Li X, Huang ZJ, Liu Z. A geometric approach for multi-degree spline. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 27(4): 841–850 July 2012. DOI 10.1007/s11390-012-1268-2

A Geometric Approach for Multi-Degree Spline

Xin Li[1] (李　新), Zhang-Jin Huang[2] (黄章进), and Zhao Liu[1] (刘　昭)

[1]*School of Mathematical Science, University of Science and Technology of China, Hefei 230026, China*

[2]*School of Computer Science, University of Science and Technology of China, Hefei 230026, China*

E-mail: {lixustc; zhuang}@ustc.edu.cn; zhaoliu@mail.ustc.edu.cn

Received January 17, 2011; revised May 2, 2012.

Abstract Multi-degree spline (MD-spline for short) is a generalization of B-spline which comprises of polynomial segments of various degrees. The present paper provides a new definition for MD-spline curves in a geometric intuitive way based on an efficient and simple evaluation algorithm. MD-spline curves maintain various desirable properties of B-spline curves, such as convex hull, local support and variation diminishing properties. They can also be refined exactly with knot insertion. The continuity between two adjacent segments with different degrees is at least C^1 and that between two adjacent segments of same degrees d is C^{d-1}. Benefited by the exact refinement algorithm, we also provide several operators for MD-spline curves, such as converting each curve segment into Bézier form, an efficient merging algorithm and a new curve subdivision scheme which allows different degrees for each segment.

Keywords spline, B-spline, multi-degree spline, merging

1 Introduction

Non-uniform rational B-splines (NURBS) have been used for modeling free-form curves or surfaces as an industrial standard. In computer-aided design (CAD) models, besides higher degree free-form patches, there are also lots of lower degree primitives, such as planes, spheres, cylinders and cones. In order to create a model with both higher degree free-form patches and lower degree primitives, one can represent the model with separate patches of different degrees. However the model is not mathematically watertight which will create gaps if we modify any part of the model. One can also represent each lower degree part with a higher degree. However it needs more control points and any modification to the primitives will make them no longer to be primitives. A better approach is to define a new spline which allows different degrees for each patch or segment. We call it "multi-degree spline", or "MD-spline". With MD-spline, it is possible to define a watertight arbitrary topology spline surface with different degrees. For the purposes, several important requirements for MD-splines include:

NURBS Compatible. MD-splines should be generalizations of B-splines, i.e., MD-splines specialize to B-splines in the case when all patches or segments are of the same degree;

Locally Constructed. To construct an arbitrary topology surface, one can use the subdivision surface[1] or patching with piecewise polynomial patches[2]. Both approaches locally construct the surface from a control grid with knot intervals. Thus for MD-spline, it is desirable to construct the MD-spline basis functions directly from the control grid using the local information (degrees and knot intervals);

Geometric Intuitive. MD-splines have a new parameter, degree, to modify the shape besides weights, knot intervals, and control points. So we should have a simple and intuitive way to assign and modify degrees.

The goal of this paper is to develop a simple approach to MD-spline curve, which is the first step for our ultimate goal. Given a control polygon with degrees and knot intervals, our algorithm can construct a smooth spline curve with the specified degrees.

In the following, we only focus on polynomial MD-spline curves since it is straightforward to generalize them to rational case. Note that when we consider rational form as its homogeneous one, there is not any difference between polynomial form and rational form. This is because polynomial curve is a curve with control points in $\mathbb{R}^3$, while rational form is with control points in $\mathbb{R}^4$.

Regular Paper

This work was supported by the National Natural Science Foundation of China under Grant Nos.11031007, 60903148, 60803066, the Chinese Universities Scientific Fund, the Scientific Research Foundation for the Returned Overseas Chinese Scholars of State Education Ministry of China, and the Startup Scientific Research Foundation of Chinese Academy of Sciences.

1.1 Prior Work

The splines with variable degrees were first constructed for shape-preserving interpolation in [3-5]. And later, a kind of two-degree-spline basis function is constructed in the process of degree elevation of B-spline curves in [6].

The very promising results for MD-splines were presented recently[7-8]. The authors constructed a set of basis functions for MD-spline spaces where continuity between two adjacent curve segments with degrees d_1 and d_2 could be $C^{\min(d_1,d_2)-1}$ or $C^{\min(d_1,d_2)}$ respectively.

Given a knot vector $\boldsymbol{T} = \{t_i\}$ and the degree sequence $G = \{d_i\}$ and let D be the maximal degree. For $n = 0; 1; \ldots; D$, they use an iterative process to generate a function sequence $\{N_{i,n}(t)\}_{i=-\infty}^{i=+\infty}$ over $\boldsymbol{T}$ and G. The final sequence $\{N_{i,n}(t)\}_{i=-\infty}^{i=+\infty}$ are the basis functions over $\boldsymbol{T}$ and G.

$$N_{i,n}(t) = \begin{cases} 0, & \text{if } d_i < D - n, \\ \begin{cases} 1, & \text{if } t \in [t_i, t_{i+1}), \\ 0, & \text{otherwise}, \end{cases} & \text{if } d_i = D - n, \\ \displaystyle\int_{-\infty}^{t} I_{i,n-1}(s)\mathrm{d}s, & \text{if } d_i > D - n. \end{cases} \tag{1}$$

Here

$$I_{i,n-1}(s) = \delta_{i,n-1} N_{i,n-1}(s) - \delta_{i+1,n-1} N_{i+1,n-1}(s),$$

and $\delta_{i,n} = (\int_{-\infty}^{+\infty} N_{i,n}(t)\mathrm{d}t)^{-1}$.

Noticed that in the construction, there are two global parameters D and $\delta_{i,n}$. Here $\delta_{i,n}$ is extremely difficult to compute with different degrees. Thus, one has to compute the coefficients $\delta_{i,n}$ before the definition of basis functions and any changes to the degrees need recompute the coefficients.

Moreover, we cannot apply the approaches in [7-8] to the input of a control polygon with degrees and knot intervals. This is because we should first define a spline space according to the degrees and knot intervals such that the dimension of the spline space is the number of control points. However, this is not always feasible. For example in Fig.1, the periodic curve includes four segments. However, the dimensions for the spline space defined in [7-8] are 8 and 4 respectively which are different from the number of control points, 5.

[9] provides an approach to MD-spline which has the same input as ours. It explicitly defines the Bézier control points for each segment according to the constraints on continuity and degrees. Since there is no general rule, the approach is restricted in degree one, two and three. Thus, the authors in [9] left several interesting questions for MD-splines on knot insertion and evaluation with recurrence relations.

