Projection of curves on B-spline surfaces using quadratic reparameterization

Yi-Jun Yang\textsuperscript{a,b,d,*}, Wei Zeng\textsuperscript{c}, Hui Zhang\textsuperscript{d}, Jun-Hai Yong\textsuperscript{d}, Jean-Claude Paul\textsuperscript{b,d}

\textsuperscript{a}GMSV center, King Abdullah University of Science and Technology, Thuwal 23955-6900, Saudi Arabia
\textsuperscript{b}INRIA, Villers Les Nancy 54600, France
\textsuperscript{c}Computer Science Department, Stony Brook University, Stony Brook, NY 11790, USA
\textsuperscript{d}School of Software, Tsinghua University, Beijing 100084, PR China

Abstract

Curves on surfaces play an important role in computer aided geometric design. In this paper, we present a hyperbola approximation method based on the quadratic reparameterization of Bézier surfaces, which generates reasonable low degree curves lying completely on the surfaces by using iso-parameter curves of the reparameterized surfaces. The Hausdorff distance between the projected curve and the original curve is controlled under the user-specified distance tolerance. The projected curve is $\mathcal{G}_1$ continuous, where $\mathcal{G}_1$ is the user-specified angle tolerance. Examples are given to show the performance of our algorithm.

1. Introduction

Curves lying on free form surfaces play an important role in surface blending, surface-surface intersection, surface trimming and NC tool path generation for machining surfaces. 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\textsuperscript{[15,23]}. In the past 20 years, curves on surfaces have been extensively studied\textsuperscript{[3,4,6–11,14–16,20,21,23]}. These methods can be generally categorized into two approaches. The first approach is to compute exact curves on surfaces directly\textsuperscript{[3,4,7–9,11,14,15]} while the second approach generates an approximation of the original exact curves\textsuperscript{[6,10,14–16,20,21]}. The degree of the original exact curves is considerably high, which results in computational demanding evaluations and introduces numerical instability in practice. The approximation approach uses a relatively low degree curve to approximate the original exact curve, with some constraints imposed on the approximate projected curve. However, most approximation algorithms\textsuperscript{[6,10,14–16,20,21]} generate projected curves not lying completely on the surfaces. 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.

The gap problem is also encountered when computing the intersection curve of two B-spline surfaces. A generic intersection curve of two cubic surfaces is degree 324\textsuperscript{[17]}. Hence, intersection curves can only be approximated by parametric trimming curves. Several papers\textsuperscript{[5,18,19]} have been published to address the gaps between the two adjacent surfaces, which assume there already exist two adjacent surfaces. Usually surface perturbation in the vicinity of their intersection is involved to remove the gap between the surfaces. For some CAD applications such as surface blending, this is not the case. In surface blending, a new blending surface is constructed to connect the two base surfaces at the linkage curves (see Fig. 1). In this case, the linkage curves are defined by trimming curves on the
This paper gives the explicit representation of quadratic reparameterized Bézier surfaces.

1. The original curves are approximated by iso-parameter curves of the quadratic reparameterized surfaces satisfying the given distance and angle tolerances.

2. Surface blending.

The organization of this paper is as follows. In Section 2, input and output handling is discussed. Section 3 describes how to preprocess the domain curves. Section 4 gives the explicit representation of the quadratic reparameterized surfaces. Section 5 describes how to generate the approximate hyperbolae and control the Hausdorff distance between the projected curve and the original curve under the user-specified distance tolerance. Tangent discrepancy between adjacent curves is dealt with in Section 6. Results are given in Sections 7 and 8 concludes the paper.

2. Algorithm overview

A B-spline curve is defined by

$$D(t) = \sum_{k=0}^{n_k-1} D_k N_k^d(t),$$

where $D_k$ are the control points and $N_k^d$ are the $d$th-degree B-spline basis functions. A B-spline surface in three dimensional space is defined by

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

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

1. Divide the B-spline surface into Bézier surfaces by knots insertion and approximate the domain curve with hyperbolic curves.
2. Subdivide the domain curve so that the Hausdorff distance between the mapped curves of the hyperbolae and that of the B-spline curve is under the user-specified tolerance $\epsilon_d$.
3. Subdivide the domain curve so that the projected curve is $\epsilon_f$ continuous.
4. Compute the mapped curves of the hyperbolae by evaluating the iso-parameter curves of the quadratic reparameterized Bézier surfaces.

