Computers and Mathematics with Applications 68 (2014) 706–718

Contents lists available at ScienceDirect

Computers and Mathematics with Applications

journal homepage: www.elsevier.com/locate/camwa

Semi-structured *B*-spline for blending two *B*-spline surfaces

Hongwei Lin *, Yunyang Xiong, Hongwei Liao

Department of Mathematics, State Key Lab. of CAD&CG, Zhejiang University, Hangzhou, 310027, China

ARTICLE INFO

Article history:
Received 28 November 2013
Received in revised form 2 June 2014
Accepted 19 July 2014
Available online 15 August 2014

Keywords:
Semi-structured *B*-splines
Surface blending
G^2 continuity
Optimization
Geometric design

ABSTRACT

Surface blending is a useful operation in geometric design for rounding sharp edges or corners. Meanwhile, NURBS has already become the de facto industrial standard in existing CAD/CAM systems. Therefore, it is required to study how to blend two *B*-spline surfaces. However, two arbitrary *B*-spline surfaces (called *base surfaces*) are hard to be blended with a *B*-spline surface (called *blending surface*) because the knot vectors of the two base surfaces are usually mismatched. In this paper, we proposed a *curve-based spline* representation, i.e., the *semi-structured B-spline surface*, which is generated by skinning a series of *B*-spline curves with *different* knot vectors. By assigning suitable knot vectors to the head and tail skinned curves, the semi-structured *B*-spline surface can blend two *B*-spline surfaces smoothly without disturbing them at all. We formulated the *B*-spline surface blending problem as an optimization problem with continuity constraints, and the continuity between the base and blending surfaces can reach G^2 or C^2. Examples illustrated in this paper validate the effectiveness and efficiency of our method.

1. Introduction

In geometric design, sharp edges are often required to be rounded by the surface blending operation, for aesthetic or functional reasons. The blending operation generates a *blending surface*, which connects two or more surfaces, called *base surfaces*. In addition, the curves where the blending surface meets the base surfaces are called *contact curves*. Although some existing methods can blend two parametric base surfaces, the generated parametric blending surface is hard to be exactly transformed into *B*-spline representation [1,2]. Since NURBS has already become the de facto industrial standard in existing CAD/CAM systems [3], it is needed to study how to blend two *B*-spline surfaces using the *B*-spline represented blending surface for data exchanging and further processing.

The main difficulty in blending two *B*-spline base surfaces with the *B*-spline blending surface lies that, the knot vectors of the two contact curves, which are *B*-spline curves, are mismatched in general. A direct solution for tackling this problem is to make the two knot vectors identical by knot insertion. However, it will lead to lots of superfluous knots and control points, which will complicate the following processing, and even worsen the fairness of the *B*-spline surfaces.

To conquer the difficulty in blending two *B*-spline surfaces aforementioned, in this paper, we design a *curve-based spline* representation, i.e., the *semi-structured B-spline surface*, which is generated by skinning a series of *B*-spline curves with *different* knot vectors. Therefore, the semi-structured *B*-spline can blend two *B*-spline surfaces smoothly without disturbing them at all, by assigning suitable knot vectors to the head and tail skinned curves of the semi-structured *B*-spline surface.

The structure of this paper is as follows. The related work is reviewed briefly in Section 1.1. In Section 2, we present the definition of the semi-structured *B*-spline surface, and discuss its evaluation and properties. In Section 3, the semi-structured

* Corresponding author. Tel.: +86 571 87951860 8304; fax: +86 571 88206681.
E-mail addresses: hwlin@zju.edu.cn, hwlin@zjucadcg.cn (H. Lin).

http://dx.doi.org/10.1016/j.camwa.2014.07.013
0898-1221/

B-spline surface is employed to blend two B-spline surfaces. Section 4 introduces the implementation detail. Moreover, Section 5 illustrates some examples and lists statistics. Finally, Section 6 concludes the paper.

1.1. Related work

In the following, we will briefly review the work on blending methods and surface representations.

Blending methods: Existing blending methods can be classified into two types according to the number of base surfaces: two-surface blending, and multiple surface blending. For the multiple surface blending methods, please refer to Ref. [2] and the references therein. In the following, we will briefly review the two-surface blending methods.

A classical method for the two-surface blending is the rolling-ball method, which was invented by Choi and Ju in [4] using a constant radius rolling-ball, and further extended in [5] with variable radius rolling-balls. Moreover, the constant radius rolling-ball blending surface was approximated by rational tensor-product patches [6]. The differential properties of the variable radius rolling-ball blending surface was analyzed on the basis of the theory of envelopes and discriminant sets [7]. In addition, Ref. [8] described varieties of the rolling-ball blending in kernel boundary modelers emphasizing topology, algorithms and the program structure. However, not only the rolling-ball blending surfaces can only reach G^1 continuity with the base surfaces, but the flexibility is low as well, because the generated blending surface is fixed to be a circular surface.

Moreover, Belkhatir et al. developed a method to maintain the C^1G^2 continuity of the Bézier curve shape-blending process by adjusting the control points of such curves [9]. Later, this method was extended to reach C^1G^2 continuity in blending two Bézier surfaces [10]. In [11], two parametric surfaces were blended by the Hermite interpolation with a Bézier curve or a C-Curve, respectively. To attain G^2 continuity between the base and blending surfaces, the Bézier curve is required to be quintic, and the blending surface is hard to be transformed into the B-spline representation [11]. However, by our method, to reach G^2 continuity between the base and blending surfaces, the blending semi-structured B-spline surface is just cubic, and it is easy to be transformed into the B-spline representation. In addition, a method was presented in [12] to construct G^1 Bézier surfaces over a boundary curve network with T-junctions.

To our knowledge, there is only one work which employs B-spline surface as blending surface to blend two B-spline base surfaces [13]. However, the method presented in [13] not only requires the two B-spline base surfaces have the same knot vector in the blending direction, but the generated blending surface can reach just G^1 (or C^1) continuity with base surfaces.

According to the extensive survey on blending methods [1], though there are lots of methods focused on parametric blending, a few of them can generate G^2 or higher order continuous parametric blending surface with base parametric surfaces. To raise the continuity between the blending and base surfaces to G^2 (or C^2), Bloor and Wilson developed a blending method by solving partial differential equations [14,15]. However, the analytical solution of a partial differential equation is hard to be obtained, and it is usually solved by numerical methods to get approximate solution. Although the method presented in [16] can generate blending with high-order continuity in theory, it needs higher-order derivatives of the surface to be blended, whose closed form formula may not be computationally viable.

Moreover, Erich Hartmann presented a blending method to generate G^n parametric blending surfaces by a linear combination of base surfaces along one of the common parameters [17]. This method was improved by Song and Wang [18], developing a partial reparameterization method of base surfaces. Although these methods can generate blending surfaces with high order continuity with base surfaces, the generated blending surface can only be approximately transformed into a NURBS representation in general.

Surface representations: The semi-structured B-spline is a generalization of the classical B-spline. In the past years, B-spline has been extended to several representations, including the semi-regular B-spline [19,20], T-spline [21], and the spline proposed by Hayes [22,23]. While the semi-structured B-spline is a curve-based spline, the semi-regular B-spline [19,20] and T-spline [21] are point-based splines, and allow a knot line to terminate inside the parametric domain. On the other hand, though the spline proposed by Hayes [22,23] and the semi-structured B-spline are both curve-based splines, and both permit different knot vectors in one parametric direction, Hayes' spline needs the numbers of knots in different knot vectors are identical, while semi-structured B-spline allows that different knot vectors have different number of knots. Finally, though Hu et al. studied the problem of subdividing a Bézier patch along a Bézier function [24,25], they did not note the semi-structure of the subdivided patches.

2. Semi-structured *B*-spline surfaces

It is well known that the control points of a B-spline surface can be arranged as a grid. Each row of the grid has the same number of control points; so does each column. Therefore, the classical B-spline surface is a patch-based spline with structured control grid. Unlike the classical B-spline surface, the semi-structured B-spline surface is a *curve-based* spline surface, which is defined as follows.

Definition 1. The semi-structured B-spline surface is represented as,

$$\boldsymbol{P}(u,v)=\sum_{i=0}^{m}\boldsymbol{P}_i(v)B_{i,k}(u;U)=\sum_{i=0}^{m}\left(\sum_{j=0}^{n_i}\boldsymbol{P}_{ij}B_{j,l}(v;V_i)\right)B_{i,k}(u;U),\quad (u,v)\in[u_s,u_e]\times[v_s,v_e], \tag{1}$$

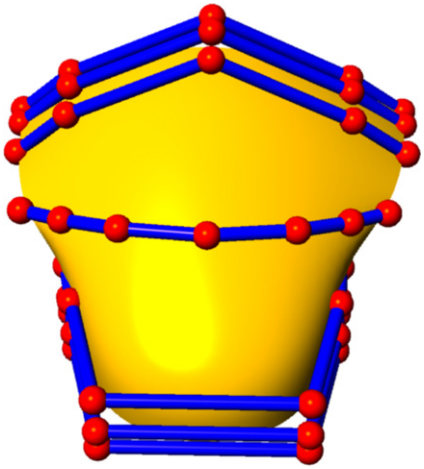

Fig. 1. A semi-structured B-spline surface with its control points.

where, $\boldsymbol{P}_i(v) = \sum_{j=0}^{n_i} \boldsymbol{P}_{ij} B_{j,l}(v; V_i)$ is called a v-control curve, $\{B_{i,k}(u; U),\ i = 0, 1, \ldots, m\}$ are B-spline basis functions of order k, defined on the knot vector,

$$U = \{u_0, u_1, \ldots, u_{m+k}\};$$

$\{B_{j,l}(v; V_i),\ j = 0, 1, \ldots, n_i\}$ are B-spline basis functions of order l, defined on the knot vector,

$$V_i = \{v_0^i, v_1^i, \ldots, v_{n_i+l}^i\}.$$

It can be seen from the above definition that, the knot vectors V_i of *v-control curves* $\boldsymbol{P}_i(v),\ i = 0, 1, \ldots, m$ can be different.

