Pruned Bézier Curves

Phillip J. Barry
Computer Science Department
University of Minnesota
4-192 EE/CSci Building, 200 Union St. SE
Minneapolis, Minnesota 55455

Tony D. DeRose
Department of Computer Science and Engineering, FR-35
University of Washington
Seattle, WA 98195

Ronald N. Goldman
Computer Science Department
University of Waterloo
Waterloo, Ontario
Canada N2L 3G1

Abstract

The de Casteljau algorithm for evaluating Bézier curves can be represented as a simple data-flow graph where nodes represent either control points or linear interpolation steps. By modifying this graph using an operation called "pruning," we generate new curve schemes called "pruned Bézier curves" that retain many properties of Bézier curves, but have smaller data-flow graphs, and hence can be computed using fewer linear interpolation steps. Many properties of pruned Bézier curves can be determined simply by inspecting the shape of the data-flow graph. In particular, we show that if the frontier of the graph does not oscillate (in a certain easily determined way), then the corresponding curve scheme is variation diminishing.

Keywords: Bézier curves, the de Casteljau algorithm, Pólya curves

1 Introduction

Bézier curves are used frequently in computer aided geometric design because of their many attractive geometric properties [6,9]. In this paper we introduce and examine a related class of curves which we call "pruned Bézier curves."

The Bézier curve with control points $P_0, ..., P_n$ is defined by

$$B(t) = \sum_{k=0}^{n} P_k b_k^n(t) \qquad t \in [0,1]$$

where $b_k^n(t) = \binom{n}{k} t^k (1-t)^{n-k}$ are the Bernstein basis functions of degree n. An elegant method for evaluating points on this curve is the de Casteljau algorithm [9]. To evaluate the curve at parameter value t, let

$$P_k^0(t) = P_k \quad k = 0, \ldots, n$$

$$P_k^r(t) = (1-t)P_k^{r-1}(t) + tP_{k+1}^{r-1}(t)$$
$$r = 1, \ldots, n; \quad k = 0, \ldots, n-r. \qquad (1)$$

Then $B(t) = P_0^n(t)$. The de Casteljau algorithm can be interpreted as a sequence of nested linear interpolations, and can, in fact, be used as the *definition* of the curve. That is, given a set of points $\{P_k\}$, the Bézier curve they define is the curve consisting of the points $P_0^n(t)$ generated by the algorithm.

The nested linear interpolation steps described by the de Casteljau algorithm can be summarized nicely in a triangular data-flow graph as in Figure 1 for the case of cubic curves ($n = 3$).

Circles in the figure denote linear interpolations, while squares denote constant values (the control points).

In this paper we will show that if certain linear interpolation steps are removed from the graph we still get interesting curves; these curves retain many of the properties of Bézier curves, but are simpler than Bézier curves in certain ways. We call these new curves "pruned Bézier curves" because they arise from "pruning away" parts of the graph associated with the de Casteljau algorithm.

The class of pruned Bézier curves is quite large, containing Bézier curves, a generalization of Ball's cubic

[0]This work was supported in part by the National Science Foundation under grant numbers DMC-8802949 and CCR-8957323, the Digital Equipment Corporation under the Faculty Program / Incentives for Excellence, and the Natural Science and Engineering Research Council of Canada, grant number OGP 0036825.

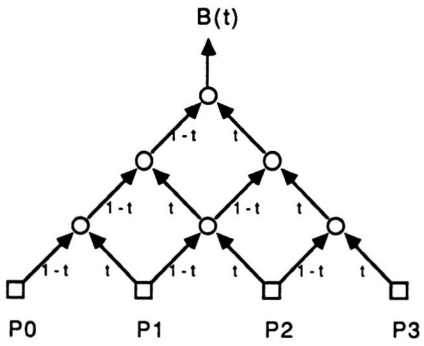

Figure 1: A diagram of the de Casteljau algorithm for the cubic case ($n = 3$).

curves [1], a curve scheme closely related to the power representation, and many new curve representations. Despite this generality, a substantial number of properties are shared by these schemes. Our study of this class provides simple and uniform proofs for analyzing these schemes. We show that many of the properties of a pruned Bézier curve follow from simple examination of its defining data-flow graph. For instance, an important but generally difficult to prove attribute is the variation diminishing property. We show that if the frontier (i.e., the lower boundary) of the graph does not oscillate in a certain easily determined way, then the curve scheme is variation diminishing.

The purpose of this paper, therefore, is to introduce and discuss pruned Bézier curves. Our motivation is theoretical rather than practical. Our goal is to continue the work done by two of the authors in [4] in examining interesting generalizations of Bézier curves; we hope that the study of these generalizations serves to unify, simplify, and provide insight into the theory of curves in computer graphics and related fields.

In Section 2, we define more rigorously the rules used to prune the de Casteljau graph. We call a pruned de Casteljau graph a "bush," a term motivated by the shape of the frontier of the graph. In Section 3, we show that pruned Bézier curves share many of the properties of Bézier curves. In Section 4, we discuss a few conversion algorithms for pruned Bézier curves. Section 5 contains some specializations and possible generalizations. Finally, in Section 6 we summarize our work.

