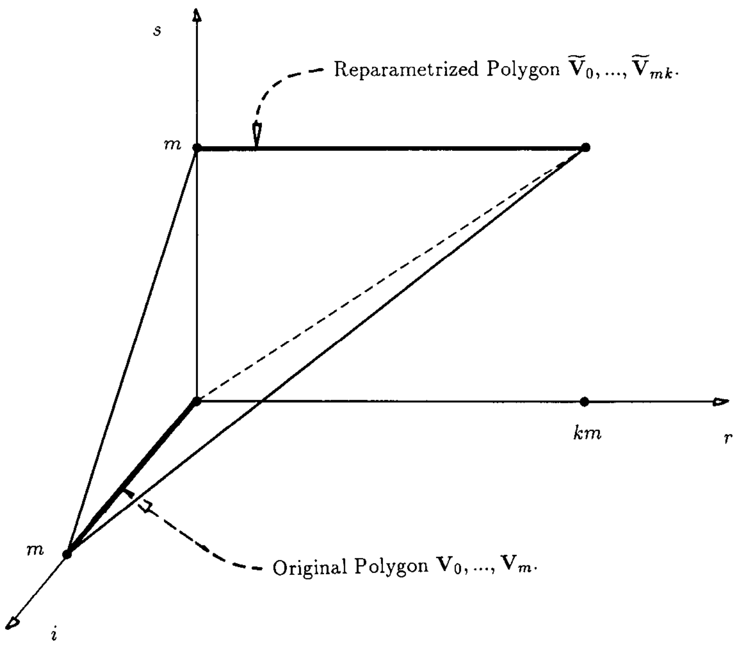
$$\tilde{\mathbf{V}}_r=\mathbf{V}_{0,r}^{[m]},\qquad r=0,\ldots,mk,$$

where the points $\mathbf{V}_{i,r}^{[s]}$ are as defined in Theorem 3.1.

PROOF. Set $s = m$ in Theorem 3.1. □

Theorem 3.1 and Corollary 3.2 together define an algorithm, called the *product algorithm,* for computing the control points of the reparameterized curve $\tilde{\mathbf{S}} = \mathbf{S} \circ f$. The name *product algorithm* was chosen to emphasize that the product formula from Lemma 2.1 plays a key role in the development of the algorithm.

The algorithm proceeds by building a tetrahedral arrangement of points $\mathbf{V}_{i,r}^{[s]}$, as shown in Figure 5. The construction of the arrangement begins by placing the control polygon $\mathbf{V}_0, \ldots, \mathbf{V}_m$ along the bottom edge of the tetrahedron (i.e., the $s = 0$ level). The tetrahedron is then filled in one level at a time until the points



s
Reparametrized Polygon $\tilde{V}_0, \ldots, \tilde{V}_{mk}$.
m
km
r
m
Original Polygon $V_0, \ldots, V_m$.
i

Fig. 5. Tetrahedral arrangement of points used in Theorem 3.1.

on the edge of the tetrahedron corresponding to $s = m$ are computed. The points appearing on this edge form the control polygon of $\tilde{\mathbf{S}}(u)$. For completeness, a pseudocode version is given in Figure 6.

It is interesting to note that the Boehm–Sablonnière algorithm [11] for computing the Bézier polygon for a B-spline curve has a similar computational structure: The B-spline polygon is placed along an edge of a tetrahedral arrangement of points, the tetrahedral arrangement is computed, and the Bézier polygon is retrieved from the skew edge of the tetrahedron. One should not attempt to read too much into this interpretation though since there are substantial differences between the algorithms. For instance, the Boehm–Sablonnière algorithm uses two recurrence relations instead of one to build the tetrahedral arrangement of points.

3.1 The Blossom Algorithm for Curves

The product algorithm is relatively efficient computationally and quite easy to implement, but the geometric intuition it provides is somewhat limited. To gain more geometric intuition into the problem, we now informally describe a variant of the product algorithm called the *blossom algorithm.* The term *blossom* is explained shortly, and a rigorous proof of the algorithm appears later in this section.

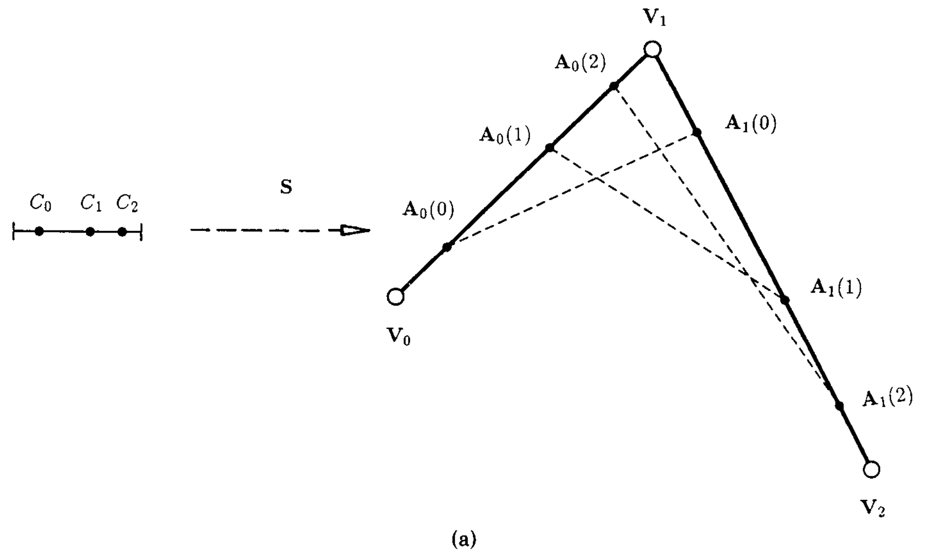
We begin by noting that the Bernstein coefficients of the function f can be viewed geometrically simply by plotting them as points in the domain interval of $\mathbf{S}$, as shown in Figure 7a, where $C_0 = 0.2$, $C_1 = 0.6$, and $C_2 = 0.85$. The blossom

```
Input: A control polygon $\mathbf{V}_0, \ldots, \mathbf{V}_m$ defining a Bézier curve $\mathbf{S}$.
and set of Bernstein coefficients $C_0, \ldots, C_k$ defining a polynomial $f$.

Output: A control polygon $\tilde{\mathbf{V}}_0, \ldots, \tilde{\mathbf{V}}_{mk}$ defining the composed
(reparameterized) curve $\tilde{\mathbf{S}} = \mathbf{S} \circ f$.

for $i \leftarrow 0$ to $m$ do
    $\mathbf{V}_{i,0}^{[0]} \leftarrow \mathbf{V}_i$
endfor
for $s \leftarrow 1$ to $m$ do
    for $r \leftarrow 0$ to $ks$ do
        for $i \leftarrow 0$ to $m - s$ do
            $\mathbf{V}_{i,r}^{[s]} \leftarrow 0$
            $j_{\min} = \max\{0, r - k\}$
            $j_{\max} = \min\{r, k(s-1)\}$
            for $j \leftarrow j_{\min}$ to $j_{\max}$ do
                $\mathbf{V}_{i,r}^{[s]} \leftarrow \mathbf{V}_{i,r}^{[s]} + \binom{k(s-1)}{j}\binom{k}{r-j}\left\{(1 - C_{r-j})\mathbf{V}_{i,j}^{[s-1]} + C_{r-j}\mathbf{V}_{i+1,j}^{[s-1]}\right\}$
            endfor
            $\mathbf{V}_{i,r}^{[s]} \leftarrow \frac{\mathbf{V}_{i,r}^{[s]}}{\binom{ks}{r}}$
        endfor
    endfor
endfor
for $r \leftarrow 0$ to $mk$ do
    $\tilde{\mathbf{V}}_r \leftarrow \mathbf{V}_{0,r}^{[m]}$
endfor
```