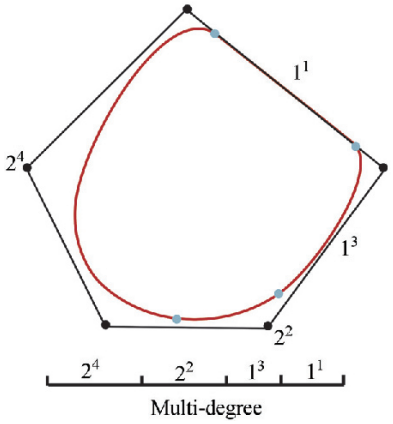


Fig.1. Knot interval for MD-spline curves.

For these reasons, we need find some new method to define MD-splines in order to achieve our ultimate goal.

1.2 Contribution

In the present paper, we provide a new definition by a de Boor-like evaluation algorithm. We prove that MD-spline defined in this way can be refined exactly with knot insertion. We also prove that the continuity between two adjacent segments with different degrees is at least C^1 and the continuity between two adjacent segments of same degrees d is C^{d-1}. MD-splines maintain many desirable properties of B-spline curves, such as convex hull, local support and variation diminishing properties. Compared with the approach in [9], our approach has several advantages.

- Our approach has an exact knot interval splitting algorithm, i.e., we can refine the curve exactly by knot insertion.
- Our approach is based on a de-Boor-like evaluation algorithm, however [9] has to convert each segment into Bézier form.
- Unlike the approach in [9], ours has no restriction on the degrees.

A Chinese character "Tian" is represented with an MD-spline of 29 control points in Fig.2 which includes 6 cubic curves segments, 22 quadratic curves segments and 11 linear curves segments. However, if we represent the font with B-splines, we need 87 control points, which is illustrated in Fig.2(b).

1.3 Overview

The remainder of the paper is constructed as follows. Section 2 provides some preparations for our construction. The de Boor-like evaluation algorithm for MD-spline is discussed in Section 3. And then we provide

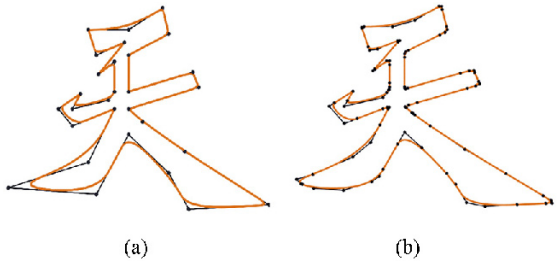


Fig.2. Chinese character "Tian"" with MD-splines (a) and B-splines (b) respectively.

knot insertion algorithm and apply the algorithm for Bézier extraction and knot removal. In Section 5, we provide some applications, including merging and MD-subdivision algorithm. The final section is the conclusion and future work.

2 Preparations

A knot interval[9] is the difference between two adjacent knots in a knot vector, which represents the parameter length of a curve segment. In conventional B-splines, knot intervals are assigned to vertices for even degree splines and assigned to edges of the control polygon for odd degree splines. Knot intervals are basically just an alternative notation for representing knot vectors, but knot interval notation is more closely coupled to the control polygon and has more geometric meaning. Fig.3 shows two simple knot interval examples for a quadratic curve and a cubic curve. As our ultimate goal is to construct watertight arbitrary topological surface, we use knot intervals instead of knot vectors.

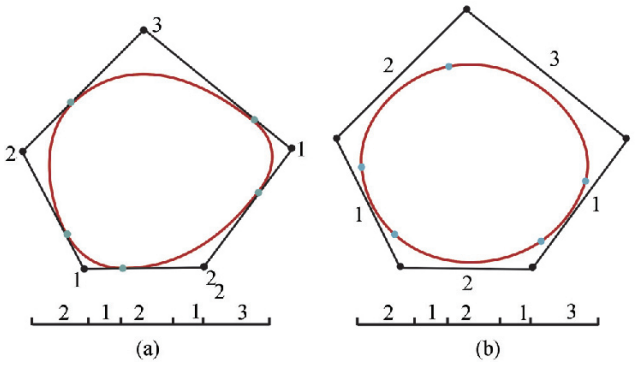


Fig.3. Knot interval for B-spline curves. (a) Even degree. (b) Odd degree.

Same as B-splines, each non-zero knot interval corresponds to a curve segment. But in MD-splines, curve segments are not required to all that have the same degree and so each knot interval carries a superscript which specifies the degree of its corresponding curve segment, see Fig.1.

Suppose $T = \{k_i\}_{i=0}^{m}$ is a non-negative real number sequence for knot interval and $D = \{d_i\}_{i=0}^{m}$ is a bounded positive integer sequence for degrees. We assume the vertices of the control polygon are $P_i, i = 0, \ldots, n$ and edges are E_i with two end points P_i and P_{i+1}. Before stating the construction, we define some terminology.

Definition 1 *The support edges of a degree d_i knot interval k_i are d_i connected edges. Denote the first index of the support edge as s_i for the i-th knot interval, then the support edges are $E_{s_i}, \ldots, E_{s_i+d_i-1}$. Here $s_i = j - \lfloor \frac{d}{2} \rfloor$, if the knot interval is assigned for vertex P_j (if d_i is even) or edge E_j (if d_i is odd). A knot interval is contribution interval of an edge if the edge is a support edge of the knot interval,*

The degree assignment for the MD-spline in the present paper is constrained by the following rules.

Rule for Degree Assignment. If k_b and $k_e, (b < e)$ are contribution intervals for edge E_i, then $k_j, j = b+1, \ldots, e-1$ are also the contribution intervals for the edge E_i.

The constraint may seem a little strange. In fact, we can illustrate this by a simple example in Fig.4. For two adjacent curve segments, the edge P_0P_1 is the support edge for the knot interval 2^4, and it should also be the support edge for all the inner knot intervals, such as 1^1. However, this is not the case for Fig.4(a). But if we assign the knot interval 1^1 to edge P_0P_1 such as that in Fig.4(b), then it is a valid one.

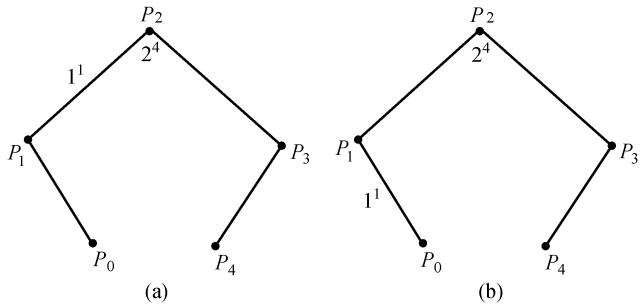


Fig.4. Explanation about the constraint. (a) Invalid assignment. (b) Valid assignment.

