A Recursive Evaluation Algorithm for a Class of Catmull-Rom Splines

Phillip J. Barry and Ronald N. Goldman
Computer Graphics Laboratory
Computer Science Dept., Univ. of Waterloo
Waterloo, Ontario, Canada N2L 3G1

Abstract: It is known that certain Catmull-Rom splines [7] interpolate their control vertices and share many properties such as affine invariance, global smoothness, and local control with B-spline curves; they are therefore of possible interest to computer aided design. It is shown here that another property a class of Catmull-Rom splines shares with B-spline curves is that both schemes possess a simple recursive evaluation algorithm. The Catmull-Rom evaluation algorithm is constructed by combining the de Boor algorithm for evaluating B-spline curves with Neville's algorithm for evaluating Lagrange polynomials. The recursive evaluation algorithm for Catmull-Rom curves allows rapid evaluation of these curves by pipelining with specially designed hardware. Furthermore it facilitates the development of new, related curve schemes which may have useful shape parameters for altering the shape of the curve without moving the control vertices. It may also be used for constructing transformations to Bézier and B-spline form.

Categories and Subject Descriptors: G.1.1 [**Numerical Analysis**]: Interpolation — *Spline and piecewise polynomial interpolation*; G.1.2 [**Numerical Analysis**]: Approximation — *Spline and piecewise polynomial approximation*; I.3.5 [**Computer Graphics**]: Computational Geometry and Object Modeling — *Curve and surface representation*

Additional Key Words and Phrases: B-spline, Catmull-Rom spline, de Boor algorithm, Lagrange polynomial, Neville's algorithm, recursive evaluation algorithm

1 Introduction

In computer aided design designers often use B-spline curves:

$$S(t) = \sum_i N_i^n(t) V_i \tag{1}$$

where the V_i are points called *control vertices* and the $N_i^n(t)$ are degree n (order $n+1$) piecewise polynomial blending functions called *B-splines* or *B-spline basis functions*. These blending functions depend on a set of knots $\{t_i\}$. By altering the control vertices, and perhaps the knots, the designer is able to manipulate the shape of the curve [13].

B-spline curves are useful for many reasons, among them

— piecewise polynomials: B-spline curves are piecewise polynomial. They are therefore easy to store and manipulate, while often being faster to compute and analyze than single polynomials because usually one can use lower degree piecewise polynomials.

— differentiability: B-spline curves have a high degree of smoothness. *Usually the curve will be $n-1$ times continuously differentiable*, although by introducing multiple knots it can be designed to be less smooth.

— local control: altering a single control vertex affects only a limited portion of the curve rather than the entire curve. This is because the blending function $N_i^n(t)$ is 0 outside of the interval $[t_i, t_{i+n+1})$.

— affine invariance: the B-spline basis functions are normalized to sum to 1. This implies that B-spline curves are invariant under affine transformations. Thus these curves depend only on the relative geometry of their control vertices, and not on any absolute coordinate system.

— recursive evaluation algorithm: There is a simple, numerically stable, recursive evaluation algorithm for B-spline curves called the de Boor algorithm [4], which computes points along $S(t)$ without explicitly evaluating $N_i^n(t)$.

While B-spline curves do have many desirable properties, there are some desirable features they lack. For example, B-spline curves are approximating curves; that is, they approximate the shape and position of the control polygon (the polygon obtained by connecting the control vertices in order with line segments), but in general they do not interpolate the control vertices. For some applications in computer aided geometric design it is desirable for the control vertices to actually lie *on* the curve. Many known curve schemes interpolate their control vertices, however these curve schemes usually have other drawbacks. Natural cubic splines (see, e.g., [5] or [6]), for example, do not have local control or a known recursive evaluation algorithm.

Certain curve schemes have been developed which retain many B-spline curve properties while incorporating additional features. Catmull-Rom splines are one such scheme. Catmull and Rom noted that, in any curve scheme, one can replace control vertices with functions, and thus get a more general scheme [7]. Various choices of these functions impart different desirable properties to the curve. In particular, they observed that certain choices led to interpolatory curves. Although Catmull and Rom discussed a more general case, we will restrict our attention to an important class of Catmull-Rom splines obtained by combining B-spline basis functions and Lagrange interpolating polynomials. These Catmull-Rom splines, which we shall define more precisely below, have many nice features. They are piecewise polynomial, have local support, are invariant under affine transformations, and have certain differentiability and interpolatory properties. The purpose of this paper is to show that they also have a recursive evaluation algorithm similar to the de Boor algorithm for B-splines.

This new result is interesting for many reasons. First, by employing this algorithm and specially designed hardware one can evaluate and render such curves very rapidly [10]. Second, the fact that Catmull-Rom splines possess such an evaluation algorithm places them in a class of curves called "piecewise recursive curve schemes" [2]. These schemes generate new curves which may not only retain certain properties of Catmull-Rom splines but also possess shape parameters, scalars which affect the shape of a curve without moving the control vertices. Third, general results for recursive curve schemes provide ways of transforming Catmull-Rom splines to Bézier or B-spline form. Finally, the fact that Catmull-Rom splines have a recursive evaluation algorithm of this sort makes them noteworthy because very few well-known piecewise polynomial schemes have such an evaluation algorithm.

This paper is structured as follows: in Section 2 we will give more details concerning Catmull-Rom splines and then introduce the recursive evaluation algorithm and make some observations about it. In Section 3 we give some concrete examples by looking at a few cubic Catmull-Rom splines. Section 4 contains concluding remarks.