2 Pruned Bézier Curves

Examine in more detail the graph of the de Casteljau algorithm shown in Figure 1. The edges emanating from each of the nodes refer to the left and right parents, and the edges incident upon a node emanate from the left and right children. The *root* of the graph is the node with no parents; the *leaves* of the graph are the nodes

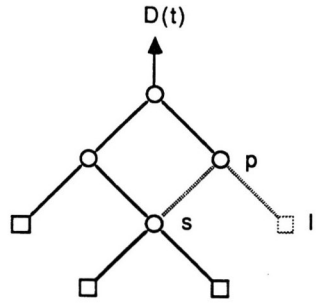

Figure 2: The leaf ℓ is pruned by removing its parent p and the two downward edges out of p.

with no children; the *internal nodes* are the nodes with exactly two children. Two nodes are said to be *siblings* if they have a common parent. The *depth* of the graph is defined to be the number of edges traversed in the longest path from the root to a leaf. Control points $P_0, \ldots, P_n$ are assigned left-to-right to the leaves of the graph and are successively blended together using linear interpolation until the point on the Bézier curve emerges from the root.

We want to modify this graph in such a way that we still obtain a degree n curve which possesses most of the features that make Bézier curves so popular. At the same time we wish to make the graph less complex by pruning away a number of the linear interpolation steps. The graphs resulting from careful pruning lead to curve schemes that share many of the properties of Bézier curves.

Pruning is an operation that reduces the number of internal nodes (and hence the number of linear interpolations) without altering the depth of the graph or the number of leaves. To make these ideas more precise, we say that a leaf ℓ is *eligible for pruning* if:

(i) ℓ has exactly one parent, and

(ii) ℓ has a sibling which has exactly two parents.

An eligible leaf ℓ is pruned as follows. Let p denote the parent of ℓ, and let s denote ℓ's sibling (see Figure 2). To prune ℓ, remove from the graph the node ℓ and the edges $\ell \rightarrow p$ and $s \rightarrow p$.

A graph obtained by applying zero or more pruning operations to a Casteljau graph is called a *bush*. The rules for eligibility ensure that bushes generated from a de Casteljau graph of depth n have the following attributes:

(a) each bush has exactly $n + 1$ leaves.

(b) each bush has at least one leaf with the property that a path from that leaf to the root traverses exactly n edges.

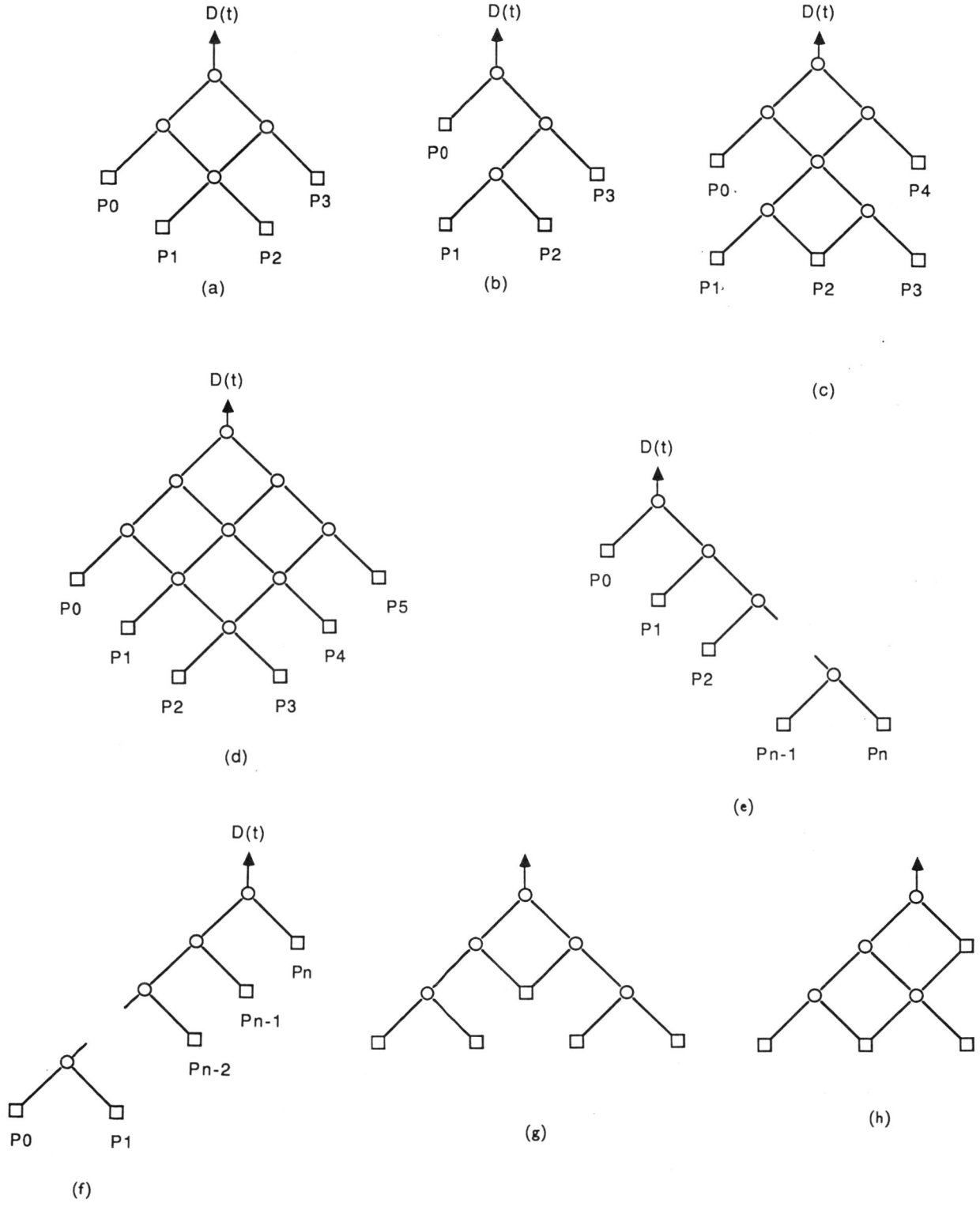

Figure 3: Valid and invalid bushes: figures (a) – (f) are valid bushes, (g) and (h) are not.

