ELSEVIER

Computer-Aided Design 34 (2002) 337–345

COMPUTER-AIDED DESIGN

www.elsevier.com/locate/cad

Short Communication

The conversion of a dynamic B-spline curve into piecewise polynomials in power form

D.-S. Kim[a,*], J. Ryu[a], H.C. Lee[b], H. Shin[c]

[a]*Department of Industrial Engineering, Hanyang University, 17 Haengdang-Dong, Seongdong-Ku, Seoul, 133-791, South Korea*
[b]*Department of Industrial Engineering, Hongik University, Seoul, South Korea*
[c]*Department of Industrial Engineering, KAIST (Korea Advanced Institute of Science and Technology), Taejon, South Korea*

Received 21 March 2000; revised 12 January 2001; accepted 22 January 2001

Abstract

The evaluation of points and the computations of inflection points or cusps on a curve are often necessary in CAGD applications. When a curve is represented in a B-spline form, such computations can be made easier once it is transformed into a set of piecewise polynomial curves in power form. The usual practice of the transformation of a B-spline curve into a set of piecewise polynomial curves in power form is done either by a knot refinement followed by basis conversions, or by applying a Taylor expansion on each knot span of a B-spline curve.

Presented in this paper is a new algorithm to convert a B-spline curve into a set of piecewise polynomial curves in power form. Experiment shows that the proposed algorithm significantly outperforms the conventional approach when one or more control points of a B-spline curve are continuously moving. © 2002 Elsevier Science Ltd. All rights reserved.

Keywords: Dynamic curve; B-spline; Polynomial; Power form; Basis conversion; Taylor expansion

1. Introduction

In computer graphics and the applications of geometric modeling, it is often necessary to manipulate B-spline curves or surfaces by converting the B-spline representation into a set of piecewise polynomial curves or surfaces in power form. Once a curve is represented in power form, a point evaluation can be made faster due to Horner's rule even though the issue of stability remains [1]. It is also known that faster computation of the characteristic points on a curve, such as inflection points and cusps, can be facilitated by the conversion of a B-spline curve into a set of piecewise polynomial curves in power form. Note that the subdivision of a parametric curve at these characteristic points facilitates the fast computation of intersection points between curves [2]. In addition, this form of curve is supported by IGES as an entity type 112 [3].

Due to the relative advantages of implicit representation of curves or surfaces over parametric one in some geometric calculations such as a point inclusion problem, it is sometimes necessary that a parametric form be converted into an implicit form [4]. The implicitization process, which uses a resultant, usually requires the geometry to be represented in power form [5]. Since this operation is computationally demanding, the reduction of computation should not be ignored.

Discussed in this paper is the transformation of a B-spline curve into a set of piecewise polynomials in power form, which is known to be a tedious task [6]. Two types of curves are considered for the problem: static and dynamic curves. In this paper, a curve is called dynamic when one or more of the control points of the curve are moving. Otherwise, a curve is called static.

There are a few existing approaches to this problem on static cases. Since a static B-spline curve can be converted into a set of piecewise Bézier curves by a knot refinement [7–11], applying a basis conversion operation to each piece will produce a set of piecewise polynomials in power form. This approach is called the KR-approach in this paper. The same problem can also be solved by applying Taylor expansion of the B-spline curve at the beginning of each knot span with the appropriate number of terms depending on the degree of the curve [12], which is denoted by the TE-approach in this paper.

In this paper, however, we propose a different approach using the properties of basis functions. Note that a power form of B-spline curve can be easily obtained if the basis functions are maintained in power form. When one or more control points are moving, the multiplication of the

* Corresponding author. Tel.: +82-2-2290-0472; fax: +82-2-2292-0472.
E-mail address: dskim@hangyang.ac.kr (D.-S. Kim).