Fig. 6. The product algorithm for curves.

algorithm proceeds as follows:

(1) On each leg of the original polygon $\mathbf{V}_0, \ldots, \mathbf{V}_m$, draw the images of $C_0, \ldots, C_k$, treated as points in $\mathbf{S}$'s domain, under the affine map that carries 0 to the starting vertex of the leg and carries 1 to the ending vertex. Label the image of C_{j1} on the leg $\mathbf{V}_i\mathbf{V}_{i+1}$ with $\mathbf{A}_i(j_1)$. Stated algebraically, $\mathbf{A}_i(j_1) = (1 - C_{j1})\mathbf{V}_i + C_{j1}\mathbf{V}_{i+1}$. Figure 7a depicts the case for $m = k = 2$, that is, for a quadratic reparameterization of a quadratic curve.
(2) Connect corresponding images on adjacent legs. That is, for each i, draw a line segment between $\mathbf{A}_i(j_1)$ and $\mathbf{A}_{i+1}(j_1)$. This results in $k + 1$ polygons, each with m vertices. For the case of $m = k = 2$, there are 3 polygons, each with 2 vertices, as shown in Figure 7a.
(3) For each of the polygons produced in step (2), repeat steps (1) and (2), labeling the newly constructed points with two arguments. For instance, $\mathbf{A}_0(0, 2)$ is the image of C_2 on the zeroth leg of the polygon produced by connecting the images of C_0, as shown in Figure 7b. Algebraically, $\mathbf{A}_0(0, 2) = (1 - C_2)\mathbf{A}_0(0) + C_2\mathbf{A}_1(0)$.
(4) Repeat the above steps of creating polygons, marking images of the C's, etc., until points with m arguments $\mathbf{A}_0(j_1, \ldots, j_m)$ are produced.
(5) The rth control point of the reparameterized curve can now be constructed by forming a convex combination of all points with m arguments whose arguments sum to r. Thus, in the case of $m = k = 2$, the zeroth control point



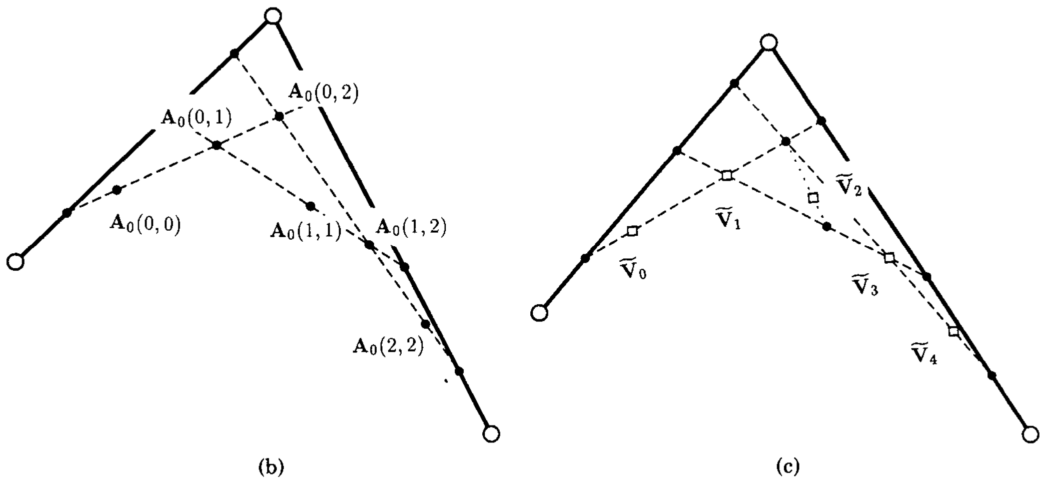


Fig. 7. Quadratic reparameterization of a quadratic curve.

$\tilde{\mathbf{V}}_0$ is simply the point $\mathbf{A}_0(0, 0)$. The first control point $\tilde{\mathbf{V}}_1$ is a convex combination of the points $\mathbf{A}_0(0, 1)$ and $\mathbf{A}_0(1, 0)$; however, it can be shown that the **A**'s are symmetric with respect to permutation of their arguments, implying that $\mathbf{A}_0(0, 1) = \mathbf{A}_0(1, 0)$, which in turn implies that $\tilde{\mathbf{V}}_1 = \mathbf{A}_0(0, 1)$, as shown in Figure 7c. The situation for $\tilde{\mathbf{V}}_2$ is somewhat more interesting. It is formed by a convex combination of the points $\mathbf{A}_0(0, 2)$, $\mathbf{A}_0(1, 1)$, and $\mathbf{A}_0(2, 0)$, but owing to symmetry, this is equivalent to a convex combination of the two points $\mathbf{A}_0(0, 2)$ and $\mathbf{A}_0(1, 1)$. The specific convex combination in this case is such that $\tilde{\mathbf{V}}_2$ divides the segment $\mathbf{A}_0(0, 2)\mathbf{A}_0(1, 1)$ into relative distances 2:1, as shown in Figure 7c. The convex combination required in the general case will be described in Claim 3.3.

Remarks. One way to think of evaluation of a Bézier curve $\mathbf{S}(t)$ for a fixed value $t = t^*$ is to compose $\mathbf{S}$ with a constant function $f(u) = t^*$. Indeed, when f is a constant function (i.e., when $f(u) = C_0$), the blossom algorithm reduces to the de Casteljau algorithm for constructing the point $\mathbf{S}(C_0)$; thus, the blossom algorithm is actually a generalization of the de Casteljau algorithm.

When f is a linear function, the composition algorithms provide methods for performing arbitrary linear subdivision of the curve. In particular, if f has Bernstein coefficients C_0 and C_1, the algorithms compute the control points for the portion of $\mathbf{S}$ generated when $\mathbf{S}$'s parameter ranges over the interval $[C_0, C_1]$. Goldman [8] has previously described an alternative (more efficient) method for accomplishing this task based on degenerate Bézier tetrahedrons.

To prove the correctness of the blossom algorithm, we begin by defining some new quantities $\Delta_i(a_1, \ldots, a_l; \mathbf{V})$ recursively by

$$\Delta_i(a_1, \ldots, a_l; \mathbf{V}) = \begin{cases} (1 - a_1)\mathbf{V}_i + a_1\mathbf{V}_{i+1} & \text{if } l = 1, \\ (1 - a_l)\Delta_i(a_1, \ldots, a_{l-1}; \mathbf{V}) & \\ \qquad + a_l\Delta_{i+1}(a_1, \ldots, a_{l-1}; \mathbf{V}) & \text{otherwise.} \end{cases} \tag{3.12}$$

The Δ's are actually quite closely related to the constructed points $\mathbf{A}$ in the above informal description. More precisely, for any $l > 0$,

$$\mathbf{A}_i(j_1, \ldots, j_l) = \Delta_i(C_{j_1}, \ldots, C_{j_l}; \mathbf{V}).$$

Ramshaw [10] calls $\Delta_i(a_1, \ldots, a_l; \mathbf{V})$ the *blossom* of the Bézier curve $\sum_j \mathbf{V}_{i+j}B_j^l(u)$. Indeed, when all a's are identical and equal to u, we have the identity $\Delta_i(u, \ldots, u; \mathbf{V}) = \sum_j \mathbf{V}_{i+j}B_j^l(u)$. It was previously mentioned that the $\mathbf{A}$'s are symmetric with respect to permutation of their arguments, which is equivalent to the fact, proved by Ramshaw [10], that the blossom $\Delta_i(a_1, \ldots, a_l; \mathbf{V})$ is symmetric with respect to permutation of the a's.