2 Catmull-Rom splines and their recursive evaluation algorithm

This section is divided into three parts. In Subsection 2.1 we define the class of Catmull-Rom splines with which we are concerned and discuss the properties of the splines in this class. In Subsection 2.2 we introduce the recursive evaluation algorithm for this class of curves. Subsection 2.3 contains a discussion of some aspects of the algorithm.

2.1 Catmull-Rom splines

Catmull and Rom noted that if one began with a curve scheme

$$D(t) = \sum_i F_i(t)V_i \tag{2}$$

one could replace the control vertices V_i by functions $V_i(t)$ which may depend on new control vertices. The resulting curve scheme will be more general than the original scheme, since $V_i(t)$ can be thought of as a generalization of V_i. This generality can be exploited to endow the new curve with special properties; in particular Catmull and Rom observed that special choices of $V_i(t)$ will result in $D(t)$ interpolating certain points.

We will now center our attention on an important subclass of this class of curves. We will employ a notation more suitable for our needs than that used in [7].

Suppose one wishes to generalize B-spline curves. Let the blending functions $F_i(t)$ in (2) be the B-spline basis functions $N_i^n(t)$ with knot set $\{t_i\}$. One can choose the $V_i(t)$ to be the Lagrange interpolating polynomials of degree m which interpolate the control vertices $P_{i-m}, \ldots, P_i$ at any distinct nodes $s_{i-m}, \ldots, s_i$, respectively. Unlike the knots, we do not require the nodes to be increasing. Although Catmull and Rom equated the nodes s_j with the knots t_i, we will allow the nodes to be arbitrary, as long as they are distinct. Analogous to the interpolation result in [7], if $s_j \in [t_{j+n}, t_{j+m+1})$, then $D(s_j) = P_j$ (the spline is interpolatory). Note this requires $m \geq n$. (If t_{j+m+1} is not a knot of multiplicity $n+1$, then $D(s_j) = P_j$ if $s_j \in [t_{j+n}, t_{j+m+1}]$.) Regardless of whether or not the spline is interpolatory, it will always be a degree $n+m$ piecewise polynomial, $n-1$ times continuously differentiable, affine invariant, and have local control. All these properties are inherited from properties of the B-spline basis functions $N_i^n(t)$ and the Lagrange interpolating polynomials $V_i(t)$.

The Catmull-Rom splines in this class can be written either as

$$D(t) = \sum_i N_i^n(t)V_i(t) \tag{3}$$

or, collecting coefficients of P_i, as

$$D(t) = \sum_i G_i(t)P_i \tag{4}$$

for some functions $G_i(t)$. The $G_i(t)$ are called the Catmull-Rom blending functions and are sums of products of certain B-spline and Lagrange basis functions. More specifically if we let $L_{j,i}^m(t)$ denote the Lagrange cardinal function which has nodes $s_{j-m}, \ldots, s_j$ and which is 1 at s_i, $j-m \leq i \leq j$, then

$$V_i(t) = \sum_{w=i-m}^{i} L_{i,w}^m(t)P_w \tag{5}$$

and it is then not difficult to see that

$$G_i(t) = \sum_{w=i}^{i+m} N_w^n(t)L_{w,i}^m(t). \tag{6}$$

When $s_j \in [t_{j+n}, t_{j+m+1})$ for all j, then these blending functions satisfy the cardinal conditions $G_i(s_j) = \delta_{ij}$ since if $j \neq i$ then either $L_{w,i}^m(s_j) = 0$ or $N_w^n(s_j) = 0$ while if $j = i$ then $L_{w,i}^m(s_j) = 1$ for $w = i, \ldots, i+m$ and $\sum_{w=i}^{i+m} N_w^n(s_j) = 1$.

For Catmull-Rom splines we have a set of knots $\{t_i\}$ and a set of nodes $\{s_j\}$. To simplify the number of parameters, we can express the s's in terms of the t's. For example, we can equate the nodes with the knots. If for all j we set $s_j = t_{j+k}$ for any integer k such that $n \leq k \leq m$ ($n \leq k \leq m+1$ if all the knots have multiplicity 1), we will not destroy any interpolation present. Another possibility is to equate the Lagrange nodes to the B-spline nodes (cf. [5, p.214]).

If the B-spline has multiple knots, equating knots and nodes will create a singularity in the Lagrange curve. The usual method of overcoming this difficulty is to use a scheme which also interpolates certain derivative values; however, such a scheme will not have a recursive evaluation algorithm of the type developed below. Thus, when the B-spline has multiple knots, one cannot totally equate the Lagrange nodes to the B-spline knots. One can do so partially as long as no assignments are made resulting in singularities.

Some examples of Catmull-Rom splines and their blending functions will be shown in Section 3.

2.2 The recursive evaluation algorithm

We now introduce a recursive evaluation algorithm for this class of Catmull-Rom curves. To do this, we call to mind two other recursive evaluation algorithms — the above mentioned de Boor algorithm [4] for B-spline curves, and Neville's algorithm (cf. [6]) for Lagrange polynomials.

