# Transformation of a Dynamic B-spline Curve Into Piecewise Power Basis Representation

Joonghyun Ryu[1], Youngsong Cho[1] and Deok-Soo Kim[1]

[1] Department of Industrial Engineering, Hanyang University
17 Haengdang-Dong, Seongdong-Ku, Seoul, 133-791, South Korea
dskim@hanyang.ac.kr

**Abstract.** In the fields of computer aided geometric design and computer graphics, B-spline curves and surfaces are often adopted as a geometric modelling tool and their evaluation is frequently required for a various geometric processing. In this paper, we present a new algorithm to convert B-spline curves into piecewise polynomials in power form. The proposed algorithm considers recursive B-spline basis function to be a pile of linear functions and collects all the necessary linear functions in each knot span. Then, algorithm computes the power form representation of B-spline basis functions and the required transformation of B-spline curve is obtained through the linear combination of B-splines in power form and the corresponding control points in each knot span.

## 1. Introduction

In the applications of computer graphics and computer aided geometric design, shapes are often modeled in terms of freeform curves and surfaces represented in B-spline form and their evaluation is frequently required for a various geometric processing. A B-spline curve of degree p with (m + 1) knots is defined by

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

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

$$N_{i,p}(t) = \frac{t - \mathrm{t}_i}{\mathrm{t}_{i+p} - \mathrm{t}_i} N_{i,p-1}(t) + \frac{\mathrm{t}_{i+p+1} - t}{\mathrm{t}_{i+p+1} - \mathrm{t}_{i+1}} N_{i+1,p-1}(t) \tag{2}$$

$$N_{i,0}(t) = \begin{cases} 1 & if \quad \mathrm{t}_i \le t < \mathrm{t}_{i+1} \\ 0 & otherwise \end{cases}$$

where $i$ = 0, 1, ..., $m$-$p$-1. Equation (2) shows that the evaluation of $\mathbf{C}(t)$ requires inevitably a recursive function evaluation which makes the evaluation slower. Even though a faster implementation in a non-recursive form may exist for periodic curve [2][3], a further reduction of the computation time is desirable especially when the B-spline curve changes its shape continuously by moving some of the control points.

Once a curve or surface is represented in power form, a point evaluation can be made faster due to Horner's rule even though the issue of numerical stability remains [4]. In this paper, we propose a faster algorithm for the evaluation of a B-spline curve based on the conversion of curve into a piecewise polynomial in a power form. 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 [5]. In addition, IGES supports a free-form curve as a piecewise polynomial in power form with an entity type 112 [6]. 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 to an implicit form [7]. The implicitization process, which uses a resultant, usually requires the curve to be represented in power form [8]. Since this operation is computationally demanding, the reduction of computation should not be ignored.

Discussed in this paper is the transformation of B-spline curve into a set of piecewise polynomials in power form, which is known to be a tedious task [9]. Especially, the focus of the paper is made on dynamic curves in the sense that one or more of the control points of the curves are moving. On the other hand, a curve with fixed control points is called static.

Since a static B-spline curve can be converted into a set of piecewise Bezier curves by a knot refinement [1][10][11][12][13], 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 required transformation can also be obtained through applying Taylor expansion of the B-spline at each knot span with the appropriate number of terms depending on the degree of the curve [14], which is denoted by the TE-approach in this paper.

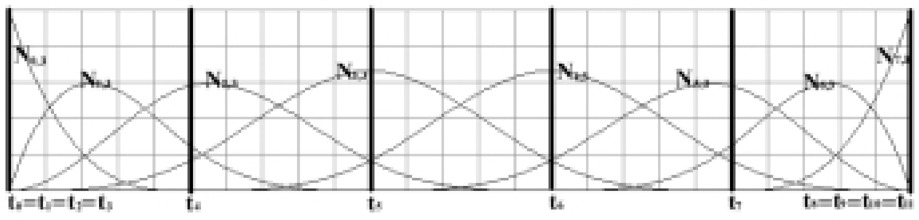
The main idea of the proposed algorithm, called a direct expansion (DE) algorithm, is as follows: after collecting all linear terms that make up the basis functions in a knot span, the algorithm directly obtains the power form representation of basis functions in the knot span by expanding the summation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