The next claim is a precise statement and proof of the blossom algorithm:

CLAIM 3.3. *The points* $\mathbf{V}_{i,r}^{[s]}$ *from Theorem* 3.1 *are convex combinations of all points* $\Delta_i(C_{i_1}, \ldots, C_{i_s}; \mathbf{V})$ *where* $i_1 + i_2 + \cdots + i_s = r$. *More precisely,*

$$\mathbf{V}_{i,r}^{[s]} = \sum_{\substack{i_1,\ldots,i_s \in \{0,\ldots,k\} \\ i_1+i_2+\cdots+i_s=r}} \mathcal{C}_r(i_1, \ldots, i_s)\Delta_i(C_{i_1}, \ldots, C_{i_s}; \mathbf{V}),$$

where $\mathcal{C}_r(i_1, \ldots, i_s)$ *are combinatorial constants given by*

$$\mathcal{C}_r(i_1, \ldots, i_s) = \frac{\binom{k}{i_1}\binom{k}{i_2} \cdots \binom{k}{i_s}}{\binom{ks}{r}}.$$

PROOF. By induction on s. The basis, $s = 1$, can be directly verified from the definition of $\mathbf{V}_{i,r}^{[1]}$:

$$\begin{aligned} \mathbf{V}_{i,r}^{[1]} &= (1 - C_r)\,\mathbf{V}_i + C_r\,\mathbf{V}_{i+1} \\ &= \Delta_i(C_r; \mathbf{V}) \\ &= \mathcal{C}_r(r)\Delta_i(C_r; \mathbf{V}) \\ &= \sum_{i_1=r} \mathcal{C}_r(i_1)\Delta_i(C_{i_1}; \mathbf{V}). \end{aligned}$$

Inductive hypothesis

$$\mathbf{V}_{i,r}^{[s-1]} = \sum_{\substack{i_1,\ldots,i_{s-1}\in\{0,\ldots,k\}\\ i_1+i_2+\cdots+i_{s-1}=r}} \mathcal{C}_r(i_1,\ldots,i_{s-1})\Delta_i(C_{i_1},\ldots,C_{i_{s-1}};\mathbf{V}).$$

Into the recursive definition of $\mathbf{V}_{i,r}^{[s]}$,

$$\mathbf{V}_{i,r}^{[s]} = \sum_j \frac{\binom{k(s-1)}{j}\binom{k}{r-j}}{\binom{ks}{r}} \{(1-C_{r-j})\mathbf{V}_{i,j}^{[s-1]} + C_{r-j}\mathbf{V}_{i+1,j}^{[s-1]}\},$$

we substitute the inductive hypothesis, once for $\mathbf{V}_{i,j}^{[s-1]}$ and once for $\mathbf{V}_{i+1,j}^{[s-1]}$, to obtain

$$\mathbf{V}_{i,r}^{[s]} = \sum_j \sum_{\substack{i_1,\ldots,i_{s-1}\in\{0,\ldots,k\}\\ i_1+i_2+\cdots+i_{s-1}=j}} \mathcal{C}_j(i_1,\ldots,i_{s-1})\frac{\binom{k(s-1)}{j}\binom{k}{r-j}}{\binom{ks}{r}}$$
$$\times\{(1-C_{r-j})\Delta_i(C_{i_1},\ldots,C_{i_{s-1}};\mathbf{V}) + C_{r-j}\Delta_{i+1}(C_{i_1},\ldots,C_{i_{s-1}};\mathbf{V})\}.$$

By definition of the Δ's, this reduces to

$$\mathbf{V}_{i,r}^{[s]} = \sum_j \sum_{\substack{i_1,\ldots,i_{s-1}\in\{0,\ldots,k\}\\ i_1+i_2+\cdots+i_{s-1}=j}} \mathcal{C}_r(i_1,\ldots,i_{s-1})\frac{\binom{k(s-1)}{j}\binom{k}{r-j}}{\binom{ks}{r}}$$
$$\times\Delta_i(C_{i_1},\ldots,C_{i_{s-1}},C_{r-j};\mathbf{V}).$$

By substituting $r - i_s$ for j, we obtain

$$\mathbf{V}_{i,r}^{[s]} = \sum_{i_s} \sum_{\substack{i_1,\ldots,i_{s-1}\in\{0,\ldots,k\}\\ i_1+i_2+\cdots+i_{s-1}=r-i_s}} \mathcal{C}_{r-i_s}(i_1,\ldots,i_{s-1})\frac{\binom{k(s-1)}{r-i_s}\binom{k}{i_s}}{\binom{ks}{r}}$$
$$\times\Delta_i(C_{i_1},\ldots,C_{i_{s-1}},C_{i_s};\mathbf{V}),$$

which can be equivalently written as

$$\mathbf{V}_{i,r}^{[s]} = \sum_{\substack{i_1,\ldots,i_s\in\{0,\ldots,k\}\\ i_1+i_2+\cdots+i_s=r}} \mathcal{C}_{r-i_s}(i_1,\ldots,i_{s-1})\frac{\binom{k(s-1)}{r-i_s}\binom{k}{i_s}}{\binom{ks}{r}}$$
$$\times\Delta_i(C_{i_1},\ldots,C_{i_{s-1}},C_{i_s};\mathbf{V}). \qquad (3.13)$$

By noting that

$$\mathcal{C}_r(i_1,\ldots,i_s) = \mathcal{C}_{r-i_s}(i_1,\ldots,i_{s-1})\frac{\binom{k(s-1)}{r-i_s}\binom{k}{i_s}}{\binom{ks}{r}},$$

eq. (3.13) becomes

$$\mathbf{V}_{i,r}^{[s]} = \sum_{\substack{i_1,\ldots,i_s\in\{0,\ldots,k\}\\ i_1+i_2+\cdots+i_s=r}} \mathcal{C}_r(i_1,\ldots,i_s)\Delta_i(C_{i_1},\ldots,C_{i_{s-1}},C_{i_s};\mathbf{V}),$$

thus completing the proof. □

4. COMPOSITION OF BÉZIER SIMPLEXES

In this section we generalize the results of Section 3 to the case in which Bézier simplexes of arbitrary dimension are composed.

THEOREM 4.1. *If* $\mathbf{f}: \mathbb{R}^n \mapsto \mathbb{R}^N$ *and* $\mathbf{S}: \mathbb{R}^N \mapsto \mathbb{R}^d$ *are Bézier simplexes of dimension n and N, respectively,*

$$\mathbf{f}(\mathbf{u}) = \sum_{\vec{p}} \mathbf{C}_{\vec{p}} B^k_{\vec{p}}(\mathbf{u}), \qquad \mathbf{u} \in \mathbb{R}^n, \quad \vec{p} \in \mathbf{Z}^n_+, \quad \mathbf{C}_{\vec{p}} \in \mathbb{R}^N,$$

$$\mathbf{S}(\mathbf{t}) = \sum_{\vec{i}} \mathbf{V}_{\vec{i}} B^m_{\vec{i}}(\mathbf{t}), \qquad \mathbf{t} \in \mathbb{R}^N, \quad \vec{i} \in \mathbf{Z}^N_+, \quad \mathbf{V}_{\vec{i}} \in \mathbb{R}^d,$$