The generation of the semi-structured B-spline surface (1) can be considered as a skinning procedure. First, we generate a series of *v-control curves*,

$$\boldsymbol{P}_i(v) = \sum_{j=0}^{n_i} \boldsymbol{P}_{ij} B_{j,l}(v; V_i), \quad v \in [v_s, v_e],\ i = 0, 1, \ldots, m, \tag{2}$$

where each curve has different knot vector V_i, and then, the semi-structured B-spline surface (1) is the blending of these v-control curves, i.e.,

$$\boldsymbol{P}(u, v) = \sum_{i=0}^{m} \boldsymbol{P}_i(v) B_{i,k}(u; U), \quad (u, v) \in [u_s, u_e] \times [v_s, v_e]. \tag{3}$$

In this meaning, the semi-structured B-spline surface (1) is a curve-based spline. For clarity, we illustrate a semi-structured B-spline surface with its control points along each v-control curve in Fig. 1, where the numbers of the control points of its 7 pieces of v-control curves are 5, 5, 5, 7, 6, 6, 6, respectively.

Example: The following example shows that the semi-structured surface exists naturally. Consider a Bézier patch,

$$\boldsymbol{P}(u, v) = \sum_{i=0}^{m} \sum_{j=0}^{n} \boldsymbol{P}_{ij} B_i^m(u) B_j^n(v), \quad (u, v) \in [0, 1] \times [0, 1], \tag{4}$$

where, $B_i^m(u)$, and $B_j^n(v)$ are Bernstein polynomials. If its domain is divided into two sub-domains, namely D_1 and D_2 (see Fig. 2(a)), by a Bézier function,

$$v(t) = \sum_{i=0}^{k} v_i B_i^k(t), \quad t \in [0, 1],\ v_i \in [0, 1],\ i = 0, 1, \ldots, k, \tag{5}$$

with control point $(i/k, v_i)$, the original Bézier patch (4) is correspondingly divided into two sub-patches $\boldsymbol{P}_1(u, v)$ and $\boldsymbol{P}_2(u, v)$, which are semi-structured patches, respectively (Fig. 2(b)).

In fact, as illustrated in Fig. 2(b), the representations of the bottom sub-patch $\boldsymbol{P}_1(u, v)$ and the top sub-patch $\boldsymbol{P}_2(u, v)$ can be written as,

$$\boldsymbol{P}_1(u, v) = \sum_{s=0}^{n} \boldsymbol{Q}_0^s(u) B_s^n(v), \quad \text{and} \quad \boldsymbol{P}_2(u, v) = \sum_{s=0}^{n} \boldsymbol{Q}_s^{n-s}(u) B_s^n(v), \tag{6}$$

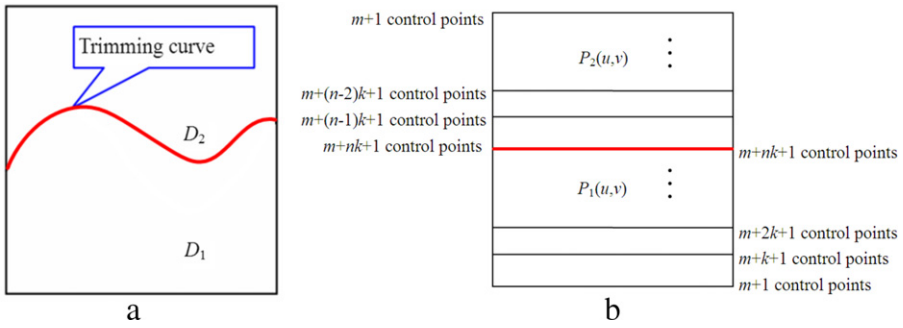

Fig. 2. Generate semi-structured Bézier surfaces by dividing the domain of a Bézier surface into two sub-domains. (a) Dividing the domain by a Bézier function. (b) The numbers of the control points in each row of the control nets of two semi-structured Bézier surfaces.

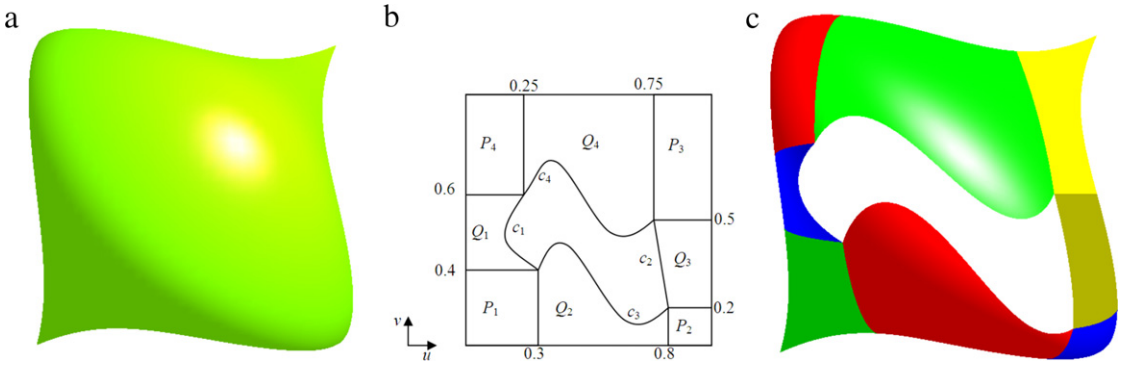

Fig. 3. The semi-structured Bézier patch can exactly represent a trimmed patch. (a) The original bi-cubic Bézier patch. (b) The parametric domain with trimming curves. (c) The trimmed patch, where Q_1, Q_2, Q_3, and Q_4 are semi-structured Bézier patches.

where,

$$\boldsymbol{Q}_j^s(u) = \sum_{l=0}^{m+sk} \sum_{i+j_1+\cdots+j_s=l} \frac{\binom{m}{i}\prod_{r=1}^{s}\binom{k}{j_r}}{\binom{m+sk}{i+j_1+\cdots+j_s}} \boldsymbol{P}_{ij}^s[v_{j_1}, v_{j_2}, \ldots, v_{j_s}] B_{i+j_1+\cdots+j_s}^{m+sk}(u) \quad \text{with } j = 0, 1, \ldots, n-s,\ s = 1, 2, \ldots, n.$$

Here, $\boldsymbol{P}_{ij}^s[v_{j_1}, v_{j_2}, \ldots, v_{j_s}]$ are the ith column blossom [26,27] of the Bézier patch (4), which is defined recursively, for each column $i = 0, 1, \ldots, m$,

$$\begin{aligned}
&(1)\ \boldsymbol{P}_{ij}^1[v_{j_1}] = (1 - v_{j_1})\boldsymbol{P}_{ij} + v_{j_1}\boldsymbol{P}_{i,j+1}, \quad j = 0, 1, \ldots, n-1;\\
&(2)\ \boldsymbol{P}_{ij}^s[v_{j_1}, v_{j_2}, \ldots, v_{j_s}] = (1 - v_{j_s})\boldsymbol{P}_{ij}^{s-1}[v_{j_1}, \ldots, v_{j_{s-1}}] + v_{j_s}\boldsymbol{P}_{i,j+1}^{s-1}[v_{j_1}, \ldots, v_{j_{s-1}}],\\
&\qquad\qquad j = 0, 1, \ldots, n-s,\ s = 2, 3, \ldots, n.
\end{aligned}$$

Referring to Eq. (6) and Fig. 2(b), the bottom sub-patch $\boldsymbol{P}_1(u, v)$ is the blending of $n + 1$ pieces of Bézier curves $\boldsymbol{Q}_0^0(u), \boldsymbol{Q}_0^1(u), \ldots, \boldsymbol{Q}_0^n(u)$. The first curve $\boldsymbol{Q}_0^0(u)$ has $m + 1$ control points; the second curve $\boldsymbol{Q}_0^1(u)$ has $m + k + 1$ control points; $\cdots$; the last curve $\boldsymbol{Q}_0^n(u)$ has $m + nk + 1$ control points. Therefore, the bottom sub-patch is a semi-structure patch. So is the top sub-patch $\boldsymbol{P}_2(u, v)$ (Fig. 2(b)). It is also the blending of $n + 1$ pieces of Bézier curves $\boldsymbol{Q}_0^n(u)$ with $m + nk + 1$ control points, $\boldsymbol{Q}_1^{n-1}(u)$ with $m + (n-1)k + 1$ control points, $\cdots$, and $\boldsymbol{Q}_n^0(u)$ with $m + 1$ control points.