To evaluate a B-spline curve $S(t)$ for $t \in [t_q, t_{q+1})$, set

$$P_i^0(t) = V_i \qquad i = q-n, \ldots, q$$

$$P_i^r(t) = \frac{t_{n+1+i-r} - t}{t_{n+1+i-r} - t_i} P_{i-1}^{r-1}(t) + \frac{t - t_i}{t_{n+1+i-r} - t_i} P_i^{r-1}(t)$$

$$r = 1, \ldots, n; \quad i = q-n+r, \ldots, q. \tag{7}$$

Then $S(t) = P_q^n(t)$. The de Boor algorithm can be represented in a triangular array as shown in Figure 1. An example using this algorithm to evaluate a B-spline curve is given in Section 3.

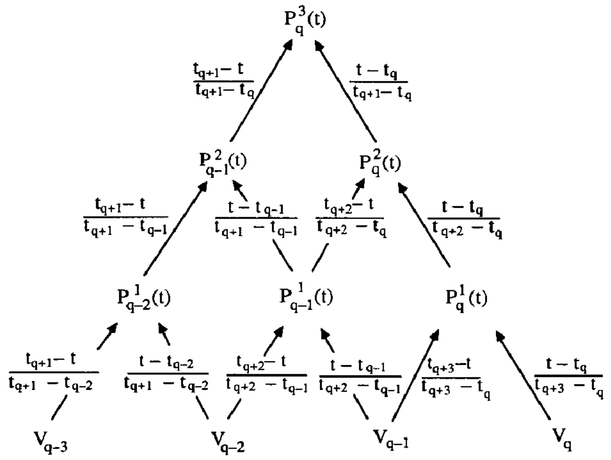
$P_q^3(t)$
$\frac{t_{q+1}-t}{t_{q+1}-t_q}$
$\frac{t-t_q}{t_{q+1}-t_q}$
$P_{q-1}^2(t)$
$P_q^2(t)$
$\frac{t_{q+1}-t}{t_{q+1}-t_{q-1}}$
$\frac{t-t_{q-1}}{t_{q+1}-t_{q-1}}$
$\frac{t_{q+2}-t}{t_{q+2}-t_q}$
$\frac{t-t_q}{t_{q+2}-t_q}$
$P_{q-2}^1(t)$
$P_{q-1}^1(t)$
$P_q^1(t)$
$\frac{t_{q+1}-t}{t_{q+1}-t_{q-2}}$
$\frac{t-t_{q-2}}{t_{q+1}-t_{q-2}}$
$\frac{t_{q+2}-t}{t_{q+2}-t_{q-1}}$
$\frac{t-t_{q-1}}{t_{q+2}-t_{q-1}}$
$\frac{t_{q+3}-t}{t_{q+3}-t_q}$
$\frac{t-t_q}{t_{q+3}-t_q}$
V_{q-3}
V_{q-2}
V_{q-1}
V_q

Figure 1: A diagram of the de Boor algorithm for evaluating a B-spline curve $S(t)$. The cubic case ($n = 3$) is shown where $t \in [t_q, t_{q+1})$. The control vertices V_i are placed at the bottom of the triangle and blended together until the point $P_q^n(t) = S(t)$ is obtained.

Neville's algorithm is a recursive algorithm for evaluating the Lagrange polynomial $L(t)$ which interpolates points $P_0, \ldots, P_m$ at nodes $s_0, \ldots, s_m$. It can be written as

$$P_i^0(t) = P_i \quad i = 0, \ldots, m$$

$$P_i^r(t) = \frac{s_i - t}{s_i - s_{i-r}} P_{i-1}^{r-1}(t) + \frac{t - s_{i-r}}{s_i - s_{i-r}} P_i^{r-1}(t)$$

$$r = 1, \ldots, m; \quad i = r, \ldots, m. \tag{8}$$

Then $L(t) = P_m^m(t)$. Figure 2 shows a diagram of Neville's algorithm as a triangular array. An example using Neville's algorithm to evaluate a Lagrange polynomial is given in Section 3.

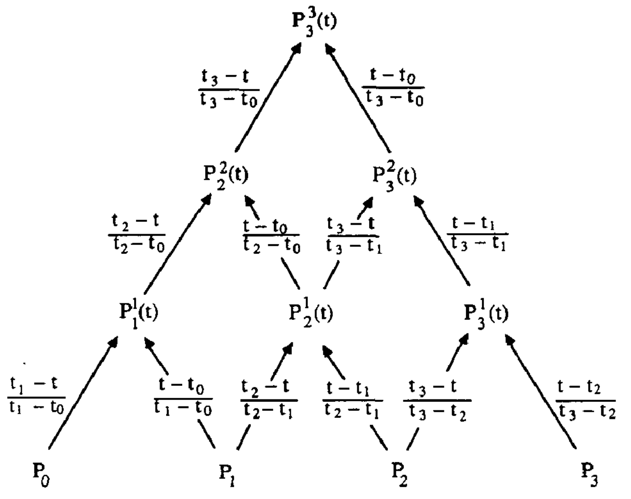
$P_3^3(t)$
$\frac{t_3 - t}{t_3 - t_0}$
$\frac{t - t_0}{t_3 - t_0}$
$P_2^2(t)$
$P_3^2(t)$
$\frac{t_2 - t}{t_2 - t_0}$
$\frac{t - t_0}{t_2 - t_0}$
$\frac{t_3 - t}{t_3 - t_1}$
$\frac{t - t_1}{t_3 - t_1}$
$P_1^1(t)$
$P_2^1(t)$
$P_3^1(t)$
$\frac{t_1 - t}{t_1 - t_0}$
$\frac{t - t_0}{t_1 - t_0}$
$\frac{t_2 - t}{t_2 - t_1}$
$\frac{t - t_1}{t_2 - t_1}$
$\frac{t_3 - t}{t_3 - t_2}$
$\frac{t - t_2}{t_3 - t_2}$
P_0
P_1
P_2
P_3

Figure 2: A diagram of Neville's algorithm for evaluating a Lagrange polynomial $L(t)$. The cubic case ($m = 3$) is shown. The control points P_i are placed at the bottom of the triangle and blended together until the point $P_m^m(t) = L(t)$ is obtained.