then, for any $s \in \{0, \ldots, m\}$,

$$\tilde{\mathbf{S}}(\mathbf{u}) \equiv \mathbf{S}(\mathbf{f}(\mathbf{u})) = \sum_{\vec{i}} B^{m-s}_{\vec{i}}(\mathbf{f}(\mathbf{u})) \sum_{\vec{r}} \mathbf{V}^{[s]}_{\vec{i},\vec{r}} B^{ks}_{\vec{r}}(\mathbf{u}), \qquad \vec{i} \in \mathbf{Z}^N_+, \quad \vec{r} \in \mathbf{Z}^n_+,$$

where the points $\mathbf{V}^{[s]}_{\vec{i},\vec{r}}, |\vec{i}| = m - s, |\vec{r}| = ks,$ *are defined recursively by*

$$\mathbf{V}^{[s]}_{\vec{i},\vec{r}} = \begin{cases} \mathbf{V}_{\vec{i}} & \text{if } s = 0, \\ \dfrac{1}{\binom{ks}{\vec{r}}} \displaystyle\sum_{\vec{j}} \binom{k(s-1)}{\vec{j}} \binom{k}{\vec{r}-\vec{j}} \mathbf{W}^{[s]}_{\vec{i},\vec{j},\vec{r}-\vec{j}} & \text{otherwise,} \end{cases}$$

with $\vec{j} \in \mathbf{Z}^n_+$, *and where*

$$\mathbf{W}^{[s]}_{\vec{i},\vec{j},\vec{r}-\vec{j}} = \sum_{\alpha=0}^{N} C^{\alpha}_{\vec{r}-\vec{j}} \mathbf{V}^{[s-1]}_{\vec{i}+\vec{e}_\alpha,\vec{j}},$$

where $C^0_{\vec{p}}, \ldots, C^N_{\vec{p}}$ *denote the barycentric coordinates of* $\mathbf{C}_{\vec{p}}$ *relative to the domain simplex of* $\mathbf{S}$.

Discussion. The following proof is essentially obtained by rewriting the proof of Theorem 3.1 using the Bézier simplex formulation of a curve instead of the standard formulation. For completeness, we now sketch the following proof:

PROOF. By induction on s. The basis, $s = 0$, is trivially true.
For convenience, we define

$$\mathbf{T}^{[s]}_{\vec{i}}(\mathbf{u}) \equiv \sum_{|\vec{j}|=ks} \mathbf{V}^{[s]}_{\vec{i},\vec{j}} B^{ks}_{\vec{j}}(\mathbf{u}). \tag{4.1}$$

Inductive hypothesis

$$\tilde{\mathbf{S}}(\mathbf{u}) = \sum_{|\vec{i}|=m-s+1} B^{m-s+1}_{\vec{i}}(\mathbf{f}(\mathbf{u})) \mathbf{T}^{[s-1]}_{\vec{i}}(\mathbf{u}). \tag{4.2}$$

If $f^0(\mathbf{u}), f^1(\mathbf{u}), \ldots, f^N(\mathbf{u})$ denote the affine coordinates of $\mathbf{f}(\mathbf{u})$ relative to the domain simplex of $\mathbf{S}$, then the recursive definition of the multivariate Bernstein polynomials states that

$$B^{m-s+1}_{\vec{i}}(\mathbf{f}(\mathbf{u})) = \sum_{\alpha=0}^{N} f^{\alpha}(\mathbf{u}) B^{m-s}_{\vec{i}-\vec{e}_\alpha}(\mathbf{f}(\mathbf{u})). \tag{4.3}$$

Using eq. (4.3), the Bernstein representation of **f**, eq. (4.1), and Lemma 2.1, eq. (4.2) can be manipulated as follows:

$$\begin{aligned}
\tilde{\mathbf{S}}(\mathbf{u}) &= \sum_{|\vec{i}|=m-s} B^{m-s}_{\vec{i}}(\mathbf{f}(\mathbf{u})) \sum_{\alpha=0}^{N} f^{\alpha}(\mathbf{u})\mathbf{T}^{[s-1]}_{\vec{i}+\vec{e}_\alpha}(\mathbf{u}) \\
&= \sum_{|\vec{i}|=m-s} B^{m-s}_{\vec{i}}(\mathbf{f}(\mathbf{u})) \sum_{\alpha=0}^{N} \left\{ \sum_{|\vec{p}|=k} C^{\alpha}_{\vec{p}} B^{k}_{\vec{p}}(\mathbf{u}) \right\} \mathbf{T}^{[s-1]}_{\vec{i}+\vec{e}_\alpha}(\mathbf{u}) \\
&= \sum_{|\vec{i}|=m-s} B^{m-s}_{\vec{i}}(\mathbf{f}(\mathbf{u})) \sum_{|\vec{p}|=k} \sum_{|\vec{j}|=k(s-1)} \left\{ \sum_{\alpha=0}^{N} C^{\alpha}_{\vec{p}} \mathbf{V}^{[s-1]}_{\vec{i}+\vec{e}_\alpha,\vec{j}} \right\} B^{k(s-1)}_{\vec{j}}(\mathbf{u}) B^{k}_{\vec{p}}(\mathbf{u}) \\
&= \sum_{|\vec{i}|=m-s} B^{m-s}_{\vec{i}}(\mathbf{f}(\mathbf{u})) \sum_{|\vec{p}|=k} \sum_{|\vec{j}|=k(s-1)} \frac{\binom{k(s-1)}{\vec{j}}\binom{k}{\vec{p}}}{\binom{ks}{\vec{j}+\vec{p}}} \mathbf{W}^{[s]}_{\vec{i},\vec{j},\vec{p}} B^{ks}_{\vec{j}+\vec{p}}(\mathbf{u}).
\end{aligned} \tag{4.4}$$

The proof is completed by regrouping terms in much the same way as in the final steps of the proof of Theorem 3.1. In particular, introduce summation indices $\vec{j}$ and $\vec{r}$ such that $\vec{r} = \vec{j} + \vec{p}$, and then use the definition of the **V**'s in terms of the **W**'s. □

COROLLARY 4.2

$$\tilde{\mathbf{S}}(\mathbf{u}) \equiv \mathbf{S}(\mathbf{f}(\mathbf{u})) = \sum_{|\vec{r}|=mk} \tilde{\mathbf{V}}_{\vec{r}} B^{mk}_{\vec{r}}(\mathbf{u}), \qquad \mathbf{u} \in \mathbb{R}^n, \quad \vec{r} \in \mathbf{Z}^n_+,$$

where

$$\tilde{\mathbf{V}}_{\vec{r}} = \mathbf{V}^{[m]}_{\vec{0},\vec{r}},$$

and $\vec{0} \in \mathbf{Z}^N_+$ *is the multiindex consisting entirely of zeros.*

PROOF. Set $s = m$ in Theorem 4.1. □

Theorem 4.1 and Corollary 4.2 together define the product algorithm for composing Bézier simplexes of arbitrary dimension. The corresponding blossom algorithm enjoys all the properties possessed by the blossom algorithm for curves (e.g., it generalizes de Casteljau's algorithm) and is described by the following definition and claim. Before the blossom algorithm can be precisely stated, the blossom of a Bézier simplex **S** of arbitrary dimension N must be defined. Just as for curves, this is done recursively:

$$\Delta_{\vec{i}}(\mathbf{a}_1, \ldots, \mathbf{a}_l; \mathbf{V}) = \begin{cases} \sum_{\alpha=0}^{N} a^{\alpha}_1 \mathbf{V}_{\vec{i}+\vec{e}_\alpha} & \text{if } l = 1, \\ \sum_{\alpha=0}^{N} a^{\alpha}_l \Delta_{\vec{i}+\vec{e}_\alpha}(a_1, \ldots, a_{l-1}; \mathbf{V}) & \text{otherwise,} \end{cases} \quad \vec{i} \in \mathbf{Z}^N_+,$$

and where $a^0_l, \ldots, a^N_l$ are the barycentric coordinates of $\mathbf{a}_l$ relative to the domain simplex of **S**.