To present a concrete example, we employ the semi-structured Bézier patch to exactly represent a trimmed patch. As illustrated in Fig. 3, the original Bézier patch $\boldsymbol{P}(u, v)$, $(u, v) \in [0, 1] \times [0, 1]$ in Fig. 3(a) is trimmed out a region enclosed by the following four Bézier functions (refer to Fig. 3(b)):

$$\begin{aligned}
&c_1 : u = 0.3B_0^2\left(\tfrac{v-0.4}{0.2}\right) + 0.1B_1^2\left(\tfrac{v-0.4}{0.2}\right) + 0.25B_2^2\left(\tfrac{v-0.4}{0.2}\right), \quad v \in [0.4, 0.6],\\
&c_2 : u = 0.8B_0^1\left(\tfrac{v-0.2}{0.3}\right) + 0.75B_1^1\left(\tfrac{v-0.2}{0.3}\right), \quad v \in [0.2, 0.5],\\
&c_3 : v = 0.4B_0^3\left(\tfrac{u-0.3}{0.5}\right) + 0.7B_1^3\left(\tfrac{u-0.3}{0.5}\right) + 0.1B_2^3\left(\tfrac{u-0.3}{0.5}\right) + 0.2B_3^3\left(\tfrac{u-0.3}{0.5}\right), \quad u \in [0.3, 0.8],\\
&c_4 : v = 0.6B_0^3\left(\tfrac{u-0.25}{0.5}\right) + 0.8B_1^3\left(\tfrac{u-0.25}{0.5}\right) + 0.3B_2^3\left(\tfrac{u-0.25}{0.5}\right) + 0.5B_3^3\left(\tfrac{u-0.25}{0.5}\right), \quad u \in [0.25, 0.75].
\end{aligned}$$

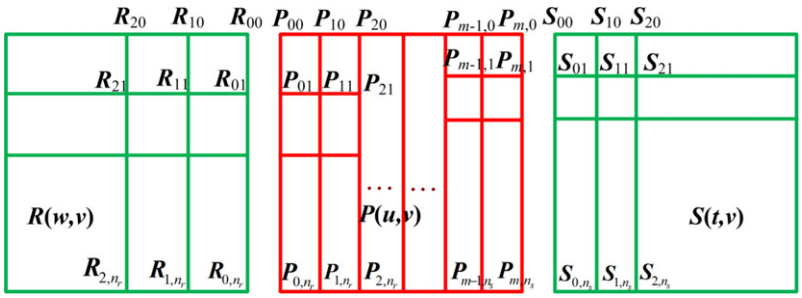

Fig. 4. The semi-structured B-spline surface $\boldsymbol{P}(u, v)$ is employed to blend two base B-spline surfaces $\boldsymbol{R}(w, v)$ and $\boldsymbol{S}(t, v)$. Note that the v-directional knot vectors of $\boldsymbol{R}(w, v)$ and $\boldsymbol{S}(t, v)$ are different.

To generate the trimmed patch, the original Bézier patch (Fig. 3(a)) is first segmented using de Casteljau algorithm at $v = 0.4$ and $v = 0.6$, producing the sub-patch $\bar{\boldsymbol{P}}(u, v)$, $(u, v) \in [0, 1] \times [0.4, 0.6]$, which contains the trimming curve c_1 (Fig. 3(b)). Then, the sub-patch $\bar{\boldsymbol{P}}(u, v)$ is segmented at the trimming curve c_1, resulting in two semi-structured patches. One of them is the patch Q_1 (Fig. 3(b) and (c)). In this way, we get the exact representation of the trimmed patch Q_1, which is a semi-structured patch. Similarly, the exact representations of the other three patches, Q_2, Q_3, and Q_4 can be produced, which are all semi-structured patches (Fig. 3(b)). In addition, the four patches P_1, P_2, P_3, and P_4 (Fig. 3(b)) can be generated using the classical de Casteljau algorithm, which are all regular Bézier patches (Fig. 3(c)).

Evaluation: The classical B-spline surface has a structured control grid, so it can be evaluated by the de Boor algorithm [28] along arbitrary parameter order, i.e., first u-direction, and then v-direction; or first v-direction, and then u-direction. Similarly, the semi-structured B-spline surface can also be evaluated by the de Boor algorithm, with the parameter order for evaluation specified by the definition of the semi-structured B-spline surface. For example, if we want to evaluate the value at (u_0, v_0) of the semi-structured B-spline surface defined by Eq. (1), the evaluation should be performed first along v-direction, calculating a series of points $\boldsymbol{P}_i(v_0)$ on the series of curves $\boldsymbol{P}_i(v)$ from Eq. (2), and then, performed along u-direction, generating the point $\boldsymbol{P}(u_0, v_0)$.

Property: It can be derived easily from the definition of the semi-structured B-spline surface (Eq. (1)) that, the semi-structured B-spline surface holds the majority of properties of the classical B-spline surface, such as, convex-hull, affine invariance, localness, etc. We only mention some differences between the semi-structured and classical B-spline in the following.

- Continuity control: According to Eqs. (1)–(3), since each curve $\boldsymbol{P}_i(v)$ has its own knot vector, the continuity at each knot of $\boldsymbol{P}_i(v)$ can be controlled independently. It is more flexible than the classical B-spline surface.
- Transformation: Moreover, the semi-structured B-spline surface can easily be transformed into the classical B-spline representation by the knot insertion.
- Derivatives: However, the semi-structured B-spline surface definition (1) also brings some inconveniences. For example, while the partial derivatives of the classical B-spline surface (1) can still be represented as B-spline forms with lower degrees, the u-directional partial derivatives of the semi-structured B-spline surface (1) are hard to be represented as B-spline forms.

3. *B*-spline surface blending

Suppose we are given two base B-spline surfaces with the same order along v-direction, (see Fig. 4),

$$\boldsymbol{R}(w, v) = \sum_{i=0}^{m_r} \sum_{j=0}^{n_r} \boldsymbol{R}_{ij} B_{j,l}(v) B_{i,k_r}(w), \quad (w, v) \in [w_s, w_e] \times [v_s, v_e], \tag{7}$$

defined on the knot vectors,

$$W = \{w_0, w_1, \ldots, w_{m_r+k_r}\}, \quad \text{and} \quad V^r = \{v_0^r, v_1^r, \ldots, v_{n_r+l}^r\}, \tag{8}$$

and,

$$\boldsymbol{S}(t, v) = \sum_{i=0}^{m_s} \sum_{j=0}^{n_s} \boldsymbol{S}_{ij} B_{j,l}(v) B_{i,k_s}(t), \quad (t, v) \in [t_s, t_e] \times [v_s, v_e], \tag{9}$$

defined on the knot vectors,

$$T = \{t_0, t_1, \ldots, t_{m_s+k_s}\}, \quad \text{and} \quad V^s = \{v_0^s, v_1^s, \ldots, v_{n_s+l}^s\}, \tag{10}$$

which have Bézier end conditions, i.e.,

$$\begin{aligned}
&w_0 = w_1 = \cdots = w_{k_r-1}, \qquad w_{m_r+k_r} = w_{m_r+k_r-1} = \cdots = w_{m_r+1};\\
&v^r_0 = v^r_1 = \cdots = v^r_{l-1}, \qquad v^r_{n_r+l} = v^r_{n_r+l-1} = \cdots = v^r_{n_r+1};\\
&t_0 = t_1 = \cdots = t_{k_s-1}, \qquad t_{m_s+k_s} = t_{m_s+k_s-1} = \cdots = t_{m_s+1};\\
&v^s_0 = v^s_1 = \cdots = v^s_{l-1}, \qquad v^s_{n_s+l} = v^s_{n_s+l-1} = \cdots = v^s_{n_s+1}.
\end{aligned}$$

The two B-spline surfaces are required to be blended with $\boldsymbol{R}(w_s, v)$ and $\boldsymbol{S}(t_s, v)$ as contact curves (Fig. 4).

Although the domains of the two contact curves are both $[v_s, v_e]$, their knot vectors are mismatched, i.e., the knot vector of one contact curve $\boldsymbol{R}(w_s, v)$ is $V^r = \{v^r_0, v^r_1, \ldots, v^r_{n_r+l}\}$, and that of the other contact curve $\boldsymbol{S}(t_s, v)$ is $V^s = \{v^s_0, v^s_1, \ldots, v^s_{n_s+l}\}$, where, $v^r_{l-1} = v^s_{l-1} = v_s$, $v^r_{n_r+1} = v^s_{n_s+1} = v_e$. Therefore, the two base B-spline surfaces (7) and (9) cannot be blended by a classical B-spline surface directly.