The following sections illustrate how to generate the projected curve.

3. Domain curves

Given a B-spline curve, we divide it into Bézier surfaces by knots insertion (see Fig. 2). Given a B-spline curve
in the parameter domain of the surface (see Fig. 3), we use hyperbolae to approximate it. First, the B-spline curve \( D(t) \) is subdivided as follows. Divide the B-spline curve (see Fig. 4) at

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

After the above subdivision, each segment of curve \( D(t) \) is a Bézier curve defined by

\[
C_i(t) = \sum_{i=0}^{n} B^n_i(t) P_i, \quad 0 \leq t \leq 1,
\]

where \( P_i = (u_i, v_i) \). Each segment is approximated by hyperbolic curves in the following sections.

### 4. Quadratic reparameterization

Before introducing quadratic reparameterization, we first study linear Möbius transformations of Bézier surfaces. A Bézier surface can be represented in the following form

\[
X(u, v) = \sum_{i=0}^{m} \sum_{j=0}^{n} B^m_i(u) B^n_j(v) P_{ij}, \quad u \in [0, 1],
\]

\[
v \in [0, 1],
\]

where \( P_{ij} \) are the control points, \( B^m_i(u) \) and \( B^n_j(v) \) are the Bernstein polynomials. For a Bézier surface, each parameter is subjected to a Möbius transformation as follows.
\[ u = u(s) = \frac{(\alpha - 1)s}{2s - s - \alpha}, \quad \alpha \in (0, 1), \] \tag{2}

and

\[ v = v(t) = \frac{(\beta - 1)t}{2\beta t - t - \beta}, \quad \beta \in (0, 1). \] \tag{3}

Applying the transformations (2) and (3) to surface (1) results in the rational Bézier surface

\[ \mathbf{X}(s, t) = \sum_{i=0}^{m} \sum_{j=0}^{n} B_i^m(s) B_j^n(t) \omega_{ij} \mathbf{P}_{ij}, \quad s \in [0, 1], \quad t \in [0, 1], \]

where \( \omega_{ij} = (1 - \alpha)^i \alpha^{m-i}(1 - \beta)^j \beta^{n-j} \). An example of Möbius transformations of a Bézier surface is given in Fig. 5. Linear Möbius transformations can not change the shape of the parameter lines. What changes is the distribution of the transformed parameter lines.

To introduce the quadratic reparameterization [24], \( \alpha \) and \( \beta \) in Eqs. (2) and (3) are not constants any more. They are redefined as linear interpolations of another parameter as follows

\[ \alpha = \alpha_1 t + \alpha_2 (1-t) \quad \text{and} \quad \beta = \beta_1 s + \beta_2 (1-s), \quad \alpha_1, \alpha_2, \beta_1, \beta_2 \in (0, 1), \] \tag{4}

\[ x \in \text{Eq. (2)} \] is defined as a linear function of \( t \), with coefficients \( \alpha_1 \) and \( \alpha_2 \), and \( \beta \) in Eq. (3) is defined as a linear function of \( s \), with coefficients \( \beta_1 \) and \( \beta_2 \). As a result of more freedom, the surface degree will be raised accordingly. Applying the quadratic transformations (2)-(4) to surface (1) results in the rational Bézier surface

\[ \mathbf{X}(s, t) = \sum_{i=0}^{m+n} \sum_{j=0}^{m+n} B_i^{m+n}(s) B_j^{m+n}(t) \omega_{ik} \mathbf{Q}_{ik}, \quad s \in [0, 1], \quad t \in [0, 1]. \]

The new surface is of degree \( (m+n) \times (m+n) \), where \( m \) and \( n \) are the degrees of the original Bézier surface in the \( u \)-direction and \( v \)-direction, respectively. The control points \( \mathbf{Q}_{ik} \) and their weights \( \omega_{ik} \) of the reparameterized surface (see [24] for detailed deduction) are as follows

\[ \mathbf{Q}_{ik} = \sum_{i=0}^{m+n} \sum_{j=max(k, 0)}^{m+n} \sum_{l=min(k, 0)}^{m+n} c_{ijkl} d_{ijkl} \mathbf{P}_{ij}, \]