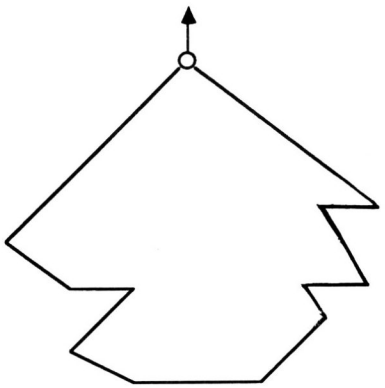

Figure 4: A schematic view of the frontier of an arbitrary bush.

Some valid and invalid examples of bushes are given in Figure 3. The term "bush" is motivated by the shape of the frontier of the graph. Imagine tracing out the frontier in left-to-right order (see Figure 4). Intuitively, the trace initially follows a downward path that may oscillate in the horizontal direction. At some point in the traversal, however, the frontier can switch to follow an upward path. Thus, there may be any number of turning points in left-right motion, but there can be at most one turning point in up-down motion.

Each of the nodes of a bush can be denoted by a pair of integers (d, i), where d is the depth of the node, and i is the number of edges oriented top left to bottom right that must be traversed in a path from the root to the node. For example, the leaves of the bush in Figure 3(a) are denoted by $(2,0)$, $(3,1)$, $(3,2)$, and $(2,2)$.

Once we have done the pruning, if we assign control points $P_0, \ldots, P_n$ at the leaves by traversing the frontier left to right, and then perform the sequence of linear interpolations described by the bush, we obtain a point at the root. We can then define a curve $D(t)$ to have this value at parameter value $t \in [0,1]$. We define the class of "pruned Bézier curves" to consist of all curves which can be obtained from bushes. Attributes (a) and (b) above insure that we get a degree n curve with exactly $n+1$ control points. Figures 5(a), 5(b), and 5(c) show curves described by the bushes of Figures 3(a), 3(b), and 3(d), respectively.

3 Properties of Pruned Bézier Curves

Bézier curves have many important properties. Among the more important ones are affine invariance, the convex hull property, interpolation of the endpoints, a simple explicit formula, non-degeneracy (i.e., the curve does not collapse to a point unless all its control points are located at a common point), the variation diminishing property, simple differentiation formulas, and simple algorithms for evaluation, degree elevation, and subdi-

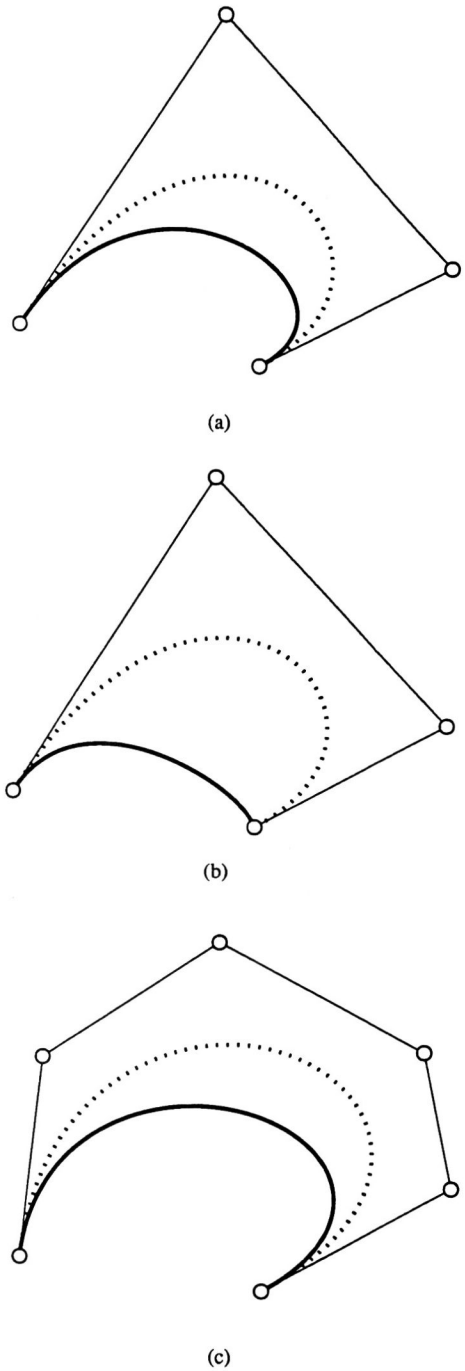

Figure 5: Pruned Bézier curves: the dotted curves are Bézier curves for comparison purposes. The solid curve of Figure (a) uses the bush of Figure 3(a), Figure (b) uses the bush of Figure 3(b), and Figure (c) uses the bush of Figure 3(d).

vision. In this section we will investigate properties of pruned Bézier curves.

To begin, notice that any pruned Bézier curve of degree n can be written as

$$D(t) = \sum_{k=0}^{n} P_k d_k^n(t) \qquad t \in [0,1]$$

where the P_k are the control points and the $d_k^n(t)$ are "blending functions." The function $d_k^n(t)$ can be computed by multiplying the labels on each path from the leaf associated with P_k to the root, and then summing over all paths. Each time a left edge is traversed a factor of $1-t$ enters the product, and each time a right edge is traversed a factor of t enters. Since for fixed k all the products are the same, we can simply multiply this product by the number of paths. For the bush of Figure 3(c), for example, each of the two paths to P_3 encounters one left edge and three right ones; thus, $d_3^4(t) = 2t^3(1-t)$. Many properties of the curve can be proved by examining properties of the functions $d_k^n(t)$.