However, using the semi-structured B-spline surface $\boldsymbol{P}(u, v)$, the two base B-spline surfaces (7) and (9) can be blended naturally, without disturbing the two base surfaces. Actually, we want to generate a blending semi-structured B-spline surface which stitches continuously (i.e., C^1, G^1, C^2, or G^2) with the two base surfaces, while its quality is as high as possible. In geometric design, the quality of a surface $\boldsymbol{P}(u, v)$ is generally measured by the strain energy $E(\boldsymbol{P}(u, v))$, i.e., the integral of the squared principal curvatures over the parameter space [29]. The smaller the energy $E(\boldsymbol{P}(u, v))$, the better the quality of the surface $\boldsymbol{P}(u, v)$. Because the computation of the strain energy is considerable, it is usually approximated by the thin plate energy [29], i.e.,

$$E(\boldsymbol{P}(u, v)) = \int_{u_s}^{u_e}\int_{v_s}^{v_e}\left(\left\|\frac{\partial^2\boldsymbol{P}(u, v)}{\partial u^2}\right\|^2 + 2\left\|\frac{\partial^2\boldsymbol{P}(u, v)}{\partial u\partial v}\right\|^2 + \left\|\frac{\partial^2\boldsymbol{P}(u, v)}{\partial v^2}\right\|^2\right)dudv. \tag{11}$$

Therefore, the generation of the blending semi-structured B-spline surface can be formulated as a constrained optimization problem:

$$\begin{aligned}
&\min_{\boldsymbol{P}_{ij}} E(\boldsymbol{P}(u, v))\\
&s.t.\ \text{continuity condition},
\end{aligned} \tag{12}$$

where, the objective function is the thin plate energy (11) of the semi-structured B-spline surface $\boldsymbol{P}(u, v)$.

To make the blending reach G^2 or C^2, the semi-structured B-spline surface $\boldsymbol{P}(u, v)$ is defined as follows (Fig. 4):

$$\boldsymbol{P}(u, v) = \sum_{i=0}^{m}\boldsymbol{P}_i(v)B_{i,k}(u; U) = \sum_{i=0}^{m}\left(\sum_{j=0}^{n_i}\boldsymbol{P}_{ij}B_{j,l}(v; V_i)\right)B_{i,k}(u; U), \quad (u, v) \in [0, 1]\times[v_s, v_e], \tag{13}$$

where, $n_0 = n_1 = n_2 = n_r$, and $V_0 = V_1 = V_2 = V^r$ (8); $n_{m-2} = n_{m-1} = n_m = n_s$, and $V_{m-2} = V_{m-1} = V_m = V^s$ (10); on the other hand,

$$U = \{\underbrace{0, \ldots, 0}_{k}, u_1, u_2, \ldots, u_{m-k+1}, \underbrace{1, \ldots, 1}_{k}\}, \quad (m \geq k+1).$$

In other words, the blending semi-structured B-spline surface $\boldsymbol{P}(u, v)$ (13) is so constructed that, the order and knot vector of the first three v-control curves $\boldsymbol{P}_i(v)$, $i = 0, 1, 2$, are the same as those of the contact curve on the base surface $\boldsymbol{R}(w, v)$; the order and knot vector of the last three v-control curves $\boldsymbol{P}_i(v)$, $i = m-2, m-1, m$, are the same as those of the contact curve on the base surface $\boldsymbol{S}(t, v)$ (Fig. 4). In our implementation, u_i are chosen uniformly, i.e., $u_i = \frac{i}{m-k+2}$, $i = 1, 2, \ldots, m-k+1$.

In general, the less control points the blending surface has, the better its quality is. Then, the control points of a blending surface should be as few as possible to generate a high quality blending surface. Therefore, in our implementation, we employ two kinds of blending surfaces: One has no middle v-control curve; the other has just one middle v-control curve $\boldsymbol{P}_3(v)$.

When there is one middle v-control curve $\boldsymbol{P}_3(v)$, its knot vector should reflect the influences of the first three and last three v-control curves. Therefore, the knot vector of the v-control curve $\boldsymbol{P}_3(v)$ is constructed in the following manner. First, we merge the two knot vectors

$$\{v^r_l, v^r_{l+1}, \ldots, v^r_{n_r}\}, \quad \text{and} \quad \{v^s_l, v^s_{l+1}, \ldots, v^s_{n_s}\},$$

and make the multiple knots be single, leading to a knot vector

$$V_n = \{v_1, v_2, \ldots, v_n\}.$$

Next, the inner knots of the knot vector of $\boldsymbol{P}_3(v)$ are generated by averaging adjacent knots of V_n, i.e., $\frac{v_{2i-1}+v_{2i}}{2}$. Finally, the so generated partial knot vector is made to be Bézier end conditions by adding l-multiple v_s and l-multiple v_e. After constructing the knot vector of $\boldsymbol{P}_3(v)$, its control points can be determined by solving the optimization problem (12).

Geometric continuity condition: The optimization problem (12) is constrained by the continuity condition [30]. In the following, we list the sufficient conditions for G^0, G^1, and G^2 continuity between the base surfaces (7), (9) and the blending surface (13) (refer to Fig. 4), and append their derivation in the Appendix.

- G^0 or C^0 continuity condition:
  (1.1) $\boldsymbol{P}_{0j} = \boldsymbol{R}_{0j},\ j = 0, 1, \ldots, n_r$,
  (1.2) $\boldsymbol{P}_{m,j} = \boldsymbol{S}_{0j},\ j = 0, 1, \ldots, n_s$;
- G^1 continuity condition: In addition to (1.1) and (1.2), they should also satisfy:
  (2.1) $\boldsymbol{P}_{1j} = \frac{\alpha^2 u_1 (k_r - 1)}{(w_{k_r} - w_s)(k-1)}(\boldsymbol{R}_{0j} - \boldsymbol{R}_{1j}) + \boldsymbol{P}_{0j},\ j = 0, 1, \ldots, n_r$,
  (2.2) $\boldsymbol{P}_{m-1,j} = \frac{\beta^2 (1 - u_{m-3})(k_s - 1)}{(t_{k_s} - t_s)(k-1)}(\boldsymbol{S}_{0j} - \boldsymbol{S}_{1j}) + \boldsymbol{P}_{m,j},\ j = 0, 1, \ldots, n_s$;
- G^2 continuity condition: In addition to (1.1), (1.2), and, (2.1), (2.2), they should also satisfy:
  (3.1) $\boldsymbol{P}_{2j} = \frac{\alpha^4 u_1 (k_r - 1)(k_r - 2)}{(w_{k_r} - w_s)(k-1)(k-2)}(\boldsymbol{R}_{0j} - 2\boldsymbol{R}_{1j} + \boldsymbol{R}_{2j}) + 2\boldsymbol{P}_{1j} - \boldsymbol{P}_{0j},\ j = 0, 1, \ldots, n_r$,
  (3.2) $\boldsymbol{P}_{m-2,j} = \frac{\beta^4 (1 - u_{m-3})(k_s - 1)(k_s - 2)}{(t_{k_s} - t_s)(k-1)(k-2)}(\boldsymbol{S}_{0j} - 2\boldsymbol{S}_{1j} + \boldsymbol{S}_{2j}) + 2\boldsymbol{P}_{m-1,j} - \boldsymbol{P}_{m,j},\ j = 0, 1, \ldots, n_s$.

where α, β are two variables related to the reparameterization. Specifically, conditions (1.1), (2.1), and (3.1) present the continuity conditions between the base surface $\boldsymbol{R}(w, v)$ (7) and the blending surface $\boldsymbol{P}(u, v)$ (13); conditions (1.2), (2.2), and (3.2) designate the continuity conditions between $\boldsymbol{S}(t, v)$ and $\boldsymbol{P}(u, v)$.

Parametric continuity condition: Accordingly, when $\alpha = \beta = 1$, the geometric continuity conditions become parametric continuity conditions.

4. Implementation

In our implementation, we take $k = 4$ in $\boldsymbol{P}(u, v)$ (13).

Solution with C^2 condition: If there is no middle v-control curve, the control points of the first three and last three v-control curves can be determined directly by the parametric continuity conditions. However, if there are some middle v-control curves, their control points should be generated by solving the optimization problem (12).

The constrained optimization problem (12) can be made constraint free by directly substituting the C^1 or C^2 condition into the object function (11). Since the object function after substitution is quadratic, it can be settled by solving its normal equation, i.e., a linear system of equations. In the following, we present the linear system for solving the optimization problem (12) with C^2 continuity conditions (i.e., with $\alpha = \beta = 1$). The linear system for solving the problem (12) with C^1 continuity conditions (i.e., with $\alpha = \beta = 1$) can be derived similarly.