To evaluate a Catmull-Rom spline we could naively first use Neville's algorithm to calculate the functions $V_i(t)$ and then apply the de Boor algorithm to obtain a point on the curve $D(t)$. See Figure 3 for a diagram of this procedure. Note that although Figure 3 does present a recursive evaluation algorithm for Catmull-Rom curves, this algorithm is neither as efficient nor as elegant as, e.g., the de Boor algorithm for B-spline curves since the de Boor algorithm can be represented in a more compact triangular form (Figure 1).

What is remarkable about Catmull-Rom splines is that this naive recursive evaluation algorithm can be simplified extensively because there is substantial overlap in evaluating succesive $V_i(t)$'s. Let $P_{ji}^r(t)$ be the point $P_j^r(t)$ computed in evaluating $V_i(t)$ by Neville's algorithm. Then, since $V_i(t)$ and $V_{i+1}(t)$ share the nodes $s_{i-m+1}, \ldots, s_i$ and the control vertices $P_{i-m+1}, \ldots, P_i$, the parameters used in calculating $P_{j+1,i}^r(t)$ are identical to the corresponding parameters used in computing $P_{j,i+1}^r(t)$ when $j < m$. Therefore $P_{j+1,i}^r(t) = P_{j,i+1}^r(t)$ for $j < m$. In particular the points $P_{m-1,i+1}^r(t)$ are found in evaluating $V_i(t)$ and therefore do not need to be recomputed in evaluating $V_{i+1}(t)$. If we store these points, then (see Figure 2) we need only compute the m new values $P_{m,i+1}^r(t)$ $r = 1, \ldots, m$ to evaluate $V_{i+1}(t)$.

This redundancy leads to a simpler and less costly evaluation algorithm for Catmull-Rom splines. This algorithm can be represented by a triangular diagram, an example of which is shown in Figure 4, similar to the diagrams in Figures 1 and 2.

We now write down explicitly the recursive evaluation algorithm for Catmull-Rom splines. To evaluate the Catmull-Rom curve $D(t)$ for $t \in [t_q, t_{q+1})$, set

$$P_i^0(t) = P_i \quad i = q - n - m, \ldots, q$$

$$P_i^r(t) = \frac{s_i - t}{s_i - s_{i-r}} P_{i-1}^{r-1}(t) + \frac{t - s_{i-r}}{s_i - s_{i-r}} P_i^{r-1}(t)$$

$$r = 1, \ldots, m; \quad i = q - n - m + r, \ldots, q.$$

$$P_i^r(t) = \frac{t_{n+1+i-r} - t}{t_{n+1+i-r} - t_i} P_{i-1}^{r-1}(t) + \frac{t - t_i}{t_{n+1+i-r} - t_i} P_i^{r-1}(t)$$

$$r = m+1, \ldots, m+n; \quad i = q-n-m+r, \ldots, q. \quad (9)$$

Then $D(t) = P_q^{m+n}(t)$.

To verify that $P_q^{m+n}(t)$ does indeed equal $D(t)$, merely note that the first stage of the algorithm produces the Lagrange curves $P_i^m(t) = V_i(t)$ $i = q - n, \ldots, q$ and the second stage applies the de Boor algorithm to these curves.

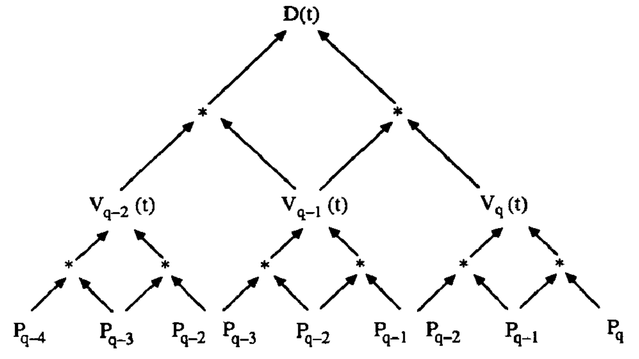

Figure 3: One possible means of evaluating Catmull-Rom splines is to evaluate each of the $V_i(t)$ separately by Neville's algorithm, and then blend these curves together by the de Boor algorithm to get $D(t)$. Such a technique can be represented as shown (for the case $n = m = 2$).