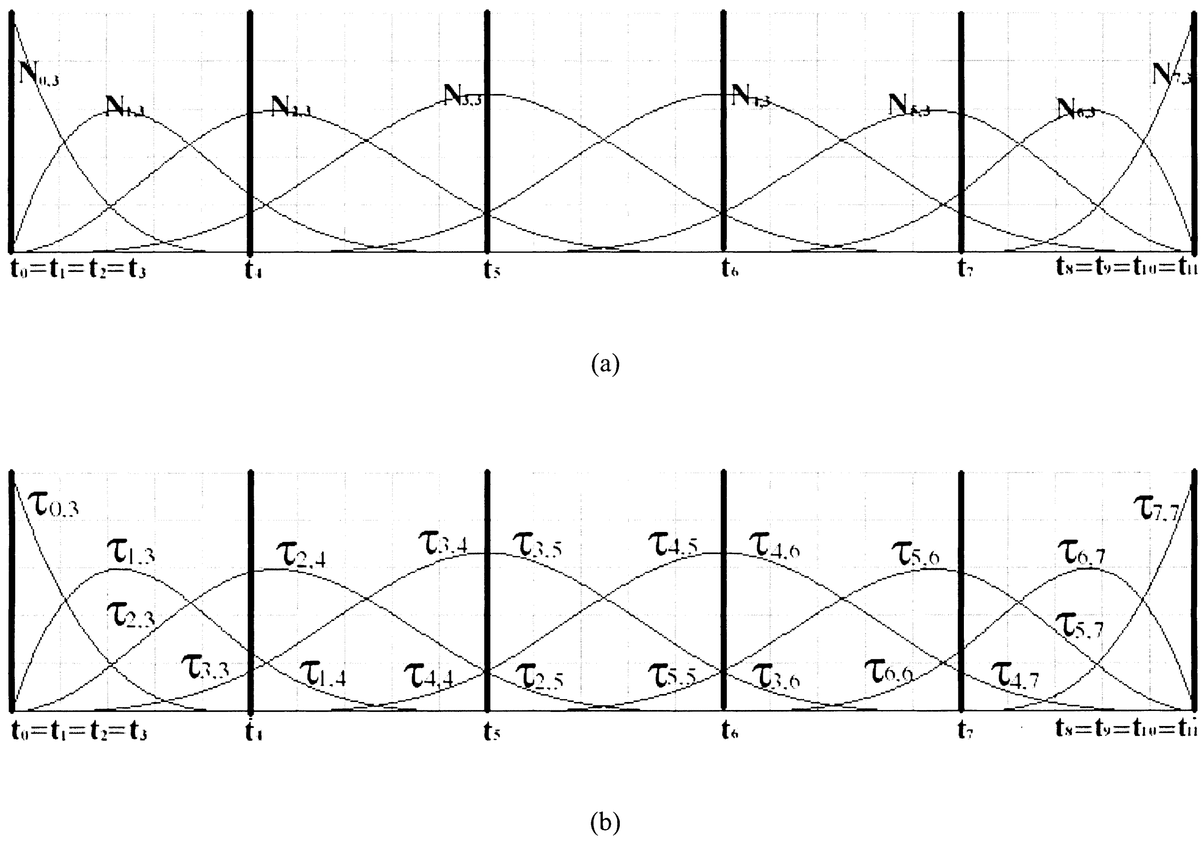


Fig. 1. An example of basis splines and truncated basis functions ($p = 3$, $m = 11$).

displacement of control points with the corresponding basis functions only need to be calculated for updating the curve. Since the basis functions of a B-spline curve are spline curves themselves, it is necessary to maintain a set of power form polynomial curves for a basis function conveniently. One way to obtain the power form of a basis function is applying Taylor expansion of the basis function at the beginning of each knot span appropriately. Note that the convenient form of the derivatives of basis functions is readily available [1,11]. In this paper, we present a different, yet more efficient, approach for this problem as well.

The main idea of the proposed algorithm, called a direct expansion (DE) algorithm, is as follows: after locating the coefficients of all linear terms that make up the basis functions in a knot span, the algorithm directly obtains the power form representation of basis functions in the knot span by expanding the summation of products of appropriate linear terms. Then, the polynomial curves in power form in the knot span can easily be obtained by summing up the multiplications of the basis functions in power form with corresponding control points. Repeating this operation for each knot span, all of the polynomials of a B-spline curve are transformed into a set of piecewise polynomials in power form.

Through experiments, it has been observed that the proposed DE algorithm significantly outperforms the existing approaches for the case of dynamic curves. Hence, the proposed algorithm can be very useful for the curve and surface implicitization as well as the computation of intersections when the curves or surfaces are dynamically changing.

This paper is organized as follows: Section 2 provides the basic introduction to basis functions of the B-spline curve and introduces the key concepts to develop the algorithm. In Sections 3 and 4, the computations of the power form polynomial curve segments of static and dynamic B-spline curves are discussed. In Section 5, the experiment results for the dynamic curve are presented.

2. Representation of basis functions

A B-spline curve of degree p with $(m+1)$ knots is defined by

$$\mathbf{C}(t) = \sum_{i=0}^{m-p-1} \mathbf{P}_i N_{i,p}(t) \qquad \text{for} \qquad 0 \le t \le 1 \tag{1}$$

where $\mathbf{P}_i$ and $N_{i,p}(t)$ are control points and B-spline basis functions of degree p on a knot vector $\mathbf{U} = \{0,\ldots,0,t_{p+1},\ldots,t_{m-p-1},1,\ldots 1\}$, $t_i \le t_{i+1}$, respectively [11]. In Eq. (1), the basis function $N_{i,p}(t)$ is defined as the following recurrence formula.

$$N_{i,p}(t) = \frac{t - t_i}{t_{i+p} - t_i} N_{i,p-1}(t) + \frac{t_{i+p+1} - t}{t_{i+p+1} - t_{i+1}} N_{i+1,p-1}(t)$$
$$N_{i,0}(t) = \begin{cases} 1 & \text{if} \quad t_i \le t < t_{i+1} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

where $i = 0,1,\ldots, m - p - 1$. It is known that Eq. (2) forms a triangular scheme [11]. For the convenience of discussion, we will use $N_{i,p}$ instead of $N_{i,p}(t)$.

Definition 1. A *truncated basis function*, $\tau_{i,w}(t)$, $i = w - p, w - p + 1,\ldots,w$, $w = p,p + 1,\ldots,m - p - 1$, is an active polynomial segment of $N_{i,p}$, in $[t_w,t_{w+1})$.

Fig. 1(a) shows cubic B-spline basis function with $m = 11$, and Fig. 1(b) illustrates the corresponding truncated basis functions in each knot span.

Note that there are $p + 1$ truncated basis functions in $[t_i,t_{i+1})$ for a B-spline curve of degree p. If we collect all the truncated basis functions with same i-value, then we can obtain a B-spline basis function. Thus, B-spline basis function $N_{i,p}$ can be represented by the following equation.

$$N_{i,p} = \sum_{w=i}^{i+p} \tau_{i,w}(t) N_{w,0} \tag{3}$$

Let Eq. (2) be rewritten as follows

$$N_{i,p} = h_{i,p}(t) N_{i,p-1} + v_{i,p}(t) N_{i+1,p-1} \tag{4}$$

where

$$h_{i,p}(t) = \frac{t - t_i}{t_{i+p} - t_i}, \quad v_{i,p}(t) = \frac{t_{i+p+1} - t}{t_{i+p+1} - t_{i+1}} \tag{5}$$

In the triangular scheme, a directed edge from $N_{i,j}$ to $N_{k,l}$ is called a *horizontal edge* and has an *edge value* of $h_{i,j+1}(t)$ if $k = i$ and $l = j + 1$. Similarly, a directed edge is called a *vertical edge* and has an *edge value* of $v_{i-1,j+1}(t)$ if $k = i - 1$ and $l = j + 1$.

