Filling n-sided regions with G^1 triangular Coons B-spline patches

Kan-Le Shi · Jun-Hai Yong · Jia-Guang Sun · Jean-Claude Paul · He-Jin Gu

Abstract Filling n-sided regions is an essential operation in shape and surface modeling. Positional and tangential continuities are highly required in designing and manufacturing. We propose a method for filling n-sided regions with untrimmed triangular Coons B-spline patches, preserving G^1 continuity exactly. The algorithm first computes a central point, a central normal, the central, and the corner derivative vectors. Then the region is split into n triangular areas by connecting the central point to each corner of the boundary. These inner curves and all cross-boundary derivatives are computed fulfilling G^1 compatibility conditions. And finally, the triangular patches are generated in the Coons B-spline form, one boundary of which is regressed to the central vertex. Neither positional nor tangential error is introduced by this method. And only one degree elevation is needed.

K.-L. Shi (✉) · J.-H. Yong · J.-G. Sun · J.-C. Paul
School of Software, Tsinghua University, Beijing 100084, P.R. China
e-mail: skl03@mails.tsinghua.edu.cn

K.-L. Shi
Department of Computer Science and Technology, Tsinghua University, Beijing 100084, P.R. China

K.-L. Shi · J.-H. Yong · J.-G. Sun · J.-C. Paul
Key Laboratory for Information System Security, Ministry of Education of China, Beijing 100084, P.R. China

K.-L. Shi · J.-H. Yong · J.-G. Sun · J.-C. Paul
Tsinghua National Laboratory for Information Science and Technology, Beijing 100084, P.R. China

J.-C. Paul
INRIA, Nancy, France

H.-J. Gu
Jiangxi Academy of Sciences, Nanchang 330029, P.R. China

Keywords n-sided region filling · Triangular Coons B-spline surface · G^1 continuity · CAD

1 Introduction

N-sided region filling is a recurring operation in geometric designing and modeling. It aims to provide a trimmed surface or untrimmed surface patches that interpolate the region's boundary and the cross-boundary derivatives. Since B-spline has become a popular representation for 3D models in industry, filling n-sided regions using B-spline surfaces has been widely discussed in the past two decades.

The existing practical methods for filling n-sided regions mainly fall into three categories. The first category uses a single patch as the blending surface, with both planar domains [1–3] and nonplanar domains [4, 5]. The second category uses an energy minimization method or constrained-optimization method to solve trimmed surfaces. These methods, which are integrated in some industrial modeling systems, such as CATIA and SolidWorks, are more flexible, but introduce the positional and tangential errors. The trimmed surfaces generated by these methods contain large numbers of control points and complicated imprecise boundary. And all of these methods are time consuming. Besides, the operations of stitching, patching, and blending also become more complicated than the ones for untrimmed surfaces. The third category splits the region into several quadrilateral patches by connecting a central vertex of the region to the middle point of each boundary curve [6–8]. Piegl and Tiller [8] and Yang [9] used untrimmed Coons patches to fill n-sided regions with B-spline boundary. Their methods preserve G^0 continuity, but introduce

tangential error on the boundary and the inner curves, incurring the well-known *twist compatibility problem*. Besides, subdivision surfaces are also used to fill n-sided regions. Karčiauskas and Peters [10] introduced a bicubic polar subdivision scheme with C^1 continuity, and it was extended to C^2 [11, 12]. The main target of these methods is to smooth meshes. Continuity between the subdivision surface and the neighboring surfaces is still a challenge in applying their methods to vertex blending. Hwang [13] directly used subdivision surfaces to fill holes with G^1 continuity. Li [14] then achieved C^2 continuity. The mass computation and large memory assignment were also introduced by these methods. Thus, these non-NURBS/B-spline methods have rarely been applied in industrial CAD/CAM systems.

Positional and tangential continuities are highly required in the phases of designing and manufacturing. For industrial application, B-spline/NURBS compatibility should also be considered. But most popular methods incur positional or tangential errors. The aim of this paper is to provide a method for filling n-sided regions with n triangular untrimmed B-spline surfaces, preserving G^0 and G^1 continuities exactly. All curves and surfaces are represented in standard B-spline form. This method has the following advantages over the existing ones:

- G^0 and G^1 continuities are achieved exactly. There is no positional nor tangential error on the common boundary of adjacent patches. It also has theoretical and practical value of handling the twist compatibility problem for incompatible corners without introducing error.
- The degree of the generated surface is only one larger than the degree of corresponding boundary curve.
- The generated surfaces are simple, as they are all untrimmed Coons B-spline surfaces, which benefit data exchange. The required storage space of our method is usually smaller than the existing methods of the second category.
- This method has geometric parameters to control the shape and the quality of the generated patches.
- Since the algorithm contains no iteration or large matrix calculation, it is efficient.

The organization of the remaining paper is as follows. In Sect. 2, the main framework of the algorithm is presented. Section 3 focuses on how to compute a central point, a central normal, and the central derivative vectors. Section 4 contains details on how to generate inner curves and all cross-boundary derivatives. In Sect. 5, Coons patches are computed. Examples in Sect. 6 underline the advantages of this method. And finally, Sect. 7 concludes the paper.