More precise, for two knot intervals k_i, k_{i+1},

1) If both d_i and d_{i+1} are even and k_i is assigned for vertex P_j and k_{i+1} is assigned for vertex P_k, then

$$2(j-k) \leqslant d_i - d_{i+1} \leqslant 2(k-j);$$

2) If d_i is even and d_{i+1} is odd, and k_i is assigned for vertex P_j and k_{i+1} is assigned for edge E_k, then

$$2(j-k) - 1 \leqslant d_i - d_{i+1} \leqslant 2(k-j) + 1;$$

3) If d_i is odd and d_{i+1} is even and k_i is assigned for edge E_j and k_{i+1} is assigned for vertex P_k, then

$$2(j-k) + 1 \leqslant d_i - d_{i+1} \leqslant 2(k-j) - 1;$$

4) If both d_i and d_{i+1} are odd and k_i is assigned for edge E_j and k_{i+1} is assigned for edge E_k, then

$$2(j-k) \leqslant d_i - d_{i+1} \leqslant 2(k-j).$$

In the rest of the paper, we assume the sequence T will always be a knot interval sequence and the sequence D will always be a degree sequence of T satisfying the constraint above. Suppose s_i is the index of the first support edge for the i-th knot interval, and the indices of the contribution intervals for edge E_i are from b_i to e_i ($b_i < e_i$), i.e., edge E_i is the support edge of the knot intervals $k_j, j = b_i + 1, \ldots, e_i$. Besides, we define a set of real parameters by $t_0 = 0$ and $t_j = \sum_{i=0}^{j} \frac{k_i}{d_i}$.

3 Evaluation Algorithm

An MD-spline curve in present paper is defined by a de Boor-like evaluation algorithm (shown in Fig.5) from the given control polygon with a specified degree sequence D and a knot interval sequence T.

Require: An MD-spline control polygon with degrees and knot intervals and a parameter $t, 0 \leqslant t \leqslant k_i$ for the i-th curve segment
Ensure: Point on MD-splines at t
1: $t \longleftarrow t_i + \dfrac{t}{d_i}$
2: **for** $j = 0$ to d_i **do**
3: $R_j^0 \longleftarrow P_{s_i+j}$
4: **end for**
5: **for** $j = 1$ to d_i **do**
6: **for** $k = 0$ to $d_i - j$ **do**
7: $beg = b_{s_i+j+k-1}$
8: $end = e_{s_i+k}$
9: $\delta = \dfrac{t - t_{beg}}{t_{end} - t_{beg}}$
10: $R_k^j = R_{k+1}^{j-1} \times \delta + R_k^{j-1} \times (1-\delta)$
11: **end for**
12: **end for**
13: **return** $R_0^{d_i}$

Fig.5. Evaluation algorithm for MD-splines.

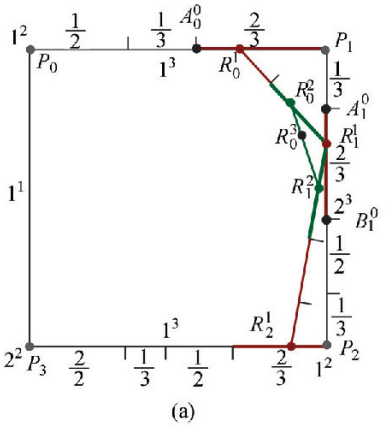
The key idea is that a degree d_i curve segment is contributed by $d_i + 1$ control points. We can specify these control points similar to B-splines. If the degree is even and assigned for vertex P_j or the degree is odd and assigned for edge E_j, then we specify the points to be $P_{s_i}, \ldots, P_{s_i+d_i}$, here $s_i = j - \lfloor \frac{d_i}{2} \rfloor$. And then we can compute the point on the curve segment at some parameter with d_i-level affine combination of these control points. The challenge of the approach is how to set the coefficients for affine combination such that we can smoothly connect all the curve segments and refine exactly by knot insertion.

We can see that our algorithm is very similar to traditional de Boor algorithm for B-spline except two main differences. First, the contribution intervals for each edge are different from those for B-splines which have fixed patterns. Second, the ratio for splitting each edge segment is different from that for B-spline, which is also considered with the degrees (t_j is associated with the degree for each curve segment).

Remark 1. *For each knot interval k_i, it is the contribution interval for edge $E_j, j = s_i, \ldots, s_i + d_i - 1$. For any $j = s_i, \ldots, s_i + d_i - 1$, $b_j < i \leqslant e_j$. Thus, in lines $7 \sim 10$ in the algorithm, the indices beg is always less than end.*

Remark 2. *If we let P_i be one and all the other control points be 0 and then we can define a blending function B_i which can be regarded as the i-th basis function for MD-spline (the linear independency of these blending functions will be proved in the next section in Theorem 2). According to the definition of MD-spline, the basis functions satisfy all the basic properties of B-spline, such as non-negative, partition of unity, local support and convex hull.*

In the following, we provide a simple example in Fig.6 to calculate the point at $t = \frac{2}{3}$ for the knot interval 2^3. The support edges for the knot interval are e_0, e_1 and e_2. And for edge e_0, the contribution intervals



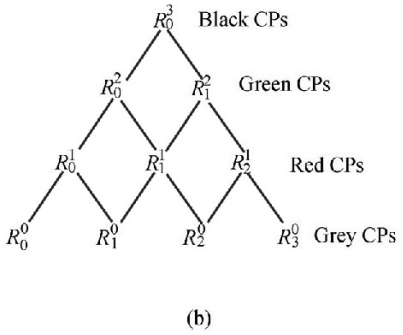


Fig.6. Illustration of the de Boor-like algorithm for the first example in Fig.8. (b) shows the pyramid diagram for an evaluation of cubic curve segment. CP: control point.

are 1^2, 1^3 and 2^3. So in the first iterator,

$$R_0^1 = \frac{\frac{2}{3}(1-t)}{\frac{1}{2}+\frac{1}{3}+\frac{2}{3}}P_0 + \frac{\frac{1}{2}+\frac{1}{3}+\frac{2}{3}t}{\frac{1}{2}+\frac{1}{3}+\frac{2}{3}}P_1 = \frac{8}{27}P_0 + \frac{19}{27}P_1.$$

And we can compute R_1^1 and R_2^1 in the similar fashion.

Then we can compute R_0^2, R_1^2 in the second iteration. For example, for R_0^2, the contribution intervals for the edge are 1^3 and 2^3, so

$$R_0^2 = \frac{\frac{2}{3}(1-t)}{\frac{1}{3}+\frac{2}{3}}R_0^1 + \frac{\frac{1}{3}+\frac{2}{3}t}{\frac{1}{3}+\frac{2}{3}}R_1^1 = \frac{2}{9}R_0^1 + \frac{7}{9}R_1^1.$$

In the third and last iterations,

$$R_0^3 = (1-t)R_0^2 + tR_1^2 = \frac{1}{3}R_0^2 + \frac{2}{3}R_1^2,$$

which is the point of the MD-spline curve at the parameter. The process is illustrated in Fig.6 with different colors on each level.