It turns out that the truncated basis function is composed of the summation of the products of $h_{i,p}(t)$'s and $v_{i,p}(t)$'s. The proposed algorithm collects all $h_{i,p}(t)$'s and $v_{i,p}(t)$'s of each truncated basis function of a knot span, and efficiently performs the necessary summations of multiplications between appropriate $h_{i,p}(t)$'s and $v_{i,p}(t)$'s so that the result should be the desired truncated basis functions arranged in power form in the knot span. Then, the B-spline curve in the knot span can easily be transformed into a polynomial curve in power form by the summation of the multiplications between appropriate control points and truncated basis functions. Thus, a B-spline curve can be transformed into a set of piecewise polynomial curves in power form by repeating the above-mentioned operation for each knot span.

Therefore, the fundamental question is how to efficiently extract all the truncated basis functions in the power form of each knot span. The a priori knowledge of each constituent $h_{i,p}(t)$ and $v_{i,p}(t)$ of a truncated basis function would enable us to exploit the repeated form for calculating a truncated basis function. Let us focus our interest on one knot span for the time being. Fig. 2(a) is the directed graph, which illustrates a subset of triangular scheme in a knot span $[t_3,t_4)$ and shows the necessary computations for four truncated basis functions in a knot span $[t_3,t_4)$. Arrows in the figure should be interpreted as appropriate linear polynomials, which are defined by Eq. (5) and eventually comprise four truncated basis functions in $[t_3,t_4)$. The root and four leaf nodes correspond to a knot span and four truncated basis functions, respectively.

Definition 2. A *path* π from $N_{i,j}$ to $N_{k,l}$ is the set of directed edges consisting of a path from $N_{i,j}$ to $N_{k,l}$, and a path set $\Pi = \{\pi_a, a = 1,2,\ldots,n\}$ where n is the number of possible paths.

Definition 3. Given $N_{i,0}$, $N_{j,p}$, and Π, a *graph primitive* is $\bigcup_a \pi_a$.

The union operation in the previous definition implies the superposition of all paths so that the result should yield a topologically merged graph. It turns out that a truncated basis function corresponds to a graph primitive such as Fig. 2(b). In $[t_3,t_4)$, for example, there are four truncated basis functions: $\tau_{0,3}(t)$, $\tau_{1,3}(t)$, $\tau_{2,3}(t)$, and $\tau_{3,3}(t)$. Each truncated basis function, $\tau_{i,3}(t)$, $i = 0,1,2$, and 3, corresponds to the summation of the product of all the linear polynomials in paths from $N_{3,0}$ to $N_{i,3}$ $(i = 0,1,2,3)$, respectively. In Fig. 2(b), for example, there are three possible paths π_1, π_2 and π_3 from $N_{3,0}$ to $N_{1,3}$, and therefore the truncated basis function $\tau_{1,3}(t)$ can be obtained as follows

$$\tau_{1,3}(t) = v_{2,1}(t)v_{1,2}(t)h_{1,3}(t) + v_{2,1}(t)h_{2,2}(t)v_{1,3}(t) + h_{3,1}(t)v_{2,2}(t)v_{1,3}(t) \tag{6}$$

Theorem 1. $\tau_{i,w}(t)$, $i = w - p, w - p + 1,\ldots,w$, $w = p,p + 1,\ldots,m - p - 1$, for $[t_w,t_{w+1})$ can be represented as

$$\tau_{i,w}(t) = \sum_{k=1}^{n_p} P_k(t) \qquad P_k(t) = \prod_{l_1=1}^{n_h} h_{p_h,q_h}^{l_1}(t) \prod_{l_2=1}^{n_v} v_{p_v,q_v}^{l_2}(t) \tag{7}$$

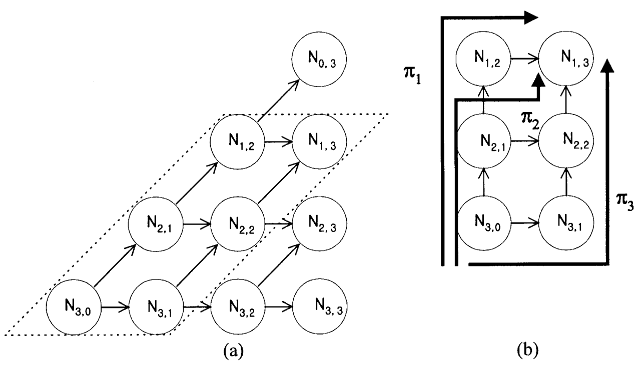


Fig. 2. A graph primitive corresponding to $\tau_{1,3}(t)$.

where n_p is the number of all the possible paths from $N_{w,0}$ to $N_{i,p}$, $i = w - p,\ldots,w$. n_h and n_v are the number of horizontal and vertical directed edges in each path, respectively, and l_1 and l_2 are arbitrary indices. Note that $n_h + n_v = p$, and p_h, p_v, q_h, and q_v are appropriate integers needed to define horizontal and vertical directed edges in each knot span.