and

\[ \tilde{\omega}_{ik} = \sum_{i=max(k, 0)}^{m+n} \sum_{j=max(k, 0)}^{m+n} c_{ijkl} d_{ijkl}. \]

where

\[ R = \left( \begin{array}{ccc} m & n & m \cdot n \\ k_1 & k_2 & m+n \end{array} \right), \]

and

\[ c_{ik} = \sum_{j=0}^{m+n} \left( \begin{array}{ccc} m-k-i & n-j & m \cdot n \cdot j+i \\ k_1 & k_2 & m+n \end{array} \right), \]

and

\[ d_{ik} = \sum_{j=0}^{m+n} \left( \begin{array}{ccc} n-k-i & n-j & m \cdot n \cdot j+i \\ k_1 & k_2 & m+n \end{array} \right). \]

In [24], we give the detailed derivation of the quadratic reparameterization. However, the coefficients \( c_{ik} \) and \( d_{ik} \) of the control points and weights are deduced only for degree \( 2 \times 2 \) and \( 3 \times 3 \) Bézier surfaces. Explicit representations of \( c_{ik} \) and \( d_{ik} \) for Bézier surfaces of any degree are first given in this paper. The coefficients \( c_{ik} \) and \( d_{ik} \) for degree \( 2 \times 2 \) and \( 3 \times 3 \) Bézier surfaces are listed in Appendix A. To give a better understanding of how the parameterization changes in a quadratic reparameterization, an example is given in Fig. 6. As a consequence of introducing additional free parameters, quadratic reparameterization can change the shape of the parameter lines as well as the distribution of the parameter lines. The quadratic reparameterization would not change the shape of the surface. What changes is the surface parameterization. Letting the 3D mapped curves on the surface fixed, the domain curve lying in the parameter domain of the original surface differs from the domain curve lying in the parameter domain of the original surface.

![Fig. 5. Möbius reparameterization of a Bézier surface: (a) Bézier surface and its polynomial parameterization and (b) Bézier surface and its Möbius reparameterized parameterization with the coefficients \( \alpha = 0.30 \) and \( \beta = 0.54 \).](image-url)
reparameterized surface according to the coefficients. An example is given in Fig. 7. A vertical segment in the parameter domain of the reparameterized surface corresponds to a hyperbola in the parameter domain of the original surface.

In order not to raise the degree of the projected curve, our method is to utilize the iso-parameter curves of the reparameterized surface (the domain curve in the parameter domain of the reparameterized surface is a horizontal or vertical line segment while the corresponding domain curve in the parameter domain of the original surface is a hyperbolic curve) to approximate the target curve. As the reparameterized surface is of degree \((m+n)/C^2\) \((m+n)\), the iso-parameter curves are of degree \(m+n\). By introducing quadratic reparameterization, a certain form of hyperbolic curves can be utilized to approximate the target curves in the original surface parameter domain, which can be transformed into vertical or horizontal line segments in the parameter domain of the reparameterized surface by properly choosing the reparameterization coefficients. The details of generating such a hyperbola are described in the next section.

5. Approximate hyperbolic curves

In this section, hyperbolae are utilized to approximate the target curves in the parameter domain of the original surface. According to end point locations of the target curve, approximate hyperbolic curves are constructed as follows. For the sake of computation simplicity, the Bézier surface is first divided or extended according to the two end points such that the two end points lie on opposite edges of the parameter domain of the new surface. If the two end points lie on a horizontal or vertical line, the curve is approximated with a line segment connecting the end points directly. Thus we suppose that the end points of the curve don’t lie on a horizontal or vertical line in the following hyperbola construction procedure.

After the above process of surface extension or subdivision, the two end points of the target curve lie on opposite boundaries of the new surface parameter domain (see Fig. 8). The procedure of how to approximate such a target curve with an appropriate hyperbola is described as follows. Supposing the 3D mapped curve fixed, each iso-parameter line (see Fig. 7b) lying in the parameter domain of the quadratic reparameterized surface corresponds to a hyperbola (see Fig. 7a) in the parameter domain of the original surface. First the optimal iso-parameter line lying in the parameter domain of the quadratic reparameterized surface corresponds to a hyperbola (see Fig. 7a) in the parameter domain of the original surface. First the optimal iso-parameter line lying in the parameter domain of the quadratic reparameterized surface is computed as follows. For simplicity, let the coefficients \(b_2\) and \(b_1\) in Eq. (4) be the same and denote them as \(b\). Here we give the solution of how to approximate the target curve 23 with a hyperbolic curve. Approximating the target curve 24 with a hyperbolic curve can be handled similarly. How to approximate the target curve 23 with a hyperbolic curve is described as follows. The implicit
representation of the approximate hyperbolic curve in the original surface parameter domain is as follows
\[ c_1 u v + c_2 u + c_3 v + c_4 = 0, \]  
(5)