CLAIM 4.3. *The points* $\mathbf{V}^{[s]}_{\vec{i},\vec{r}}$ *from Theorem* 4.1 *are convex combinations of all points* $\Delta_{\vec{i}}(\mathbf{C}_{\vec{i}_1}, \ldots, \mathbf{C}_{\vec{i}_s}; \mathbf{V})$ *where* $\vec{i}_1 + \vec{i}_2 + \cdots + \vec{i}_s = \vec{r}$. *More precisely,*

$$\mathbf{V}^{[s]}_{\vec{i},\vec{r}} = \sum_{\substack{|\vec{i}_1| = |\vec{i}_2| = \cdots = |\vec{i}_s| = k \\ \vec{i}_1+\vec{i}_2+\cdots+\vec{i}_s=\vec{r}}} \mathcal{C}_{\vec{r}}(\vec{i}_1, \ldots, \vec{i}_s)\Delta_{\vec{i}}(\mathbf{C}_{\vec{i}_1}, \ldots, \mathbf{C}_{\vec{i}_s}; \mathbf{V}),$$

$$\vec{i} \in \mathbf{Z}^N_+, \vec{r}, \vec{i}_1, \ldots, \vec{i}_s \in \mathbf{Z}^n_+,$$

where $C_{\vec{r}}(\vec{i}_1, \ldots, \vec{i}_s)$ *are combinatorial constants given by*

$$C_{\vec{r}}(\vec{i}_1, \ldots, \vec{i}_s) = \frac{\binom{k}{\vec{i}_1}\binom{k}{\vec{i}_2} \cdots \binom{k}{\vec{i}_s}}{\binom{ks}{\vec{r}}}.$$

PROOF. The proof is strictly analogous to the proof of Claim 3.3. □

A specific example for the case in which $n = 1$, $N = 2$, $d = 2$, and $m = k = 2$ is presented in Section 5.1.

Remarks. Just as for curves, evaluation of a Bézier simplex can be accomplished by composition with a constant function. Also, linear subdivision, that is, the extraction of an arbitrary subsimplex, can be accomplished by composing with a degree 1 simplex of equal dimension.

5. APPLICATIONS

As mentioned in Section 1, quite a number of problems in CAGD can be viewed as functional composition, implying that the composition algorithms can be used in their solution. As was pointed out earlier, some simple examples include evaluation, subdivision, and polynomial reparameterization. This section is devoted to describing more fully two other applications of the composition algorithms.

5.1 Free-Form Deformations

Bézier [2] and Sederberg and Parry [13] have described a method of geometric modeling in which objects are deformed by polynomial maps from $\mathbb{R}^2$ to $\mathbb{R}^2$, or from $\mathbb{R}^3$ to $\mathbb{R}^3$. For instance, let $\mathbf{Q}$ be a Bézier curve in the plane, and $\mathbf{D}$ be a polynomial map from $\mathbb{R}^2$ to $\mathbb{R}^2$. The deformed curve $\tilde{\mathbf{Q}}$ is then defined to be $\mathbf{D} \circ \mathbf{Q}$, as depicted in Figure 1. Since deformations are accomplished via composition, either the product algorithm or the blossom algorithm can be used to compute directly the control polygon for a curve that has undergone a free-form deformation, given that the deformation is represented as a Bézier simplex.

As a specific example, consider the situation shown in Figure 8a for the case of a quadratic curve $\mathbf{Q}$, having control points $\mathbf{q}_{(0,2)}$, $\mathbf{q}_{(1,1)}$, and $\mathbf{q}_{(2,0)}$,[2] deformed by a quadratic deformation $\mathbf{D}$, having control points $\mathbf{d}_{(i_0,i_1,i_2)}$ and domain triangle uvw. The following steps are required by the blossom algorithm for composing a Bézier triangle ($N = 2$) and a Bézier curve ($n = 1$):

(1) Draw the image of $\mathbf{Q}$'s control polygon under the affine map that carries the triangle uvw into the triangle $\mathbf{d}_{(2,0,0)}\mathbf{d}_{(1,1,0)}\mathbf{d}_{(1,0,1)}$, labeling the image of $\mathbf{q}_{(0,2)}$, $\mathbf{q}_{(1,1)}$, and $\mathbf{q}_{(2,0)}$ by $\mathbf{A}_{(1,0,0)}(0)$, $\mathbf{A}_{(1,0,0)}(1)$, and $\mathbf{A}_{(1,0,0)}(2)$, respectively. Do the same for the triangles $\mathbf{d}_{(1,1,0)}\mathbf{d}_{(0,2,0)}\mathbf{d}_{(0,1,1)}$ and $\mathbf{d}_{(1,0,1)}\mathbf{d}_{(0,1,1)}\mathbf{d}_{(0,0,2)}$, labeling the images with $\mathbf{A}$'s subscripted with (0, 1, 0) and (0, 0, 1), respectively (the indices on the $\mathbf{A}$'s have been chosen to correspond to the blossom values used in Claim 4.3). The result of this step is nine points labeled as shown in Figure 8b.

[2] The Bézier simplex form of the curve is used here to emphasize the use of the blossom algorithm for simplexes.

(2) Find the image of **Q**'s control polygon under the affine map that carries the triangle uvw into the triangle $\mathbf{A}_{(1,0,0)}(0)\mathbf{A}_{(0,1,0)}(0)\ \mathbf{A}_{(0,0,1)}(0)$. Label the images of $\mathbf{q}_{(0,2)}$, $\mathbf{q}_{(1,1)}$, and $\mathbf{q}_{(2,0)}$ with $\mathbf{A}_{(0,0,0)}(0,\ 0)$, $\mathbf{A}_{(0,0,0)}(0,\ 1)$, and $\mathbf{A}_{(0,0,0)}(0,\ 2)$, respectively, as shown in Figure 8c. Do the same for triangles formed by **A**'s with arguments equal to 1 (as shown in Figure 8d) and 2. The result of this step is nine points $\mathbf{A}_{(0,0,0)}(j,p)$, with $j = 0,\ 1,\ 2$, and $p = 0,\ 1,\ 2$. Notice, however, that only six of the nine points are distinct, as shown in Figure 8e. This is due to the fact that the **A**'s are symmetric with respect to permutation of their arguments, a property that follows from their connection to blossom values.

(3) The rth control point $\tilde{\mathbf{q}}_{(r,4-r)}$ for the deformed curve $\tilde{\mathbf{Q}} = \mathbf{D} \circ \mathbf{Q}$ is now constructed by forming an affine combination of all points $\mathbf{A}_{(0,0,0)}(j,p)$ where $j + p = r$. Due to the large degree of symmetry in this case, most of the affine combinations collapse into simple assignment:

$$\begin{aligned}
\tilde{\mathbf{q}}_{(0,4)} &= \mathbf{A}_{(0,0,0)}(0,\ 0),\\
\tilde{\mathbf{q}}_{(1,3)} &= \mathbf{A}_{(0,0,0)}(0,\ 1),\\
\tilde{\mathbf{q}}_{(2,2)} &= \tfrac{1}{3}\mathbf{A}_{(0,0,0)}(0,\ 2) + \tfrac{2}{3}\mathbf{A}_{(0,0,0)}(1,\ 1),\\
\tilde{\mathbf{q}}_{(3,1)} &= \mathbf{A}_{(0,0,0)}(1,\ 2),\\
\tilde{\mathbf{q}}_{(4,0)} &= \mathbf{A}_{(0,0,0)}(2,\ 2),
\end{aligned}$$

as shown in Figure 8f.