Fig.7 is one more font example of MD-spline with our new de Boor-like algorithm. The font includes seven cubic curve segments, nine quadratic curve segments and six linear curve segments.

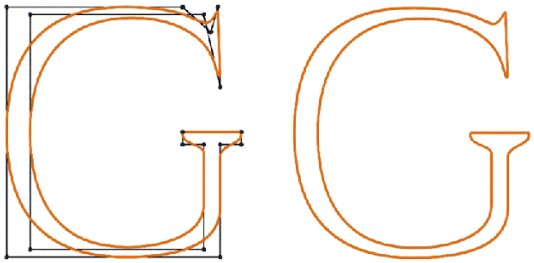


Fig.7. Font "G" is represented with MD-spline.

MD-splines have advantages of decreasing the number of control points for the same objects. Fig.8 shows more examples with MD-splines. The first one in Fig.8(a) is comprised of three cubic curve segments, three quadratic curve segments and one linear segment. The dots on the curves indicate junction points between adjacent curve segments. The second one shows an MD-spline comprised of three cubic segments and one linear segment. In this case, the cubic segments are C^1 with the linear segment and C^2 with the other cubic segments. The third one shows an MD-spline with four cubic segments and one quadratic curve segment. All pairs of neighboring curve segments are C^1 while neighboring cubic segments are C^2. The second and third rows show the corresponding representations with B-splines and Bézier form.

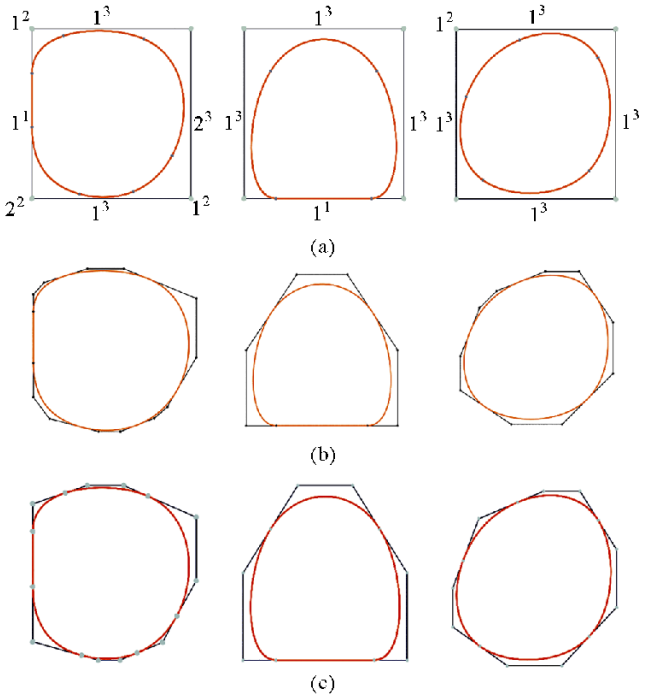


Fig.8. (a) MD-splines curves with four control points, where knot intervals are same as those in [9]. (b) Corresponding B-splines representation with 14, 8, 10 control points. (c) Corresponding Bézier representation.

4 Knot Insertion Algorithm

Knot insertion in terms of knot intervals can be thought of as splitting a knot interval into two knot intervals with the same degree. In this section, we provide the knot insertion algorithm (shown in Fig.9) for MD-spline which is very similar to that for traditional B-spline.

Require: An MD-spline control polygon with degrees and knot intervals and a parameter $t, 0 \leqslant t \leqslant k_i$ in the i-th curve segment
Ensure: A new MD-spline control polygon
1: Insert all the control points and assign the knot intervals and degrees
2: $t \longleftarrow t_i + \frac{t}{d_i}$
3: **for** $j = 0$ to $d_i - 1$ **do**
4: $\quad Q_{s_i+j} = P_{s_i+j} \times \frac{t-t_{b_j}}{t_{e_j}-t_{b_j}} + P_{s_i+j-1} \times \frac{t_{e_j}-t}{t_{e_j}-t_{b_j}}$
5: **end for**

Fig.9. Knot insertion algorithm for MD-splines.

Suppose we have an MD-spline curve with a control polygon P_j, $j = 0, \ldots, n$, a knot interval sequence T and a degree sequence D. We want to split knot interval k_i with degree d_i into two intervals t and $k_i - t$ for

$t \in [0, k_i]$ with both degree d_i. The algorithm is listed as below.

In line one, for each edge E_j, $j = s_i, \ldots, s_i + d_i - 1$, we insert new control points Q_j. If d_i is odd and is assigned for edge E_k, then we assign edge $Q_{k-1}Q_k$ knot interval t and assign edge Q_kQ_{k+1} knot interval $k_i - t$. If d_i is even and is assigned for edge P_k, then we assign vertex Q_{k-1} knot interval t and assign vertex Q_k with knot interval $k_i - t$. The other knot intervals and degrees are replicated from the origin control polygon.

We illustrate the algorithm with the example in Fig.10, in which we split the knot interval 2^3 into two knot intervals $t = \frac{2}{3}$ and $\frac{4}{3}$ with the same degree. In the example of Fig.10, the new control points are inserted on the $t = \frac{2}{3}$ of the way along each segment corresponding to knot intervals 2^3, such as Q_0, Q_1 and Q_2. Edges Q_0Q_1 and Q_1Q_2 will be assigned knot intervals $\frac{2}{3}$ and $\frac{4}{3}$ respectively. Fig.10(b) is the MD-spline curve with the new control polygon after knot insertion.

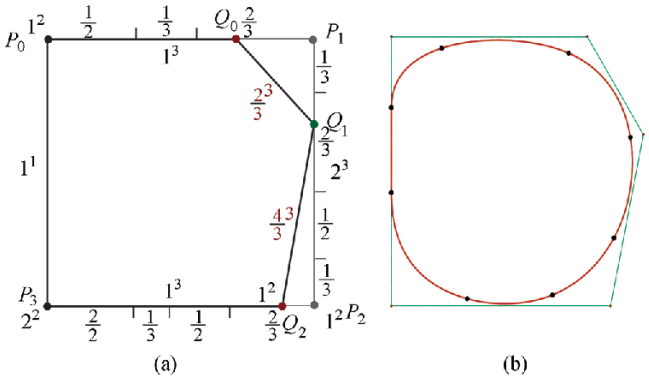


Fig.10. Illustration of exact refinement algorithm for the first example in Fig.8.

The following theorem proves that the MD-spline curves defined by the control polygons after and before knot insertion are identical curves. We call it exact refinement property.