where
\[
\begin{align*}
  c_1 &= -x_2 - s - 2s x_1 - 2s x_2 + \beta x_1 + \beta x_2 + 2s \beta + 2s x_2, \\
  c_2 &= -2s x_2 - s \beta - x_2 \beta + 2s \beta x_2 + s + x_2, \\
  c_3 &= -2s \beta - x_2 + \beta s x_1 + \beta s x_2 + s, \\
  c_4 &= -sx_2 \beta - s + s \beta + s x_2.
\end{align*}
\]  
(6)

The explicit (control points and their weights) representation of the hyperbola can be easily deduced from Eq. (5) when the coefficients are determined. The coefficients are computed as follows.

1. Let
\[ s = (P_{0u} + P_{1u})/2. \]  
(7)

2. Suppose \( P_{0u} = 1 \) and \( P_{1u} = 0 \). That is, the start point lies on the top line of the parameter domain and the end point lies on the bottom line of the parameter domain, respectively. To let the end points of the target curve and the hyperbolic curve be the same, compute the coefficients \( x_1 \) and \( x_2 \) as follows
\[
\begin{align*}
  x_1 &= \frac{u(P_{1u} - 1)}{P_{0u} - P_{1u}}, \\
  x_2 &= \frac{u(P_{0u} - 1)}{P_{1u} - P_{0u} - 1}.
\end{align*}
\]

If \( P_{0u} = 0 \) and \( P_{1u} = 1 \), exchange coefficients \( x_1 \) and \( x_2 \).

3. Coefficient \( \beta \) of the optimal curve can be computed using an iterative method. For illustration, an example is given in Fig. 9. We sample \( \beta \) and the red one is the target curve. The hyperbola can be represented by the following equation
\[ C_2(t) = \frac{(1-t)^3 P_0 + 2(1-t) t \omega_2 P_1 + t^2 P_2}{(1-t)^3 + 2(1-t) t \omega_1 + t^2}. \]  
(8)

From coefficients \( s, x_1, x_2, \beta \) and Eq. (6), we can get the control points and the weights of the approximate curve (8) as follows
\[
\begin{align*}
  P_0 &= \left( \frac{c_2}{c_1}, 0 \right), \quad \omega_0 = 1, \\
  P_1 &= \left( \frac{2c_1 + c_3}{c_1 + 2c_2}, \frac{c_2}{c_1 + 2c_2} \right), \quad \omega_1 = \left| \frac{c_1 + 2c_2}{2\sqrt{c_1^2 + c_2^2}} \right|, \\
  P_2 &= \left( \frac{c_1 + c_2}{c_1 + c_2}, 1 \right), \quad \omega_2 = 1.
\end{align*}
\]

The Hausdorff distance between the approximate curve (a hyperbola) and the target curve (a Bézier curve) can be obtained using an iterative method [25].

A good approximation algorithm should control the Hausdorff distance between the approximate curve and the target curve. To control the Hausdorff distance between the mapped curves of the approximate curve and the target curve under the user-specified tolerance \( \epsilon_D \), we must subdivide the target Bézier curves whose Hausdorff distance to their approximate curves is more than the 2D tolerance \( d \) in the parameter domain, which has been calculated from user-specified tolerance \( \epsilon_D \) in Section 4 of [23] as follows
\[ d = \max_{i,h,k} [\| P_{h,i+1} - P_{h,k} \| + \max_{i,h,k} [\| P_{i,h+1} - P_{i,k} \|]. \]  
(9)

where \( P_{ij} \) are the control points of the located Bézier surface, \( m \) and \( n \) are the degrees of the Bézier surface in the \( u \)-direction and \( v \)-direction, respectively. If \( H(C_2(t), C_2(t)) \leq d \), the Hausdorff distance between \( C_2(t) \) and \( C_1(t) \) is controlled under the user-specified tolerance \( \epsilon_D \), where \( C_2(t) \) and \( C_1(t) \) are the mapped curves of \( C_2(t) \) and \( C_1(t) \) on their located surfaces, respectively. If this condition holds for all Bézier curves in the parameter domain, the Hausdorff distance between the 3D projected curve and the 3D original curve is controlled under \( \epsilon_D \).

