Available online at www.sciencedirect.com
ScienceDirect
Computer-Aided Design ▮ (▮▮▮▮) ▮▮▮–▮▮▮
COMPUTER-AIDED DESIGN
www.elsevier.com/locate/cad

Approximate computation of curves on B-spline surfaces

Yi-Jun Yang[a,b,c,*], Song Cao[a,c], Jun-Hai Yong[a,c], Hui Zhang[a,c], Jean-Claude Paul[a,c,d], Jia-Guang Sun[a,b,c], He-jin Gu[e]

[a] *School of Software, Tsinghua University, Beijing, China*
[b] *Department of Computer Science and Tech., Tsinghua University, Beijing, China*
[c] *Key Laboratory for Information System Security, Ministry of Education, China*
[d] *INRIA, France*
[e] *Jiangxi Academy of Sciences, Nanchang, China*

Received 15 April 2007; accepted 22 October 2007

Abstract

Curves on surfaces play an important role in computer-aided geometric design. Because of the considerably high degree of exact curves on surfaces, approximation algorithms are preferred in CAD systems. To approximate the exact curve with a reasonably low degree curve which also lies completely on the B-spline surface, an algorithm is presented in this paper. The Hausdorff distance between the approximate curve and the exact curve is controlled under the user-specified distance tolerance. The approximate curve is ε_T–G^1 continuous, where ε_T is the user-specified angle tolerance. Examples are given to show the performance of our algorithm.
© 2007 Elsevier Ltd. All rights reserved.

Keywords: Approximation; B-spline; Curves on surfaces; Curve approximations

1. Introduction

Curves lying on free form surfaces play an important role in surface blending, surface–surface intersection, surface trimming and numerical control (NC) tool path generation for machining surfaces [17]. Particularly in surface blending, the blending surface connects the two base surfaces at the linkage curves (see Fig. 1). In this case, the linkage curves are defined by trimming curves on the base surfaces which rarely coincide with the iso-parameter lines. To ensure positional continuity between the blending and base surfaces, linkage curves are usually first computed in the parametric domain, and then represented as the mapping of the domain curves on the base surfaces [1,2,23].

An explicit and control point based representation of a curve on a surface is needed in many circumstances, such as using the curve as a boundary curve of another surface [17]. Also many geometric properties derived from the control polygon, like the

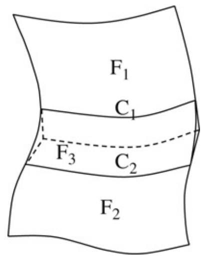


Fig. 1. Surface blending. $\mathbf{F}_1$ and $\mathbf{F}_2$ are two base surfaces, $\mathbf{F}_3$ is the blending surface while $\mathbf{C}_1$ and $\mathbf{C}_2$ are two linkage curves.

convex hull, can be computed directly. To compute the exact curve on a free form surface in control point representation, many algorithms [3,4,8–10,12,16,17] have been presented. However, the degrees of exact curves are considerably high, which results in computationally demanding evaluations and introduces numerical instability. Mapping a dth-degree B-spline curve onto a $p \times q$th-degree B-spline surface will result in a curve of degree $(p+q)d$.

To overcome the problems of the exact, explicit representation, many approximation algorithms have been presented [7,11,16–18,21,22]. Recently, Flöry [7] presented a B-spline

* Corresponding author at: School of Software, Tsinghua University, Beijing, China. Tel.: +86 0 10 62795442; fax: +86 0 10 62795460.
E-mail address: yangyijuny@gmail.com (Y.-J. Yang).

0010-4485/$ - see front matter © 2007 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cad.2007.10.011

fitting algorithm using squared distance minimization to approximate unordered points lying on a B-spline surface. Some constraints such as on-manifold, one-sided fitting or excluding regions can be imposed on the fitting curve. Instead of just simply approximating discrete points, Renner [17] utilized the full geometrical information contained in the exact representation to generate high quality curves. The Hausdorff distance between the approximate curve and the exact curve can be controlled under the user-specified tolerance.

To the authors' knowledge, all presented approximation algorithms [7,11,16–18,21,22] generate curves not lying completely on the surface. If such a curve is used as a boundary curve of another surface, gaps may occur between the two surfaces, which is not acceptable in many CAD applications such as surface blending and surface–surface intersection. In surface blending, if the linkage curves are not completely on the base surfaces, the blending surface and the base surfaces are not even G^0 continuous (see Fig. 1). So attention should be paid to the approximation algorithms that generate low degree curves lying completely on the free form surfaces, which is the aim of our paper.

In order to generate low degree approximate curves lying completely on the B-spline surface, we first approximate the domain curve with a polyline (a continuous line composed of one or more line segments) and divide the B-spline surface into Bézier surfaces. In many applications such as surface–surface intersection, what we obtain is not a curve but some discrete points in the parametric domain. In this case, the polyline that connects the discrete points sequentially follows. Then the domain curve is subdivided such that the Hausdorff distance between the approximate curve and the exact curve is controlled under the tolerance ε_D and the approximate curve is ε_T–G^1 continuous. Here ε_D is the user-specified distance tolerance while ε_T is the user-specified angle tolerance. By connecting all the mutual points of adjacent sub-curves in the parametric domain sequentially, we obtain the resultant polyline, the number of whose line segments depends on the tolerances ε_D and ε_T. Finally, the polyline is projected to the Bézier surfaces. The spatial approximate curve is ε_T–G^1 continuous at the mutual points of adjacent sub-curves. Approximate continuity first proposed by DeRose and Mann [5] has been applied in many CAD applications such as surface interpolation [5,20], surface blending [6], terrain modelling [13] and n-sided hole filling [15,23]. To date, the experience from presented algorithms [5,6,13,15,20,23] suggests that approximate continuity will virtually simplify the surface modelling activities without degrading the quality of the finished product. The degree of the approximate curve is $p+q$, where p and q are the u-direction and v-direction degrees of the B-spline surface, respectively. The main contributions of our work are as follows.

1. The approximate curves generated by our algorithm lie completely on the B-spline surface.
2. The Hausdorff distance between the approximate curve and the exact curve can be controlled under the user-specified tolerance.

The organization of this paper is as follows. In Section 2, input and output handling is discussed. Section 3 describes how to generate the initial approximate polyline. Section 4 contains details on how to control the Hausdorff distance between the approximate curve and the exact curve under the user-specified tolerance. In Section 5, how to generate an ε_T–G^1 continuous curve is described. Results are given in Section 6, and Section 7 concludes the paper.

2. Algorithm overview

A B-spline curve is defined by

$$\mathbf{D}(t) = \sum_{k=0}^{n_d-1} \mathbf{D}_k N_k^d(t),$$

where $\mathbf{D}_k$ are the control points and N_k^d are the dth-degree B-spline basis functions. A B-spline surface in three-dimensional space is defined by

$$\mathbf{S}(u,v) = \sum_{i=0}^{n_u-1} \sum_{j=0}^{n_v-1} \mathbf{P}_{i,j} N_i^p(u) N_j^q(v),$$

where $\mathbf{P}_{i,j}$ are the control points, and $N_i^p(u)$, $N_j^q(v)$ are the pth-degree and qth-degree B-spline basis functions, respectively. Assume that we have a B-spline curve $\mathbf{D}(t)$ lying completely in the parametric domain of the surface $\mathbf{S}$. Let $\widetilde{\mathbf{D}}(t)$ denote the image curve obtained by substituting $\mathbf{D}(t)$ into the surface equation of $\mathbf{S}$. We attempt to obtain a spatial low degree curve $\widetilde{\mathbf{C}}(t)$ lying completely on the surface $\mathbf{S}$ to approximate the curve $\widetilde{\mathbf{D}}(t)$. The main algorithm flow is described as follows.

1. Divide the B-spline surface into some Bézier surfaces by the insertion of knots, and approximate the domain curve with a polyline.
2. Subdivide the domain curve so that the Hausdorff distance between the mapped curve of the polyline and that of the B-spline curve is under the user-specified tolerance ε_D.
3. Subdivide the domain curve so that the spatial approximate curve is ε_T–G^1 continuous.
4. Project the polyline to the B-spline surface.

The following sections illustrate how to generate the approximate curve.

3. Approximate polyline

We first define several notations that will be used in the following sections.

Definition 1. Given a point $\mathbf{P}$ and a polyline $\mathbf{C}(t)$, the distance between $\mathbf{P}$ and $\mathbf{C}(t)$ is $\|\mathbf{P} - \mathbf{C}(t)\| = \min_{\mathbf{Q}\in\mathbf{C}(t)} \|\mathbf{P} - \mathbf{Q}\|$.