We now list and discuss the properties of pruned Bézier curves:

Affine Invariance: Pruned Bézier curves are affine invariant, depending only on the relative geometry of their control points and not on any external coordinate system, because the computation of each node in the bush involves only affine combinations. Since a curve scheme is affine invariant if and only if its blending functions sum to one, we deduce that $\sum_k d_k^n(t) = 1$.

Convex hull: Since $t \in [0,1]$, computing a point at any node consists of taking a convex combination of the points at its children. Thus a point at any node is a convex combination of the points at the leaves of the subgraph rooted at that node. In particular, the curve will always be a convex combination of its control points.

Interpolates endpoints: The curve always interpolates the first control point P_0 at $t = 0$ and the last control point, P_n, at $t = 1$. This interpolation occurs because to reach the root from any point other than P_0 we must traverse an edge labelled "t," and to reach the root from any point other than P_n we must traverse an edge labelled "$1-t$."

Explicit formula: From the bush we can find explicit formulas for the functions $d_k^n(t)$. Although the formulas will vary from bush to bush, each $d_k^n(t)$ will always equal a constant times a power of t times a power of $(1-t)$. More specifically, a blending function $d_k^n(t)$ associated with a leaf (m_k, i_k) will be of the form

$$d_k^n(t) = a_k t^{i_k}(1-t)^{m_k - i_k} = \frac{a_k}{\binom{m_k}{i_k}} b_{i_k}^{m_k}(t),$$

where a_k is the number of distinct paths from the root to the leaf (m_k, i_k).

Evaluation algorithm: Pruned Bézier curves have a recursive evaluation algorithm. Indeed, this is how they are defined. The algorithm for a pruned Bézier curve cannot be written down in equation form as compactly as the de Casteljau algorithm, but the evaluation algorithm can be concisely programmed. Since bushes typically have fewer internal nodes than de Casteljau graphs, pruned Bézier curves can be computed using fewer linear interpolation steps than required for ordinary Bézier curves.

The de Casteljau algorithm can be implemented in hardware to allow very rapid evaluation of a large number of Bézier curves [8]. The basic idea is to use a different processor to do the evaluations which occur at each interior node, running them in parallel pipe-lined fashion to achieve high throughput. A similar technique will work for pruned Bézier curves. Further, since there are fewer interior nodes for a pruned Bézier curve, hardware implementation of the evaluation algorithm for these curves can be less costly than the corresponding implementation for Bézier curves.

Other evaluation algorithms for Bézier curves, such as nested multiplication [13], can also be applied to pruned Bézier curves, although nowhere near as neatly.

Differentiation: The major practical drawback of pruned Bézier curves is that in general they do not share the derivative properties of Bézier curves. Only certain pruned Bézier curves share some of the properties. For example, Bézier curves have the property that the first derivative at $t = 0$ depends only on the first two control points. The reason for this dependence is that each basis function $b_k^n(t) = \binom{n}{k} t^k (1-t)^{n-k}$ is divisible by t^2 if $k \geq 2$. Only if we prune in such a manner that t^2 divides $d_k^n(t)$ for $k \geq 2$ will $D'(0)$ depend only on the first two control points. Analogous results hold for other derivatives and at the other endpoint.

Degree Lowering: A degree n Bézier curve may actually be of degree less than n if its control points are in certain configurations. It is not easy to tell by inspection whether or not a Bézier curve does have lower degree or not. However, there is a result for pruned Bézier curves which states a simple sufficient condition for when a pruned Bézier curve is of lower degree:

Theorem 1 *Let $D(t)$ be a pruned Bézier curve defined by a graph of depth n, and suppose that all the control points located at the leaves of deepest level coincide. Then the curve has degree less than n.*

Proof. Note that if all the points on any level are constant and coincide, the points located at the nodes on the next highest level are constant. Therefore the degree of the curve is at most the deepest level whose points do not coincide. In particular, a depth n curve, all of whose level n control points coincide, has degree less than n. □

Nondegeneracy: A pruned Bézier curve cannot collapse to a single point unless all its control points coincide. This is because the blending functions described

by bushes are linearly independent, as the next theorem shows.

Theorem 2 *The blending functions $d_0^n(t), ..., d_n^n(t)$ defined by a bush β of depth n form a basis for the polynomials of degree n.*

Proof.[1] Since there are the correct number of functions, we must simply show that the functions are independent. We use induction on the number of nodes pruned. If no nodes are pruned the blending functions are the Bernstein basis functions. Now suppose we prune the k^{th} leaf (m_k, i_k) from a bush β to obtain a new bush β'. Let the basis functions for β be denoted by $g_j(t)$, and the blending functions for β' by $h_j(t)$. Since only the k^{th} leaf is pruned, the explicit formulas for $h_j(t)$ and $g_j(t)$ differ by at most a positive constant unless $j = k$. Now in pruning the k^{th} leaf we removed two edges, one leading to the k^{th} leaf, and one leading to either the $k-1^{st}$ or the $k+1^{st}$ leaf (but not both). Suppose it was "pruning from the right;" that is, suppose an edge leading to the $k-1^{st}$ leaf is removed. Then

$$h_k(t) = c_1 t^{i_k - 1}(1-t)^{m_k - i_k}$$

$$= c_1 t^{i_k}(1-t)^{m_k - i_k} + c_1 t^{i_k - 1}(1-t)^{m_k - i_k + 1}$$

where c_1 is a positive constant. Notice that the first summand in the right hand side is a positive multiple of $g_k(t)$. The second summand is a nonzero multiple of the product of the labels on any path in β from the root to the node $(m_k, i_k - 1)$ (which is in β since it is the sibling of the pruned node). Now suppose we ran the recursive evaluation algorithm for β with all control points associated with leaves which are descendents of the node $(m_k, i_k - 1)$ given the value 1, and all other control points set to 0. Then, because only affine combinations are used, the value at $(m_k, i_k - 1) = 1$. Thus the resulting curve is both a nonnegative linear combination of its basis functions and a positive multiple of $t^{i_k - 1}(1-t)^{m_k - i_k + 1}$; these two expressions must be equal. Therefore the matrix M such that

$$(h_0(t) \dots h_n(t)) = (g_0(t) \dots g_n(t))M$$

has positive entries on the diagonal, and zeros elsewhere except in one column. Such a matrix is non-singular, and the result follows. □