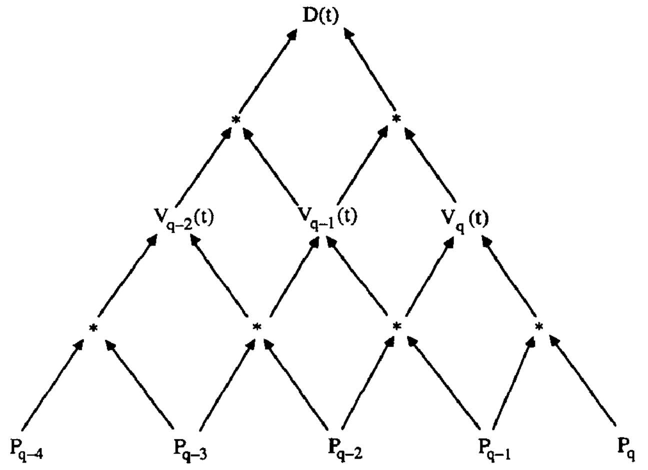

Figure 4: A diagram of the recursive evaluation algorithm for Catmull-Rom curves. The case shown is $n = m = 2$. The lower levels in Figure 3 can be combined to yield the algorithm shown here. This algorithm is theoretically less complicated and computationally less expensive than the one in Figure 3.

2.3 A discussion of the algorithm

In this subsection we will discuss a few important aspects of the recursive evaluation algorithm for Catmull-Rom splines.

As with B-splines, it is possible to use the standard tensor product construction to obtain rectangular tensor product Catmull-Rom surfaces. Such surfaces will possess geometric properties similar to those possessed by Catmull-Rom curves, and will also possess an analogous recursive evaluation algorithm. Whether one can construct similar surfaces for triangular domains is still an open question.

We mentioned above that one of the reasons the recursive evaluation algorithm for Catmull-Rom curves is important is that it allows fast evaluation of the curve. If one used a processor to compute each new point $P_i^r(t)$ and ran the processors in parallel, one could calculate points on the curve extremely rapidly [10]. A slightly more complicated multiprocessor architecture would allow rapid evaluation of Catmull-Rom tensor product surfaces. Such architectures would permit real-time interactive design using Catmull-Rom tensor product surfaces and/or large numbers of Catmull-Rom splines.

Shape parameters are scalars which affect the shape of the curve without moving the control vertices (cf. [3]). They are useful for introducing features such as tension into a curve. By altering a single shape parameter one can often obtain a curve which would otherwise have required simultaneously changing the positions of many control vertices. Catmull-Rom splines do not have shape parameters as such, although it may be possible to develop useful shape parameters based on the knots and/or the nodes. (Another possibility — *visually continuous* Catmull-Rom splines — has been studied in [9].) We can, however, use the recursive evaluation algorithm to construct curve schemes which have shape parameters and which are related to Catmull-Rom splines. One can modify the recursive evaluation algorithm so that for some choices of r, i and shape parameters $a_{r,i}, b_{r,i}$,

$$P_i^r(t) = \frac{a_{r,i} - t}{a_{r,i} - b_{r,i}} P_{i-1}^{r-1}(t) + \frac{t - b_{r,i}}{a_{r,i} - b_{r,i}} P_i^{r-1}(t) \qquad (10)$$

while the remainder of the algorithm stays the same. The resulting curves will still have a recursive evaluation algorithm, be piecewise polynomial, have local support, and be affine invariant. Depending on how the shape parameters are introduced, the curves may retain differentiability and interpolatory properties. Again, the effects of such shape parameters need further study.

Another reason that a recursive evaluation algorithm is important is that it can be applied to develop transformation formulas. By using the recursive evaluation algorithm and techniques such as degree raising and duality [1] or a method called "blossoming" [12], it is possible to transform Catmull-Rom curves to B-spline or Bézier form. Further research is needed to investigate the efficiency of these transformation techniques and to determine whether these methods can be generalized to yield other transformations algorithms.

As with B-splines (cf. [12]), the triangular arrays representing the recursive evaluation algorithm for contiguous segments of a Catmull-Rom spline mesh together. If we overlay the triangular arrays for two adjacent segments of the spline so that the shared control points lie directly on top of one another, then the functions on overlapping edges of the diagram will be identical [2]. This makes Catmull-Rom splines especially noteworthy since very few piecewise polynomial schemes have a simple evaluation algorithm, let alone one of this special form.

There are certain properties possessed by the de Boor algorithm which are not possessed by the Catmull-Rom evaluation algorithm. Since the de Boor algorithm calculates new points by taking convex combinations of old points, it is numerically stable. However, any reasonably smooth interpolatory curve will not lie in the convex hull of its control vertices; therefore the evaluation algorithm for Catmull-Rom splines may be unstable. In particular instability can occur when there is a large variation in the spacing between the knots or the nodes. This instability is a consequence of a similar problem in Neville's algorithm.

Another difference is that there is a closer connection between the de Boor algorithm and the B-spline basis functions than between the evaluation algorithm for Catmull-Rom splines and the Catmull-Rom basis functions.

For example, one can derive the Cox-Mansfield-de Boor recurrence relationship [4,8] for the B-spline basis functions from the de Boor algorithm. Similarly one can derive a recursion formula for the Catmull-Rom blending functions from the recursive evaluation algorithm for Catmull-Rom curves. However, in the recursion formula for B-splines, B-splines are computed directly from lower degree B-splines, while the Catmull-Rom blending functions are not computed directly from lower degree Catmull-Rom blending functions. Again this happens because Neville's algorithm does not compute the Lagrange basis functions from a Lagrange basis of lower degree.

3 Examples

In this section we will examine a few cubic Catmull-Rom splines and their recursive evaluation algorithms. We take the B-spline knots to be uniformly spaced with $t_i = i$, define the Lagrange nodes by $s_j = t_{j+m}$, and consider the evaluation algorithm for the interval [0,1]. For each example we will provide a set of four graphs: a diagram of the recursive evaluation algorithm, a spline curve, an example using the recursive evaluation algorithm to evaluate a point on the curve, and a graph of one of the blending functions. In the graphs showing evaluation, the only point $P_i^r(t)$ which lies on the curve is $P_0^3(t)$. That some of the other points appear to lie on the curve is merely coincidental. Also, since the knot spacing in the examples is uniform, all other blending functions are translates of the one shown in the blending function graphs.