An example for the distance between one point and a polyline composed of just one line segment on a plane is given in Fig. 2. The two lines perpendicular to the line segment $(\mathbf{P_1}, \mathbf{P}_2)$ (denoted as $\mathbf{C}(t)$) and through the end points divide the plane into three areas: $\mathbf{D}_1$, $\mathbf{D}_2$ and $\mathbf{D}_3$. For all points in area $\mathbf{D}_1$ ($\mathbf{D}_3$), the closest point on $\mathbf{C}(t)$ is $\mathbf{P_1}(\mathbf{P}_2)$. Thus we have $\|\mathbf{M_1} -$

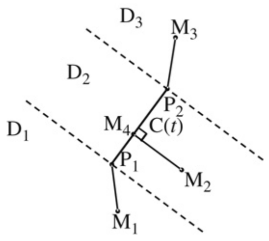


Fig. 2. The distances between points and lines.

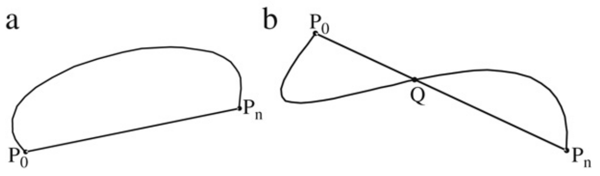


Fig. 3. The categories of Bézier curves: (a) a single-sided curve; (b) a double-sided curve.

$\mathbf{C}(t)\| = \|\mathbf{M_1} - \mathbf{P_1}\|$ and $\|\mathbf{M_3} - \mathbf{C}(t)\| = \|\mathbf{M_3} - \mathbf{P_2}\|$ where point $\mathbf{M_1} \in \mathbf{D}_1$ and point $\mathbf{M_3} \in \mathbf{D}_3$. For an arbitrary point $\mathbf{M}_2$ in area $\mathbf{D}_2$, the closest point on $\mathbf{C}(t)$ is the foot of the perpendicular from the point to $\mathbf{C}(t)$, denoted as $\mathbf{M}_4$. Thus we have $\|\mathbf{M_2} - \mathbf{C}(t)\| = \|\mathbf{M_2} - \mathbf{M_4}\|$.

Definition 2. Given a curve $\mathbf{C_1}(t)$ and a polyline $\mathbf{C_2}(t)$, the directed Hausdorff distance from $\mathbf{C_1}(t)$ to $\mathbf{C_2}(t)$ is $h(\mathbf{C_1}(t), \mathbf{C_2}(t)) = \max_{\mathbf{P}\in\mathbf{C_1}(t)} \|\mathbf{P} - \mathbf{C_2}(t)\|$. The Hausdorff distance between $\mathbf{C_1}(t)$ and $\mathbf{C_2}(t)$ is $H(\mathbf{C_1}(t), \mathbf{C_2}(t)) = \max(h(\mathbf{C_1}(t), \mathbf{C_2}(t)), h(\mathbf{C_2}(t), \mathbf{C_1}(t)))$.

$h(\mathbf{C_1}(t), \mathbf{C_2}(t)) = h(\mathbf{C_2}(t), \mathbf{C_1}(t))$ does not hold in general. Here we classify the Bézier curves into two categories as follows.

1. Single-sided curves: the Bézier curves that are on only one side of their corresponding line segments (see Fig. 3(a)). The line segment connecting the end points of a Bézier curve defines two half planes. The single-sided curve is a curve contained in just one half plane.
2. Double-sided curves: the Bézier curves that are on both sides of their corresponding line segments (see Fig. 3(b)).

If $\mathbf{C_1}(t)$ is a single-sided Bézier curve and $\mathbf{C_2}(t)$ is the line segment that connects the end points of $\mathbf{C_1}(t)$, we have $h(\mathbf{C_1}(t), \mathbf{C_2}(t)) \geq h(\mathbf{C_2}(t), \mathbf{C_1}(t))$ (see Appendix for the demonstration). Thus $h(\mathbf{C_1}(t), \mathbf{C_2}(t)) = H(\mathbf{C_1}(t), \mathbf{C_2}(t))$. For simplicity, without explicit explanation, we use $h(\mathbf{C_1}(t), \mathbf{C_2}(t))$ as the Hausdorff distance between $\mathbf{C_1}(t)$ and $\mathbf{C_2}(t)$.

Definition 3. Given a polyline $\mathbf{C}(t)$, the d tolerance boundary is composed of all the points $\mathbf{P}$ that satisfy $\|\mathbf{P} - \mathbf{C}(t)\| = d$.

Given a B-spline surface, we divide it into Bézier surfaces by the insertion of knots (see Fig. 4). Given a B-spline curve $\mathbf{D}(t)$ in the parametric domain of the surface (see Fig. 5), we use a polyline to approximate it. First, the B-spline curve $\mathbf{D}(t)$ is subdivided as follows.

1. Divide the B-spline curve (see Fig. 6) at

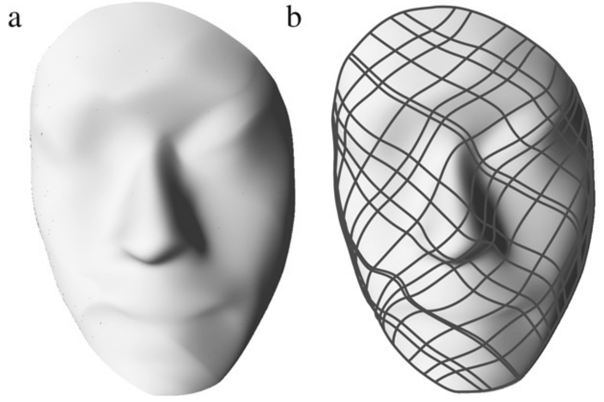


Fig. 4. B-spline surface division: (a) the original B-spline surface; (b) the resultant Bézier surfaces.

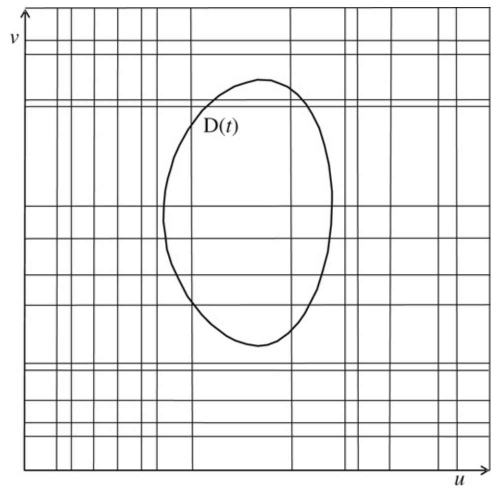


Fig. 5. B-spline curve in the surface domain.

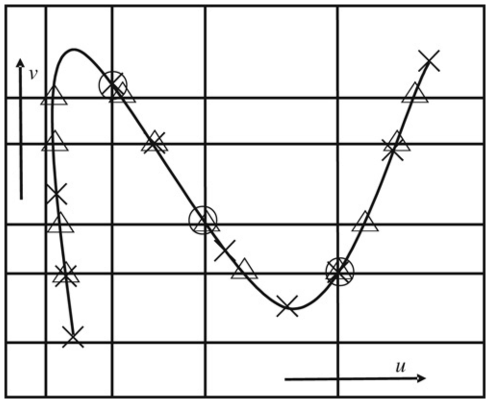


Fig. 6. Approximate polyline: $\times$ denotes the knot position of $\mathbf{D}(t)$, $\bigcirc$ denotes the position where the curve crosses a knot value in the u-direction, $\triangle$ denotes the position where the curve crosses a knot value in the v-direction.

. knot positions of $\mathbf{D}(t)$;
. parameter values where $\mathbf{D}(t)$ crosses a knot value in the u-direction or a knot value in the v-direction.

2. A Bézier curve $\mathbf{C}_1(t)$ is defined by

$$\mathbf{C}_1(t) = \sum_{i=0}^{n} B_i^n(t)\mathbf{P}_i, \quad 0 \leq t \leq 1, \tag{1}$$

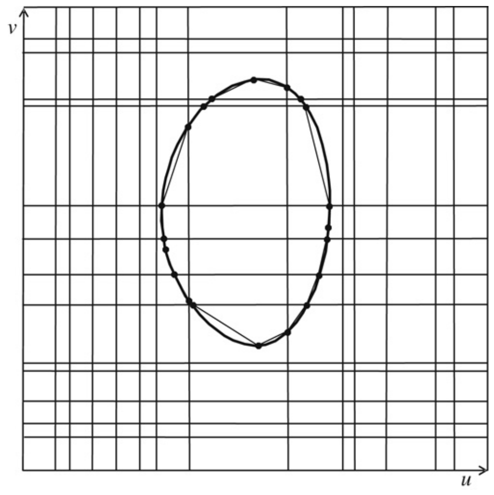


Fig. 7. The initial polyline. The dots are the split points of the B-spline curve, which are also the end points of the line segments.