Variation Diminishing: Bézier curves have important shape preserving properties in that they mimic the shape of the control polygon (the piecewise linear interpolant to the control points). One of these shape preserving properties is the variation diminishing property. A curve scheme is said to be variation diminishing if any line (or plane if the curve is a space curve) intersects the curve no more often than it intersects the control polygon.

Pruned Bézier curves are not always variation diminishing. (Construct, for example, a curve using the bush

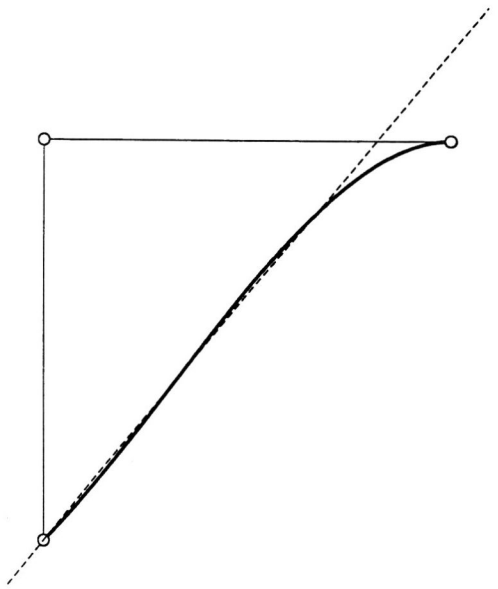

Figure 6: Variation diminishing counterexample for bushes: the dotted line intersects the curve three times and the control polygon twice.

from Figure 3(b) and the control points $P_0 = (1,1), P_1 = (0,1), P_2 = P_3 = (0,0)$. Then there exist lines such as $y = \frac{11}{9}x$ which intersect the curve three times, but the control polygon only twice, as shown in Figure 6.) However, by strengthening the eligibility rules for pruning, a subclass of bushes known as *hedges* can be generated which always yield curves that are variation diminishing. In particular, hedges result if a leaf ℓ is considered eligible for pruning when, in addition to conditions (i) and (ii) above, we require:

(iii) ℓ's sibling is also a leaf.

Once again, the term "hedge" is motivated by the shape of the frontier of the graph. The frontier of a hedge is characterized by no turning points in left-right motion, and at most one turning point in up-down motion, as shown schematically in Figure 7. The bushes of Figures 3(a), 3(d), 3(e), and 3(f) are therefore hedges.

That the restriction to hedges is sufficient for the variation diminishing property is demonstrated by the next theorem.

Theorem 3 *Hedges yield pruned Bézier curves that are variation diminishing.*

Proof. Let M be the matrix which transforms the basis functions $b_k^n(t) = \binom{n}{k} t^k (1-t)^{n-k}$ for Bézier curves into the functions $d_k^n(t)$ for a pruned Bézier curve associated with a hedge. That is, let M be such that

$$(d_0^n(t) \cdots d_n^n(t)) = (b_0^n(t) \cdots b_n^n(t))M.$$

To prove the result, it suffices to show that M is strictly totally positive [10]. (A matrix is strictly totally positive

[1] We thank a referee for suggesting this proof.

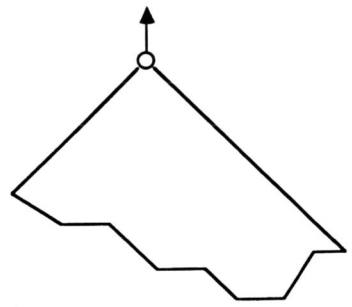

Figure 7: A schematic view of the frontier of an arbitrary hedge.

if all its minors are nonnegative, and at least one minor of each order is positive).

We will do this in the following way. Let $H[0]$ denote the de Casteljau graph which we pruned (using the strengthened rules) to obtain the curve in question. Let $H[1]$ be the hedge after the first node is pruned, $H[2]$ the hedge after the first two nodes are pruned, and, in general, $H[i]$ the hedge after the first i nodes are pruned. If q nodes in all are pruned, then the final hedge is $H[q]$.

Let M_i be the matrix transforming the blending functions for $H[i-1]$ to the blending functions for $H[i]$. Then $M = M_1 \cdots M_q$. Since a product of totally positive matrices is totally positive [12], it suffices to show that each matrix M_i is totally positive.

Now let $g_j(t)$ be the blending functions from $H[i-1]$ and let $h_j(t)$ be the blending functions from $H[i]$, $j = 0, \ldots, n$. Suppose in going from $H[i-1]$ to $H[i]$ we remove the k^{th} leaf of $H[i-1]$. Since all other leaves remain, $g_j(t) = h_j(t)$ if $j \neq k-1, k, k+1$ because we do not pass through any of the removed edges to get from the root to the corresponding leaves. Now in pruning the k^{th} leaf, we removed two edges, one leading to the k^{th} leaf, and one leading to either the $k-1^{st}$ or the $k+1^{st}$ leaf (but not both). Suppose we pruned from the right. The paths leading to the $k+1^{st}$ leaves in $H[i]$ and $H[i-1]$ are then the same and $h_{k+1}(t) = g_{k+1}(t)$.