Specifically, the solution of the optimization problem (12) with C^2 continuity constraints can be obtained by solving the linear system as follows:

$$AX^T = -BN^T, \tag{14}$$

where, $X = [\boldsymbol{P}_{30}, \boldsymbol{P}_{31}, \ldots, \boldsymbol{P}_{3,n_3}, \ldots, \boldsymbol{P}_{m-3,0}, \boldsymbol{P}_{m-3,1}, \ldots, \boldsymbol{P}_{m-3,n_{m-3}}]$, are the unknown control points, and,

$$N = [\boldsymbol{P}_{00}, \boldsymbol{P}_{01}, \ldots, \boldsymbol{P}_{0,n_0}, \boldsymbol{P}_{10}, \ldots, \boldsymbol{P}_{1,n_1}, \ldots, \boldsymbol{P}_{20}, \ldots, \boldsymbol{P}_{2,n_2}, \boldsymbol{P}_{m-2,0}, \ldots, \boldsymbol{P}_{m-2,n_{m-2}}, \ldots, \boldsymbol{P}_{m,0}, \ldots, \boldsymbol{P}_{m,n_m}],$$

are determined by the C^2 continuity conditions. Moreover,

$$A = C_{uu}^T C_{uu} + 2C_{uv}^T C_{uv} + C_{vv}^T C_{vv}, \qquad B = C_{uu}^T D_{uu} + 2C_{uv}^T D_{uv} + C_{vv}^T D_{vv},$$

where,

$$C_{uu} = [a_{30}, \ldots, a_{3,n_3}, \ldots, a_{m-3,0}, \ldots, a_{m-3,n_{m-3}}],$$
$$C_{vv} = [b_{30}, \ldots, b_{3,n_3} \ldots, b_{m-3,0}, \ldots, b_{m-3,n_{m-3}}],$$
$$C_{uv} = [c_{30}, \ldots, c_{3,n_3} \ldots, c_{m-3,0}, \ldots, c_{m-3,n_{m-3}}],$$
$$D_{uu} = [a_{00}, \ldots, a_{0,n_0}, \ldots, a_{20}, \ldots, a_{2,n_2}, a_{m-2,0}, \ldots, a_{m-2,n_{m-2}}, \ldots, a_{m,0}, \ldots, a_{m,n_m}],$$
$$D_{vv} = [b_{00}, \ldots, b_{0,n_0}, \ldots, b_{20}, \ldots, b_{2,n_2}, b_{m-2,0}, \ldots, b_{m-2,n_{m-2}}, \ldots, b_{m,0}, \ldots, b_{m,n_m}],$$
$$D_{uv} = [c_{00}, \ldots, c_{0,n_0}, \ldots, c_{20}, \ldots, c_{2,n_2}, c_{m-2,0}, \ldots, c_{m-2,n_{m-2}}, \ldots, c_{m,0}, \ldots, c_{m,n_m}],$$

with,

$$a_{ij} = \int_{u_s}^{u_e}\int_{v_s}^{v_e} B_i''(u; U)B_j(v; V_i)dudv,$$
$$b_{ij} = \int_{u_s}^{u_e}\int_{v_s}^{v_e} B_i(u; U)B_j''(v; V_i)dudv,$$
$$c_{ij} = \int_{u_s}^{u_e}\int_{v_s}^{v_e} B_i'(u; U)B_j'(v; V_i)dudv.$$

Solution with G^1 or G^2 conditions: After substituting the G^1 or G^2 continuity condition into the object function (11), it becomes quartic function or degree eight function, respectively. Meanwhile, the constrained optimization problem (12) changes to unconstrained. We employ the *Matlab* function *fminunc* [31], which uses the BFGS Quasi-Newton method with a cubic line search procedure [32], to solve the unconstrained optimization problem. In addition, the solution with C^2 continuity conditions are taken as the initial value input to the function *fminunc*. With the desirable initial value, *fminunc* just needs several steps to converge to a local minimum in all of the following blending results with the geometric continuity.

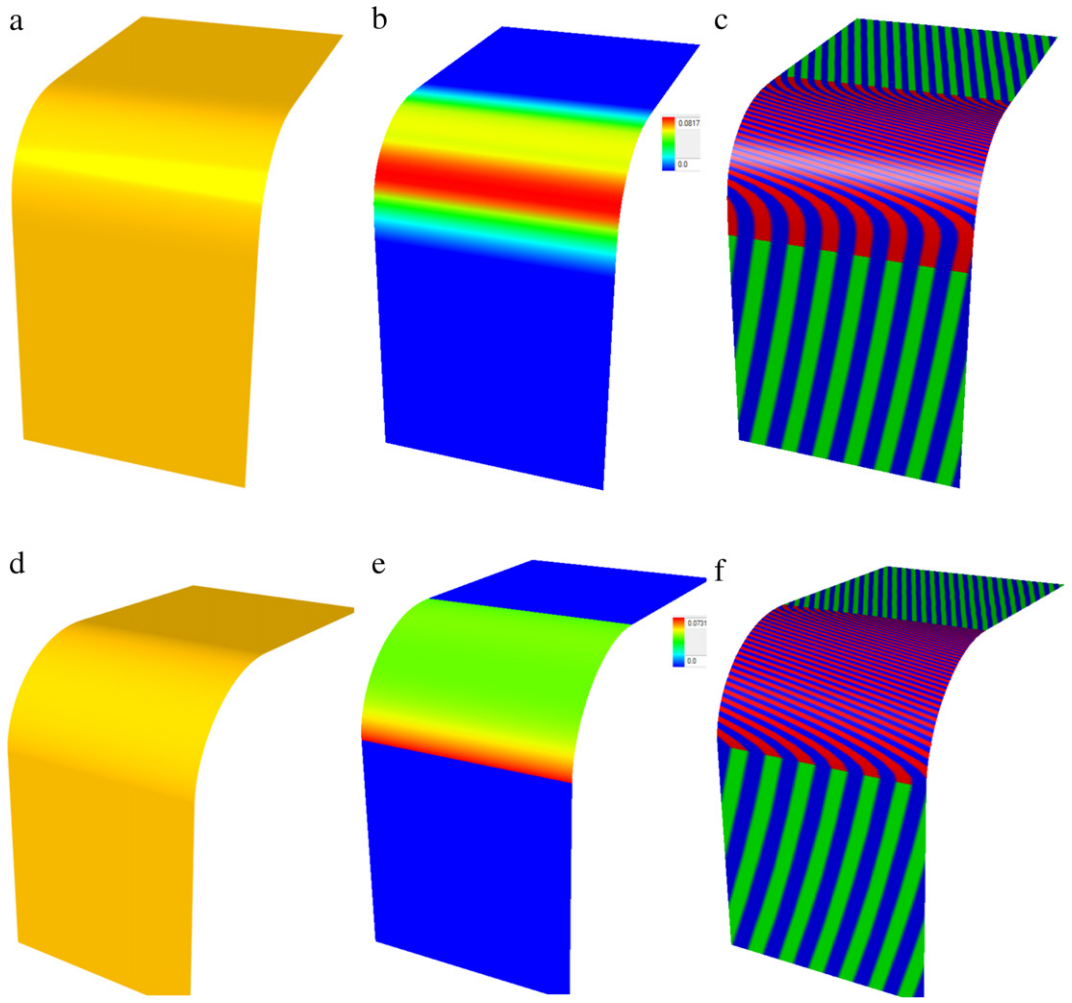

Fig. 5. Two planar patches are blended by our method (a, b, c) and the method developed in [13] (d, e, f), respectively. While our method can reach C^2 continuity, the method in [13] can attain just G^1 continuity. (a, d) Rendering result; (b, e) mean curvature on the base and blending surfaces; (c, f) zebra on the base surfaces (in green) and blending surface (in red). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

5. Blending results

The semi-structured B-spline blending method developed above has been implemented with Visual C++ and Matlab, and run on a PC with Intel Core2 Quad CPU Q9400 2.66 GHz and 4 G memory. We tested our algorithm by a lot of examples, and listed the statistics in Table 1, including model size, computational time, surface energy, and range of mean curvatures. In all of the examples, the base surfaces and blending surface are all taken as bi-cubic surfaces, since the cubic curve and surface are the most frequently employed in the geometric design.

The blending results by our method is illustrated in Figs. 5, 6, and 8–11. Each of the blending surfaces in Figs. 5, 6, 8, and 9 has one middle v-control curve, and the blending surfaces in Figs. 10 and 11 have not the middle v-control curve. In each example, besides the rendering result, we also display the mean curvatures on the base and blending surfaces to demonstrate the bending energy, and the zebra to validate the C^2 (G^2) continuity between the base and blending surfaces. Moreover, our method is compared with the method developed in [13] (refer to Fig. 5(d)–(f)), and the method presented in [17] (see Fig. 7).

In Fig. 5, two planar patches (in green) are blended by a piece of semi-structured B-spline surface (in red) (see Fig. 5(a)–(c)). Although the knot vectors of the two contact curves are different, one is $[0,\ 0,\ 0,\ 0,\ \frac{1}{2},\ 1,\ 1,\ 1,\ 1]$, and the other is $[0,\ 0,\ 0,\ 0,\ \frac{1}{3},\ \frac{2}{3},\ 1,\ 1,\ 1,\ 1]$, a piece of semi-structured B-spline surface can blend the two base surfaces with C^2 continuity. The mean curvature (Fig. 5(b)) and zebra (Fig. 5(c)) on the base and blending surfaces show the curvature continuity among them.

Table 1
Statistics on the semi-structured *B*-spline blending method.

Models in	Model size[a]		Time[b]	Energy[c]	Range of mean curvature
	Model 1	Model 2			
Fig. 5	6×5	6×6	0.4515	3.09×10^6	[0.0000, 0.0817]
Fig. 6	5×10	5×8	1.4599	3.15×10^6	[0.0000, 0.2581]
Fig. 8	6×6	6×5	0.4520	2.44×10^6	[0.0000, 0.3822]
Fig. 9	6×6	6×5	0.4331	3.62×10^6	[0.0000, 0.2834]
Fig. 10	10×10	8×8	1.6294	2.99×10^6	[0.0000, 0.1645]
Fig. 11	4×10	5×64	5.5418	6.78×10^6	[0.0000, 1.0613]

[a] The model size is measured by the size of the control net.
[b] Computational time is in seconds.
[c] Surface energy is measured by Eq. (11).