Note that $n_p = \frac{p!}{(w-i)!(p-w+i)!}$ and the number of graph primitives in a knot span for a curve of degree p is $(p+1)$. In Fig. 3, four graph primitives for Fig. 2 are presented. Fig. 3(a)–(d) correspond to truncated basis functions $\tau_{0,3}(t)$, $\tau_{1,3}(t)$, $\tau_{2,3}(t)$ and $\tau_{3,3}(t)$ in $[t_3,t_4)$, respectively.

According to the theorem, if all the possible paths in each graph primitive of a knot span can be enumerated, then each truncated basis function of the knot span can be obtained by summing up the products of the linear polynomials for each path. Furthermore, once the degree of curve is fixed, these paths are identical for all the knot spans. Therefore, the algorithm requires the path enumeration step only once.

The problem that enumerates all the possible paths in each graph primitive can be formulated as the enumeration of all the possible sequences of size p, which consist of fixed numbers of 0's and 1's. The numbers of 0's and 1's are those of grids of graph primitive horizontally and vertically, respectively. If there is a k numbers of 1s, then there are $\binom{p}{k}$ numbers of paths, which can be obtained by enumerating lexicographically ordered positions of 1s. Hence, the number of all paths in a knot span for a curve of degree p is

$$\sum_{j=w-p}^{w} \frac{p!}{(w-j)!(p-w+j)!} = \sum_{k=0}^{p} \frac{p!}{k!(p-k)!} = \sum_{k=0}^{p} \binom{p}{k} = 2^p \tag{8}$$

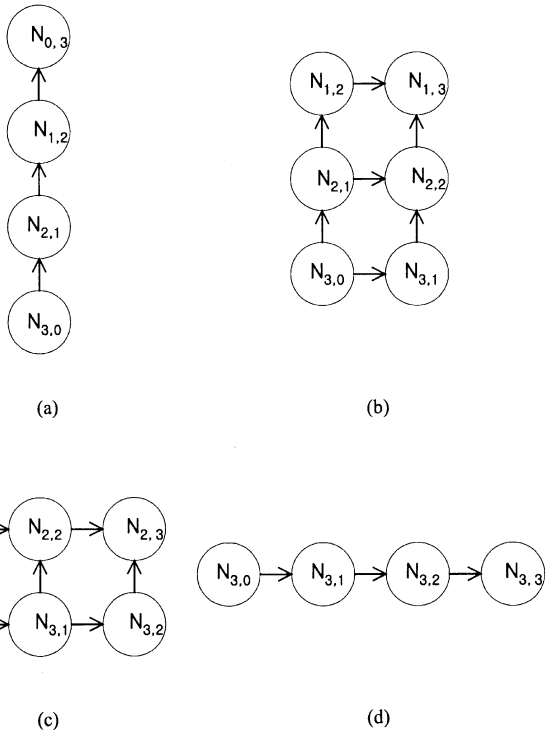

$$\begin{tikzcd} & & & N_{3,0} \arrow[r] & N_{3,1} \arrow[r] & N_{3,2} \arrow[r] & N_{3,3} \end{tikzcd}$$

Fig. 3. Four graph primitives for Fig. 2.

3. Direct expansion and experiments for static curves

Once all of the truncated basis functions are obtained in power form, the power form polynomial B-spline curve for $[t_i,t_{i+1})$ is given as

$$\mathbf{C}_i(t) = \sum_{j=i-p}^{i} \tau_{j,i}(t)\mathbf{P}_j \tag{9}$$

where $\tau_{j,i}(t)$ is a truncated basis function and $\mathbf{P}_j$ is the corresponding control point. Thus, if the previous operation is performed for each knot span, then a B-spline curve can be transformed into a set of piecewise polynomial curves in power form which can be formulated like the following.

$$\mathbf{C}(t) = \sum_{i=p}^{m-p-1} \mathbf{C}_i(t)N_{i,0} \tag{10}$$

Experiments are performed to compare the performance of the proposed DE algorithm with KR and TE-approaches. Illustrated in Fig. 4 is degree vs computation time with a fixed knot vector size of 50. It turns out that TE-approach is most efficient regardless of degree, and DE algorithm is superior to KR for degrees less than 7. Note that DE algorithm deteriorates quite rapidly compared to KR-approach. It is observed that DE shows a quadratic-like increase whereas KR and TE show only linear-like increases w.r.t. degrees. This is due to the fact that there are 2^p numbers of paths for calculating $(p+1)$ truncated basis functions in a knot span, when the degree of the curve is p. Thus, the computation time of truncated basis functions for each knot span doubles as the degree of curve increases. Fig. 5 shows that DE algorithm is comparable to TE-approach for degrees of three or four.

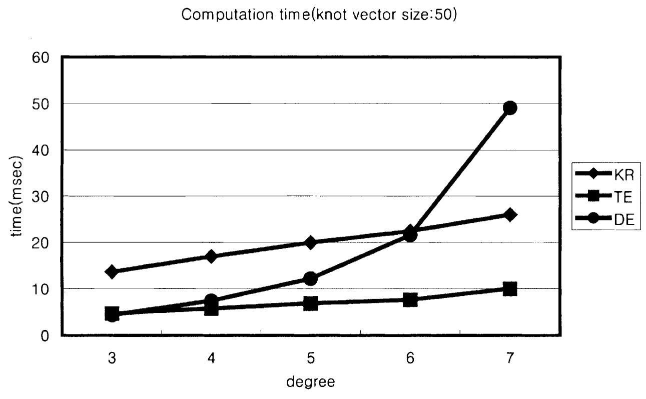


Fig. 4. Time vs degrees (knot vector size: 50).