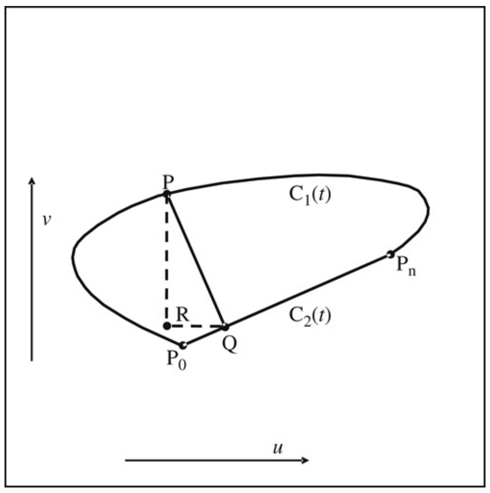


Fig. 8. Tolerance computation in the parametric domain.

where $\mathbf{P}_i = (u_i, v_i)^T$. From Eq. (1), we can obtain the parameter values of the intersection points of the Bézier curve and the line segment connecting the two end points. For curves of degree three, the parameter values of the intersection points are as follows:

$$\begin{cases} t_0 = 0 \\ t_1 = \dfrac{(u_3-u_0)v_1-(v_3-v_0)u_1-u_3v_0+u_0v_3}{(u_3-u_0)(v_1-v_2)+(v_3-v_0)(u_2-u_1)} \\ t_2 = 1. \end{cases}$$

If $t_1 \notin (0, 1)$, the curve is a single-sided curve; otherwise, it is a double-sided curve. We split double-sided curves into single-sided Bézier curves at the intersection points. For the double-sided curve shown in Fig. 3(b), we divide it into two single-sided curves at the intersection point $\mathbf{Q}$.

Then by connecting all the mutual points of adjacent Bézier curves sequentially, we obtain the initial polyline (see Fig. 7). The domain curve $\mathbf{D}(t)$ is further subdivided repeatedly to satisfy the distance and angle tolerances in the following sections.

4. Approximation precision

A good approximation algorithm should control the Hausdorff distance between the approximate curve and the exact curve. To control the Hausdorff distance between the mapped curves of the polyline and the B-spline curve under the user-specified tolerance ε_D, we must subdivide the Bézier curves whose directed Hausdorff distances to their corresponding line segments are more than the 2D tolerance d in the parametric domain, which will be calculated soon. After the curve splitting in Section 3, each Bézier curve lies inside the parametric domain of one definite Bézier surface and all the Bézier curves are single-sided. Suppose that $\mathbf{C}_1(t)$ and $\mathbf{C}_2(t)$ are the Bézier curve and its corresponding line segment, respectively. From the tolerance ε_D and the located Bézier surface, we calculate d as follows. Here we use $\widetilde{\mathbf{P}}$ to denote the spatial mapped element on the B-spline surface of a planar element $\mathbf{P}$ in the parametric domain. From Definition 2, the Hausdorff distance between the approximate curve and the exact curve is as follows.

$$H(\widetilde{\mathbf{C}}_1(t), \widetilde{\mathbf{C}}_2(t)) = \max_{\widetilde{\mathbf{P}}\in\widetilde{\mathbf{C}}_1(t)} \|\widetilde{\mathbf{P}} - \widetilde{\mathbf{C}}_2(t)\|. \tag{2}$$

Supposing $\mathbf{Q}$ to be $\mathbf{P}$'s nearest point on $\mathbf{C_2}(t)$ (see Fig. 8), we have

$$\max_{\widetilde{\mathbf{P}}\in\widetilde{\mathbf{C}}_1(t)} \|\widetilde{\mathbf{P}} - \widetilde{\mathbf{C}}_2(t)\| \le \max_{\widetilde{\mathbf{P}}\in\widetilde{\mathbf{C}}_1(t)} \|\widetilde{\mathbf{P}} - \widetilde{\mathbf{Q}}\|. \tag{3}$$

Letting $\mathbf{R}$ be the intersection point of the v-direction line through point $\mathbf{P}$ and the u-direction line through point $\mathbf{Q}$, we have

$$\max_{\widetilde{\mathbf{P}}\in\widetilde{\mathbf{C}}_1(t)} \|\widetilde{\mathbf{P}} - \widetilde{\mathbf{Q}}\| \le \max_{\widetilde{\mathbf{P}}\in\widetilde{\mathbf{C}}_1(t)} (\|\widetilde{\mathbf{P}} - \widetilde{\mathbf{R}}\| + \|\widetilde{\mathbf{R}} - \widetilde{\mathbf{Q}}\|). \tag{4}$$

From Eq. (2), and Inequalities (3) and (4), we have

$$H(\widetilde{\mathbf{C}}_1(t), \widetilde{\mathbf{C}}_2(t)) \le \max_{\widetilde{\mathbf{P}}\in\widetilde{\mathbf{C}}_1(t)} (\|\widetilde{\mathbf{P}} - \widetilde{\mathbf{R}}\| + \|\widetilde{\mathbf{R}} - \widetilde{\mathbf{Q}}\|). \tag{5}$$

Given a Bézier surface defined by

$$\mathbf{S}(u, v) = \sum_{i=0}^{m}\sum_{j=0}^{n} B_i^m(u)B_j^n(v)\mathbf{P}_{i,j}, \quad u \in [0, 1], v \in [0, 1], \tag{6}$$

Selimovic [19] derived the bound on the first-order derivative:

$$\left\|\frac{\partial \mathbf{S}}{\partial u}\right\| \le m \max_{i,h,k} \|\mathbf{P}_{i+1,h} - \mathbf{P}_{i,k}\| \tag{7}$$

and

$$\left\|\frac{\partial \mathbf{S}}{\partial v}\right\| \le n \max_{i,h,k} \|\mathbf{P}_{h,i+1} - \mathbf{P}_{k,i}\|. \tag{8}$$

From Inequality (8), we have

$$\|\widetilde{\mathbf{P}} - \widetilde{\mathbf{R}}\| \le \left|\int_{R_v}^{P_v} \frac{\partial \mathbf{S}}{\partial v} \mathrm{d}v\right|$$

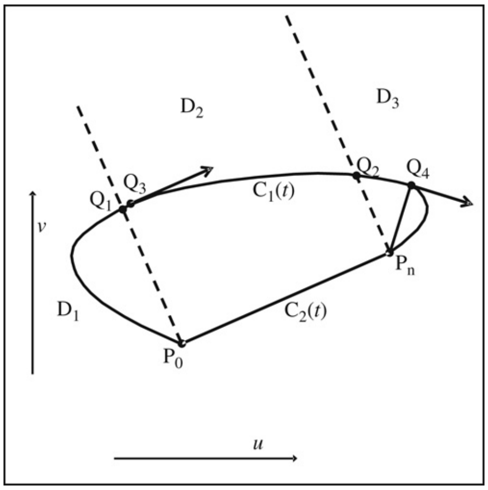


Fig. 9. Distance between a single-sided curve and its corresponding line segment.

$$\leq n\|\mathbf{P}-\mathbf{R}\| \max_{i,h,k} \|\mathbf{P}_{h,i+1}-\mathbf{P}_{k,i}\|. \tag{9}$$

From Inequality (7), we similarly have

$$\|\widetilde{\mathbf{R}}-\widetilde{\mathbf{Q}}\| \leq m\|\mathbf{R}-\mathbf{Q}\| \max_{i,h,k} \|\mathbf{P}_{i+1,h}-\mathbf{P}_{i,k}\|. \tag{10}$$

From Inequalities (5), (9) and (10), we have

$$\begin{aligned} H(\widetilde{\mathbf{C}}_1(t),\widetilde{\mathbf{C}}_2(t)) &\leq \max_{\widetilde{\mathbf{P}}\in\widetilde{\mathbf{C}}_1(t)} (\|\widetilde{\mathbf{P}}-\widetilde{\mathbf{R}}\|+\|\widetilde{\mathbf{R}}-\widetilde{\mathbf{Q}}\|) \\ &\leq \max_{\mathbf{P}\in\mathbf{C}_1(t)} (n\|\mathbf{P}-\mathbf{R}\| \max_{i,h,k}\|\mathbf{P}_{h,i+1}-\mathbf{P}_{k,i}\| \\ &\quad + m\|\mathbf{R}-\mathbf{Q}\| \max_{i,h,k}\|\mathbf{P}_{i+1,h}-\mathbf{P}_{i,k}\|) \\ &\leq \max_{\mathbf{P}\in\mathbf{C}_1(t)} (\|\mathbf{P}-\mathbf{Q}\|)(n\max_{i,h,k}\|\mathbf{P}_{h,i+1}-\mathbf{P}_{k,i}\| \\ &\quad + m\max_{i,h,k}\|\mathbf{P}_{i+1,h}-\mathbf{P}_{i,k}\|) \\ &= H(\mathbf{C}_1(t),\mathbf{C}_2(t))(n\max_{i,h,k}\|\mathbf{P}_{h,i+1} \\ &\quad -\mathbf{P}_{k,i}\|+m\max_{i,h,k}\|\mathbf{P}_{i+1,h}-\mathbf{P}_{i,k}\|) \\ &\leq \varepsilon_D. \end{aligned} \tag{11}$$

From Inequality (11), we have

$$H(\mathbf{C}_1(t),\mathbf{C}_2(t)) \leq \frac{\varepsilon_D}{n\max_{i,h,k}\|\mathbf{P}_{h,i+1}-\mathbf{P}_{k,i}\|+m\max_{i,h,k}\|\mathbf{P}_{i+1,h}-\mathbf{P}_{i,k}\|}. \tag{12}$$

Let

$$d=\frac{\varepsilon_D}{n\max_{i,h,k}\|\mathbf{P}_{h,i+1}-\mathbf{P}_{k,i}\|+m\max_{i,h,k}\|\mathbf{P}_{i+1,h}-\mathbf{P}_{i,k}\|}.$$

If $H(\mathbf{C}_1(t),\mathbf{C}_2(t)) \leq d$, the Hausdorff distance between $\widetilde{\mathbf{C}}_2(t)$ and $\widetilde{\mathbf{C}}_1(t)$ is controlled under the user-specified tolerance ε_D. If this condition holds for all Bézier curves in the parametric domain, the Hausdorff distance between the approximate curve and the exact curve is also controlled under ε_D.