## 2. Direct Expansion of A Static Curve

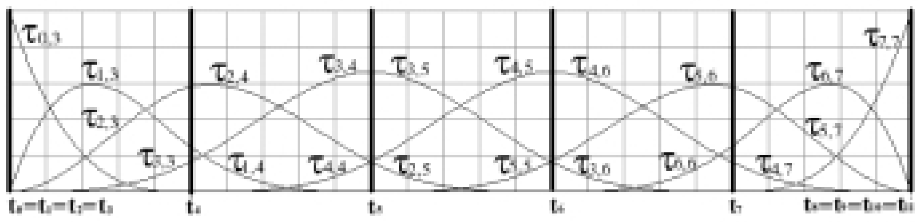
Definition 1. A truncated basis function, $\tau_{i,w}(t)$ , $i$=$w$-$p$,$w$-$p$+1,..,$w$, $w$=$p$,$p$+1,..,$m$-$p$-1, be an active polynomial segment among $N_{i,p}$'s, in $[t_w, t_{w+1})$.

Figure 1a shows cubic B-splines with $m$ = 11, and Figure 1b illustrates the corresponding truncated basis functions. 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 well-known B-spline basis function. For example, there exist four truncated basis functions $\tau_{0,3}(t)$, $\tau_{1,3}(t)$, $\tau_{2,3}(t)$ and $\tau_{3,3}(t)$ in $[t_3, t_4)$and $N_{3,3}$ consists of four truncated basis functions, $\tau_{3,3}(t)$ , $\tau_{3,4}(t)$ , $\tau_{3,5}(t)$ and $\tau_{3,6}(t)$ . Thus, B-spline basis function $N_{i,p}$ can be represented by the following equation.

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



a



b

**Fig. 1.** Cubic basis splines and truncated basis functions with five knot spans of nonzero length

The truncated basis function in power form can be computed summation of products among appropriate linear terms, which can be obtained by Equation (2) and the enumeration of 0-1 sequences [15]. Once all of the truncated basis functions are computed, 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, the power form polynomial curve for $\left[t_i, t_{i+1}\right)$ is given as

$$\mathbf{C}_i(t) = \sum_{j=i-p}^{i} \tau_{j,i}(t)\mathbf{P}_j \tag{4}$$

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

$$\mathbf{C}(t) = \sum_{i=p}^{m-p-1} \mathbf{C}_i(t) N_{i,0}(t) \tag{5}$$

The above equations show that the computation required is not less than the KR or TE. In fact, experimental result shows that TE-approach is the fastest independent of the degree of curve. Especially, DE shows a quadratic-like increase whereas KR and TE show only linear-like increases w.r.t. degrees of static B-spline curves. However, it is quite different for dynamic curves.

## 3. Direct Expansion of A Dynamic Curve

Definition 2. A dynamic B-spline curve, $\mathbf{C}_\mathrm{d}(t)$, is a B-spline curve with more than one control point moving. Thus, $\mathbf{C}_\mathrm{d}(t)$ can be represented by the following equation.

$$\mathbf{C}_\mathrm{d}(t) = \sum_{i \in I} N_{i,p}(t)\mathbf{P}_i + \sum_{j \in J} N_{j,p}(t)\tilde{\mathbf{P}}_j \tag{6}$$

where $I$ and $J$ are the index sets of fixed control points, $\mathbf{P}_i$ and moving control points , $\tilde{\mathbf{P}}_j$, respectively.

A naive approach to transform $\mathbf{C}_\mathrm{d}(t)$ to piecewise polynomials in power form is 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. Let $\mathbf{C}(t)$ be a B-spline curve before any control point moves and $\mathbf{C}_\mathrm{d}(t)$ be a dynamic curve counterpart of $\mathbf{C}(t)$. Then, $\mathbf{C}_\mathrm{d}(t)$ can be now rewritten as the following equation using difference vectors, starting at old control points and ending at new control points.

$$\mathbf{C}_\mathrm{d}(t) = \sum_{k \in K} N_{k,p}(t)\mathbf{P}_k + \sum_{j \in J} N_{j,p}(t)\mathbf{D}_j \tag{7}$$

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, Equation (7) means that $\mathbf{C}_\mathrm{d}(t)$ can be obtained by the summation of original curve $\mathbf{C}(t)$ and difference vectors multiplied by the corresponding basis functions. It is required to detect knot spans that are affected by the second term of Equation (7) so that 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 behavior of DE-algorithm is quite different. Regardless of whether control points are 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 if that a static curve, $\mathbf{C}(t)$, is provided as Equation (5) through DE algorithm, as a pre-processing tool for a dynamic curve.