Theorem 1 (Exact Refinement). *The curve derived from the control polygon after knot interval splitting algorithm is identical as the curve from the initial control polygon before refinement.*

Proof. Referring to Fig.11, suppose we split the k-th curve segment $P(u), u \in [0, k_k]$ into two curve segments with knot interval Δ and $k_k - \Delta$. Without loss of generalization, we assume the contribution control points for the curve segment are P_i^0, $i = 0, 1, \ldots, d_k$. Denote the curve segment associated with knot interval Δ is $Q(u), u \in [0, \Delta]$ and the associated control points are Q_i^0, $i = 0, \ldots, d_k$. Let P_i^j and Q_i^j, $i = 0, 1, \ldots, d_k - j$ be the control points generated from the de Boor-like algorithm in the above section when we evaluate $P(\delta)$ and $Q(\delta)$ $(\delta \leqslant \Delta)$ for curve segments $P(u)$ and $Q(u)$ respectively. Thus, what we need to prove is $P_0^{d_k} = Q_0^{d_k}$.

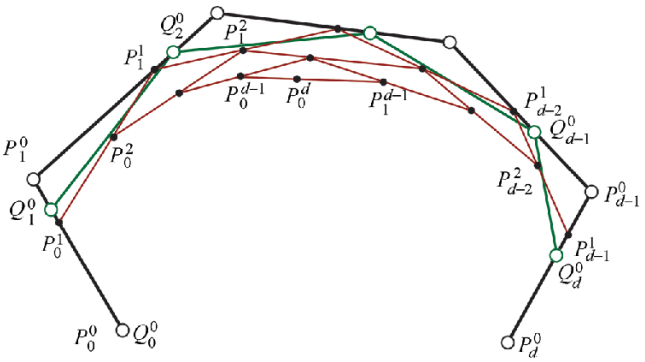


Fig.11. Notations for the control polygon evaluation algorithm and knot insertion algorithm.

Let $t = t_{k-1} + \frac{\delta}{d_k}$ and $t_q = t_{k-1} + \frac{\Delta}{d_k}$. According to the local refinement algorithm, we have

$$Q_j^0 = P_j^0 \times \frac{t_q - t_{b_j}}{t_{e_j} - t_{b_j}} + P_{j-1}^0 \times \frac{t_{e_j} - t_q}{t_{e_j} - t_{b_j}}. \tag{2}$$

According to the evaluation algorithm, for $i > 0$, the support edges for $P_i^{j-1}P_{i+1}^{j-1}$ and $Q_i^{j-1}Q_{i+1}^{j-1}$ are identical, so

$$P_i^j = P_{i+1}^{j-1} \times \frac{t - t_{b_{j+i-1}}}{t_{e_i} - t_{b_{j+i-1}}} + P_i^{j-1} \times \frac{t_{e_i} - t}{t_{e_i} - t_{b_{j+i-1}}}, \tag{3}$$

$$Q_i^j = Q_{i+1}^{j-1} \times \frac{t - t_{b_{j+i-1}}}{t_{e_i} - t_{b_{j+i-1}}} + Q_i^{j-1} \times \frac{t_{e_i} - t}{t_{e_i} - t_{b_{j+i-1}}}. \tag{4}$$

For $i = 0$, we have

$$\frac{|P_0^{l-1}P_0^l|}{|P_0^lP_1^{l-1}|} = \frac{t - t_{b_{l-1}}}{t_k - t}, \tag{5}$$

$$\frac{|Q_0^{l-1}Q_0^l|}{|Q_0^lQ_1^{l-1}|} = \frac{t - t_{b_{l-1}}}{t_q - t}. \tag{6}$$

First we prove the following lemma.

Lemma 1 *For any level $j, j = 0, 1, \ldots, d_k - 1$, P_i^j, Q_{i+1}^j, P_{i+1}^j and P_i^{j+1} are collinear and the lengths for edges $P_i^jP_i^{j+1}$, $P_i^{j+1}Q_{i+1}^j$ and $Q_{i+1}^jP_{i+1}^j$ are proportional to $t - t_{b_{j+i}} : t_q - t : t_{e_i} - t_q$.*

Proof. For $j = 0$, according to the evaluation algorithm in Section 3, it is obvious that P_i^0, Q_{i+1}^0, P_{i+1}^0 and P_i^1 are collinear and the lengths for edges $P_i^0P_i^1$, $P_i^1Q_{i+1}^0$ and $Q_{i+1}^0P_{i+1}^0$ are proportional to $t - t_{b_i} : t_q - t : t_{e_i} - t_q$.

Suppose the lemma is satisfied for level $j-1$, i.e., P_i^{j-1}, Q_{i+1}^{j-1}, P_{i+1}^{j-1} are collinear and the length for edges $P_i^{j-1}P_i^j$, $P_i^jQ_{i+1}^{j-1}$ and $Q_{i+1}^{j-1}P_{i+1}^{j-1}$ are proportional to $t - t_{b_{j+i-1}} : t_q - t : t_{e_i} - t_q$, as illustrated in Fig.12.

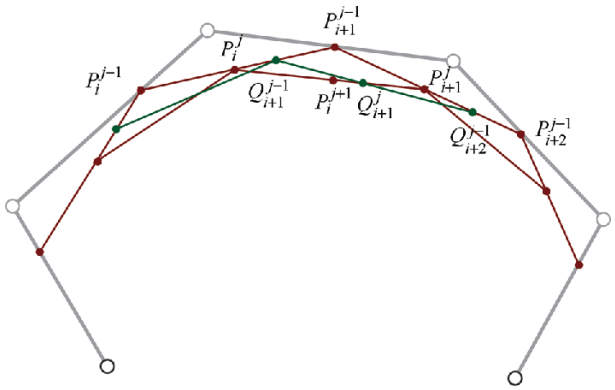


Fig.12. Illustration for the relation of P_i^j and Q_i^j during the evaluation process.

For level j, denote the intersection of edge $P_i^j P_{i+1}^j$ and edge $Q_{i+1}^{j-1}Q_{i+2}^{j-1}$ to be S_i^j, then we have the following equation according to Menelaus theory:

$$\frac{|Q_{i+2}^{j-1}S_i^j|}{|S_i^jQ_{i+1}^{j-1}|}\frac{|Q_{i+1}^{j-1}P_i^j|}{|P_i^jP_{i+1}^{j-1}|}\frac{|P_{i+1}^{j-1}P_{i+1}^j|}{|P_{i+1}^jQ_{i+2}^{j-1}|}=1. \tag{7}$$

Here $|*|$ is the length of the associated edge. And according to the assumption for level $j-1$ and (4),

$$\frac{|Q_{i+2}^{j-1}S_i^j|}{|S_i^jQ_{i+1}^{j-1}|}=\frac{t-t_{b_{j+i}}}{t_{e_{i+1}}-t}=\frac{|Q_{i+2}^{j-1}Q_{i+1}^j|}{|Q_{i+1}^jQ_{i+1}^{j-1}|}, \tag{8}$$

which leads to,

$$S_i^j=Q_{i+1}^j.$$

For level j, P_i^j, Q_{i+1}^j, P_{i+1}^j and P_i^{j+1} are also collinear.