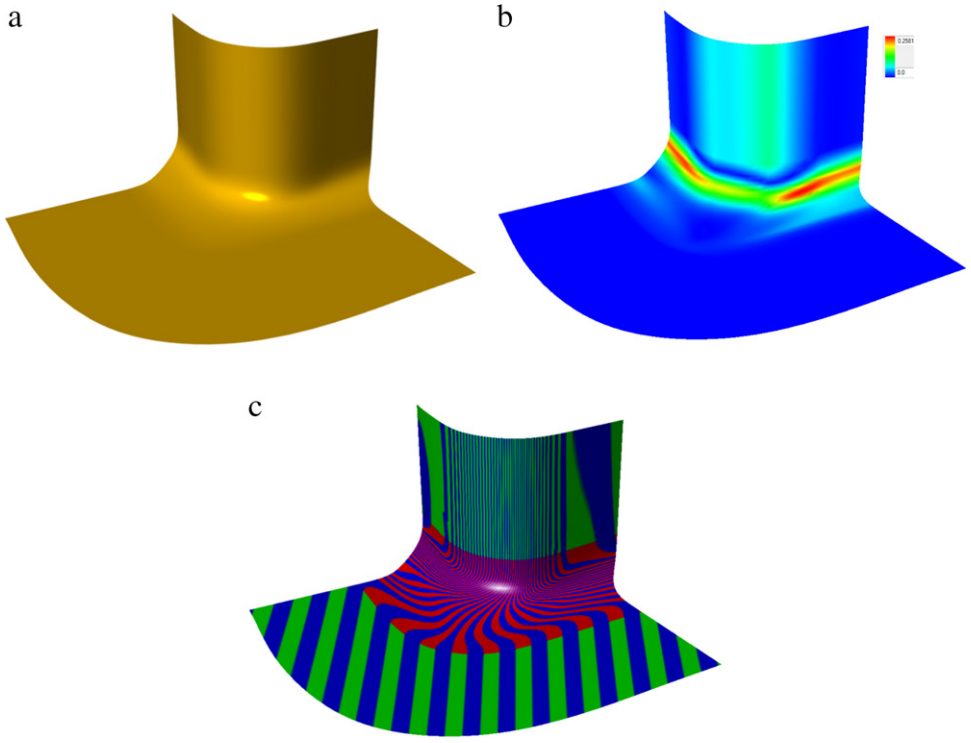

Fig. 6. Blend a planar patch and a curved surface with a semi-structured *B*-spline surface (in red) with G^2 continuity. (a) Rendering result; (b) mean curvature on the base surfaces and blending surface; (c) zebra on the base surfaces (in green) and blending surface (in red). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Moreover, our method is compared with the one developed in [13]. Since the method in [13] requires that the two contact curves have the same knot vectors, we change the representation of the two base planar patches, and make their v-directional knot vectors both be $[0,\ 0,\ 0,\ 0,\ \frac{1}{2},\ 1,\ 1,\ 1,\ 1]$. The blending results are illustrated in Fig. 5(d)–(f). It can reach just G^1 continuity between base and blending surfaces, and the curvature discontinuity is clearly revealed by the mean curvature distribution in Fig. 5(e) and the zebra in Fig. 5(f).

In Fig. 6, a planar patch and a curved surface are blended by a piece of semi-structured *B*-spline surface with G^2 continuity. The knot vector of the contact curve on the planar patch is

$$[0,\ 0,\ 0,\ 0,\ 0.15,\ 0.28,\ 0.48,\ 0.55,\ 0.70, 0.86,\ 1,\ 1,\ 1,\ 1];$$

the knot vector of the contact curve on the curved surface is

$$[0,\ 0,\ 0,\ 0,\ 0.22,\ 0.39,\ 0.61,\ 0.76,\ 1,\ 1,\ 1,\ 1].$$

The G^2 continuity among the base and blending surfaces is validated by the mean curvature distribution (Fig. 6(b)), and the zebra on them (Fig. 6(c)). In this example, the Matlab function *fminunc* costs 5 iterations to generate the blending result with G^2 continuity.

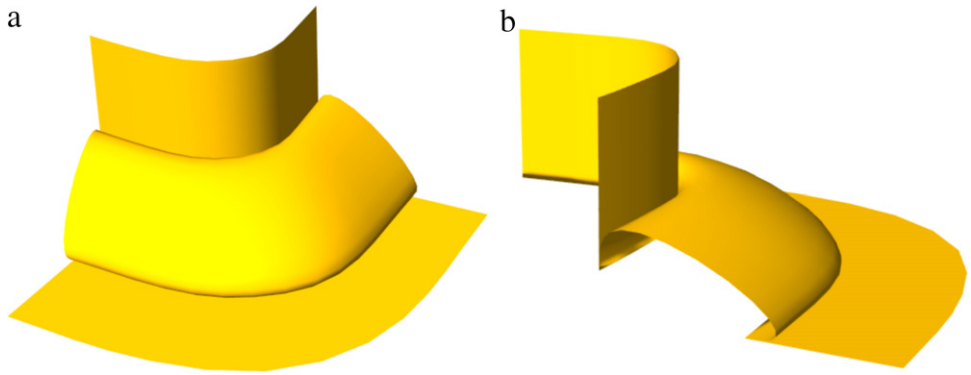

Fig. 7. Blend the planar patch and the curved surface, same as those in Fig. 6, by the method presented in [17]. The blending result lies in the convex hull of the two base surfaces, and then the blending is along the reverse direction. (a) One side view; (b) the other side view.

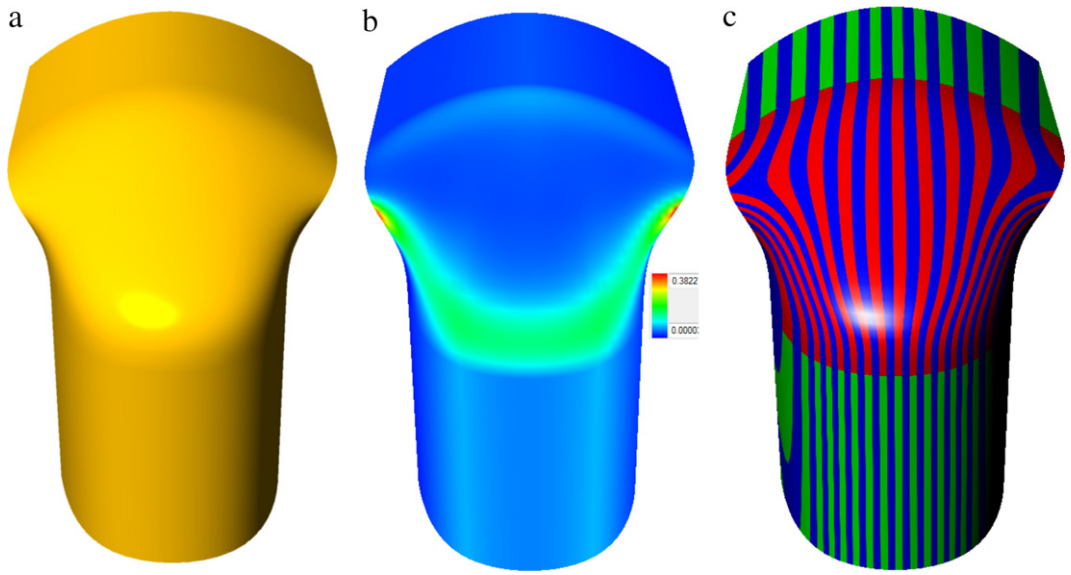

Fig. 8. Blend two part cylindrical patches of different radius by a semi-structured B-spline surface with C^2 continuity. (a) Rendering result; (b) mean curvature distribution on the base and blending surfaces; (c) zebra on the base surfaces (in green) and blending surface (in red). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

On the other hand, we also blend the planar patch and the curved patch, same as those in Fig. 6, with the method presented in [17]. Since the blending result by the method in [17] is the convex combination of portions of two base surfaces, it lies in the convex hull of the two base surfaces, and then the blending is along the reverse direction (see Fig. 7).

In Figs. 8 and 9, two part cylindrical patches of different radius are blended by semi-structured B-spline surfaces with different postures, respectively. The knot vector of the contact curve on the cylindrical patch with larger radius is [0, 0, 0, 0, 0.3, 0.7, 1, 1, 1, 1]; that on the cylindrical patch with smaller radius is [0, 0, 0, 0, 0.5, 1, 1, 1, 1]. Although the two postures differ greatly, the two semi-structured B-spline blending surfaces are both made C^2 continuity with the corresponding base surfaces, which are validated by the mean curvature distribution (Figs. 8(b), 9(b)), and zebra (Figs. 8(c), 9(c)) on them.