To compute $H(\mathbf{C}_1(t),\mathbf{C}_2(t))$, the single-sided Bézier curve is divided into sub-curves by $\mathbf{C}_2(t)$'s perpendicular lines through its end points. An example is given in Fig. 9. The curve $\mathbf{C}_1(t)$ is divided into three sub-curves at the intersection points $\mathbf{Q}_1$ and $\mathbf{Q}_2$. For the sub-curve in $\mathbf{D}_2$, the farthest point $\mathbf{Q}_3$ to the line segment is the point whose tangent is parallel to the line $\mathbf{C}_2(t)$ if it exists, or otherwise, one of the end points. For the sub-curve in $\mathbf{D}_3$, the farthest point $\mathbf{Q}_4$ to $\mathbf{P}_n$ is the point whose tangent is perpendicular to the line $(\mathbf{P}_n,\mathbf{Q}_4)$ if it exists, or otherwise, the end point $\mathbf{Q}_2$. A similar method is used for the sub-curve in $\mathbf{D}_1$. In all, to compute the Hausdorff distance between the Bézier curve and its corresponding line segment, we need to compute the following candidate points:

1. The intersection points of the Bézier curve and the lines which are perpendicular to the corresponding line segment through its end points.
2. The point on the Bézier curve whose tangent is parallel to the corresponding line segment. The derivative of the curve (1) is

$$\mathbf{C}_1'(t)=n\sum_{i=0}^{n-1}B_i^n(t)(\mathbf{P}_{i+1}-\mathbf{P}_i),\quad 0\leq t\leq 1. \tag{13}$$

Let $\mathbf{V}$ denote the vector $(v_0-v_n,u_n-u_0)^T$, which is perpendicular to the line. Because the derivative is parallel to the line $(\mathbf{P}_0,\mathbf{P}_n)$, we have

$$\mathbf{C}_1'(t)\cdot\mathbf{V}=0. \tag{14}$$

From Eqs. (13) and (14), we have

$$\sum_{i=0}^{n-1}B_i^n(t)(\mathbf{P}_{i+1}-\mathbf{P}_i)\cdot\mathbf{V}=0. \tag{15}$$

Eq. (15) is a polynomial of variable t. Roots of Eq. (15) in (0, 1) are parameter values of the candidate points.

3. The points on the Bézier curve whose tangents are perpendicular to the lines from them to the end points. For the end point $\mathbf{P}_0$, the parameter values of the candidate points satisfy the following equation:

$$\mathbf{C}_1'(t)\cdot(\mathbf{C}_1(t)-\mathbf{P}_0)=0. \tag{16}$$

Roots of Eq. (16) whose corresponding points lie inside area $\mathbf{D}_1$ are the candidate parameter values. For the end point $\mathbf{P}_n$, the parameter values of the candidate points satisfy the following equation:

$$\mathbf{C}_1'(t)\cdot(\mathbf{C}_1(t)-\mathbf{P}_n)=0. \tag{17}$$

Roots of Eq. (17) whose corresponding points lie inside area $\mathbf{D}_3$ are parameter values of the candidate points. Eqs. (16) and (17) can be solved using an iterative method.

Select the farthest point to the line from candidate points calculated above. Letting d_1 denote the distance between the farthest point and the line, we have $d_1=H(\mathbf{C}_1(t),\mathbf{C}_2(t))$. If $d_1\leq d$ holds for all Bézier curves, we get the final polyline; otherwise, each Bézier curve that fails, which is single-sided due to the curve splitting in Section 3, is handled as follows.

Step 1. The Bézier curve is subdivided at the farthest point.
Step 2. If the sub-curves are double-sided, they are split at the intersection points of the sub-curves and their corresponding line segments.

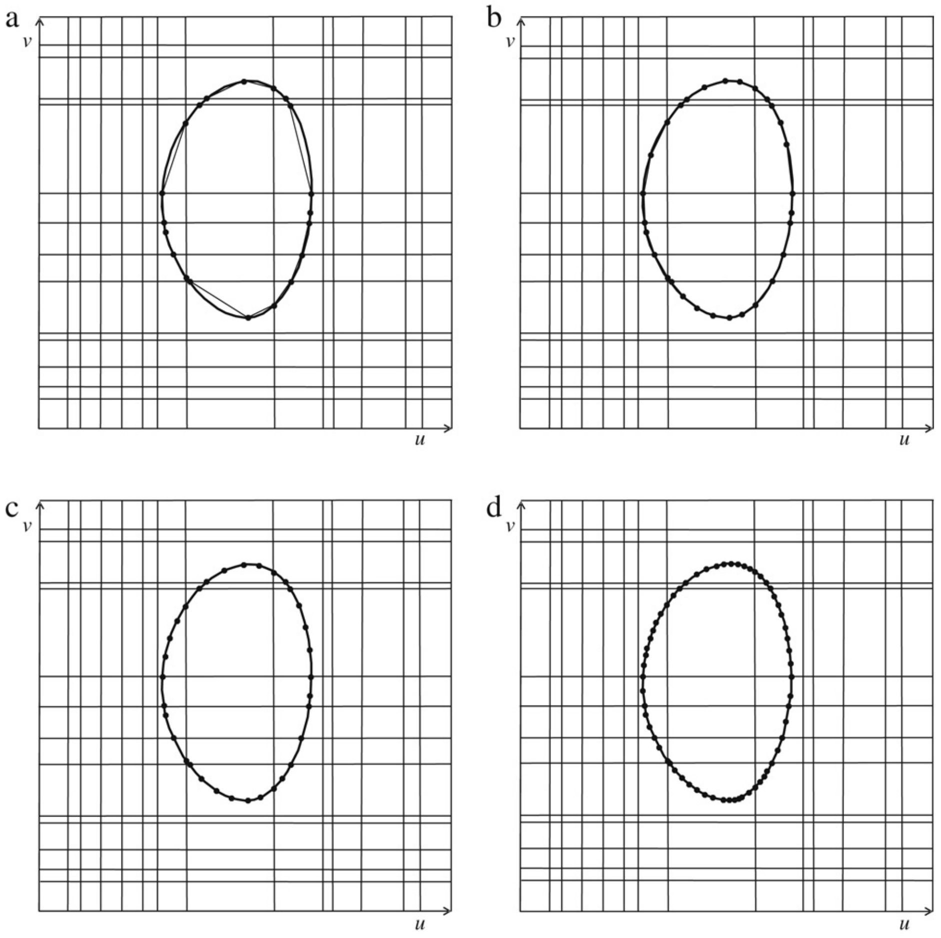


Fig. 10. Approximate polylines for (a) $\varepsilon_D = 0.5$, (b) $\varepsilon_D = 0.1$, (c) $\varepsilon_D = 0.01$, (d) $\varepsilon_D = 0.001$.

Step 3. Check whether $d_1 \le d$ holds for all generated sub-curves. For each sub-curve that fails, go to Step 1.

After the splitting procedure above, the Hausdorff distance between the approximate curve obtained in this section and the exact curve is under the user-specified tolerance ε_D. Also all the planar Bézier curves in the parametric domain are single-sided, which is a necessary condition for Theorem 1 in the next section. For the B-spline curve shown in Figs. 5 and 10(a)–(d) show the approximate polylines for $\varepsilon_D = \{0.5, 0.1, 0.01, 0.001\}$. The B-spline curve is located in the parametric domain of the surface shown in Fig. 4(a).