For cubics we have four possibilities for n and m. If we choose $n = 3$, $m = 0$ we will obtain the usual cubic B-spline which is twice continuously differentiable. The graphs for this case are shown in Figure 5. The case $n = 2, m = 1$ produces a spline which is C^1. Figure 6 shows the illustrations for this case.

The case $n = 1, m = 2$ illustrates some elegant properties of Catmull-Rom splines. The figures for this case are presented in Figure 7. The resulting spline is guaranteed to be C^0 and interpolating. It is, however, actually C^1. Indeed, for any Catmull-Rom spline if for all j we let $s_j = t_{j+k}$ for any integer k such that $n < k \leq m$,

$$\begin{tikzcd}
 & & & P_0^3(t) & & & \\
 & & P_{-1}^2(t) \arrow[ur, "1-t"] & & P_0^2(t) \arrow[ul, "t"'] & & \\
 & P_{-2}^1(t) \arrow[ur, "\frac{1-t}{2}"] & & P_{-1}^1(t) \arrow[ul, "\frac{t+1}{2}"'] \arrow[ur, "\frac{2-t}{2}"] & & P_0^1(t) \arrow[ul, "\frac{t}{2}"'] & \\
P_{-3} \arrow[ur, "\frac{1-t}{3}"] & & P_{-2} \arrow[ul, "\frac{t+2}{3}"'] \arrow[ur, "\frac{2-t}{3}"] & & P_{-1} \arrow[ul, "\frac{t+1}{3}"'] \arrow[ur, "\frac{3-t}{3}"] & & P_0 \arrow[ul, "\frac{t}{3}"']
\end{tikzcd}$$

Figure 5a

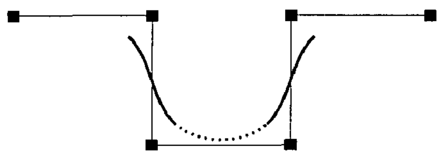

Figure 5b

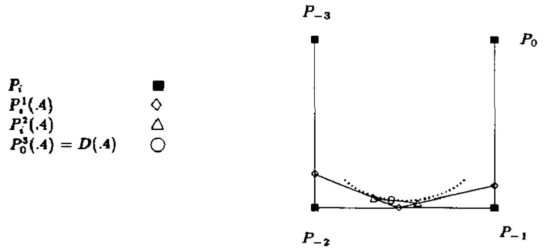

Figure 5c

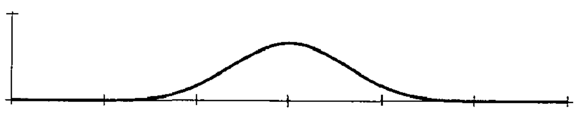

Figure 5d

Figure 5: Various figures for cubic Catmull-Rom splines with $n = 3$, $m = 0$, $t_i = i$. Since $m = 0$ this is a C^2 B-spline curve. Figure 5a shows a diagram of the evaluation algorithm for $t \in [0, 1]$, Figure 5b shows an example of such a spline along with its control polygon, Figure 5c shows evaluation of the spline at $t = .4$, and Figure 5d shows a blending function $G_i(t)$.