If the Hausdorff distance between the approximate curve and the target curve is larger than \( d \), subdivide the target curve at the shoulder point \( C_i(0.5) \). The subdivision procedure is performed repeatedly until the distance tolerance is met for all sub-curves. For the B-spline curve shown in Fig. 3 which is located in the parameter domain of the surface shown in Figs. 2a, 10 shows the approximate hyperbolic and polyline (see the method presented in [23]) curves for \( \epsilon_D = (0.01, 0.001) \), respectively. Also the number of the approximate curves shown in Fig. 10 is listed in Table 1. From Table 1, we can see that the hyperbola approximation method decreases the number of subdivisions evidently compared with the polyline approximation method. How to generate an \( \epsilon_D = G^3 \) continuous projected curve that lies completely on the surface is described in the next section.
6. $\varepsilon_T$-$C^1$ continuous curves

Curves are said to be $\varepsilon_T$-$C^1$ continuous if the maximum tangent discrepancy between any pair of adjacent curves (see Fig. 11) is bounded by the angle tolerance $\varepsilon_T$. The end derivatives of the mapped curve are computed as follows. By substituting approximate curve equation $C_2(t)$ into located surface Eq. (1), we have

$$\frac{dS}{dt}(u, v) = m \sum_{i=0}^{n-1} \sum_{j=0}^{n} B^m_i(u)B^n_j(v)(P_{i+1,j} - P_{i,j}),$$  \hspace{1cm} (12)

and

$$\frac{dS}{dv}(u, v) = n \sum_{i=0}^{m-1} \sum_{j=0}^{n} B^m_i(u)B^n_j(v)(P_{i,j+1} - P_{i,j}).$$  \hspace{1cm} (13)

Letting $t = 0$ and $t = 1$, from Eqs. (11)–(13), we obtain the two end derivatives of the mapped curve $C_2(t)$. If the angle (denoted as $\alpha$) between the end derivatives at the mutual point of two adjacent mapped curves exceeds the angle tolerance $\varepsilon_T$, the Bézier curve with the greater deviation from its corresponding line segment connecting the end points is subdivided at the farthest point. This operation is performed repeatedly until $\alpha < \varepsilon_T$ holds for all the adjacent hyperbolae. After the splitting operations, we should guarantee that Hausdorff distance between the 3D projected curve and the 3D original curve remains under the tolerance $\varepsilon_D$ for each Bézier curve in the parameter do-

![Fig. 10. Approximate hyperbolic and polyline curves: Hyperbolae for: (a) $\varepsilon_D = 0.01$, (b) $\varepsilon_D = 0.0001$ and polylines for (c) $\varepsilon_D = 0.01$, and (d) $\varepsilon_D = 0.0001$.](image-url)
The subdivision process is preformed repeatedly until the angle and distance tolerances are satisfied for all the Bézier curves.

To show the superiority of the hyperbola approximation method over the polyline approximation method in generating $C^1$ continuous curves, an example is given in Fig. 11. For the three Bézier curves (black curves) shown in Fig. 11a, hyperbolae (red curves) and a polyline (green curves) are generated to approximate the Bézier curves, respectively. From Fig. 11a, we can see that the initial hyperbolae not only approximate the original curves better than the polylines but also have smaller tangent discrepancy between adjacent curves in the parameter domain. As a result, after the approximation curves are projected onto a Bézier surface, the 3D projection curves of the hyperbolae will have smaller tangent discrepancy correspondingly (see Fig. 11b). With the angle tolerance considered, for the B-spline curve shown in Fig. 3 which is located in the parameter domain of the surface shown in Fig. 2a, the number of the approximate curve segments are listed in Table 2.

After the final approximate hyperbolae are obtained, we map them to the spatial B-spline surface by computing $s$-$t$ iso-parameter curves of the reparameterized surface. The $s$ parameter values can be obtained from Eq. (7) directly.