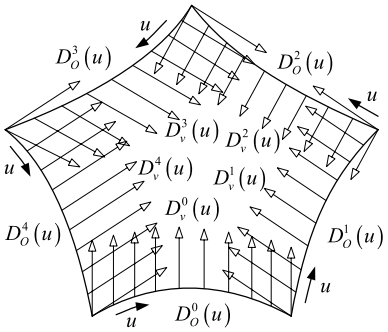

Fig. 1 The input boundary and its cross-boundary derivatives: $D_O^i(u)$ are the boundary curves and $D_v^i(u)$ are the cross-boundary derivative curves, where $i = 0, 1, \ldots, 4$

2 Algorithm framework

In this paper, all the boundary curves and their derivative curves are represented in B-spline form as follows:

$$D_w^i(u) = \sum_{r=0}^{p_i} N_{r,p_i}(u) D_w^i[r],$$
$$i = 0, 1, \ldots, n-1; w \in \{O, v\}; u \in [0, 1], \quad (1)$$

where $N_{p_i,r}(t)$ is the B-spline basis function, i is the subscript of the boundary, and w is the type of cross-boundary derivatives (O denotes the zeroth-order derivative, or the boundary position, and v denotes the 1st-order cross-boundary derivative). In the rest of the paper, the square brackets always denote the subscript of one curve's control points. As shown in Fig. 1, the positive parametric u-direction of the boundary is anticlockwise, and the corresponding v-direction is pointing to the center of the n-sided region. The boundary conditions can be manually specified or derived from the neighboring surfaces. We denote

$$D^i_{\underbrace{u \ldots u}_{s} w} \triangleq \frac{\partial^s D_w^i}{\partial u^s},$$

where w in the left part can be omitted if $w = O$.

Piegl and Tiller's method [8] requires that

$$\begin{cases} D_u^i(1) = \alpha_i D_v^j(0) \\ D_v^i(1) = \beta_i D_u^j(0) \end{cases},$$

where $j = (i+1) \bmod n$, and $\alpha_i, \beta_i \in \mathbb{R}$. This limitation is overcome by our algorithm. An along-boundary derivative vector and the corresponding cross-boundary derivative vector (e.g., $D_u^i(1)$ and $D_v^j(0)$) do not need to be parallel. Instead, these four vectors only need to be coplanar. That is,

$$\text{rank}[D_u^i(1), D_v^j(0), D_v^i(1), D_u^j(0)] < 3.$$

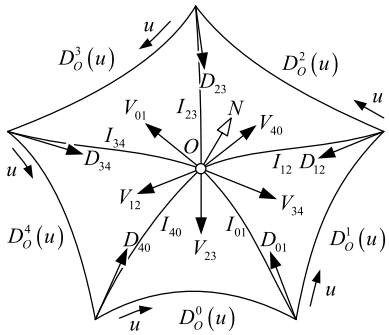

Fig. 2 Central point O, central normal vector N, the corner derivative vectors D_{ij}, the central derivative vectors V_{ij}, and the inner curves I_{ij} of the triangular partition

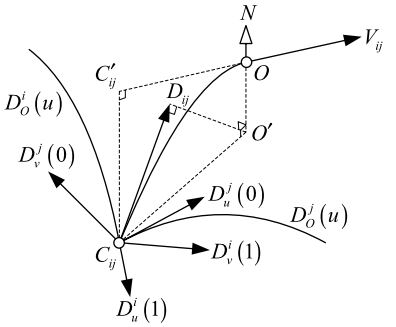

Fig. 3 Computing central and corner derivative vectors: O' is the projection of O to the plane spanned by $D_u^i(1)$ and $D_u^j(0)$, and C'_{ij} is the projection point of the corner C_{ij} to the central tangential plane

We denote

$$j = (i+1) \bmod n, \quad \text{and,} \quad k = (i+n-1) \bmod n.$$

The main steps of the algorithm are as follows (see Fig. 2).

1. Compute a central point O and a central normal vector N of the region using the specified boundary conditions (discussed in Sect. 3).
2. Compute the central derivative vectors V_{ij} and the corner derivative vectors D_{ij} (discussed in Sect. 3).
3. Solve compatibility conditions for each corner of triangular patches to construct inner curves I_{ij} and the corresponding cross-boundary derivatives L_{ij}, R_{ij}, B_{ij} and C_{ij} (discussed in Sect. 4). These derivative curves and the boundary curves of the triangular subregions fulfill all G^1 compatibility conditions exactly.
4. Construct the final Coons patches with fully compatible boundary conditions constructed in the previous step (discussed in Sect. 5).

In the following sections, the details of these steps are derived.