Next consider $h_{k-1}(t)$. There are fewer paths leading to the $k-1^{st}$ leaf in $H[i]$ than there are leading to the same node in $H[i-1]$, but the product of the labels along any of these paths is the same, so $h_{k-1}(t) = c_0 g_{k-1}(t)$ where $0 < c_0 < 1$.

Now examine $h_k(t)$. From the explicit formula, there exists a positive integer c_1 and nonnegative integers r, s such that $h_k(t) = c_1 t^r (1-t)^s$. Further, there exists a positive integer c_2 such that $g_{k-1}(t) = c_2 t^r (1-t)^{s+1}$; also $g_k(t) = c_1 t^{r+1}(1-t)^s$ since the only way to get to the k^{th} leaf of $H[i-1]$ is through the k^{th} leaf of $H[i]$. This implies that

$$h_k(t) = \frac{c_1}{c_2} g_{k-1}(t) + g_k(t).$$

Therefore the matrix M_i has positive entries on the diagonal, one positive entry on the superdiagonal, and zeros elsewhere. Such a matrix is totally positive.

The analysis is similar for pruning from the left where an edge leading to the $k+1^{st}$ node in $H[i-1]$ is removed. Therefore the result follows. □

4 Conversion Algorithms

Of the many possible conversions, we will restrict ourselves to two: pruned Bézier to Bézier conversion, and subdivision.

Conversion from pruned Bézier form to Bézier form: It is possible to represent every pruned Bézier curve in standard Bézier form. In fact, the proof of Theorem 3 provides an algorithm for the construction of the matrix M that transforms the Bernstein basis into the basis defined by a hedge. This same matrix transforms the control points $P_0, \ldots, P_n$ for a pruned Bézier curve (defined by a hedge) into Bézier control points $V_0, \ldots, V_n$ according to:

$$(V_0 \cdots V_n)^T = M(P_0 \cdots P_n)^T,$$

where superscript T denotes matrix transpose. The factorization of M into q sparse matrices provides the basis for an iterative algorithm for the computation of the Bézier polygon. If we set $(P_0^{[0]} \cdots P_n^{[0]}) = (P_0 \cdots P_n)$ and compute $(P_0^{[i+1]} \cdots P_n^{[i+1]})$ from $(P_0^{[i]} \cdots P_n^{[i]})$ according to

$$(P_0^{[i+1]} \cdots P_n^{[i+1]})^T = M_{q-i}(P_0^{[i]} \cdots P_n^{[i]})^T,$$

then $(V_0 \cdots V_n) = (P_0^{[q]} \cdots P_n^{[q]})$. The structure of M_{q-i} implies that there exists an integer k and constants a, b summing to one, such that $P_j^{[i+1]} = P_j^{[i]}$ if $j \neq k$, and

$$P_k^{[i+1]} = \begin{cases} aP_{k-1}^{[i]} + bP_k^{[i]} & \text{if } H[q-i-1] \text{ is pruned from the left} \\ aP_k^{[i]} + bP_{k+1}^{i} & \text{if } H[q-i-1] \text{ is pruned from the right.} \end{cases}$$

Geometrically, the polygon $(P_0^{[i+1]} \cdots P_n^{[i+1]})$ is obtained from $(P_0^{[i]} \cdots P_n^{[i]})$ by "corner cutting", as shown in Figure 8 for the case of pruning from the left.

Unfortunately, Theorem 3, and hence the corner cutting algorithm, is only appropriate for hedges, so it is necessary to develop a method for computing the Bézier representation of an arbitrary pruned Bézier curve. The method we now present is based on a general explicit expression for the components of the matrix M that transforms the Bernstein basis into an arbitrary pruned Bézier basis. The key ingredient in the construction is provided by the next lemma.

Lemma 1 *Let $m \leq n$, and let $0 \leq i \leq m$. Then*

$$b_i^m(t) = \sum_{j=0}^{n} b_j^n(t) a_{ji}$$

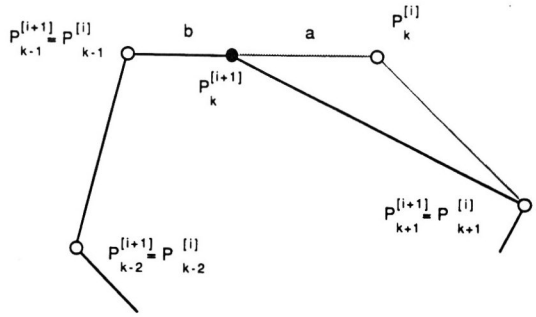

Figure 8: Corner cutting from the left.

where

$$a_{ji} = \begin{cases} \frac{\binom{n-m}{j-i}\binom{m}{i}}{\binom{n}{j}} & \text{if } j = i, \ldots, n-m+i \\ 0 & \text{otherwise.} \end{cases}$$

Proof. The proof requires nothing more than manipulation of the explicit formula for Bernstein polynomials together with the fact that Bernstein polynomials form a partition of unity. The later fact implies that $b_i^m(t) = b_i^m(t)\sum_{j=0}^{n-m} b_j^{n-m}(t)$. Explicit manipulation reveals that $b_i^m(t)b_j^{n-m}(t) = \frac{\binom{n-m}{j}\binom{m}{i}}{\binom{n}{i+j}} b_{i+j}^n(t)$. Thus,

$$b_i^m(t) = \sum_{j=0}^{n-m} \frac{\binom{n-m}{j}\binom{m}{i}}{\binom{n}{i+j}} b_{i+j}^n(t).$$