The more interesting case for applications is the deformation of Bézier triangles by Bézier tetrahedrons. For completeness, a pseudocode statement of the product algorithm is given in Figure 9, specialized to the computation of the control net of a Bézier triangle **T** deformed by a tetrahedron **D**.

5.2 Geometric Continuity of Bézier Curves

The study of geometric continuity (a.k.a. visual continuity) is an active area of current research (see, e.g., [1], [6], and [9]), so we shall not belabor it here. Rather, the purpose of this section is to show how the algorithms for functional composition can be used in the solution of a problem in geometric continuity.

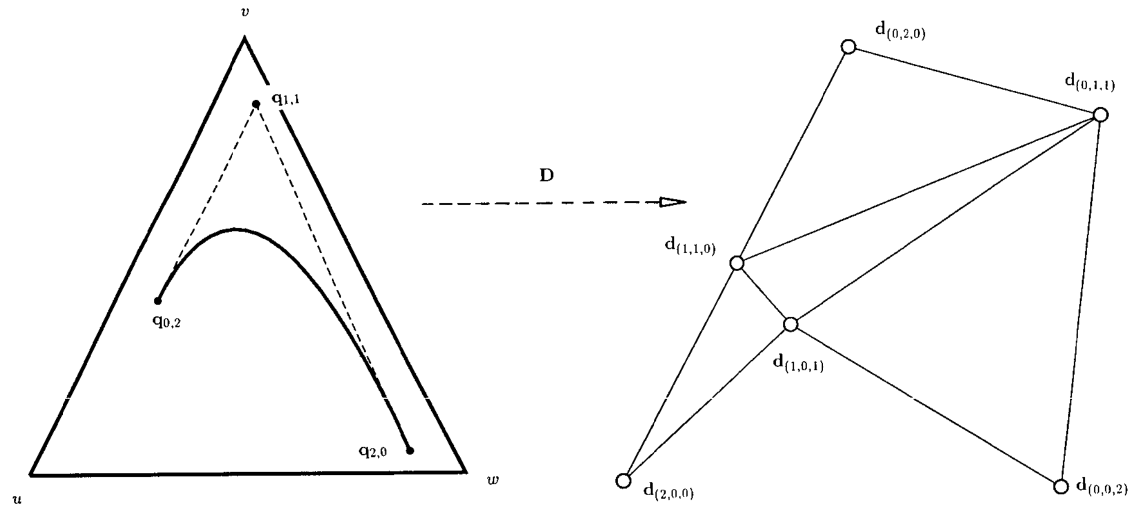
Before the problem can be accurately stated, some terminology and a few definitions are required. Let $\mathbf{Q}(u)$ and $\mathbf{P}(t)$ be two Bézier curves of degree m meeting at a common point such that $\mathbf{Q}(1) = \mathbf{P}(0)$. These curves are said to meet with *parametric continuity* of order k, denoted C^k, if their first k derivatives match at the common point; that is, if

$$\frac{d^i\mathbf{Q}(1)}{du^i} = \frac{d^i\mathbf{P}(0)}{dt^i}, \qquad i = 1, \ldots, k.$$

Parametric continuity can be generalized in the following way: Let $f\colon \mathbb{R} \mapsto \mathbb{R}$ be a polynomial function of degree k satisfying

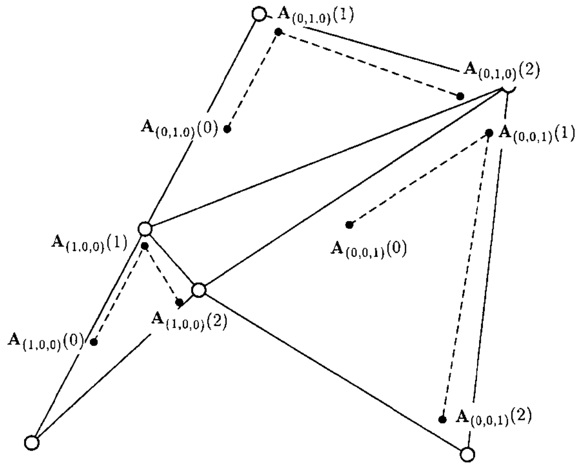
(i) $f(0) = 1$,
(ii) $f'(0) > 0$,

where a prime denotes differentiation. Such a function is called a *change of variables*. We say that **Q** and **P** as above meet with geometric continuity of order



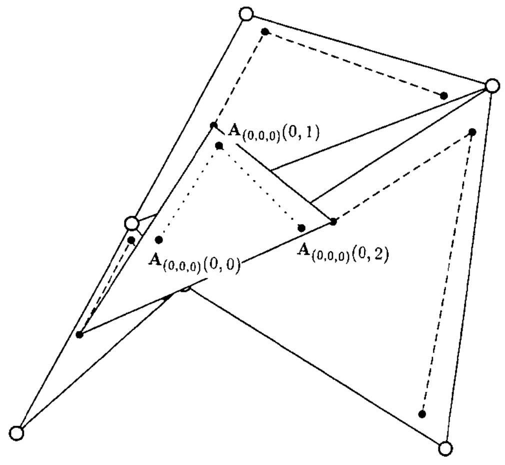
v
$q_{1,1}$
$q_{0,2}$
$q_{2,0}$
u
w
$\mathbf{D}$
$\mathbf{d}_{(0,2,0)}$
$\mathbf{d}_{(0,1,1)}$
$\mathbf{d}_{(1,1,0)}$
$\mathbf{d}_{(1,0,1)}$
$\mathbf{d}_{(2,0,0)}$
$\mathbf{d}_{(0,0,2)}$

(a)



$\mathbf{A}_{(0,1,0)}(1)$
$\mathbf{A}_{(0,1,0)}(2)$
$\mathbf{A}_{(0,1,0)}(0)$
$\mathbf{A}_{(0,0,1)}(1)$
$\mathbf{A}_{(1,0,0)}(1)$
$\mathbf{A}_{(0,0,1)}(0)$
$\mathbf{A}_{(1,0,0)}(2)$
$\mathbf{A}_{(1,0,0)}(0)$
$\mathbf{A}_{(0,0,1)}(2)$

(b)



$\mathbf{A}_{(0,0,0)}(0,1)$
$\mathbf{A}_{(0,0,0)}(0,0)$
$\mathbf{A}_{(0,0,0)}(0,2)$

(c)