### Table 1
Comparison I between algorithms presented in [23] and this paper.

<table>
<thead>
<tr>
<th>Distance tolerance</th>
<th>Number of polyline segments</th>
<th>Number of hyperbolae</th>
</tr>
</thead>
<tbody>
<tr>
<td>$1 \times 10^{-1}$</td>
<td>44</td>
<td>43</td>
</tr>
<tr>
<td>$1 \times 10^{-2}$</td>
<td>74</td>
<td>45</td>
</tr>
<tr>
<td>$1 \times 10^{-3}$</td>
<td>214</td>
<td>64</td>
</tr>
<tr>
<td>$1 \times 10^{-4}$</td>
<td>669</td>
<td>110</td>
</tr>
<tr>
<td>$1 \times 10^{-5}$</td>
<td>2118</td>
<td>230</td>
</tr>
</tbody>
</table>

### Table 2
Comparison II between algorithms presented in [23] and this paper.

<table>
<thead>
<tr>
<th>Distance/angle</th>
<th>Number of polyline segments</th>
<th>Number of hyperbolae</th>
</tr>
</thead>
<tbody>
<tr>
<td>$1 \times 10^{-1}/10^0$</td>
<td>96</td>
<td>45</td>
</tr>
<tr>
<td>$1 \times 10^{-2}/1^0$</td>
<td>443</td>
<td>73</td>
</tr>
<tr>
<td>$1 \times 10^{-3}/1^0$</td>
<td>106</td>
<td>45</td>
</tr>
<tr>
<td>$1 \times 10^{-4}/1^0$</td>
<td>448</td>
<td>69</td>
</tr>
<tr>
<td>$1 \times 10^{-5}/1^0$</td>
<td>214</td>
<td>64</td>
</tr>
<tr>
<td>$1 \times 10^{-3}/1^0$</td>
<td>459</td>
<td>70</td>
</tr>
<tr>
<td>$1 \times 10^{-4}/1^0$</td>
<td>669</td>
<td>110</td>
</tr>
<tr>
<td>$1 \times 10^{-5}/1^0$</td>
<td>700</td>
<td>110</td>
</tr>
<tr>
<td>$1 \times 10^{-3}/1^0$</td>
<td>2118</td>
<td>230</td>
</tr>
<tr>
<td>$1 \times 10^{-5}/1^0$</td>
<td>2118</td>
<td>230</td>
</tr>
</tbody>
</table>

Also, similar $t$ parameter values can be obtained for $t$ iso-parameter curves. As the reparameterized surface is of degree $m + n$, where $m$ and $n$ are the degrees of the B-spline surface in the $u$-direction and $v$-direction, respectively, the final 3D mapped curves are rational curves with degree $m + n$.

### 7. 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 $P_0 = (0.1, 0.1)^T$, $P_1 = (0.5, 1.8)^T$, and $P_2 = (0.8, 0.1)^T$ is mapped onto a biquadratic Bézier surface with 3 by 3 control points. The original curve on the surface [15] has nine control points, as shown in Fig. 12a. The degree reduction algorithm introduced in [15] is implemented to generate one projected curve (see Fig. 12b), where the tolerance is

---

Fig. 11. Tangent discrepancy between curves: (a) three curves (the black curves) and its hyperbolic (the red curves) and polyline (the green lines) approximations in the parameter domain; (b) angles between the end derivatives of the adjacent mapped curves. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)
The results of the polyline and hyperbola approximation methods are given in Fig. 12c and d, respectively, where the distance tolerance is set to $10^{-3}$ and the angle tolerance is set to $10^3$. Results of the four algorithms are given in Table 3. The curves generated by the polyline approximation method, the hyperbola approximation method and the exact algorithm lie completely on the surface, while the degree of the curves generated by the polyline and hyperbola approximation methods is 4, much lower than that of the exact 8th-degree curve. Compared with the polyline approximation method, less subdivisions are introduced in the hyperbola approximation method, which is even less than the degree reduction method. Though the curves generated by the degree reduction algorithm are very close to the surface, they do not lie completely on the surface.