The final result is obtained by re-indexing the summation so that j ranges from i to $n-m+i$ rather than from 0 to $n-m$. □

The entries of M are essentially the coefficients found in Lemma 1. To establish the precise relationship, let $d_0^n(t), \ldots, d_n^n(t)$ be the basis functions described by an arbitrary bush of depth n having leaves denoted by $(m_0, i_0), (m_1, i_1), \ldots, (m_n, i_n)$. From the explicit formula, the k^{th} function $d_k^n(t)$ is a multiple of $b_{i_k}^{m_k}(t)$. That is, there exits a constant f_k^n such that $d_k^n(t) = f_k^n b_{i_k}^{m_k}(t)$. Using Lemma 1, there exist constants a_{jk} such that

$$d_k^n(t) = f_k^n \sum_{j=0}^{n} b_j^n(t) a_{jk}.$$

Therefore, the matrix M that transforms $b_0^n(t), \ldots, b_n^n(t)$ into $d_0^n(t), \ldots, d_n^n(t)$, and hence pruned Bézier control points into Bézier control points, has entries M_{jk} given by

$$M_{jk} = f_k^n a_{jk}. \tag{2}$$

Subdivision: Subdivision, the computation of a control polygon to describe a part of a curve, is one of the most useful techniques for rendering and analyzing curves. It is well known that Bézier curves can be subdivided using de Casteljau's algorithm [9]. In particular, the points $P_0^0(t), P_0^1(t), \ldots, P_0^n(t)$ computed in Equation 1 reproduce the portion of the Bézier curve $B(t)$ traced out on the interval $[0, t]$. One possible way to subdivide a pruned Bézier curve is to convert it to Bézier form, subdivide the Bézier representation, then convert back to pruned Bézier form. However, this does not take advantage of the special properties of pruned Bézier curves; also, it can involve a large number of computations.

Another possibility does exist for curves arising from hedges. Pruned Bézier curves arising from hedges can easily be represented as Pólya curves [5]. Therefore, Polya curve algorithms [2,5], including the Pòlya subdivision techniques, can be applied to these curves. How practical these algorithms are is an open question. We will not discuss these techniques further here; however, we will describe how to represent any pruned Bézier curve arising from a hedge as a Pólya curve.

Pólya curves are defined by $2n$ parameters $\alpha_0, \ldots, \alpha_{n-1}$, $\beta_0, \ldots, \beta_{n-1}$, and $n+1$ control points. Let r and s be the respective smallest and largest integers such that $m_j = n$ for $r \leq j \leq s$. A curve from a hedge with the property that $m_j = m_{j+1} \Rightarrow m_j = n$ can be represented as a Pólya curve with $\beta_j = \infty$ $j \geq r$, $\alpha_j = \infty$ $j \geq s$, all other $\alpha_j = 0$, $\beta_j = 1$, and the *same* control points as the pruned Bézier curve.

There also exists a homogeneous variant of Pólya curves where the α_i are replaced by pairs (α_i, w_i^α) and the β_i by (β_i, w_i^β). Any curve arising from a hedge can be represented by such a Pólya curve with (i) $(\alpha_j, w_j^\alpha) = (-1, 0)$ if $m_j = m_{j+1} + 1$, (ii) $(\alpha_j, w_j^\alpha) = (0, 1)$ else, (iii) $(\beta_{n-j-1}, w_{n-j-1}^\beta) = (1, 0)$ if $m_j = m_{j+1} - 1$, (iv) $(\beta_{n-j-1}, w_{n-j-1}^\beta) = (1, 1)$ else, and (v) the Pólya control points equal to $c_k P_k$ for some constants c_k.

5 Specializations and Generalizations

The properties discussed in Section 3 hold for any pruned Bézier curve. The one exception to this is the variation diminishing property which holds for hedges, but not in general. Other special cases have additional features. In this section we mention a couple noteworthy pruned Bézier curve schemes. It is also possible to generalize pruned Bézier curves in several ways. A few of these extensions are mentioned in this section as well.

Ball's basis: In [1] Ball used the following cubic basis in the development of a software package for computer aided design:

$$d_0^3(t) = (1-t)^2$$
$$d_1^3(t) = 2t(1-t)^2$$
$$d_2^3(t) = 2(1-t)t^2$$
$$d_3^3(t) = t^2$$

The bush for this basis is shown in Figure 3(a).

Some properties of curves generated with this basis are (1) if $P_1 = P_2$ then the curve is actually a quadratic,

(2) the derivative at $t = 0$ is $2(P_1 - P_0)$, and at $t = 1$ is $2(P_3 - P_2)$, (3) the curves are variation diminishing.

Ball's basis can be generalized to arbitrary degree n by pruning all nodes in the subgraphs rooted at the nodes $(\lceil \frac{n+3}{2} \rceil, 0)$ and $(\lceil \frac{n+3}{2} \rceil, \lceil \frac{n+3}{2} \rceil)$. The resulting hedge for $n = 5$ is shown in Figure 3(d), and a curve of this type is shown in Figure 5(c). These curves have recently been studied in [11,14], although not in the context of pruned Bézier curves. In particular, Goodman and Said show that these curves are variation diminishing and that they can readily be converted to Bézier form. (These results follow easily when these curves are viewed as being defined by hedges.) Furthermore, the k^{th} derivatives, for $k = 0, \ldots, \lfloor \frac{n}{2} \rfloor$, at $t = 0$ depends only on the vertices $P_0, ..., P_k$. An analogous result holds at $t = 1$. This derivative behavior makes it easy to join curve segments with either parametric or geometric continuity. For instance, cubic curves of this type can easily be connected with G^1 continuity and quintic curves can easily be connected with G^2 continuity.