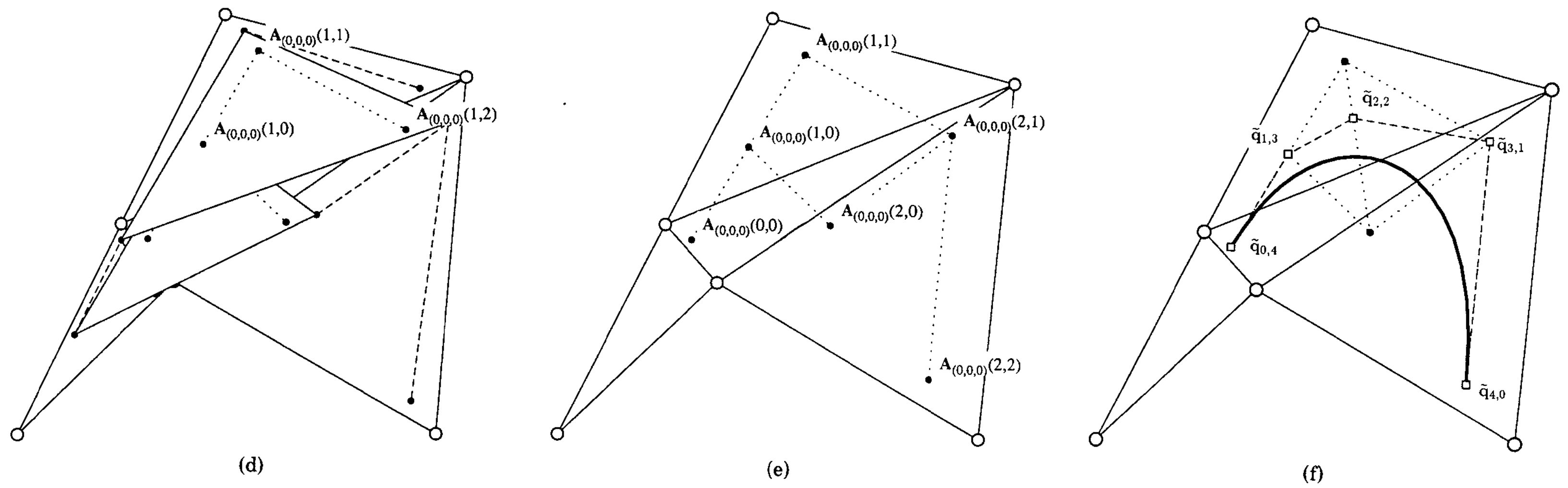


Fig. 8. Quadratic deformation of a quadratic curve.

```
Input: A control net $\{\mathbf{P}_{\vec{p}}\}_{|\vec{p}|=k}$ describing a Bézier triangle $\mathbf{T}$,
and a control net $\{\mathbf{d}_{\vec{i}}\}_{|\vec{i}|=m}$ describing a Bézier tetrahedron $\mathbf{D}$ (the deformation).

Output: A control net $\{\tilde{\mathbf{P}}_{\vec{r}}\}_{|\vec{r}|=mk}$ describing $\mathbf{D} \circ \mathbf{T}$ (the deformed triangle).

Comment: The multiindices $\vec{i}$ and $\vec{e}_\alpha$ have four components ($\vec{i}, \vec{e}_\alpha \in \mathbf{Z}^3_+$);
all other multiindices have three components ($\in \mathbf{Z}^2_+$).

Comment: $P^0_{\vec{p}}, ..., P^3_{\vec{p}}$ are the barycentric coordinates of $\mathbf{P}_{\vec{p}}$ relative to
the domain tetrahedron of $\mathbf{D}$.

for all $\vec{i}$ such that $|\vec{i}| = m$ do
    $\mathbf{d}^{[0]}_{\vec{i},(0,0,0)} \leftarrow \mathbf{d}_{\vec{i}}$
endforall

for $s \leftarrow 1$ to $m$ do
    for all $\vec{i}$ such that $|\vec{i}| = m - s$ do
        for all $\vec{r}$ such that $|\vec{r}| = ks$ do
            $\mathbf{d}^{[s]}_{\vec{i},\vec{r}} \leftarrow 0$
            for all $\vec{j}$ such that $|\vec{j}| = k(s-1)$ do
                $\mathbf{d}^{[s]}_{\vec{i},\vec{r}} \leftarrow \mathbf{d}^{[s]}_{\vec{i},\vec{r}} + \binom{k(s-1)}{\vec{j}}\binom{k}{\vec{r}-\vec{j}} \sum_{\alpha=0}^{3} P^{\alpha}_{\vec{r}-\vec{j}}\, \mathbf{d}^{[s-1]}_{\vec{i}+\vec{e}_\alpha,\vec{j}}$
            endforall
            $\mathbf{d}^{[s]}_{\vec{i},\vec{r}} \leftarrow \frac{\mathbf{d}^{[s]}_{\vec{i},\vec{r}}}{\binom{ks}{\vec{r}}}$
        endforall
    endforall
endforall
for all $\vec{r}$ such that $|\vec{r} = mk$ do
    $\tilde{\mathbf{P}}_{\vec{r}} \leftarrow \mathbf{d}^{[m]}_{(0,0,0,0),\vec{r}}$
endforall
```

Fig. 9. Product algorithm for free-form deformations.

k, denoted G^k, *with respect to* f if the reparameterized curve $\tilde{\mathbf{Q}} = \mathbf{Q} \circ f$ meets $\mathbf{P}$ with C^k continuity at the point $\tilde{\mathbf{Q}}(0) = \mathbf{Q}(1) = \mathbf{P}(0)$. Finally, we say simply that $\mathbf{Q}$ and $\mathbf{P}$ meet with G^k continuity if there exists a change of variables with respect to which they meet G^k continuously.

The problem of particular interest in this section may now be stated as follows:

> *Given:* A control polygon $\mathbf{V}_0, \ldots, \mathbf{V}_m$ defining a Bézier curve $\mathbf{Q}$ of degree m, and a set of Bernstein coefficients $C_0, \ldots, C_k$ defining a change of variables f of degree k (for f to satisfy properties (i) and (ii), C_0 must be 1, and C_1 must be greater than 1).
>
> *Find:* The control points $\mathbf{W}_0, \ldots, \mathbf{W}_m$ of a Bézier curve $\mathbf{P}$ of degree m so that $\mathbf{Q}$ and $\mathbf{P}$ meet with G^k continuity with respect to f at the point $\mathbf{Q}(1) = \mathbf{P}(0)$.

The case in which f is a linear polynomial has an elegant solution, due to Stärk [14], based on de Casteljau's algorithm (cf. [3]). Our solution to the general problem involves two steps:

(1) Compute the first $k + 1$ vertices $\tilde{\mathbf{V}}_0, \ldots, \tilde{\mathbf{V}}_k$ of the reparameterized curve $\tilde{\mathbf{Q}} = \mathbf{Q} \circ f$. These vertices are the only ones needed since we are only interested in the first k derivatives of $\tilde{\mathbf{Q}}$ at the point $\tilde{\mathbf{Q}}(0)$.

(2) Compute the first $k + 1$ control points $\mathbf{W}_0, \ldots, \mathbf{W}_k$ of $\mathbf{P}$ so that the first k derivatives of $\mathbf{P}$ match the first k derivatives of $\tilde{\mathbf{Q}}$ at $\mathbf{P}(0) = \tilde{\mathbf{Q}}(0)$.

Step (1) can be solved using a version of either the product algorithm or the blossom algorithm that computes only the first $k + 1$ points of the reparameterized curve (since $C_0 = 1$, $\tilde{\mathbf{V}}_0$ must be equal to $\mathbf{V}_m$; so only the k vertices $\tilde{\mathbf{V}}_1, \ldots, \tilde{\mathbf{V}}_k$ actually need to be computed).