4. Direct expansion of a dynamic curve

Definition 4. A *dynamic* B-spline curve, $\mathbf{C}_\mathrm{d}(t)$ is a B-spline curve with more than one control points moving. Thus, $\mathbf{C}_\mathrm{d}(t)$ can be represented by the following equation.

$$\mathbf{C}_d(t) = \sum_{i\in I} N_{i,p}\mathbf{P}_i + \sum_{j\in J} N_{j,p}\tilde{\mathbf{P}}_j \tag{11}$$

where I and J are the index sets of fixed control points $\mathbf{P}_\mathrm{i}$ and moving control points $\tilde{\mathbf{P}}_j$, respectively. $N_{i,p}$ is the basis function.

A brute force approach to transform $\mathbf{C}_\mathrm{d}(t)$ to piecewise polynomials in power form could be to recalculate the curve segment in every knot span whenever some control points are moving. This method is obviously unsatisfactory since it wastes computing time for the knot spans with unchanged curve shape. Therefore, by locating knot spans of curve segments whose shapes are changed, the transformation can be done more efficiently. In the case of KR-approach, a knot refinement and a basis conversion are performed for all knot spans of curve segments whose shapes are changed by moving control points. Similarly, TE-approach can recalculate the coefficients of polynomial curves for the knot spans of curve segments whose shapes are changed. The derivative information and factorial evaluation are needed for each coefficient of the polynomial.

However, the computational behaviour of DE-algorithm is different. Regardless of whether a control point is moving or not, the truncated basis functions are fixed. It turns out that the computational gain of DE algorithm for a dynamic curve is more significant than that of others. Assume that a static curve, $\mathbf{C}(t)$, is provided as Eq. (10) through DE algorithm, as a pre-processing tool for a dynamic curve. Let $\mathbf{C}_\mathrm{d}(t)$ be a dynamic curve counterpart of $\mathbf{C}(t)$. $\mathbf{C}_\mathrm{d}(t)$ can be now rewritten as the following equation using difference vectors, starting at old control points and ending at new control points.

$$\mathbf{C}_d(t) = \sum_{k\in K} N_{k,p}\mathbf{P}_k + \sum_{j\in J} N_{j,p}\mathbf{D}_j \tag{12}$$

where $K \equiv I \bigcup J$. That is, $\mathbf{P}_k, k \in K$, is all control points of $\mathbf{C}(t)$, and $\mathbf{D}_j = (\tilde{\mathbf{P}}_j - \mathbf{P}_j)$ corresponds to the displacement of the moving control point. Thus, Eq. (12) means that $\mathbf{C}(t)$ can be obtained by the summation of original curve $\mathbf{C}(t)$ and difference vectors multiplied by the corresponding basis functions.

On the other hand, $\mathbf{C}_\mathrm{d}(t)$ can also be divided into two groups: the first group is the set of curve segments whose shapes are fixed, and the second is the set of curve segments whose shapes are changed by moving control points. Hence the following equation holds.

$$\mathbf{C}_d(t) = \sum_{m\in M} \mathbf{C}_m(t)N_{m,0} + \sum_{n\in N} \tilde{\mathbf{C}}_n(t)N_{n,0} \tag{13}$$

where M and N are index sets for knot spans of curve segments whose shapes are fixed and changed by moving control points, respectively. In addition, $\tilde{\mathbf{C}}_n(t)$ can be again rewritten by using truncated basis functions as follows since $\tilde{\mathbf{C}}_n(t)$ may also have both fixed and moving control points.

$$\tilde{\mathbf{C}}_n(t) = \sum_{q\in Q} \tau_{q,n}(t)\mathbf{P}_q + \sum_{r\in R} \tau_{r,n}(t)\tilde{\mathbf{P}}_r \tag{14}$$

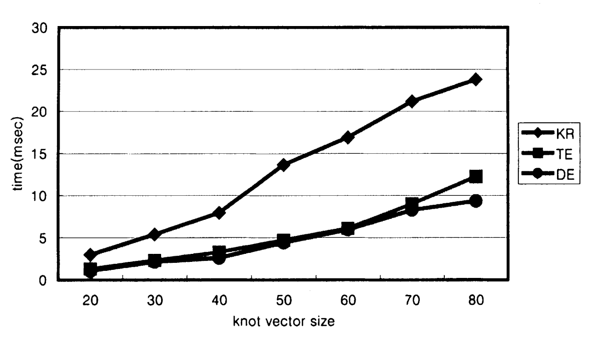
where Q and R are index sets for fixed and moving control points of $\tilde{\mathbf{C}}_n(t)$, respectively. Therefore,

$$\tilde{\mathbf{C}}_n(t) = \sum_{s\in S} \tau_{s,n}(t)\mathbf{P}_s + \sum_{r\in R} \tau_{r,n}(t)(\tilde{\mathbf{P}}_r - \mathbf{P}_r) \tag{15}$$

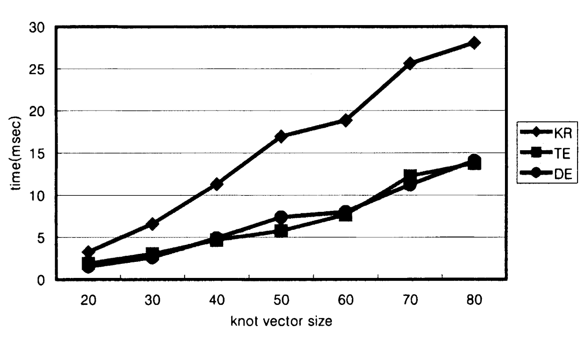
where $S \equiv Q \bigcup R$ and $|S| = p + 1$. $\mathbf{P}_s, s \in S$, is all the control points of $\mathbf{C}_n(t)$ before they move. Thus, Eq. (15) can be rewritten as Eq. (16) using difference vector and truncated basis function.