Poor Man's Basis: Consider the curve with basis functions $d_k^n(t) = t^k(1-t)$ for $k < n$ and $d_n^n(t) = t^n$. This is called the "Poor Man's Curve" since its bush (actually a hedge) has a minimal number of nodes for a degree n pruned Bézier curve. Alternatively, one can let $d_0^n(t) = (1-t)^n$ and $d_k^n(t) = t(1-t)^{n-k}$ for $k > 0$ and also get a minimal node hedge. Diagrams of these hedges are given in Figures 3(e) and 3(f). These curves are variation diminishing since they are defined by hedges, and they may be evaluated using a nested multiplication algorithm similar to Horner's method.

Other less regular minimal bushes, such as the one shown in Figure 3(b), also exist.

We now list some possible generalizations:

Surfaces: Both Bézier tensor product and triangular patch surfaces have evaluation algorithms which generalize the de Casteljau algorithm for Bézier curves. By modifying these algorithms we can get pruned Bézier surfaces. To some extent the properties of such surfaces will be similar to the properties of Bézier surfaces.

Other Triangles: There are other curve schemes which have evaluation algorithms which can be represented by triangular data flow graphs. The graphs for these curve schemes differ from de Casteljau graphs in that the labels on the edges differ. B-spline segments, Lagrange curves, and Pólya curves, as well as many other types of curves, can be represented in this way [4]. We can prune these graphs as well and generate new schemes in this manner. In fact, most of the results in this paper hold more generally for Pólya curves since the main properties of Bézier curves that were exploited in our proofs (namely, a two term recurrence formula, a simple explicit formula, and a two term degree raising formula) are also possessed by Pólya curves.

Other Modifications: We have applied one operation — pruning — to de Casteljau graphs and obtained new curve schemes. There are other operations one could apply to develop new curve schemes, for example hooking graphs together. Depending on how we hook the graphs together we could get piecewise polynomial curves (hooking graphs together side by side), or curves similar to those obtained by Catmull and Rom in [7] (by attaching to the leaf of one graph to the root of another)(see [3]).

We also need not restrict our attention to triangles. There are other curve schemes which possess evaluation algorithms which can be diagrammed in different ways. For example, Clenshaw's algorithm for orthogonal polynomials can be diagrammed as a rectangular dataflow graph. We could modify these diagrams as well to obtain new curve schemes.

6 Concluding Remarks

Given a curve scheme, there are various ways to generalize it to obtain interesting new curve schemes. In this paper we began with Bézier curves and pruned the dataflow graph of the associated de Casteljau algorithm. The resulting curves do not retain all the properties possessed by Bézier curves; however, they do retain many features, and these properties can often be derived by simple arguments referring to the pruned de Casteljau graph. These schemes also contain as specific cases not only Bézier curves, but also a generalization of Ball's basis. For these reasons, they simplify, unify, and provide insight into the theory of curves in computer graphics and geometric modeling.

References:

[1] Ball, A.A., Consurf I: Introduction of the Conic Lofting Tile, *Computer-Aided Design*, **6**, 1974, 243-249.

[2] Barry, P.J., Urn Models, Recursive Curve Schemes, and Computer Aided Geometric Design, Ph.D. dissertation, Dept. of Math., Univ. of Utah, Salt Lake City, Utah, 1987.

[3] Barry, P.J. and Goldman, R.N., A Recursive Evaluation Algorithm for a Class of Catmull-Rom Splines, *Computer Graphics — Proceedings of SIGGRAPH '88*, **22**, 1988, 199-204.

[4] Barry, P.J. and Goldman, R.N., Recursive Polynomial Curve Schemes and Computer Aided Geometric Design, *Constructive Approximation*, **6**, 1990, 65-96.

[5] Barry, P.J. and Goldman, R.N., Shape Parameter Deletion for Polya Curves, submitted for publication.

[6] Bartels, R.H., Beatty, J.C., and Barsky, B.A., *An Introduction to Splines for Use in Computer*

Graphics and Geometric Modeling, Morgan Kaufman, Los Altos, California, 1987.

[7] Catmull, E. and Rom, R., A Class of Local Interpolating Splines, in *Computer Aided Geometric Design*, R.E. Barnhill and R.F. Riesenfeld (eds.), Academic Press, New York, 1974, 317-326.

[8] Tony DeRose, Mary Bailey, Bill Barnard, Robert Cypher, David Dobrikin, Carl Ebeling, Smaragda Konstantinidou, Larry McMurchie, Haim Mizrahi, Bill Yost, The Apex: Two VLSI Designs for Generating Parametric Curves and Surfaces, *The Visual Computer*, **5**, 1989, 264-276.

[9] Farin, G., *Curves and Surfaces for Computer Aided Geometric Design*, Academic Press, San Diego, California, 1988.

[10] Goldman, R.N., Markov Chains and Computer Aided Geometric Design: Part I — Problems and Constraints, *ACM Transactions on Graphics*, **3**, 1984, 204-222.

[11] Goodman, T.N.T., and Said, H.B., Shape Preserving Properties of the Generalized Ball's Basis, submitted for publication.

[12] Karlin, S., *Total Positivity*, Stanford University Press, Stanford, California, 1968.

[13] Pavlidis, T., *Algorithms for Graphics and Image Processing*, Computer Science Press, Rockville, Maryland, 1982.

[14] Said, H.B., Recursive Algorithm for the Generalized Ball Curve, to appear in *ACM Transactions on Graphics*.