Moreover, the example in Fig. 10 demonstrates the capability of employing the semi-structured B-spline surface to blend two complete cylindrical patches. While the knot vector of one contact curve is,

$$[0,\ 0,\ 0,\ 0,\ 0.142857,\ 0.285714,\ 0.428571,\ 0.571429,\ 0.714286,\ 0.857143,\ 1,\ 1,\ 1,\ 1],$$

that of the other contract curve is,

$$[0,\ 0,\ 0,\ 0,\ 0.2,\ 0.4,\ 0.6,\ 0.8,\ 1,\ 1,\ 1,\ 1].$$

The blending meets G^2 continuity (see Fig. 10(b), (c)). It should be pointed out that, in this example, the semi-structured B-spline blending surface has no middle v-control curve. After 6 iterations, the Matlab function *fminunc* converges to the blending result with G^2 continuity.

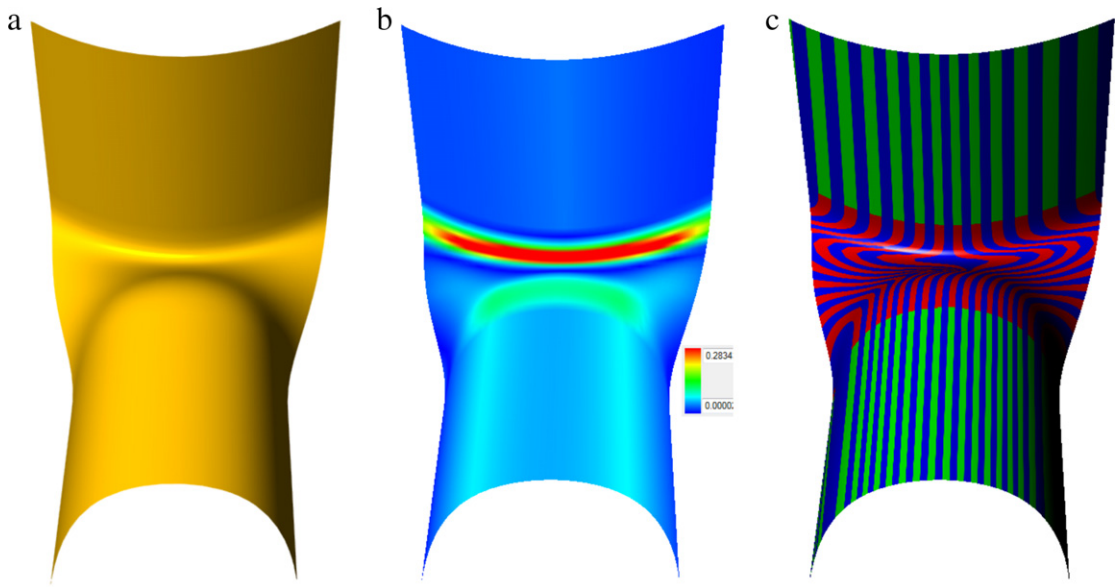

Fig. 9. Blend two part cylindrical patches, same as those in Fig. 8 but with different posture, by a piece of semi-structured B-spline surface. The blending is still reach C^2 continuity. (a) Rendering result; (b) mean curvature distribution; (c) zebra on the base surfaces (in green) and blending surface (in red). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

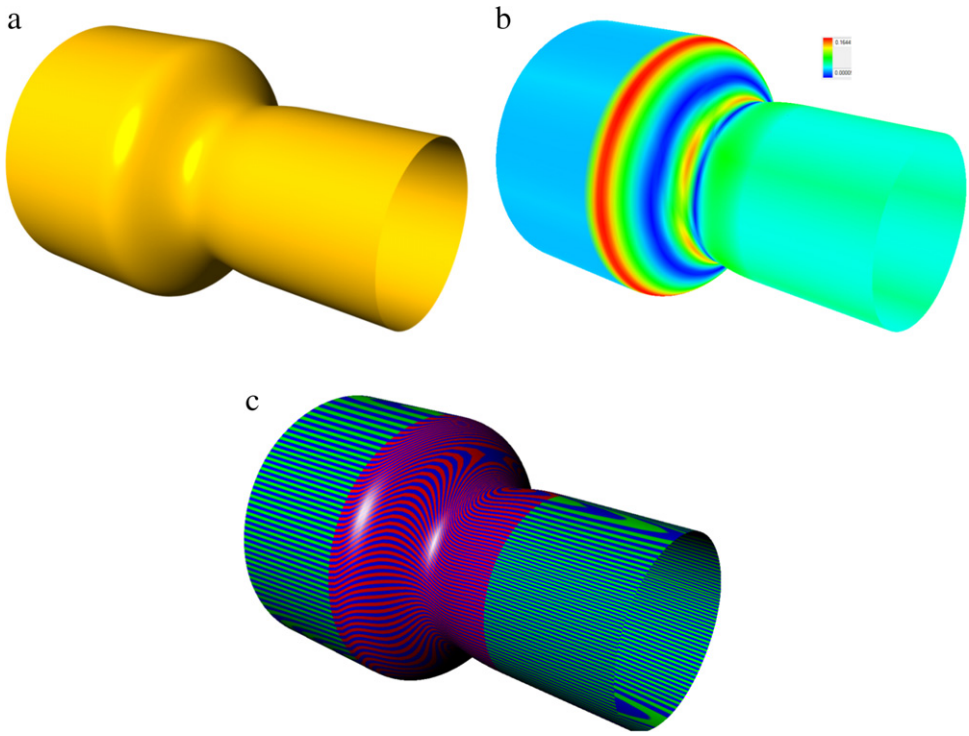

Fig. 10. Blend two complete cylindrical patches with a piece of semi-structured B-spline surface with G^2 continuity. (a) Rendering result; (b) mean curvature distribution; (c) zebra on the base surfaces (in green) and blending surface (in red). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Finally, a conical surface and a free-form surface are blended by a semi-structure B-spline surface without middle v-control curve in Fig. 11. Although the contact curve on the free-form surface has 64 control points, and the contact curve on the conical surface has 10 control points, they successfully attain G^2 continuity with the blending semi-structured B-spline surface (refer to Fig. 11(b) and (c)).

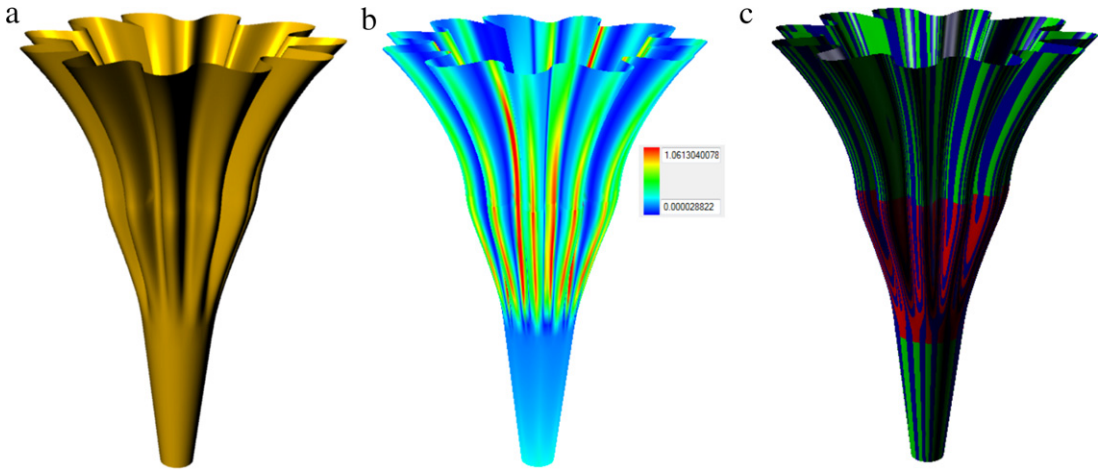

Fig. 11. Blend a conical surface and a free form surface with a piece of semi-structured B-spline surface with G^2 continuity. (a) Rendering result; (b) mean curvature distribution; (c) zebra on the base surfaces (in green) and blending surface (in red). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

6. Conclusion

In this paper, we develop the semi-structured B-spline surface, which is a curve-based spline representation generated by skinning a series of B-spline curves with different knot vectors. The so developed semi-structured B-spline surface can blend two base B-spline surfaces with mismatched knot vectors, and the continuity between the base and blending surfaces can reach G^2 and C^2. Since surface blending is an important operation in geometric design, and NURBS has become the de facto industrial standard, our algorithm will have wide applications in geometric design. However, the main difficulty in constructing the semi-structured B-spline blending surface is how to arrange its u-knots and v-knots. As a future work, a more reasonable and automatic method for arranging the u-knots and v-knots is required to be developed.

Acknowledgment

This paper is supported by Natural Science Foundation of China (Nos. 61379072, 60970150, 60933008).

Appendix

We only derive the continuity conditions between the base surface $\boldsymbol{R}(w, v)$ (7) and the blending surface $\boldsymbol{P}(u, v)$ (13), i.e., (1.1), (2.1), and (3.1). The other continuity conditions (1.2), (2.2), and (3.2) can be gotten similarly.

Since the two surfaces $\boldsymbol{R}(w, v)$ and $\boldsymbol{P}(u, v)$ should have the same contact curve, the G^0 (or C^0) continuity condition (1.1) is obvious.