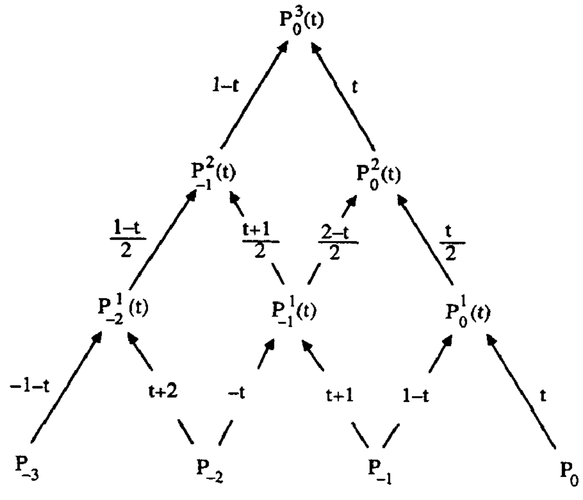

Figure 6a

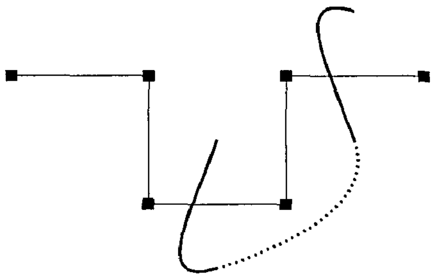

Figure 6b

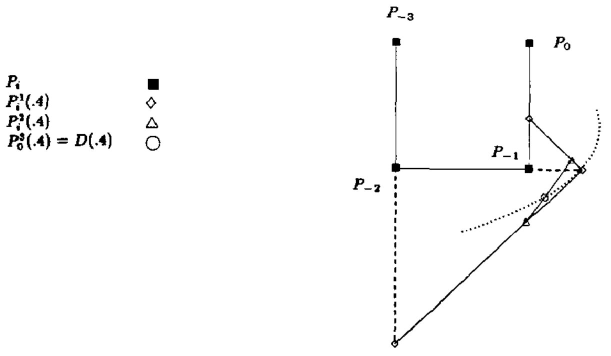

Figure 6c

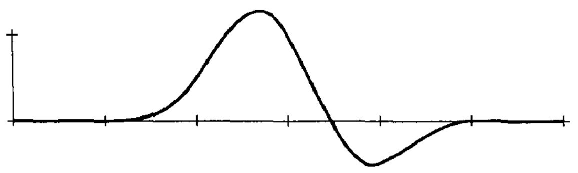

Figure 6d

Figure 6: Various figures for cubic Catmull-Rom splines with $n = 2$, $m = 1$, $t_i = i, s_i = i + 1$. The spline consists of quadratic B-splines blended with linear Lagrange curves. Figure 6a shows a diagram of the evaluation algorithm for $t \in [0, 1]$, Figure 6b shows an example of such a spline along with its control polygon, Figure 6c shows evaluation of the spline at $t = .4$, and Figure 6d shows a blending function $G_i(t)$. Note that this spline is C^1.

then the curve is C^n [2]. Note too from Figure 7 that the case $n = 1$, $m = 2$, $s_j = t_{j+2}$ is identical to the case $n = 2$, $m = 1$ with $s_j = t_{j+2}$. In general, if $n+m = 2d+1$ for any nonnegative integer d, then by comparing recursive evaluation algorithms it can be shown that the cases $n = m + 1, s_j = t_{j+m+1}$ and $n = m - 1, s_j = t_{j+m}$ are identical. Finally, the particular case here is also an Overhauser spline [11].

The final case, shown in Figure 8, is $n = 0$, $m = 3$, which produces a continuous interpolatory spline.

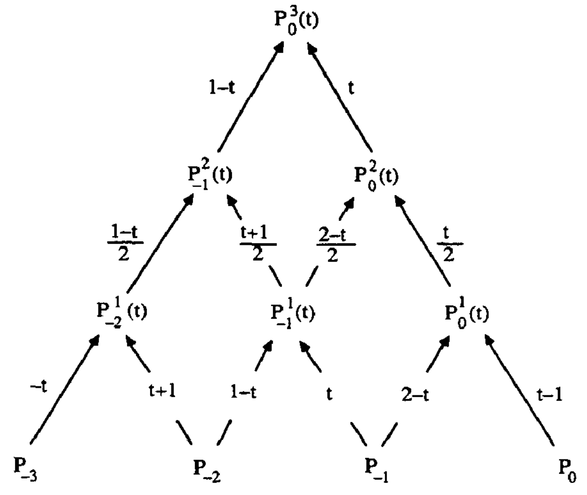

Figure 7a

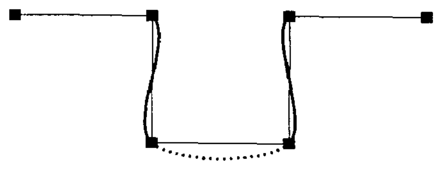

Figure 7b

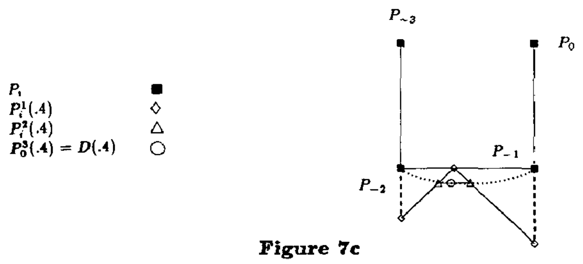

Figure 7c

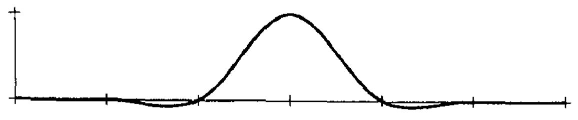

Figure 7d

Figure 7: Various figures for cubic Catmull-Rom splines with $n = 1$, $m = 2$, $t_i = i, s_i = i + 2$. This spline consists of linear B-splines blended with quadratic Lagrange polynomials. Figure 7a shows a diagram of the evaluation algorithm for $t \in [0, 1]$, Figure 7b shows an example of such a spline along with its control polygon, Figure 7c shows evaluation of the spline at $t = .4$, and Figure 7d shows a blending function $G_i(t)$. This spline is C^1 and interpolatory.