And according to Menelaus theory,

$$\frac{|P_i^jQ_{i+1}^j|}{|Q_{i+1}^jP_{i+1}^j|}=\frac{|Q_{i+2}^{j-1}P_{i+1}^{j-1}|}{|P_{i+1}^jQ_{i+2}^{j-1}|}\frac{|Q_{i+1}^{j-1}P_i^j|}{|P_{i+1}^{j-1}Q_{i+1}^{j-1}|}=\frac{t_q-t_{b_i}}{t_{e_i}-t_q}.$$

Combining (3) for the computation of P_i^{j+1} from P_i^j and P_{i+1}^j, the lengths of edges $P_i^jP_i^{j+1}$, $P_i^{j+1}Q_{i+1}^j$ and $Q_{i+1}^jP_{i+1}^j$ are proportional to $t-t_{b_i}:t_q-t:t_{e_i}-t_q$. Thus, we complete the proof. □

Now we prove the theorem by the following lemma.

Lemma 2. *For any level $j, j=0,1,\ldots,d_k$, $P_0^j=Q_0^j$.*

Proof. For $j=0$ and $j=1$, it is evident that $P_0^j=Q_0^j$. Suppose $P_0^j=Q_0^j$ for $j<l$. According to Lemma 1, we have

$$\frac{|Q_0^lP_0^{l-1}|}{|Q_0^lP_1^{l-1}|}=\frac{t_q-t_{b_{l-1}}}{t_k-t_q}. \tag{9}$$

Combining (9), (5) and (6), we have $Q_0^l=P_0^l$.

Let $j=d_k$ in Lemma 2, we have $P_0^{d_k}=Q_0^{d_k}$. □

4.1 Extracting Bézier Form

An important special case of knot interval splitting algorithm involves inserting a zero knot interval. This is the same as inserting a double knot into the knot vector because the process is the same as splitting a knot interval into two intervals, one of which is zero, and the other of which is the original knot interval. If we repeat this operation several times, we uncover the Bézier control points for each curve segment. The reason for this will be clear if you recall that a degree n Bézier curve is a special case of B-Spline curve with knot vector $(\boldsymbol{a},\boldsymbol{a},\ldots,\boldsymbol{a},\boldsymbol{b},\boldsymbol{b},\ldots,\boldsymbol{b})$ which involves $n+1$ knot vector $\boldsymbol{a}$ and $n+1$ knot vector $\boldsymbol{b}$.

Fig.13 illustrates the process of extracting Bézier form for a cubic curve segment. Fig.13(a) shows the control polygon after splitting knot interval 2^3 into 0^3, 2^3 and 0^3 using the knot split algorithm. And the green points in Fig.13(b) are the Bézier control points for the cubic curve segment.

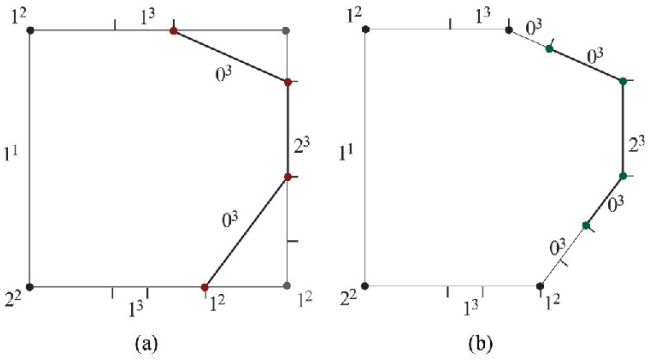


Fig.13. Illustration of extracting Bézier form for the cubic segment.

Theorem 2 (Linear Independency). *The blending functions B_i defined in Remark 2 are linear independent.*

Proof. Suppose $B_i, i=1,\ldots,n$ are n blending functions defined for a control polygon with degrees and knot intervals assignment and $\hat{B}_j, j=1,\ldots,n+1$ are $n+1$ functions defined for the control polygon with one knot interval k_t being split into two knot intervals using the knot insertion algorithm. First, we will prove that if $\hat{B}_j, j=1,\ldots,n+1$ are linear independent, then $B_i, i=1,\ldots,n$ are linear independent.

Let $\boldsymbol{B}$ and $\hat{\boldsymbol{B}}$ to be two vectors contains all B_i and $\hat{B}_j$ respectively, then there exist a matrix $\boldsymbol{M}=[m_{j,i}]$, $i=1,\ldots,n, j=1,\ldots,n+1$ such that $\boldsymbol{B}=\boldsymbol{M}\hat{\boldsymbol{B}}$. According to the knot insertion algorithm, all the elements of matrix $\boldsymbol{M}$ are 0 except the following three cases.

1) If $1\leqslant i\leqslant s_t$, then only $m_{i,i}\neq 1$;

2) If $s_t<i<s_t+d_t$, then only $m_{i,i}$ and $m_{i+1,i}$ are not 0;

3) If $i \geqslant s_t + d_t$, then only $m_{i+1,i} = 1$.

If there exist some constants λ_i such that $\lambda_i B_i = 0$. Denote $\boldsymbol{\Lambda}$ to be a vector contains all the λ_i, then we have $\boldsymbol{\Lambda M \hat{B}} = 0$. As $\hat{B}_j$ are linear independent, so we has been $\boldsymbol{\Lambda M} = 0$. From the structure of matrix $\boldsymbol{M}$ which has been discussed above, we can get that $\lambda_i = 0$. Thus B_i are linear independent.

For any MD-spline curve, we can convert it into Bézier form with multiple knots. After conversion, the functions defined from the Bézer control polygon are obvious linear independent because they are Bernstein polynomials in each interval. Thus, we can conclude that the blending functions derived from de Boor-like algorithm are also linear independent. □

Theorem 3 (Continuity). *The MD-spline defined in the present paper is at least C^1 between two adjacent curve segments with different degrees and C^{d-1} between two adjacent degree d curve segments.*

Proof. Referring to Fig.14, we first prove that two adjacent curve segments are at least C^1 between two segments with knot interval k_1^m and k_2^n, here $m \neq n$. As the knot insertion will not change the curve, we can insert several zero knot intervals such that the contribution knot intervals for edge P_0Q_0 are only k_1^m and k_2^n, such as that in Fig.14. Let $R = \frac{mk_2}{nk_1+mk_2}P_0 + \frac{nk_1}{nk_1+mk_2}Q_0$ to be a point on edge P_0Q_0. Then P_0, R and R, Q_0 are two Bézier end control points for the two curve segments with knot interval k_1^m and k_2^n respectively. It is obvious that the two curves are at least C^1.