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.

$$\mathbf{C}_\mathrm{d}(t) = \sum_{m \in M} \mathbf{C}_m(t) N_{m,0}(t) + \sum_{n \in N} \tilde{\mathbf{C}}_n(t) N_{n,0}(t) \tag{8}$$

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.

$$\tilde{\mathbf{C}}_n(t) = \sum_{q \in Q} \tau_{q,n}(t) \mathbf{P}_q + \sum_{r \in R} \tau_{r,n}(t) \tilde{\mathbf{P}}_r \tag{9}$$

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

$$\tilde{\mathbf{C}}_n(t) = \sum_{s \in S} \tau_{s,n}(t) \mathbf{P}_s + \sum_{r \in R} \tau_{r,n}(t)(\tilde{\mathbf{P}}_r - \mathbf{P}_r) \tag{10}$$

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, Equation (10) can be rewritten as Equation (11) using difference vector and truncated basis function.

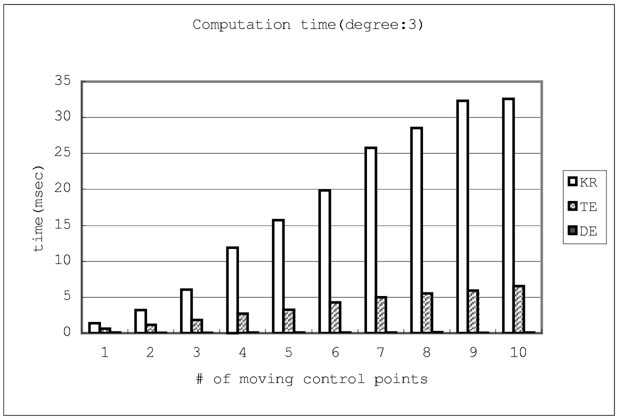
$$\tilde{\mathbf{C}}_n(t) = \mathbf{C}_n(t) + \sum_{r \in R} \tau_{r,n}(t)\mathbf{D}_r \tag{11}$$

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 form, $\tilde{\mathbf{C}}_n(t)$, can be obtained by summing the original polynomial $\mathbf{C}_n(t)$ in the form of Equation (4) and difference vector multiplied by the corresponding truncated basis function. Performing the operation in Equation (11) for the knot spans that are influenced by the moving control points completes the desired transformation for $\mathbf{C}_\mathrm{d}(t)$.

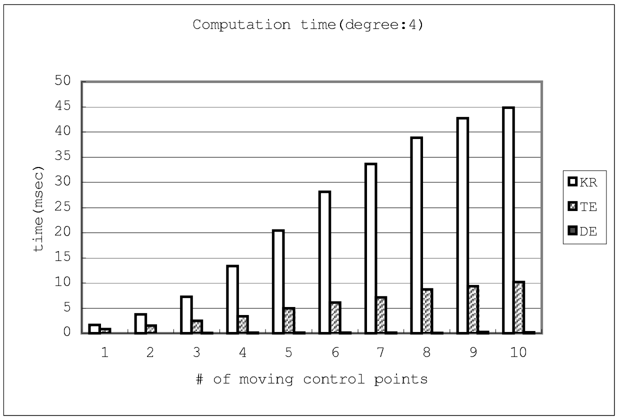
## 4. Experiments for Dynamic Curves

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 Equation (11) for all the knot spans with changed curve segments. While TE and KR-approach gets much more computational burdens, DE takes only ($p$ + 1) multiplications and ($p$ + 1) additions for a knot span. The computation time for each approach is provided in Figure 2 where pre-processing time is not considered. DE algorithm outperforms the other approaches and the computational gain of DE algorithm gets significant as the degree of curve and the number of control points increase. Since the relative time portion of DE algorithm is negligible, the trend of computation time of DE algorithm is provided separately in Figure 3. In each degree of curve, it seems that the computation time increases in linear pattern.

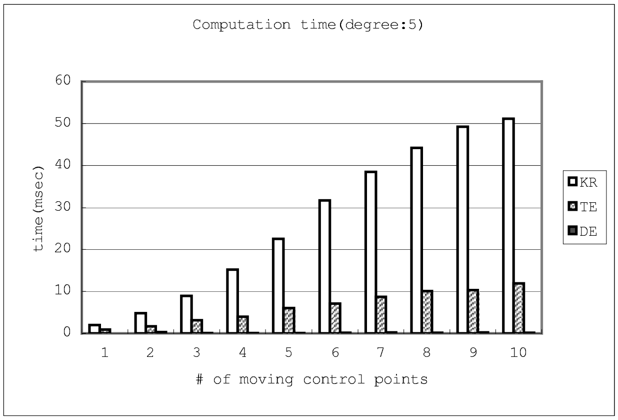
Although the implementation details may affect to the experimental results, we believe that our implementation considers the possible minimum operations for the KR and TE-approach.