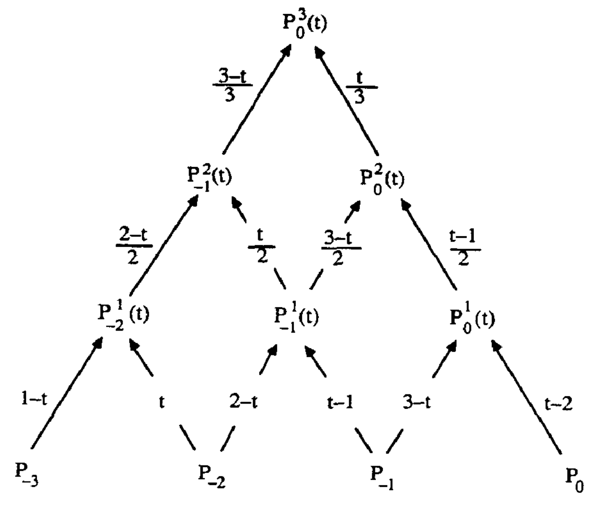

Figure 8a

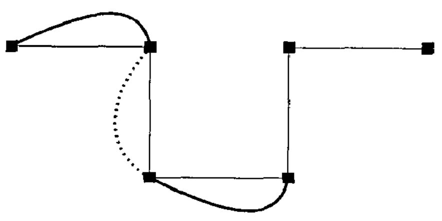

Figure 8b

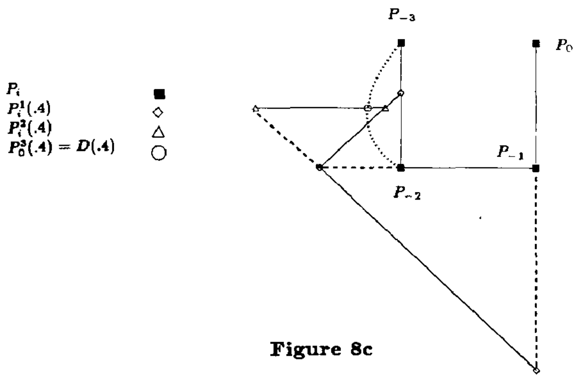

Figure 8c

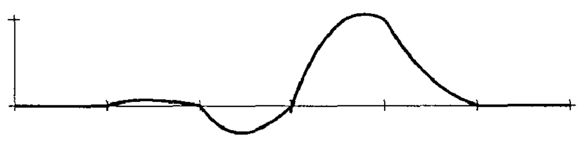

Figure 8d

Figure 8: Various figures for cubic Catmull-Rom splines with $n = 0$, $m = 3$, $t_i = i, s_i = i + 3$. Since $n = 0$ each segment of this curve is a cubic Lagrange polynomial. Figure 8a shows a diagram of the evaluation algorithm for $t \in [0, 1]$, Figure 8b shows an example of such a spline along with its control polygon, Figure 8c shows evaluation of the spline at $t = .4$, and Figure 8d shows a blending function $G_i(t)$. This spline is C^0 and interpolatory.

4 Concluding remarks

In this paper we have shown that a special class of Catmull-Rom splines possess a recursive evaluation algorithm similar to the de Boor algorithm for B-splines. This algorithm for Catmull-Rom splines is obtained by combining the deBoor algorithm for the evaluation of B-splines with Neville's algorithm for the evaluation of Lagrange curves. The recursive evaluation algorithm for Catmull-Rom splines gives us a possible means of fast evaluation. It also allows the construction of new curve schemes which retain some properties of Catmull-Rom curves and which may have useful shape parameters. Another benefit is that it facilitates derivation of some transformation techniques. Further it makes Catmull-Rom splines noteworthy since very few known piecewise polynomial schemes have a simple evaluation algorithm.

Although we have derived one new property of Catmull-Rom splines, other properties still need to be investigated. For example, what shape preservation properties do Catmull-Rom splines possess? What geometric effects can be produced by the introduction of shape parameters? And finally, are there simple algorithms for such processes as subdivision, degree elevation, or differentiation of Catmull-Rom splines?

Acknowledgments: The research reported here was partially supported by the Natural Sciences and Engineering Research Council of Canada, the University of Waterloo, and Control Data Corporation. The authors would also like to thank Dr. John Beatty of the University of Waterloo for his helpful remarks.

References

1. Barry, Phillip J., Urn models, recursive curve schemes, and computer aided geometric design, Ph.D. dissertation, Dept. of Mathematics, University of Utah, Salt Lake City, 1987.

2. Barry, Phillip J. and Goldman, Ronald N., Piecewise polynomial recursive curve schemes and computer aided geometric design, in preparation.

3. Barsky, Brian A., The beta-spline: a local representation based on shape parameters and fundamental geometric measures, Ph.D. dissertation, Computer Science Dept., University of Utah, Salt Lake City, 1981.

4. de Boor, Carl, On calculating with B-splines, *Journal of Approximation Theory* 6, (1972), 50-62.

5. de Boor, Carl, *A Practical Guide to Splines*, Springer-Verlag, New York, 1978.

6. Burden, Richard L., Faires, J.Douglas, and Reynolds, Albert C., *Numerical Analysis*, Prindle, Weber, and Schmidt, Boston, 1978.

7. Catmull, Edwin and Rom, Raphael, A class of local interpolating splines, in R.E. Barnhill and R.F. Riesenfeld (eds.) *Computer Aided Geometric Design*, Academic Press, New York, 1974, 317-326.

8. Cox, M.G., The numerical evaluation of B-splines, *J. Inst. Maths. Applics. 10*, (1972), 134-149.

9. DeRose, Anthony D. and Barsky, Brian A., Geometric continuity and shape parameters for Catmull-Rom splines, submitted for publication.

10. DeRose, Anthony D. and Holman, Thomas J., The triangle: a multiprocessor architecture for fast curve and surface generation, submitted for publication.

11. Overhauser, A.H., Analytic definition of curves and surfaces by parabolic blending, Scientific Research Staff Publication, Ford Motor Co., Detroit, Michigan, 1968.

12. Ramshaw, Lyle, *Blossoming: A Connect-the-Dots Approach to Splines*, Digital Systems Research Center, Palo Alto, California, 1987.

13. Riesenfeld, Richard F., Applications of B-spline approximation to geometric problems of computer-aided design, Ph.D. dissertation, Dept. of Systems and Information Science, Syracuse University, Syracuse, New York, 1973.