$$\tilde{\mathbf{C}}_n(t) = \mathbf{C}_n(t) + \sum_{r\in R} \tau_{r,n}(t)\mathbf{D}_r \tag{16}$$

where $\mathbf{D}_r = \tilde{\mathbf{P}}_r - \mathbf{P}_r$ is a difference vector whose value is the displacement of the moving control point. Thus, for a particular knot span, a changed curve segment in power



(a) degree: 3



(b) degree: 4

Fig. 5. Time vs different knot vector sizes.

form, $\tilde{\mathbf{C}}_n(t)$, can be obtained by summing the original polynomial curve $\mathbf{C}_n(t)$ in the form of Eq. (9) and difference vector multiplied by the corresponding truncated basis function. Performing the operation in Eq. (16) for the entire knot spans whose curve segments are changed by the moving control points completes the desired transformation for $\mathbf{C}_d(t)$.

5. Experiments for dynamic curves

Transformation of a dynamic curve into a set of piecewise polynomial curves in power form through DE algorithm consists of two steps: (i) pre-processing, and (ii) the operation of Eq. (16) for all the knot spans with changed curve segments.

Suppose one control point of B-spline curve of degree p has been moved. Then, the curve shape will be changed over $(p + 1)$ contiguous knot spans. Since $(p + 1)$ truncated basis functions for each knot span are fixed, the new power form polynomial curve can be obtained by (i) multiplying the difference vector, starting at the old control point and ending at the new control point, with corresponding truncated basis function, and (ii) adding the previous result, which is a polynomial in power form, to the old power form polynomial in the knot span. Repeating this operation for $(p + 1)$ knot spans will produce $(p + 1)$ pieces of new polynomials in power form. Note that this process is simple to program but very efficient in speed. There are $(p + 1)$ knot spans for the calculation, and each knot span takes only $(p + 1)$ multiplications and $(p + 1)$ additions. Therefore, only $O(p^2)$ operations are needed to get all the new polynomial curves since the calculation of difference vector takes only $O(1)$.

If KR is used, all of $(p + 1)$ curve segments, which will change their shapes, have to be recalculated. In other words, knot refinement for $(p + 1)$ knot spans and basis conversions for $(p + 1)$ coefficients for each knot span should be performed. Needless to say, this approach will take much more time than DE.

A similar observation can be made for TE. For each of $(p + 1)$ knot spans, new polynomial curves should be computed completely again. For a knot span, there are $(p + 1)$ numbers of coefficients to compute, and each coefficient needs the calculation of factorial function as well as derivative information. Usually, a derivative for a B-spline curve is calculated by evaluating the hodograph counterpart at an appropriate parameter value, which is again an operation for $O(p)$ if the evaluation is done based on Horner's rule. Otherwise, it is also $O(p^2)$ operation. Factorial function evaluation also takes $O(p)$ operation. Therefore, TE approach is either $O(p^3)$ or $O(p^4)$. However, to have $O(p^3)$ time behaviour, the hodograph itself should be a polynomial in power form, again.

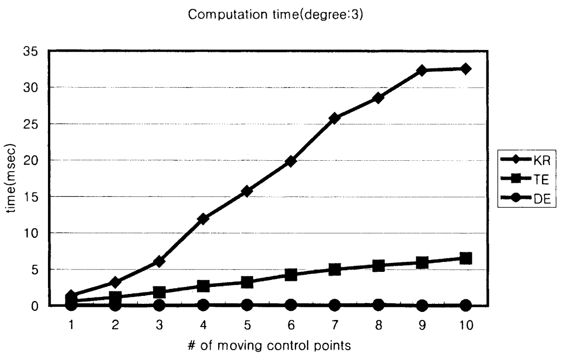
Fig. 6 illustrates experiment results for the degree of 3, 4, 5, and 6. The x-axis is the number of control points with changed coordinate values. The y-axis is the computation time taken by the re-calculation of new polynomial segments due to the moved control points. Brief descriptions of implementation for two approaches are as follows: for KR-approach, after locating the knot spans whose curve segments are changed by the moving control points, knot refinement is performed on each knot span and is followed by matrix multiplication for power basis conversion. The first stage of TE-approach is to get p hodographs of the new B-spline curve to calculate coefficients of polynomial terms. Then, after locating the knot spans whose curve segments are changed by the moving control points, piecewise polynomials in power form can be obtained through evaluating the hodograph at a particular point of each knot span. For each approach, we believe that only the minimum necessary arithmetic is performed in our experiment, even though we do not ignore the possible variations due to the implementation details.

What is shown in the figure is surprising. The time taken by the proposed DE algorithm is ignorable compared to the conventional KR and TE-approaches. For a fixed degree, the computational gain of DE algorithm gets more significant as the number of moved control points gets higher. As the degree of the curve increases, the gain increases, as was pointed earlier. Note that the scales of the y-axis are different in the figures. Since the time taken by DE algorithm is negligible compared to KR and TE-approaches, we provide Fig. 7 that shows the time behaviour of DE algorithm itself for dynamic curves. Note that it increases linearly as the number of moving control points increases.