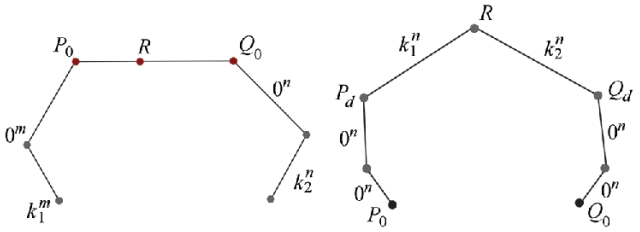


Fig.14. Illustration of continuity proof.

If two adjacent segments have same degree n, without loss of generalization, suppose $n = 2d + 1$ and the two knot intervals k_1 and k_2 are two adjacent knot intervals. As the knot insertion will not change the curve, we can extract all the curve segments into Bézier form except the curve segments for knot intervals k_1 and k_2, see Fig.14. Suppose after insertion, the control polygon is $P_0P_1 \ldots P_dRQ_d \ldots Q_1Q_0$. As the contribution intervals for the two curve segments are all of the same degrees, the definition for these curve segments are the same as B-splines with the same control polygon. Thus, they are C^{n-1} continuity. □

4.2 Knot Removal Algorithm

Knot removal algorithm is the inverse procedure of knot insertion algorithm. It is one of the key algorithms for spline which has been widely used in fitting, fairing, merging. Benefited by the exact refinement algorithm in above subsection, the knot removal algorithm is similar to that for B-spline.

Suppose we are given a control polygon P_i, $i = 0, 1, \ldots, n$ with the specified knot intervals T and degrees D. For knot interval, knot is removal means merging two adjacent knot intervals k_i and k_{i+1} with same degree into one knot interval $k_i + k_{i+1}$. After one knot is removed, we have a new control polygon Q_j, $j = 0, 1, \ldots, n-1$. Denote $\boldsymbol{P}$ and $\boldsymbol{Q}$ be two vectors which contain P_i and Q_j respectively. Using the algorithm for knot insertion, there exists a matrix $\boldsymbol{A}$ such that the control polygon $\boldsymbol{R} = [R_1, R_2, \ldots, R_n]^{\mathrm{T}} = \boldsymbol{AQ}^{\mathrm{T}}$ with the same knot intervals as P_i which defines the same MD-spline curve as that with control polygon Q_j. Knot removal algorithm will change the shape in general since the matrix $\boldsymbol{A}$ is over determined which preserves the shape if and only if P_i are identical to the control points R_i.

Now, there are lots of different possibilities to determine approximated solutions of the knot removal problem which is the same as knot removal algorithm for traditional B-spline. For more details of knot removal algorithm for B-spline, please refer [10]. What we use here is Least squares knot removal. Q_i are determined by minimizing $\sum_{i=0}^{n}(P_i - R_i)^2$, which leads to a linear system.

4.3 B-Spline Conversion

Using the Bézier extraction, we can easily to convert an MD-spline into the corresponding B-spline representation with multiple knots. First, we convert an MD-spline curve into Bézier form and the lower degree curve segments into the highest degrees. And then we remove unnecessary zero knot intervals using knot removal algorithm for B-splines. The B-splines representations in Fig.8 are the results of using the knot removal algorithm in [10].

5 Application

In this section, we provide some basic applications with MD-spline defined in Section 3, including an efficient merging algorithm and a curve subdivision scheme which allows different degrees to be assigned for the control polygon.

5.1 Merging with MD-Spline

The key application of MD-spline is that it can

merge two spline curves with different degrees into a single spline curve without degree elevation. There are two advantages for a single spline curve. One is that any modification of the curve will not change the degree for each segment and the other is that a single MD-spline curve has less control points.

The algorithm for curve merging is straightforward according to the knot removal algorithm. For example, given two MD-spline curves with Bézer end condition as Fig.15(a), for a C^0 merging, one end control point of two curves will be shared after merging. If the two end control points are distinct, then we need to perturb one of the curves or both curves.

For C^1 merging, we should remove the zero knot interval to make higher continuity. The perturbation is based on the perturbation of knot removal algorithm. Fig.15(b) is the result of C^0 merging and the result of C^1 merging is shown in Fig.15(c).

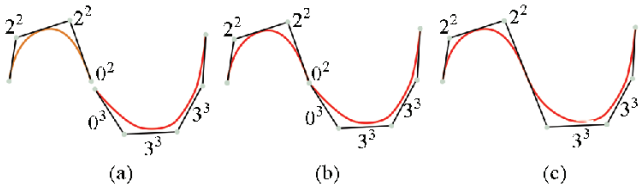


Fig.15. Simple merging example. (a) Two original curves. (b) C^0 merging. (c) C^1 merging.

5.2 MD-Subdivision

In this subsection, we provide a new type of curve subdivision scheme called MD-subdivision, which allows different degrees for curve segments based on the exact refinement algorithm. Given a control polygon with knot intervals and degrees assignment, MD-subdivision is a recurrence procedure to create a new control polygon with each knot interval being split into half. MD-subdivision is also a new scheme to combine the dual scheme (even degree) and primary scheme (odd degree). In this subsection, we only focus on the MD-subdivision algorithm for a polygon with degree 1, 2 and 3.

A subdivision scheme includes two phrases: the topological rules and the geometric rules. The topological rules for MD-subdivision scheme are based on the following two rules.

1) For each vertex with non-zero knot interval, split the control point into two new control points with the same knot interval and degree as the original control point.

2) For each edge with non-zero knot interval, insert one new control point into the edge and split the edge into two edges with the same degree and knot interval as the original edge;

The geometric rules for MD-subdivision have three different cases.

1) Geometric rule for edge points.

The edge point is the middle point of the line segment which corresponds to the knot interval assigned to the edge. Suppose edge $P_{i-1}P_i$ is assigned with knot interval k_k and the contribution intervals for the edge are $k_j, j = b_i+1, \ldots, e_i$, then the edge point Q_i for the edge is

$$Q_i = \frac{t_{e_i} - t_{\text{mid}}}{t_{e_i} - t_{b_i}} P_{i-1} + \frac{t_{\text{mid}} - t_{b_i}}{t_{e_i} - t_{b_i}} P_i,$$

here $t_{\text{mid}} = t_{k-1} + \frac{k_k}{2d_k}$.

2) Geometric rule for zero knot interval vertex.

Referring to Fig.16(a), the knot interval for vertex P_1 is zero, λ_i and μ_i are defined by the knot intervals and degree. Suppose knot interval for edge P_0P_1 is k^d, then $\lambda_1 = \frac{k}{d}$ and

$$\mu_1 = \begin{cases} \dfrac{k}{d}, & \text{if } d = 3, \\ 0, & \text{if } d = 1. \end{cases} \tag{10}$$

We can define λ_2 and μ_2 similarly. The new point R_i corresponding to the vertex is:

$$R_i = \frac{c_1 Q_{i-1} + c_1 P_i + c_2 Q_i}{c_1 + c_2 + c_3}. \tag{11}$$

Here $c_1 = (\mu_1 + \mu_2)\lambda_2$, $c_2 = (\lambda_1 + \lambda_2)(\mu_1 + \mu_2)$ and $c_3 = (\lambda_1 + \lambda_2)\mu_1$.