In the second example, a cubic, closed B-spline curve in Fig. 3 with eight control points lying in the parameter domain of the bicubic human face model in Fig. 2a is mapped onto the human face surface with 17 by 17 control points (see Fig. 13). The original curve and its segmentations [15] are shown in Fig. 13a. The result of the degree reduction algorithm in [15] is shown in Fig. 13b, where the target degree is set to 3 and the tolerance is set to $10^{-3}$. Two approximations are also computed using the polyline

![Fig. 12. Exact and approximate images of a quadratic Bézier curve on a biquadratic Bézier surface: (a) the exact curve and its control polygon, the approximate curve and its control polygon generated by (b) the degree reduction algorithm, and (c) the polyline approximation method and (d) the hyperbola approximation method.](image)

| Table 3 | Results for a curve on a Bézier surface. |
| --- | --- | --- | --- |
| Exact Degree reduction Polyn |  |  |  |
| Tolerance | $1 \times 10^{-3}$ | $1 \times 10^{-3}$ | $1 \times 10^{-3}$ |
| Degree | 8 | 3 | 4 |
| Number of control points | 9 | 16 | 33 |
| Number of segments | 1 | 7 | 8 |
| Distance to S | 0 | $0.63 \times 10^{-4}$ | 0 |
| Continuity | $C^2$ | $C^2$ | $10^{-G^3}$ |
| Processor time (ms) | 0.23 | 21 | 15 |
and hyperbola approximation methods (see Fig. 13c and d), where $\epsilon_D$ is set to $10^{-3}$ and $\epsilon_T$ is set to $10^\circ$. Results of the four algorithms are given in Table 4.

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. The original curve on the surface has more than 40 subdivisions, as shown in Fig. 14a. The result of the degree reduction algorithm is shown in Fig. 14b, where the tolerance is set to $10^{-3}$. The results of the polyline and hyperbola approximation methods are given in Fig. 14c and d, respectively, where the distance tolerance to the original curve is set to $10^{-3}$ and the angle tolerance is set to $1^\circ$. Results of the four algorithms are given in Table 5.

The approximation algorithms in [14,15] generate curves that are not completely on the B-spline surface.
Table 4
Results for a curve on a face surface.

<table>
<thead>
<tr>
<th></th>
<th>Exact</th>
<th>Degree reduction</th>
<th>Polyline</th>
<th>Hyperbola</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tolerance</td>
<td>–</td>
<td>$1 \times 10^{-3}$</td>
<td>$1 \times 10^{-3} / 10^6$</td>
<td>$1 \times 10^{-3} / 10^6$</td>
</tr>
<tr>
<td>Degree</td>
<td>18</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>Number of control points</td>
<td>757</td>
<td>220</td>
<td>1279</td>
<td>379</td>
</tr>
<tr>
<td>Number of segments</td>
<td>43</td>
<td>74</td>
<td>214</td>
<td>64</td>
</tr>
<tr>
<td>Distance to S</td>
<td>0</td>
<td>$0.47 \times 10^{-3}$</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Continuity</td>
<td>$G^1$</td>
<td>$G^1$</td>
<td>$10^0 - G^1$</td>
<td>$10^0 - G^1$</td>
</tr>
<tr>
<td>Processor time (ms)</td>
<td>21</td>
<td>155</td>
<td>229</td>
<td>276</td>
</tr>
</tbody>
</table>

Fig. 14. Exact and approximate images of a cubic domain curve on a bicubic B-spline surface of an ear face: (a) the exact curve and its segmentations, the approximate curve and its segmentations generated by (b) the degree reduction algorithm, (c) the polyline approximation method, and (d) the hyperbola approximation method.

Table 5
Results for a curve on an ear surface.