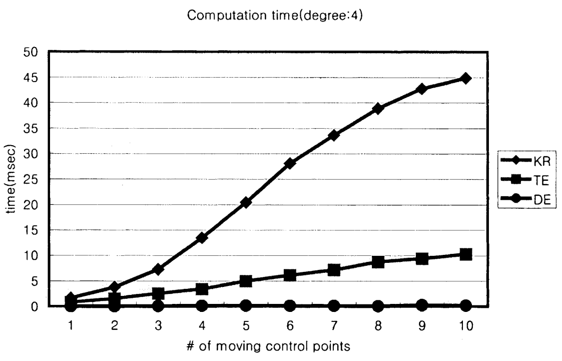
As was discussed in Section 1, there is another way to come up with the truncated basis functions using the derivative formula of B-spline basis function. In this approach, the coefficients of a truncated basis function are computed by the Taylor expansion of the derivatives of appropriate degree. In Fig. 8, the result of an experiment is provided for degrees of 3, 4, 5, and 6, to compare the performances of both approaches. In the figure, x- and y-axes represent the knot vector size and the computation time, respectively. In the figure, it is shown that the proposed approach is much faster, for all degrees, than the approach using the derivative of B-spline basis functions. The figure also shows that the computational gain increases as the degree and the size of knot vector increase.

6. Conclusions

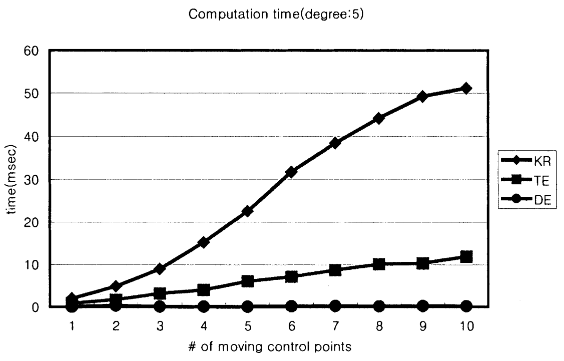
In computer graphics and CAGD, it is often necessary to manipulate B-spline curves or surfaces by converting the B-spline representation into a set of piecewise polynomial curves or surfaces in power form. In this paper, a new algorithm for converting a B-spline curve into piecewise polynomial curves in power form is presented. By defining several new concepts such as truncated basis



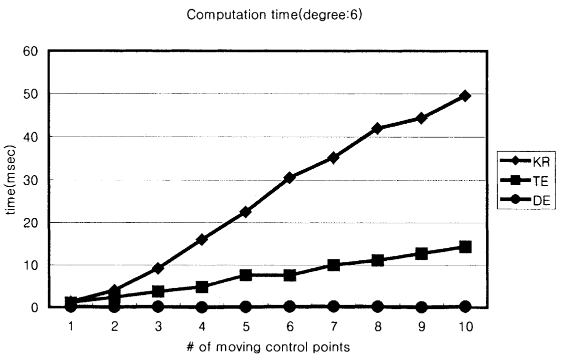
(a) degree: 3



(b) degree: 4



(c) degree: 5



(d). degree: 6

Fig. 6. Computation time vs the number of moved control points (the number of all control points of given B-spline curve is 10).

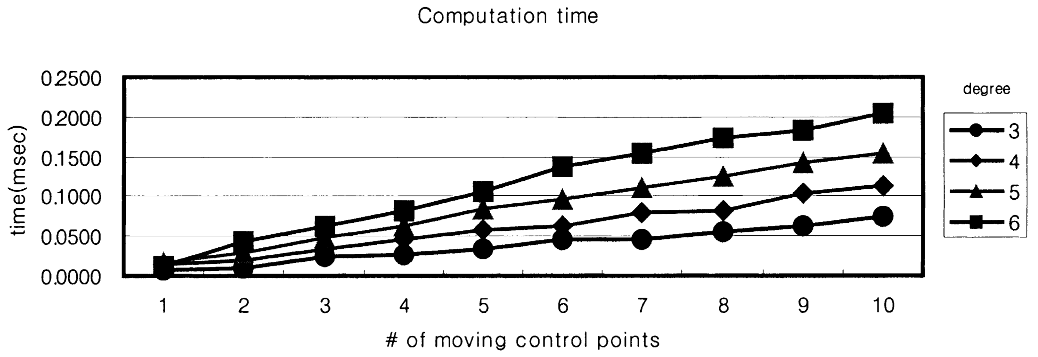


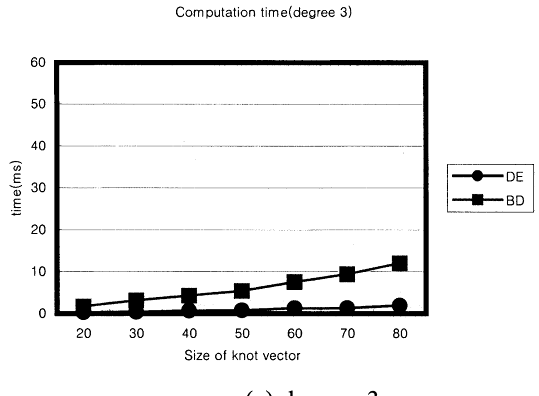
Fig. 7. Computation time vs the number of moving control points for degree 3, 4, 5 and 6.

functions, directed edges, and so on, the theoretical basis is provided.

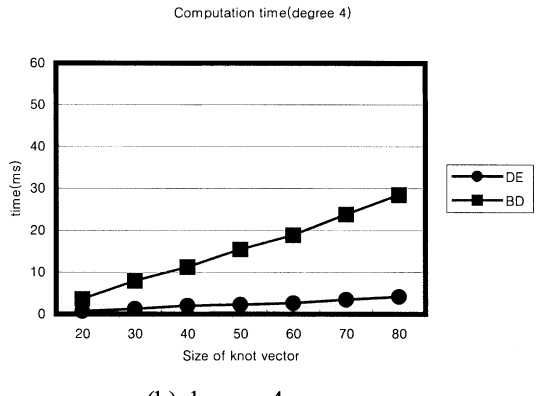
When a curve is dynamically changing its shape, the speed of computation becomes rather important. In this case, the theoretical analysis and experiment results show that DE algorithm is computationally very efficient compared to the existing approaches. It is our expectation that a similar idea can be easily extended to B-spline surfaces, and our approach will show more significant computational properties for the problem.