Comparison between user-specified distance tolerances and the resulting errors would help us to know whether the distance bound is tight or not. If we map the curve shown in Fig. 4 to the B-spline surface of degree 3×3 shown in Fig. 4, the spatial approximate curves deviate from the exact curve differently according to the user-specified tolerances. The resulting Hausdorff distances between the spatial approximate curves and the exact curve are given in Table 1. From Table 1, when the tolerances become small enough, the resulting errors are about one tenth of them. From Eq. (12), the larger the obtained 2D tolerance d is, the tighter the distance bound would be. How to compute a larger 2D tolerance for a user-specified 3D distance tolerance is left as a future work. How to generate an ε_T–G^1 continuous approximate curve that lies completely on the surface is described in the next section.

5. ε_T–G^1 continuous curves

Curves are said to be ε_T–G^1 continuous if the maximum tangent discrepancy between any pair of adjacent curves (see Fig. 11) is bounded by the angle tolerance ε_T. The end derivatives of the mapped curve are computed as follows. The line segment $\mathbf{C}_2(t)$ shown in Fig. 8 can be represented in the form of the parameter equation of variable t as follows.

$$\mathbf{C}_2(t) = \mathbf{P}_0(1-t) + t\mathbf{P}_n, \quad 0 \le t \le 1, \tag{18}$$

where $\mathbf{P}_0 = (u_0, v_0)^T$ and $\mathbf{P}_n = (u_n, v_n)^T$. By substituting Eq. (18) into Eq. (6), we have

$$\begin{aligned}\widetilde{\mathbf{C}}_2(t) &= \mathbf{S}(\mathbf{C}_2(t)) \\ &= \mathbf{S}(u_0 + t(u_n - u_0), v_0 + t(v_n - v_0)).\end{aligned} \tag{19}$$

According to the chain rule, we get from Eq. (19)

$$\begin{aligned}\widetilde{\mathbf{C}}_2'(t) &= \frac{\partial \mathbf{S}}{\partial u}\frac{\mathrm{d}u}{\mathrm{d}t} + \frac{\partial \mathbf{S}}{\partial v}\frac{\mathrm{d}v}{\mathrm{d}t} \\ &= (u_n - u_0)\frac{\partial \mathbf{S}}{\partial u}(u_0 + t(u_n - u_0), v_0 + t(v_n - v_0))\end{aligned}$$

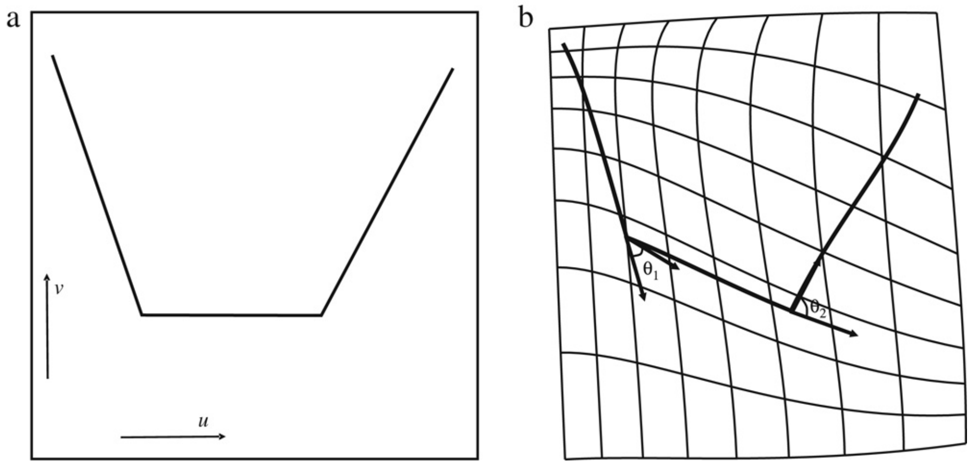


Fig. 11. Tangent discrepancy between curves: (a) three line segments in the parametric domain; (b) angles between the end derivatives of the adjacent mapped curves.

Table 1
Comparison of user-specified distance tolerances and the resulting errors

Tolerance	1	0.5	0.3	0.1	0.01	0.001	0.0001
Error	0.02798	0.02798	0.01858	0.00773	0.00102	0.00011	0.00001

$$+ (v_n - v_0)\frac{\partial \mathbf{S}}{\partial v}(u_0 + t(u_n - u_0), v_0 + t(v_n - v_0)). \quad (20)$$

For a Bézier surface defined in Eq. (6), the partial derivatives [14] are

$$\frac{\partial \mathbf{S}}{\partial u}(u, v) = m \sum_{i=0}^{m-1} \sum_{j=0}^{n} B_i^{m-1}(u) B_j^n(v)(\mathbf{P}_{i+1,j} - \mathbf{P}_{i,j}), \quad (21)$$

and

$$\frac{\partial \mathbf{S}}{\partial v}(u, v) = n \sum_{i=0}^{m} \sum_{j=0}^{n-1} B_i^{m}(u) B_j^{n-1}(v)(\mathbf{P}_{i,j+1} - \mathbf{P}_{i,j}). \quad (22)$$

Letting $t = 0$ and $t = 1$, from Eqs. (20)–(22), we obtain the two end derivatives of the mapped curve $\widetilde{\mathbf{C}}_2(t)$. If the angle (denoted as α) between the end derivatives at the mutual point of two adjacent mapped curves exceeds the angle tolerance ε_T, the Bézier curve with the greater deviation from its corresponding line segment is subdivided at the farthest point. This operation is performed repeatedly until $\alpha < \varepsilon_T$ holds for all the adjacent line segments. After the splitting operations, we should guarantee that Hausdorff distance between the approximate curve and the exact curve remain under the tolerance ε_D for each Bézier curve in the parametric domain.

Theorem 1. *For a single-sided Bézier curve in the parametric domain, if the Hausdorff distance between the approximate curve and the exact curve is under the tolerance ε_D, it also holds after splitting operations.*

Proof. After splitting operations, a polyline $\mathbf{C}_3(t)$ is generated to approximate a single-sided Bézier curve $\mathbf{C}_1(t)$ (see Fig. 12).

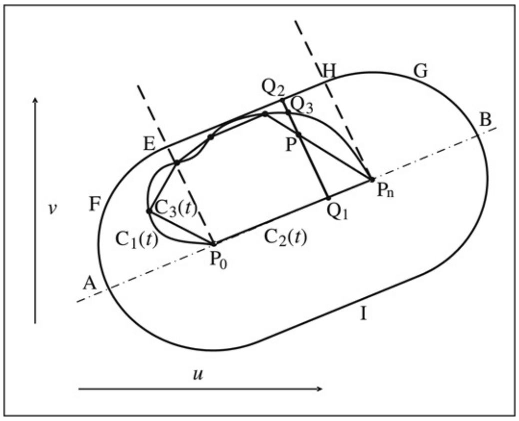


Fig. 12. Demonstration of Theorem 1.

Table 2
Results for a curve on a Bézier surface

	Exact	Degree reduction	Our algorithm
Tolerance	–	1×10^{-3}	$1 \times 10^{-3}/10°$
Degree	8	3	4
Number of control points	9	16	33
Number of segments	1	7	8
Distance to **S**	0	0.63×10^{-4}	0
Continuity	G^1	G^1	$10°$–G^1
Processor time (ms)	0.23	21	15

For the curve $\mathbf{C}_1(t)$, the d tolerance boundary of its corresponding line segment $(\mathbf{P}_0, \mathbf{P}_n)$ is the curve **AIBGHEFA**. The lines perpendicular to $\mathbf{C}_2(t)$ through points $\mathbf{P}_0$ and $\mathbf{P}_n$ intersect the tolerance boundary **AIBGHEFA** at points **E** and **H**, respectively. Because $\mathbf{C}_1(t)$ is single-sided, it lies on one side of the line $(\mathbf{P}_0, \mathbf{P}_n)$. Suppose that $\mathbf{C}_1(t)$ lies inside

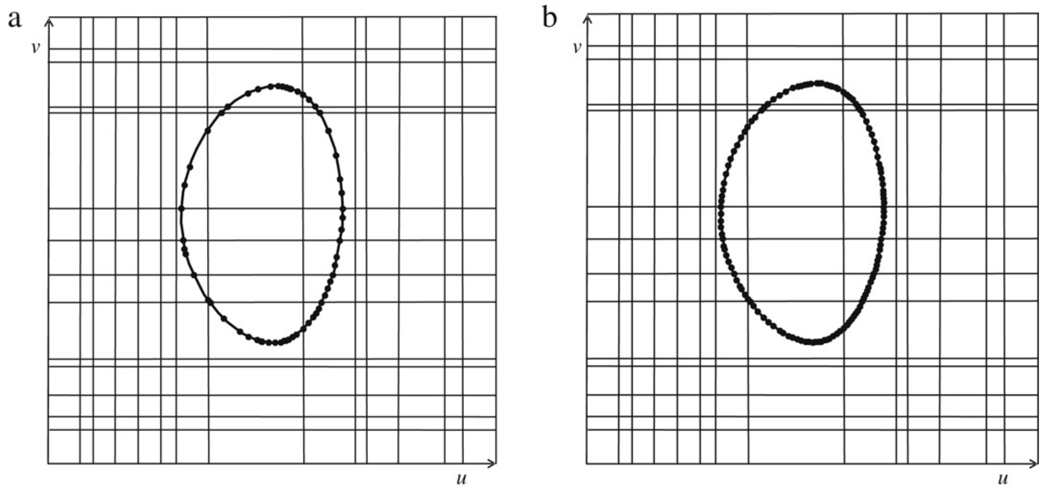


Fig. 13. Approximate polylines for $\varepsilon_D = 0.1$ and (a) $\varepsilon_T = 10°$, (b)$\varepsilon_T = 1°$.

the closed curve $\mathbf{AP_0P_nBGHEFA}$. We divide the domain inside $\mathbf{AP_0P_nBGHEFA}$ into three areas by lines $(\mathbf{P}_0, \mathbf{E})$ and $(\mathbf{P}_n, \mathbf{H})$, which are the area inside the curve $\mathbf{AP_0EFA}$, the area inside the curve $\mathbf{EP_0P_nHE}$, and the area inside the curve $\mathbf{HP_nBGH}$. Based on the division, every point inside curve $\mathbf{AP_0P_nBGHEFA}$ is demonstrated to lie inside the d tolerance boundary of the polyline $\mathbf{C}_3(t)$ as follows.