a degree: 3



b degree: 4



c degree: 5

**Fig. 2.** Computation time vs. the number of moved control points

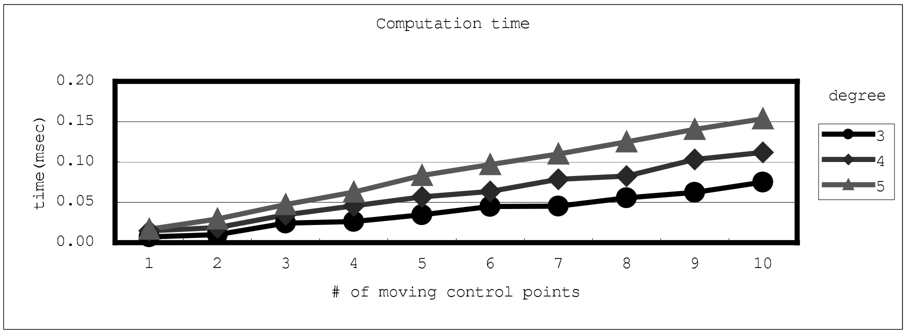


**Fig. 3.** Computation time vs. the number of moving control points for degree 3,4 and 5

## 5. Conclusions

In computer graphics and computer aided design, 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. It is 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. Once a curve is represented in power form, a point evaluation can also be made faster due to Horner's rule.

In this paper, a new algorithm for converting a B-spline curve to piecewise polynomial curves in power form is presented. We claim that the proposed algorithm outperforms the conventional KR-approach and is at least comparable with TE when the degree of the static curve is relatively low. When the curve is dynamically changing its shape, the speed of computation becomes rather important. In this case, experiments show that DE algorithm gets much more computational gain. 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.

In addition, the extensions of this algorithm to rational B-spline curves and surfaces are straightforward through the homogeneous coordinate.

## Acknowledgments

This work was supported by the Korea Science and Engineering Foundation (KOSEF) through the Ceramic Processing Research Center (CPRC) at Hanyang University.

## References

1. Piegl, L. and Tiller, W.: The NURBS Book. 2nd Ed. Springer (1995).
2. Yamaguchi, F.: Curves and Surfaces in Computer Aided Geometirc Design. Springer-Verlag (1988).
3. David F. Rogers: An introduction to NURBS:With Historical Perspective. Morgan kaufmann publishers (2000).
4. Farin, G.: Curves and Surfaces for Computer-Aided Geometric Design. 3rd Ed. Academic Press (1997).
5. Kim, D.-S., Lee, S.-W. and Shin, H.: A cocktail algorithm for planar Bezier curve intersections, Computer Aided Design, Vol. 30, No. 13, pp.1047-1051 (1998).
6. The Initial Graphics Exchange Specification (IGES). Version 5.2, ANSI Y14.26M (1993).
7. Bloomenthal, J.: Introduction to implicit surfaces. Morgan Kaufmann Publishers (1997).
8. Sederberg, T. W.: Implicit and parametric curves and surfaces for computer aided geometric design, Ph. D. Thesis. Purdue University (1983).
9. Lee, K.: Principles of CAD/CAM/CAE Systems. Addison-Wesley (1999).
10. Boehm, W. and Prautzsch, H.: The insertion algirthm, Computer-Aided Design, Vol. 12, No. 4, July, pp.58-59 (1985).

11. Boehm, W.: On the efficiency of knot insertion algorithms, Computer Aided Geometric Design, Vol. 2, Nos. 1-3, July, pp.141-143 (1985).
12. Cohen, E., Lyche, T. and Riesenfeld, R.: Discrete B-splines and subdivision techniques in computer-aided geometric design and computer graphics, Computer Graphics and Image Processing Vol. 14, No. 2, pp.87-111 (1980).
13. Goldman, R.N.: Blossomming and knot insertion algorithm for B-spline curves, Computer Aided Geometric Design, Vol.7, pp.69-81 (1990).
14. Lasser, D. and Hoschek, J.: Fundamentals of Computer Aided Geometric Design. A. K. Peters (1993).
15. Kim, D.-S., Ryu, J., Jang, T., Lee, H., and Shin, H.: Conversion of a B-spline curve into a set of piecewise polynomial curves in a power form, Korea Israel Bi-National Conference on Geometric Modeling and Computer Graphics in the World Wide Web Era, pp.195-201, Seoul, Korea, 1999.