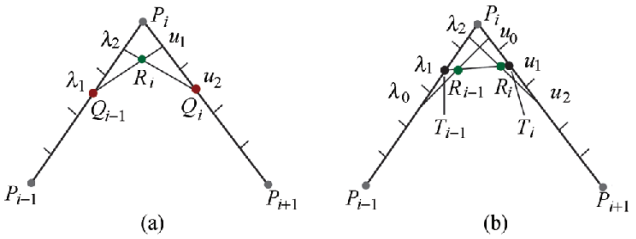


Fig.16. Illustration of MD-subdivision.

3) Geometric rule for non-zero knot interval vertex.

Referring to Fig.16(b), the knot interval for P_i is not zero. λ_j and μ_j are defined by the knot intervals similarly except λ_1 and μ_1 are defined by knot interval for control point P_i. Suppose T_{i-1} and T_i are the middle points of the segments which correspond to the knot interval assigned to P_i, then the two new points R_{i-1} and R_i corresponding to the vertex are:

$$R_{i-1} = \alpha T_{i-1} + (1-\alpha) T_i, \tag{12}$$

$$R_i = \beta T_i + (1-\beta) T_{i-1}. \tag{13}$$

Here

$$\alpha = \frac{(\lambda_0 + 2\lambda_1 + 2\lambda_2)(\mu_0 + \mu_1)}{(\lambda_0 + 2\lambda_1 + 2\lambda_2)(\mu_0 + \mu_1) + \mu_0(\lambda_0 + \lambda_1)},$$
$$\beta = \frac{(2\mu_0 + 2\mu_1 + \mu_2)(\lambda_1 + \lambda_2)}{(2\mu_0 + 2\mu_1 + \mu_2)(\lambda_1 + \lambda_2) + \lambda_2(\mu_1 + \mu_2)}.$$

Fig.17 is the result of the first six levels of subdivision for the first example in Fig.8.

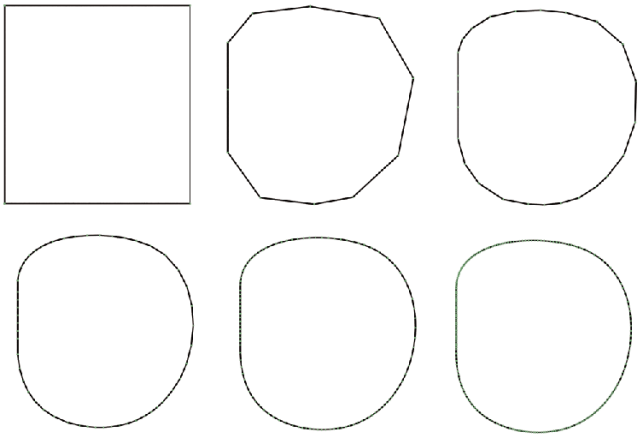


Fig.17. First six levels subdivision for the first example in Fig.8.

6 Conclusions and Future Work

This paper provides a new way to define MD-spline curves based on a de Boor-like evaluation algorithm for a control polygon with specified knot intervals and degrees. MD-spline curves defined in this way maintain many desirable properties of B-spline curves, such as convex hull, local support and variation diminishing. The MD-spline curves have less control points than B-splines for a model with different degrees. And the continuity between two adjacent segments with different degrees is at least C^1 and the continuity between two adjacent segments of same degrees d is C^{d-1}. We also apply MD-splines to merge B-spline curves with different degrees to a single spline curve and provide a new subdivision curve scheme which allows degree assignment with one, two and three.

The main weakness of the present paper is that the continuity between two adjacent segments with different degrees is C^1. How to generalize the idea to achieve higher continuity is one issue of our future work. Others include to generalize the idea to surface with arbitrary topology and to study general merging algorithm for MD-spline surface. How to generalize the MD-subdivision scheme to arbitrary degrees with unify rules is also an interesting problem.

References

[1] Sederberg T W, Zheng J M, Sewell D, Sabin M. Non-uniform recursive subdivision surfaces. In *Proc. the 25th SIGGRAPH*, Orlando, USA, July 1998, pp.387-394.

[2] Peters J. Patching Catmull-Clark meshes. In *Proc. the 27th SIGGRAPH*, New Orleans, Louisiana, USA, July 2000, pp.255-258.

[3] Kaklis P D, Pandelis D G. Convexity-preserving polynomial splines of non-uniform degree. *IMA Journal of Numerical Analysis*, 1990, 10(2): 223-234.

[4] Costantini P. Variable degree polynomial splines. In *Curves and Surfaces with Applications in CAGD*, Rabut C, Le Mehaute A, Schumaker L L (Eds.), Nashville: Vanderbilt University Press, 1997, pp.85-94.

[5] Costantini P. Curve and surface construction using variable degree polynomial splines. *Computer Aided Geometric Design*, 2000, 17(5): 419-446.

[6] Wang G Z, Deng C Y. On the degree elevation of B-spline curves and corner cutting. *Computer Aided Geometric Design*, 2007, 24(2): 90-98.

[7] Shen W Q, Wang G Z. A basis of multi-degree splines. *Computer Aided Geometric Design*. 2010, 27(1): 23-35.

[8] Shen W Q, Wang G Z. Changeable degree spline basis functions. *Journal of Computational and Applied Mathematics*, 2010, 234(8): 2516-2529.

[9] Sederberg T W, Zheng J M, Song X W. Knot intervals and multi-degree splines. *Computer Aided Geometric Design*, 2003, 20(7): 455-468.

[10] Lyche T, Morken K. Knot removal for parametric B-spline curves and surfaces. *Computer Aided Geometric Design*, 1987, 4(3): 217-230.

Xin Li was born in Hubei, China, in 1980. He received the B.S. and Ph.D. degrees in computational mathematics from University of Science and Technology of China (USTC), in 2002 and 2008, respectively. Currently he is an associate professor in the School of Mathematical Science, USTC. His research interests include computer aided geometric design and computer graphics.

Zhang-Jin Huang received the B.S. and Ph.D. degrees in computational mathematics from USTC, Hefei in 1999 and 2005, respectively. He is currently an associate professor in the School of Computer Science and Technology, USTC. His research interests include computer graphics, computer aided geometric design, and GPU parallel computing.

Zhao Liu received the B.Sc. degree from the University of Science and Technology of China in 2009. His research interests include computer aided geometric design.