Case 1. For an arbitrary point $\mathbf{P}$ lying inside the curve $\mathbf{AP_0EFA}$, the nearest point on $\mathbf{C}_2(t)$ is $\mathbf{P}_0$. Then we have $\|\mathbf{P} - \mathbf{C}_3(t)\| \le d$. Thus each point lying inside the curve $\mathbf{AP_0EFA}$ is inside the d tolerance boundary of $\mathbf{C}_3(t)$.

Case 2. Similarly, each point $\mathbf{P}$ lying inside the curve $\mathbf{HP_nBGH}$ can be demonstrated to lie inside the d tolerance boundary of $\mathbf{C}_3(t)$.

Case 3. The line through point $\mathbf{P}$, which is perpendicular to $\mathbf{C}_2(t)$, intersects curves $\mathbf{C}_2(t)$, $(\mathbf{E}, \mathbf{H})$ and $\mathbf{C}_3(t)$ at points $\mathbf{Q}_1$, $\mathbf{Q}_2$ and $\mathbf{Q}_3$, respectively. For each point $\mathbf{P}$ lying inside $\mathbf{EP_0P_nHE}$, the nearest point on $\mathbf{C}_2(t)$ is $\mathbf{Q}_1$. The length of line $(\mathbf{Q}_1, \mathbf{Q}_2)$ is d, so the distance between points $\mathbf{P}$ and $\mathbf{Q}_3$ is not greater than d. Thus $\|\mathbf{P} - \mathbf{C}_3(t)\| \le d$ holds for all points inside $\mathbf{EP_0P_nHE}$.

In all, $h(\mathbf{C}_3(t), \mathbf{C}_1(t)) \le d$ holds. Using a similar method, we can demonstrate that $h(\mathbf{C}_1(t), \mathbf{C}_3(t)) \le d$. Thus $H(\mathbf{C}_3(t), \mathbf{C}_1(t)) \le d$. As all the curves above lie inside the parametric domain of one definite Bézier surface, we have $H(\widetilde{\mathbf{C}}_1(t), \widetilde{\mathbf{C}}_3(t)) \le \varepsilon_D$. □

After the approximate precision operations in Section 4, the Bézier curves are all single-sided. From Theorem 1, after the splitting operations in this section, the Hausdorff distance between the approximate curve and the exact curve will remain under the tolerance ε_D. For the B-spline curve shown in Fig. 5, Fig. 13(a)–(b) show the approximate polylines for $\varepsilon_D = 0.1$ and $\varepsilon_T = \{10°, 1°\}$.

After the final approximate polyline is obtained, we map it to the spatial B-spline surface by composition of Bézier simplexes. Eq. (19) is a composition of Eqs. (18) and (6), which are in the Bézier form. Blossoming techniques [3,4] are utilized to get the mapped curves in the Bézier form of the approximate line segments. The degree of the mapped curves is $p+q$, where p and q are the u-direction degree and v-direction degree of the B-spline surface, respectively.

6. Results

To show the performance of the algorithm presented in this paper, three examples are given below, which are all implemented in the environment with Intel Pentium IV CPU 2.0 GHz, 1G Memory, Microsoft Windows XP, and Microsoft Visual C++ 6.0.

In the first example, a quadratic curve with 3 control points $\mathbf{P}_0 = (0.1, 0.1)^T$, $\mathbf{P_1} = (0.5, 1.8)^T$ and $\mathbf{P}_2 = (0.8, 0.1)^T$ is mapped onto a biquadratic Bézier surface with 3 by 3 control points. The control points of the Bézier surface are

$$\begin{cases}\mathbf{P}_{00} = (0, 2, -1)^T \\ \mathbf{P}_{10} = (1, 1, -2)^T \\ \mathbf{P}_{20} = (1, 0, -3)^T,\end{cases} \qquad \begin{cases}\mathbf{P}_{01} = (2.5, 1, 0)^T \\ \mathbf{P}_{11} = (1, 0, -0.5)^T \\ \mathbf{P}_{21} = (1, -1, -2)^T,\end{cases}$$

$$\text{and} \quad \begin{cases}\mathbf{P}_{02} = (1, 0, 1.5)^T \\ \mathbf{P}_{12} = (2.5, -1, 0)^T \\ \mathbf{P}_{22} = (-0.51, -2, -1)^T.\end{cases}$$

The exact curve on the surface [17] has 9 control points, as shown in Fig. 14(b). The degree reduction algorithm introduced in [17] is implemented to generate one approximate curve (see Fig. 14(c)), where the tolerance is set to 10^{-3}. The result of our algorithm is given in Fig. 14(d), where the distance tolerance is set to 10^{-3} and the angle tolerance is set to 10°. Results of the three algorithms are given in Table 2. The curves generated by our algorithm and the exact algorithm lie completely on the surface, while the degree of the curve generated by our algorithm is 4, much lower than that of the exact 8th-degree curve. Though the curve generated by the degree reduction algorithm is very close to the surface, it does not lie completely on it. In addition, a comparison of the processor time taken by the degree reduction algorithm and our algorithm shows the

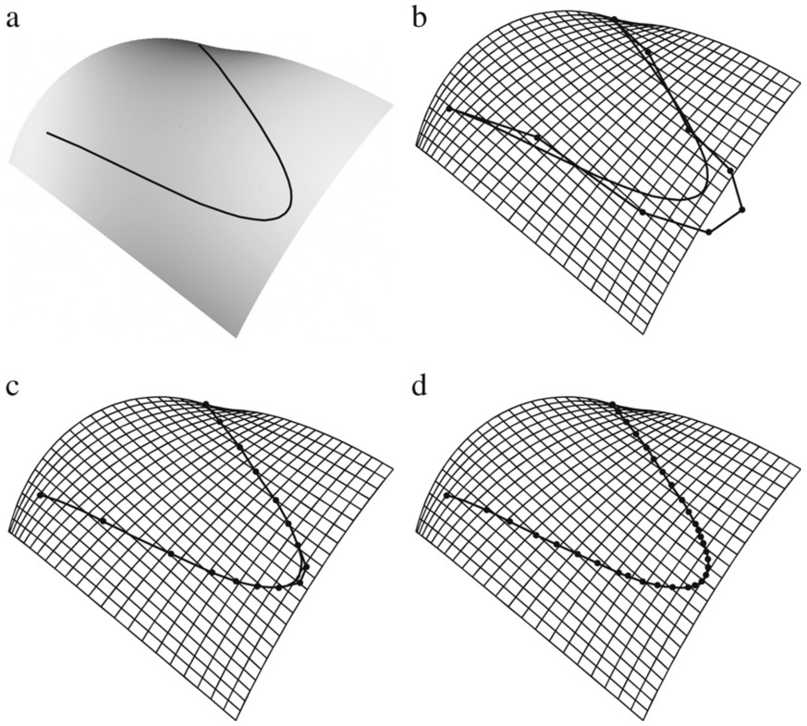


Fig. 14. Exact and approximate images of a quadratic Bézier curve on a biquadratic Bézier surface: (a) the exact curve; (b) the exact curve and its control polygon; the approximate curve and its control polygon generated by (c) the degree reduction algorithm and (d) our algorithm.

high efficiency of our algorithm. Though more subdivisions are needed, our algorithm is even faster than Renner's degree reduction algorithm.

In the second example, the cubic, closed B-spline curve in Fig. 5 with 10 control points lying in the parametric domain of bicubic human face model in Fig. 4(a) is mapped onto the human face surface with 17 by 17 control points (see Fig. 15(a)). The exact curve and its control polygon [17] are shown in Fig. 15(b). The result of the degree reduction algorithm in [17] is shown in Fig. 15(c), where the target degree is set to 3 and the tolerance is set to 10^{-3}. One approximation is also computed using our presented algorithm (see Fig. 15(d)), where ε_D is set to 10^{-3} and ε_T is set to 1°. Results of the three algorithms are given in Table 3.