Acknowledgements

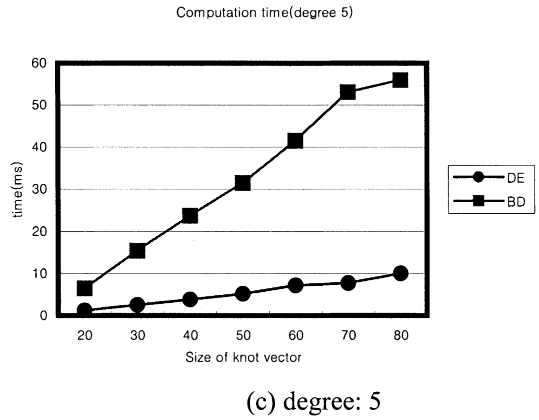
This work was supported in part by the Korea Science and Engineering Foundation (KOSEF) through the Ceramic



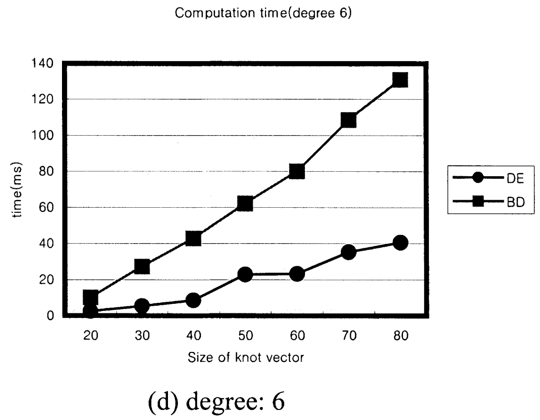
(a) degree: 3



(b) degree: 4



(c) degree: 5



(d) degree: 6

Fig. 8. Computation time of truncated basis functions for degree 3, 4, 5 and 6.

Processing Research Center (CPRC) at Hanyang University. The first author also thanks to A. Fischer, Technion-Israle Institute of Technology, for suggesting the possibility of applying the idea to dynamic B-spline curves and surfaces.

References

[1] Farin G. Curves and surfaces for computer-aided geometric design. 3rd ed. Academic Press, 1997.
[2] Kim D-S, Lee S-W, Shin H. A cocktail algorithm for planar Bezier curve intersections. Computer Aided Design 1998;30(13):1047–51.
[3] The Initial Graphics Exchange Specification (IGES), Version 5.2, ANSI Y14.26M, 1993.
[4] Bloomenthal J. Introduction to implicit surfaces. Morgan Kaufmann Publishers, Inc, 1997.
[5] Sederberg TW. Implicit and parametric curves and surfaces for computer aided geometric design, PhD Thesis, Purdue University, 1983.
[6] Lee K. Principles of CAD/CAM/CAE systems. Addison-Wesley, 1999.
[7] Boehm W, Prautzsch H. The insertion algorithm. Computer-Aided Design 1985;12(4):58–9.
[8] Boehm W. On the efficiency of knot insertion algorithms. Computer Aided Geometric Design 1985;2(1–3):141–3.
[9] Cohen E, Lyche T, Riesenfeld R. Discrete B-splines and subdivision techniques in computer-aided geometric design and computer graphics. Computer Graphics and Image Processing 1980;14(2):87–111.
[10] Goldman RN. Blossomming and knot insertion algorithm for B-spline curves. Computer Aided Geometric Design 1990;7:69–81.
[11] Piegl L, Tiller W. The NURBS book. Springer, 1995.
[12] Lasser D, Hoschek J. Fundamentals of computer aided geometric design. A. K. Peters, 1993.

Deok-Soo Kim is an associate professor in the Department of Industrial Engineering, Hanyang University, Korea. Before he joined the university in 1995, he worked at Applicon, USA, and Samsung Advanced Institute of Technology, Korea. He received a B.S. from Hanyang University, Korea, an M.S. from the New Jersey Institute of Technology, USA, and a Ph.D. from The University of Michigan, USA, in 1982, 1985 and 1990, respectively. His current research interests are in the streaming of 3D shapes on Internet, computational geometry, and geometric modeling and its applications.

Joonghyun Ryu received both BS and MS from the Department of Industrial Engineering, Hanyang University, Korea in 1997 and 1999, respectively. He is currently enrolled in PhD program. His research interests are in geometric modeling, computational geometry, and optimization.

Hyun Chan Lee received a BS degree from Seoul National University in 1978, an MS degree from KAIST in 1980, and the PhD degree in Industrial and Operations Engineering from the University of Michigan in 1988. Before he joined the University of Michigan, he worked for the Pusan Steel Pipe Inc. In 1988, he joined the Korea Electronics and Telecommunications Research Institute as a head of design automation section. He is now an associate professor of Hongik University, Seoul, Korea, in the department of Information and Industrial Engineering. His research interests include CAD/CAM, surface modeling, computer graphics, computational geometry, engineering database, and product information management.

Hayong Shin is an assistant professor in the Department of Industrial Engineering at KAIST (Korea Advanced Institute of Science and Technology). Before joining KAIST, he worked for DaimlerChrysler Corp., Cubic-Tek Co. and LG Electronics, developing commercial and in-house CAD/CAM software. He received a BS from Seoul National University in 1985, an MS and a PhD from KAIST in 1987 and 1991, all in industrial engineering. His main research interests are in the area of geometric modeling, tool path generation, process planning, and computational geometry. He can be reached at hyshin@mail.kaist.ac.kr