<table>
<thead>
<tr>
<th></th>
<th>Exact</th>
<th>Degree reduction</th>
<th>Polyline</th>
<th>Hyperbola</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tolerance</td>
<td>–</td>
<td>$1 \times 10^{-3}$</td>
<td>$1 \times 10^{-3} / 10^6$</td>
<td>$1 \times 10^{-3} / 10^6$</td>
</tr>
<tr>
<td>Degree</td>
<td>18</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>Number of control points</td>
<td>825</td>
<td>227</td>
<td>1225</td>
<td>571</td>
</tr>
<tr>
<td>Number of segments</td>
<td>47</td>
<td>89</td>
<td>204</td>
<td>96</td>
</tr>
<tr>
<td>Distance to S</td>
<td>0</td>
<td>$0.46 \times 10^{-3}$</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Continuity</td>
<td>$G^1$</td>
<td>$G^1$</td>
<td>$1^0 - G^1$</td>
<td>$1^0 - G^1$</td>
</tr>
<tr>
<td>Processor time (ms)</td>
<td>22</td>
<td>232</td>
<td>308</td>
<td>356</td>
</tr>
</tbody>
</table>
Both the polyline and hyperbola approximation methods generate projected curves lying completely on the surface. From Tables 3–5, compared with the polyline approximation method, the hyperbola approximation method reduces the number of the subdivisions to an acceptable extent, which can be comparable with that of the degree reduction method. Furthermore, by using a quadratic reparameterization technique, the hyperbola approximation method reduces the number of curve subdivisions without degree elevation, while the generated curves lie 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 the hyperbola approximation method is comparable with that of the degree reduction algorithm in [15], which satisfies the real-time needs of CAD applications. Also the curves generated in our algorithm are $C^1$-continuous. For many applications such as NC-machining and surface blending, it is sufficient for surfaces to be $C^1$-continuous. Engineering practice suggests that there is no ambiguity within acceptable working tolerance. The geometric discrepancies in the $C^1$ surface models are very small and may be blended out in the subsequent manufacturing processes.

8. Conclusions

Based on the quadratic reparameterization of Bézier surfaces, an approximation algorithm for computing a curve on a B-spline surface has been presented in this paper. First the initial hyperbolic approximation of the domain curve is generated. The domain curve is then subdivided repeatedly until the Hausdorff distance between the projected curve and the original curve is under the distance tolerance, after which the angle deviation between the projected curve and the original curve is controlled under the user-specified tolerance by another subdivision procedure. The main technique of our method is that we utilize iso-parameter curves of the reparameterized surfaces to reduce the curve subdivisions while keeping the degree of the mapped curves unchanged compared with the polyline approximation method. Compared with original curves, the degree of curves generated by our algorithm is much lower. Moreover our algorithm generates curves that lie completely on the B-spline surface, which is indispensable for many CAD applications.

Acknowledgments

The research was supported by Chinese 973 Program (2010CB328001) and the National Science Foundation of China (60625202). The fourth author was supported by ANR-NSFC (6091130368) and the Fok Ying Tung Education Foundation (111070).

Appendix A

The coefficients $c_{jk}$ and $d_{jk}$ of the quadratic reparameterization for a $2 \times 2$ degree Bézier surface are listed as follows.

\[
\begin{align*}
&c_{0,0} = a_0^2, \\
&c_{1,0} = a_0^2a_1, \\
&c_{2,0} = a_0^2a_2, \\
&c_{3,0} = a_0^2a_3, \\
&c_{0,1} = -a_0(2a_2 - 1), \\
&c_{1,1} = -2a_0a_1 + 2a_2 + a_3, \\
&c_{2,1} = -a_0(2a_1 - 1), \\
&c_{3,1} = a_0^2, \\
&c_{0,2} = (a_2 - 1)^2, \\
&c_{1,2} = (a_2 - 1)(a_1 - 1), \\
&c_{2,2} = (a_1 - 1)^2, \\
&c_{3,2} = a_2^2, \\
&d_{0,0} = \beta_0^2, \\
&d_{1,0} = \beta_0^2\beta_1, \\
&d_{2,0} = \beta_0^2, \\
&d_{3,0} = \beta_0^2, \\
&d_{0,1} = -\beta_0(2\beta_2 - 1), \\
&d_{1,1} = -2\beta_0\beta_1 + 2\beta_2 + \beta_3, \\
&d_{2,1} = -\beta_0(\beta_1 + 1), \\
&d_{3,1} = \beta_0^2, \\
&d_{0,2} = (2\beta_2 - 1)^2, \\
&d_{1,2} = (2\beta_2 - 1)(2\beta_1 - 1), \\
&d_{2,2} = (2\beta_1 - 1)^2, \\
&d_{3,2} = (\beta_1 - 1)^2, \\
&d_{0,3} = (\beta_2 - 1)^3, \\
&d_{1,3} = (\beta_1 - 1)^2(\beta_2 - 1), \\
&d_{2,3} = (\beta_1 - 1)^2, \\
&d_{3,3} = (\beta_1 - 1)^3.
\end{align*}
\]

References