The solution to step (2) requires more effort. Recall that, if $\mathbf{G}(u)$ is a Bézier curve of degree m with control points $\mathbf{g}_0, \ldots, \mathbf{g}_m$, then $d^i\mathbf{G}(0)/du^i$ can be written as in [12]:

$$\frac{d^i\mathbf{G}(0)}{du^i} = \binom{m}{i} i!\,\delta^i\mathbf{g}_0, \tag{5.1}$$

where the difference operator δ^i is defined recursively by

$$\delta^i\mathbf{g}_j = \begin{cases} \mathbf{g}_j & \text{if } i = 0, \\ \delta^{i-1}\mathbf{g}_{j+1} - \delta^{i-1}\mathbf{g}_j & \text{otherwise.} \end{cases} \tag{5.2}$$

The solution to step (2) requires that

$$\frac{d^i\tilde{\mathbf{Q}}(0)}{du^i} = \frac{d^i\mathbf{P}(0)}{dt^i}, \qquad i = 1, \ldots, k. \tag{5.3}$$

Into eq. (5.3) substitute the difference operator form of the derivatives from eq. (5.1) to obtain

$$\binom{mk}{i} i!\,\delta^i\tilde{\mathbf{V}}_0 = \binom{m}{i} i!\,\delta^i\mathbf{W}_0, \qquad i = 1, \ldots, k. \tag{5.4}$$

Thus, we seek control points $\mathbf{W}_0, \ldots, \mathbf{W}_k$, satisfying

$$\delta^i\mathbf{W}_0 = \frac{\binom{mk}{i}}{\binom{m}{i}}\,\delta^i\tilde{\mathbf{V}}_0, \qquad i = 1, \ldots, k.$$

Using the recursive definition of the difference operator, it can be verified (by induction) that

$$\delta^i\mathbf{W}_0 = \mathbf{W}_i - \sum_{j=1}^{i} \delta^{i-j}\mathbf{W}_{j-1}. \tag{5.5}$$

Substituting eq. (5.5) into eq. (5.4) and then solving for $\mathbf{W}_i$ result in

$$\mathbf{W}_i = \sum_{j=1}^{i} \delta^{i-j}\mathbf{W}_{j-1} + \frac{\binom{mk}{i}}{\binom{m}{i}}\,\delta^i\tilde{\mathbf{V}}_0. \tag{5.6}$$

Equation (5.6) is quite useful in that the right side of the equation involves only the unknown points $\mathbf{W}_j$, where $j < i$. Thus, the point $\mathbf{W}_i$ can be calculated once the points $\mathbf{W}_0, \ldots, \mathbf{W}_{i-1}$ are known. Since $\mathbf{W}_0$ must be equal to $\tilde{\mathbf{V}}_0$, $\mathbf{W}_1$ can be computed from eq. (5.6); then $\mathbf{W}_2$ can be computed, and so on, until all $k + 1$ points $\mathbf{W}_0, \ldots, \mathbf{W}_k$ are obtained.

Using this procedure, $\mathbf{Q}$ and $\mathbf{P}$ are guaranteed to meet with G^k continuity with respect to f, regardless of the placement of the remaining control points $\mathbf{W}_{k+1}, \ldots, \mathbf{W}_m$ of $\mathbf{P}$.

In the special case in which f is linear and step (1) is solved by the blossom algorithm, we find that the procedure above does not reduce to Stärk's method. It is not hard to show that the procedure above computes the same set of points as would be computed by Stärk's algorithm, but some points are computed more than once. This occurs because f is constrained to satisfy $f(0) = 1$. This specialized knowledge is built directly into the de Casteljau algorithm and hence into Stärk's algorithm. The composition algorithms, on the other hand, can assume no specialized knowledge of f's behavior, causing them to do more work in cases in which f is "special."

6. SUMMARY

Two algorithms for determining the control net of a Bézier simplex defined by the composition of two other Bézier simplexes have been derived. One of the algorithms (the product algorithm) is more efficient for machine implementation, whereas the other (the blossom algorithm) provides somewhat more geometric intuition.

These algorithms have been shown to have application to the following problems in CAGD: the evaluation, subdivision, and polynomial reparameterization of Bézier simplexes, the computation of the control net of a Bézier simplex that has undergone free-form deformation by another Bézier simplex, and the joining of Bézier curves with geometric continuity of arbitrary order.

Although this paper has dealt only with the composition of Bézier simplexes, similar algorithms can be constructed for the composition of Bézier tensor product forms, as well as for the mixed composition of tensor product forms and Bézier simplexes.

ACKNOWLEDGMENTS

I would like to thank the referees for their careful reading and thoughtful comments. This paper has benefited greatly from their diligence.

REFERENCES

1. BARSKY, B. A., AND DEROSE, T. D. Geometric continuity for parametric curves. Tech. Rep. UCB/CSD 84/205, Computer Science Div., Electrical Engineering and Computer Sciences Dept., Univ. of California, Berkeley, Oct. 1984.
2. BÉZIER, P. General distortion of an ensemble of biparametric surfaces. *Comput.-Aided Des. 10*, 2 (Mar. 1978), 116–120.
3. BOEHM, W., FARIN, G., AND KAHMANN, J. A survey of curve and surface methods in CAGD. *Comput.-Aided Geom. Des. 1*, 1 (July 1984), 1–60.
4. DE BOOR, C. B-form basics. In *Geometric Modeling: Algorithms and New Trends*, G. Farin, Ed. SIAM, Philadelphia, Pa., 1987, pp. 131–148.
5. DEROSE, T. D. Geometric continuity: A parametrization independent measure of continuity for computer aided geometric design. Ph.D. dissertation, Computer Sciences Div., Dept. of Electrical Engineering and Computer Sciences, Univ. of California, Berkeley, Aug. 1985. (Available as Tech. Rep. UCB/CSD 86/255, Computer Sciences Div., Dept. of Electrical Engineering and Computer Sciences, Univ. of California, Berkeley.)
6. DEROSE, T. D., AND BARSKY, B. A. An intuitive approach to geometric continuity for parametric curves and surfaces. In *Proceedings of Graphics Interface '85* (Montreal, May 27–31). 1985, pp. 343–351. Revised version published in *Computer-Generated Images—The State of the Art*, N. Magnenat-Thalmann and D. Thalmann, Eds. Springer-Verlag, New York, 1985, pp. 159–175.

Extended abstract in *Proceedings of the International Conference on Computational Geometry and Computer-Aided Design* (New Orleans, La., June 5–8). 1985, pp. 71–75.

7. FARIN, G. Triangular Bernstein–Bézier patches. *Comput-Aided Geom. Des. 3*, 2 (Aug. 1986), 83–127.
8. GOLDMAN, R. N. Using degenerate Bézier triangles and tetrahedra to subdivide Bézier curves. *Comput.-Aided Des. 14*, 6 (Nov. 1982), 307–311.
9. HERRON, G. Techniques for visual continuity. In *Geometric Modeling: Algorithms and New Trends*, G. Farin, Ed. SIAM, Philadelphia, Pa., 1987, pp. 163–174.
10. RAMSHAW, L. Blossoming: A connect-the-dots approach to splines. Res. Rep. 19, Systems Research Center, Digital Equipment Corp., Palo Alto, Calif., June 1987.
11. SABLONNIÉRE, P. Spline and Bézier polygons associated with a polynomial spline curve. *Comput.-Aided Des. 10*, 4 (1978), 257–261.
12. SCHWARTZ, A. J. Subdividing Bézier curves and surfaces. In *Geometric Modeling: Algorithms and New Trends*, G. Farin, Ed. SIAM, Philadelphia, Pa., 1987, pp. 55–66.
13. SEDERBERG, T. W., AND PARRY, S. R. Free-form deformation of solid geometric models. In *SIGGRAPH '86 Conference Proceedings* (Dallas, Tex., Aug. 18–22). ACM, New York, 1986, pp. 151–160.
14. STÄRK, E. Mehrfach differenzierbar Bézier–Kurven and Bézier Flächen. Diss., Technische Univ. Braunschweig, Braunschweig, West Germany, 1976.

Received September 1987; revised January 1988; accepted January 1988