In the third example, a cubic curve consisting of 20 segments is mapped onto a bicubic B-spline tiger ear surface with 19 by 19 control points (see Fig. 16(a)). The exact curve on the surface has more than 800 control points, as shown in Fig. 16(b). The result of the degree reduction algorithm is shown in Fig. 16(c), where the tolerance is set to 10^{-3}. The result of our algorithm is given in Fig. 16(d), where the distance tolerance to the exact curve is set to 10^{-3} and the angle tolerance is set to 1°. Results of the three algorithms are given in Table 4.

The approximation algorithms in [16,17] generate curves that are not completely on the B-spline surface. From Tables 2–4, though our algorithm needs more subdivisions than Renner's approximation and exact algorithms, it generates a considerably lower degree curve than the exact algorithm, while the generated curve also lies completely on the B-spline surface, which is indispensable for many CAD applications, such as surface blending and surface–surface intersection. The time cost of our algorithm is comparable with that of the degree reduction algorithm in [17], which satisfies the real-time needs of CAD applications. Also the curves generated in our algorithm are $\varepsilon - G^1$ continuous. For many applications such as NC-machining and surface blending, it is sufficient for surfaces to be $\varepsilon - G^1$ continuous. Engineering practice suggests that there is no ambiguity within acceptable working tolerance. The geometric discrepancies in the $\varepsilon - G^1$ surface models are very small and may be blended out in the subsequent manufacturing processes.

7. Conclusions

An approximation algorithm for computing a curve on a B-spline surface has been presented. First the initial polyline of the domain curve is generated. The domain curve is then subdivided repeatedly until the Hausdorff distance between the approximate curve and the exact curve is under the distance tolerance, after which the angle deviation between the mapped curves is controlled under the user-specified tolerance by another subdivision procedure. Compared with exact curves, the degree of the generated curves of our algorithm is much lower. Moreover, different from previous approximation algorithms, our algorithm generates curves that lie completely on the B-spline surface, which is indispensable for many CAD applications.

Acknowledgements

The research was supported by Chinese 973 Program (2004CB719400), and the National Science Foundation of

Table 3
Results for a curve on a face surface

	Exact	Degree reduction	Our algorithm	
Tolerance	–	1×10^{-3}	$1 \times 10^{-3}/10°$	$1 \times 10^{-3}/1°$
Degree	18	3	6	6
Number of control points	332	215	499	1159
Number of segments	20	73	83	193
Distance to **S**	0	0.58×10^{-3}	0	0
Continuity	G^1	G^1	$10°$–G^1	$1°$–G^1
Processor time (ms)	9	150	79	196

Table 4
Results for a curve on an ear surface

	Exact	Degree reduction	Our algorithm	
Tolerance	–	1×10^{-3}	$1 \times 10^{-3}/10°$	$1 \times 10^{-3}/1°$
Degree	18	3	6	6
Number of control points	825	227	739	1225
Number of segments	47	89	123	204
Distance to **S**	0	0.46×10^{-3}	0	0
Continuity	G^1	G^1	$10°$–G^1	$1°$–G^1
Processor time (ms)	22	232	121	308

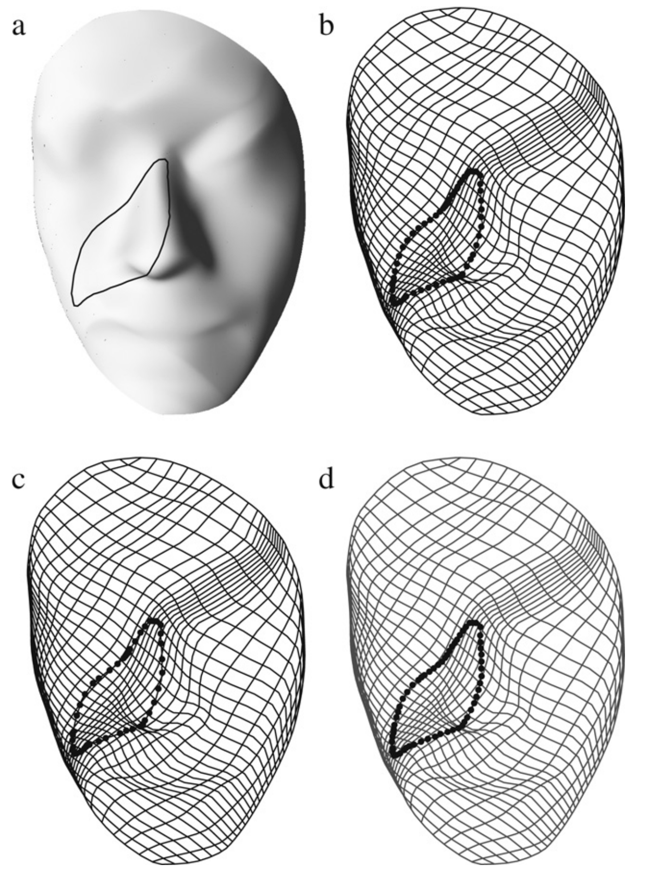


Fig. 15. Exact and approximate images of a cubic domain curve on a bicubic B-spline surface of a human face: (a) the exact curve; (b) the exact curve and its control polygon; the approximate curve and its control polygon generated by (c) the degree reduction algorithm and (d) our algorithm.

China (60625202, 60533070). The third author was supported by the project sponsored by a Foundation for the Author of National Excellent Doctoral Dissertation of PR China (200342), and a Program for New Century Excellent Talents in University (NCET-04-0088).

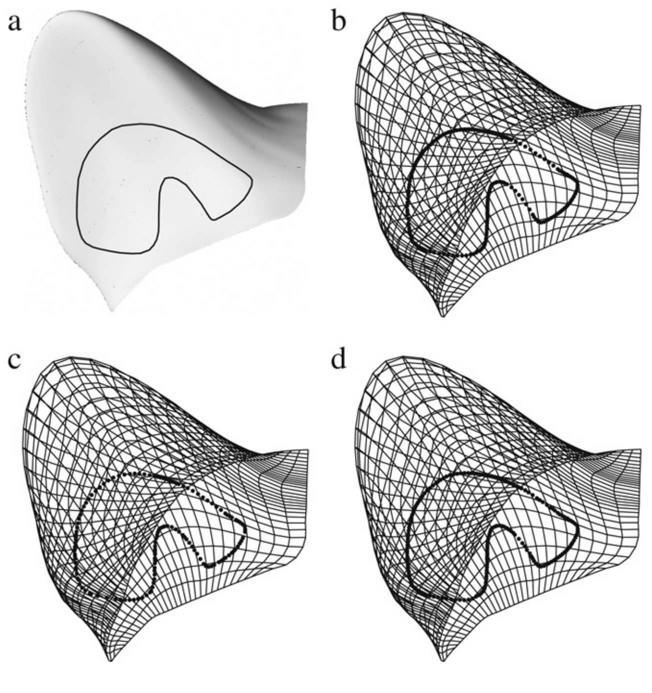


Fig. 16. Exact and approximate images of a cubic domain curve on a bicubic B-spline surface of an ear face: (a) the exact curve; (b) the exact curve and its control polygon; the approximate curve and its control polygon generated by (c) the degree reduction algorithm and (d) our algorithm.

Appendix

Theorem 2. *Given a Bézier curve* $\mathbf{C_1}(t)$ *and the corresponding line segment* $\mathbf{C_2}(t)$ *connecting the end points of curve* $\mathbf{C_1}(t)$*, the*

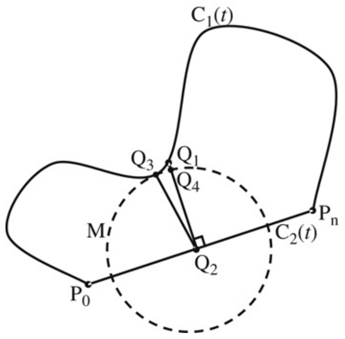


Fig. 17. The Hausdorff distance.

directed Hausdorff distance from $\mathbf{C_1}(t)$ *to* $\mathbf{C_2}(t)$ *is equal to the Hausdorff distance between them.*