According to Ref. [30], we have,

- The necessary and sufficient condition for two adjacent surfaces $\boldsymbol{P}(u, v)$ (13) and $\boldsymbol{R}(w, v)$ (7) joining G^1-continuously along a common boundary curve $\boldsymbol{C}(v)$ is that, there exist functions $p_1(v)$ and $q_1(v)$ such that the following equation holds along $\boldsymbol{C}(v)$:

$$\frac{\partial \boldsymbol{P}}{\partial u} = p_1(v)\frac{\partial \boldsymbol{R}}{\partial w} + q_1(v)\frac{\partial \boldsymbol{R}}{\partial v}. \tag{15}$$

- The necessary and sufficient conditions for the above two surfaces $\boldsymbol{P}(u, v)$ and $\boldsymbol{R}(w, v)$ joining G^2-continuously along $\boldsymbol{C}(v)$ are that, in addition to Eq. (15), there exist functions $p_2(v)$ and $q_2(v)$ such that the following equation holds along $\boldsymbol{C}(v)$:

$$\frac{\partial^2 \boldsymbol{P}}{\partial u^2} = p_2(v)\frac{\partial \boldsymbol{R}}{\partial w} + q_2(v)\frac{\partial \boldsymbol{R}}{\partial v} + p_1^2(v)\frac{\partial^2 \boldsymbol{R}}{\partial w^2} + 2p_1(v)q_1(v)\frac{\partial^2 \boldsymbol{R}}{\partial w \partial v} + q_1^2(v)\frac{\partial^2 \boldsymbol{R}}{\partial v^2}. \tag{16}$$

Letting $p_1(v) = \alpha^2$, $q_1(v) = p_2(v) = q_2(v) = 0$, Eqs. (15) and (16) change to,

$$\left.\frac{\partial \boldsymbol{P}(u, v)}{\partial u}\right|_{u=0} = \alpha^2 \left.\frac{\partial \boldsymbol{R}(w, v)}{\partial w}\right|_{w=w_s}, \tag{17}$$

$$\left.\frac{\partial^2 \boldsymbol{P}(u, v)}{\partial u^2}\right|_{u=0} = \alpha^4 \left.\frac{\partial^2 \boldsymbol{R}(w, v)}{\partial w^2}\right|_{w=w_s}. \tag{18}$$

Therefore, by Eq. (17) and,

$$\left.\frac{\partial \boldsymbol{P}(u,v)}{\partial u}\right|_{u=0} = \frac{k-1}{u_1}\sum_{j=0}^{n_r}(\boldsymbol{P}_{1j}-\boldsymbol{P}_{0j})B_{j,l}(v;V^r),$$

$$\left.\frac{\partial \boldsymbol{R}(w,v)}{\partial w}\right|_{w=w_s} = \frac{k_r-1}{w_{k_r}-w_s}\sum_{j=0}^{n_r}(\boldsymbol{R}_{1j}-\boldsymbol{R}_{0j})B_{j,l}(v;V^r),$$

we get the condition (2.1); by Eq. (18) and,

$$\left.\frac{\partial^2 \boldsymbol{P}(u,v)}{\partial u^2}\right|_{u=0} = \frac{(k-1)(k-2)}{u_1}\sum_{j=0}^{n_r}(\boldsymbol{P}_{2j}-2\boldsymbol{P}_{1j}+\boldsymbol{P}_{0j})B_{j,l}(v;V^r),$$

$$\left.\frac{\partial^2 \boldsymbol{R}(w,v)}{\partial w^2}\right|_{w=w_s} = \frac{(k_r-1)(k_r-2)}{w_{k_r}-w_s}\sum_{j=0}^{n_r}(\boldsymbol{R}_{2j}-2\boldsymbol{R}_{1j}+\boldsymbol{R}_{0j})B_{j,l}(v;V^r),$$

we get the condition (3.1).

References

[1] J. Vida, R.R. Martin, T. Varady, A survey of blending methods that use parametric surfaces, Comput.-Aided Des. 26 (5) (1994) 341–365.
[2] K.L. Shi, J.H. Yong, J.G. Sun, J.C. Paul, G^n blending multiple surfaces in polar coordinates, Comput.-Aided Des. 42 (6) (2010) 479–494.
[3] L.A. Piegl, W. Tiller, The NURBS Book, Springer Verlag, 1997.
[4] B.K. Choi, S.Y. Ju, Constant-radius blending in surface modelling, Comput. Aided Des. 21 (4) (1989) 213–220.
[5] J.H. Chuang, W.C. Hwang, Variable-radius blending by constrained spine generation, Vis. Comput. 13 (7) (1997) 316–329.
[6] R.A. Farouki, R. Sverrisson, Approximation of rolling-ball blends for free-form parametric surfaces, Comput.-Aided Des. 28 (11) (1996) 871–878.
[7] G. Lukács, Differential geometry of G^1 variable radius rolling ball blend surfaces, Comput. Aided Geom. Design 15 (6) (1998) 585–613.
[8] I.C. Braid, Non-local blending of boundary models, Comput.-Aided Des. 29 (2) (1997) 89–100.
[9] B. Belkhatira, A. Kouibiab, M. Pasadasc, D. Sbibihd, A. Zidnae, Geometric continuity C^1G^2 of blending curves, Int. J. Contemp. Math. Sci. 3 (30) (2008) 1451–1460.
[10] A. Kouibia, Miguel Pasadas, Driss Sbibih, Ahmed Zidna, Bachir Belkhatir, Geometric continuity C^1G^2 of blending surfaces, Comput.-Aided Des. 45 (3) (2013) 733–738.
[11] Bachir Belkhatir, Ahmed Zidna, Construction of flexible blending parametric surfaces via curves, Math. Comput. Simul. 79 (12) (2009) 3599–3608.
[12] Min-jae Oh, Kittichai Suthunyatanakit, Sung Ha Park, Tae-wan Kim, Constructing G^1 Bézier surfaces over a boundary curve network with t-junctions, Comput.-Aided Des. 44 (7) (2012) 671–686.
[13] B. Belkhatir, D. Sbibih, A. Zidna, G^1 blending B-spline surfaces and optimization, in: Modelling, Computation and Optimization in Information Systems and Management Sciences, 2008, pp. 458–467.
[14] M.I.G. Bloor, M.J. Wilson, Generating blend surfaces using partial differential equations, Comput.-Aided Des. 21 (3) (1989) 165–171.
[15] M.I.G. Bloor, M.J. Wilson, Using partial differential equations to generate free-form surfaces, Comput.-Aided Des. 22 (4) (1990) 202–212.
[16] D.J. Filip, Blending parametric surfaces, ACM Trans. Graph. 8 (3) (1989) 164–173.
[17] E. Hartmann, Parametric G^n blending of curves and surfaces, Vis. Comput. 17 (1) (2001) 1–13.
[18] Q. Song, J. Wang, Generating G^n parametric blending surfaces based on partial reparameterization of base surfaces, Comput.-Aided Des. 39 (11) (2007) 953–963.
[19] J. Gravesen, Semi-regular B-spline surfaces: generalized lofting by B-splines, in: Proceedings of the International Conference on Curves and Surfaces in Geometric Design, AK Peters, Ltd., 1994, pp. 225–232.
[20] F. Weller, B-spline Surfaces with Knot Segments, Citeseer, 1994.
[21] T.W. Sederberg, D.L. Cardon, G.T. Finnigan, N.S. North, J. Zheng, T. Lyche, T-spline simplification and local refinement, in: ACM Transactions on Graphics (TOG), Vol. 23, ACM, 2004, pp. 276–283.
[22] J.G. Hayes, New shapes from bicubic splines, in: Proceedings CAD 74, Guildford, fiche 36G/37A, IPC Business Press, Imperial College, London, 1974.
[23] J. Hayes, Curved knot lines and surfaces with ruled segments, in: Numerical Analysis, 1982, pp. 140–156.
[24] Shi-Min Hu, Guo-Zhao Wang, Tong-Guang Jin, Generalized subdivision of Bézier surfaces, Graph. Models Image Process. 58 (3) (1996) 218–222.
[25] Hu Shi-Min, Sun Jia-Guang, Wang Guo-Zhao, Generalized subdivision of Bézier surfaces and its applications, Chinese J. Comput. 22 (3) (1999) 290–295.
[26] Lyle Ramshaw, Béziers and B-splines as multiaffine maps, in: Theoretical Foundations of Computer Graphics and CAD, Springer, 1988, pp. 757–776.
[27] Lyle Ramshaw, Blossoms are polar forms, Comput. Aided Geom. Design 6 (4) (1989) 323–358.
[28] C. De Boor, A Practical Guide to Splines, Vol. 27, Springer Verlag, 2001.
[29] X. Wang, F.F. Cheng, B.A. Barsky, Energy and B-spline interproximation, Comput.-Aided Des. 29 (7) (1997) 485–496.
[30] X. Ye, Y. Liang, H. Nowacki, Geometric continuity between adjacent Bézier patches and their constructions, Comput. Aided Geom. Des. 13 (6) (1996) 521–548.
[31] Matlab Optimization Toolbox, User's Guide, Version 5, The Math Works, Incorporated, 2010.
[32] J.E. Dennis, R.B. Schnabel, Numerical Methods for Unconstrained Optimization and Nonlinear Equations, Vol. 16, Society for Industrial Mathematics, 1996.