Proof. Given a Bézier curve $\mathbf{C_1}(t)$ and the corresponding line segment $\mathbf{C_2}(t)$ connecting the end points of curve $\mathbf{C_1}(t)$, the directed Hausdorff distance from $\mathbf{C_2}(t)$ to $\mathbf{C_1}(t)$ is $h(\mathbf{C_2}(t), \mathbf{C_1}(t)) = \max_{\mathbf{Q_2}\in\mathbf{C_2}(t)} \|\mathbf{Q_2} - \mathbf{C_1}(t)\|$. For an arbitrary point $\mathbf{Q_2} \in \mathbf{C_2}(t)$, there exists a point $\mathbf{Q_1} \in \mathbf{C_1}(t)$ such that

$$\|\mathbf{Q_2} - \mathbf{C_1}(t)\| \le \|\mathbf{Q_1} - \mathbf{C_2}(t)\|, \tag{23}$$

which will be demonstrated soon. If Condition (23) holds for every point $\mathbf{Q_2} \in \mathbf{C_2}(t)$, we have

$$\max_{\mathbf{Q_2}\in\mathbf{C_2}(t)} \|\mathbf{Q_2} - \mathbf{C_1}(t)\| \le \max_{\mathbf{Q_1}\in\mathbf{C_1}(t)} \|\mathbf{Q_1} - \mathbf{C_2}(t)\|,$$

which is

$$h(\mathbf{C_2}(t), \mathbf{C_1}(t)) \le h(\mathbf{C_1}(t), \mathbf{C_2}(t)). \tag{24}$$

From Definition 2, we have

$$\begin{aligned} &H(\mathbf{C_1}(t), \mathbf{C_2}(t)) \\ &\quad = \max(h(\mathbf{C_1}(t), \mathbf{C_2}(t)), h(\mathbf{C_2}(t), \mathbf{C_1}(t))). \end{aligned} \tag{25}$$

From Eqs. (24) and (25), we have

$$H(\mathbf{C_1}(t), \mathbf{C_2}(t)) = h(\mathbf{C_1}(t), \mathbf{C_2}(t)). \tag{26}$$

Condition (23) is demonstrated as follows. For an arbitrary point $\mathbf{Q_2} \in \mathbf{C_2}(t)$ (see Fig. 17), a circle $\mathbf{M}$ centered at $\mathbf{Q_2}$ touches the Bézier curve $\mathbf{C_1}(t)$ at the closest point $\mathbf{Q_3}$ (which is the closest point on curve $\mathbf{C_1}(t)$ to point $\mathbf{Q_2}$). The perpendicular line of line segment $\mathbf{C_2}(t)$ through point $\mathbf{Q_2}$ intersects circle $\mathbf{M}$ and curve $\mathbf{C_1}(t)$ at points $\mathbf{Q_4}$ and $\mathbf{Q_1}$, respectively. Because there exists no point of curve $\mathbf{C_1}(t)$ lying inside the circle $\mathbf{M}$ (from the definition of the closest point), we have $\|\mathbf{Q_2} - \mathbf{Q_3}\| = \|\mathbf{Q_2} - \mathbf{Q_4}\| \le \|\mathbf{Q_1} - \mathbf{Q_2}\|$. From Definition 1, $\|\mathbf{Q_2} - \mathbf{C_1}(t)\| = \min_{\mathbf{Q}\in\mathbf{C_1}(t)} \|\mathbf{Q_2} - \mathbf{Q}\| = \|\mathbf{Q_2} - \mathbf{Q_3}\|$ and $\|\mathbf{Q_1} - \mathbf{C_2}(t)\| = \min_{\mathbf{Q}\in\mathbf{C_2}(t)} \|\mathbf{Q_1} - \mathbf{Q}\| = \|\mathbf{Q_1} - \mathbf{Q_2}\|$. Condition (23) follows. Thus Condition (26) holds. □

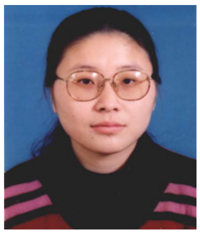
References

[1] Chuang JH, Lin CH, Hwang WC. Variable-radius blending of parametric surfaces. The Visual Computer 1995;11(10):513–25.
[2] Chuang JH, Hwang WC. Variable-radius blending by constrained spine generation. The Visual Computer 1997;13(7):316–29.
[3] DeRose T. Composing Bézier simplexes. ACM Transaction on Graphics 1988;7(3):198–221.
[4] DeRose T, Goldman R, Hagen H, Mann S. Functional composition algorithm via blossoming. ACM Transaction on Graphics 1993;12(2): 113–35.
[5] DeRose T, Mann S. An approximately G^1 cubic surface interpolant. In: Lyche T, Schumaker L, editors. Mathematical methods in computer aided geometric design, vol. II. Academic Press; 1992. p. 185–96.
[6] Elsas PAV, Vergeest JSM. Fast approximate G^1 surface blending to support interactive sculptured feature design. In: Pratt MJ, Sriram RD, Wozny MJ, editors. Product modeling for computer integrated design and manufacture. London: Chapman & Hall; 1997. p. 149–61.
[7] Flöry S, Hofer M. Constrained curve fitting on manifolds. Computer-Aided Design [in press].
[8] Hu Y-P, Sun T-C. Moving a B-spline surface to a curve — A trimmed surface matching algorithm. Computer-Aided Design 1997;29(6):449–55.
[9] Knuth D. The art of computer programming-fundamental algorithms, vol. 1. Reading (MA): Addison-Wesley; 1969.
[10] Lee E. Computing a chain of blossom, with application to products of splines. Computer Aided Geometric Design 1994;11(6):597–620.
[11] Lee I-K. Curve reconstruction from unorganized points. Computer Aided Geometric Design 2000;17(2):161–77.
[12] Liu W, Mann S. An optimal algorithm for expanding the composition of the polynomials. ACM Transaction on Graphics 1997;16(2):155–78.
[13] Pferifer N. 3D terrain models on the basis of a triangulation. Geowissenschaftliche Mitteilungen, Heft 2002.
[14] Piegl LA, Tiller W. The NURBS book. 2nd ed. New York: Springer; 1997.
[15] Piegl LA, Tiller W. Filling n-sided regions with NURBS patches. The Visual Computer 1999;15(2):77–89.
[16] Renner G, Weiß V. Curves on surfaces: Analysis and new solutions. In: Cripps R, editor. The mathematics of surfaces, vol. VIII. Information Geometers; 1998. p. 57–71.
[17] Renner G, Weiß V. Exact and approximate computation of B-spline curves on surfaces. Computer-Aided Design 2004;36(4):351–62.
[18] Saux E, Daniel M. An improved Hoschek intrinsic parametrization. Computer Aided Geometric Design 2003;20(8–9):513–21.
[19] Selimovic I. New bounds on the magnitude of the derivative of rational Bézier curves and surfaces. Computer Aided Geometric Design 2005; 22(4):321–6.
[20] Tookey R, Ball A. Approximate G^1 continuous interpolation of a quadrilateral network of rational cubic curves. Computer-Aided Design 1996;28(12):1007–16.
[21] Wang WP, Pottmann H, Liu Y. Fitting B-spline curves to point clouds by curvature-based squared distance minimization. ACM Transaction on Graphics 2006;25(2):214–38.
[22] Yang HP, Wang WP, Sun JG. Control point adjustment for B-spline curve approximation. Computer-Aided Design 2004;36(7):639–52.
[23] Yang YJ, Yong JH, Zhang H, Paul JC, Sun JG. A rational extension of Piegl's method for filling n-sided holes. Computer-Aided Design 2006; 38(11):1166–78.

Yi-Jun Yang is a Ph.D. student in the Department of Computer Science and Technology at Tsinghua University, China. He received his B.S. and M.S. in Computer Science from Shandong University of China in 2000 and 2003, respectively. His research interests are computer-aided design and computer graphics.

Song Cao is a senior student in the School of Software at Tsinghua University, China. His research interests include artificial intelligence, human computer interaction, robotic operation and computer-aided design.

Jun-Hai Yong is currently a professor in the School of Software at Tsinghua University, China. He received his B.S. and Ph.D. in computer science from Tsinghua University, China, in 1996 and 2001, respectively. He held a visiting researcher position in the Department of Computer Science at Hong Kong University of Science & Technology in 2000. He was a post-doctoral fellow in the Department of Computer Science at the University of Kentucky from 2000 to 2002. His research interests include computer-aided design, computer graphics, computer animation, and software engineering.



Hui Zhang is an associate professor in the School of Software at Tsinghua University, China. She received her B.Sc. and Ph.D. in computer science from Tsinghua University, China, in 1997 and 2003, respectively. Her research interests are computer-aided design and computer graphics.

Jean-Claude Paul is a senior researcher at CNRS and INRIA (France), and currently visiting professor at Tsinghua University. He received his Ph.D. in Mathematics from Paris University in 1976. His research interests are numerical analysis, physics-based modelling and computer-aided design.

Jia-Guang Sun is a professor in the School of Software at Tsinghua University, China. His research interests are computer graphics, computer-aided design, computer-aided manufacturing, product data management and software engineering.

He-jin Gu is a senior professor at Jiangxi Academy of Sciences. He was a visiting professor at the State University of New York in USA from 2005 to 2006. He held a visiting professor position at School of Software, Tsinghua University in 2007. He has obtained three Awards of Science and Technology from the ministries of China and Jiangxi Province. His research interests include computer-aided design, computer graphics, and software engineering.