3 Central point, central normal and derivative vectors

The central point and the central normal vector (see Fig. 2) can be computed using Piegl and Tiller's [8] method. That is,

$$l_i = \frac{1}{4}\left(\left\|D_O^j(1) - D_O^j(0)\right\| + \left\|D_O^k(1) - D_O^k(0)\right\|\right),$$

$$O = \frac{1}{n}\sum_{i=0}^{n-1}\left(l_i \operatorname{Norm}(D_v^i(0.5)) + D_O^i(0.5)\right),$$

$$N = \operatorname{Norm}\left(\frac{1}{n}\sum_{i=0}^{n-1}\left(D_O^i(0.5) - O\right) \times \left(D_O^j(0.5) - O\right)\right),$$

where $\operatorname{Norm}(\mathbf{x}) = \mathbf{x}/\|\mathbf{x}\|$. In application, (O, N) plays a decisive role in controlling the shape of the patches. This algorithm, as simple as it is, almost always yields an acceptable central point and normal [8]. The formulae above only provide an optional (or default) setting. These two vectors also can be specified interactively by designers.

Once O and N are decided, we can compute the two kinds of important derivative vectors: (a) the corner derivative vector D_{ij}, which is the derivative vector of the corresponding inner curve $I_{ij}(v)$ at the corner point; and (b) the central derivative vector V_{ij}, which is the derivative vector at the central point (see Figs. 2 and 3). D_{ij} divides the corresponding corner into two symmetrical parts. Thus, the direction of D_{ij} is

$$\tilde{D}_{ij} = \operatorname{Norm}\left(-\operatorname{Norm}\left(D_u^i(1)\right) + \operatorname{Norm}\left(D_u^j(0)\right)\right).$$

Denote the plane spanned by the two along-boundary derivative vectors $D_u^i(1)$ and $D_u^j(0)$ is Π, and O' is the projection point of O to this plane. We choose one D_{ij}, which makes the distance between $(C_{ij} + D_{ij})$ and O' the closest (see Fig. 3). Thus, we have

$$D_{ij} = \tilde{D}_{ij} \cdot \left(\tilde{D}_{ij} \cdot (O - C_{ij})\right),$$

where $C_{ij} = D_O^i(1) = D_O^j(0)$. V_{ij} can be chosen analogously. We project the corner point C_{ij} to the central tangential plane. Then, the vector from the projection point C'_{ij} to the central point O is V_{ij}. That is,

$$V_{ij} = (O - C_{ij}) - N \cdot ((O - C_{ij}) \cdot N).$$

The algorithm for computing D_{ij} and V_{ij} ensures that D_{ij} is always the angular bisector of the corner, and that the negative direction of V_{ij} is always pointing to the corner.

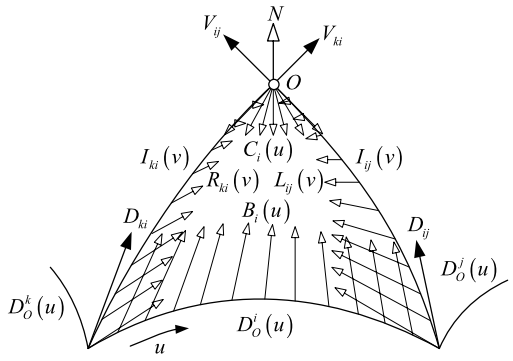

Fig. 4 A triangular patch and its cross-boundary derivatives: $D_O^i(u)$, $I_{ij}(v)$, O, and $I_{ki}(v)$ are the boundary curves; and $B_i(u)$, $L_{ij}(v)$, $C_i(u)$, and $R_{ij}(v)$ are their corresponding cross-boundary derivative curves

In the author's experience, that yields logical and acceptable results.

4 Inner curves and cross-boundary derivatives

Since O, N, D_{ij}, and V_{ij} are all computed in the previous section, the inner curves and cross-boundary derivative curves of each triangular patch are discussed in this section. As shown in Fig. 4, we represent the triangular B-spline surface using a standard quadrilateral B-spline form, one boundary of which is regressed to the central point O. The cross-boundary derivative curves of the four boundary, which are $L_{ij}(v)$, $R_{ij}(v)$, $B_i(u)$, and $C_i(u)$, should satisfy G^1 compatibility conditions. In this section, we solve these constraints in order to get sufficient boundary conditions for constructing these curves.

First, we denote a group of curves $T_{ij}(v)$, which are named the *common cross-boundary derivative curves*. And, we suppose that

$$L_{ij}(v) = p_{ij}(v)I'_{ij}(v) + q_{ij}(v)T_{ij}(v), \tag{2}$$

$$R_{ij}(v) = r_{ij}(v)I'_{ij}(v) + s_{ij}(v)T_{ij}(v), \tag{3}$$

where $p_{ij}(v)$, $q_{ij}(v)$, $r_{ij}(v)$ and $s_{ij}(v)$ are all scalar functions. The following equation can be derived from (2) and (3). That is,

$$\begin{aligned} &s_{ij}(v)L_{ij}(v) - q_{ij}(v)R_{ij}(v) \\ &\quad = (s_{ij}(v)p_{ij}(v) - q_{ij}(v)r_{ij}(v))I'_{ij}(v). \end{aligned}$$

Thus, the two cross-boundary derivatives $L_{ij}(v)$ and $R_{ij}(v)$, and the along-boundary derivative $I'_{ij}(v)$, are always coplanar. According to the G^1-continuity condition [15–17], Equations (2) and (3) ensure that the adjacent patches are always G^1-continuous on the common boundary $I_{ij}(v)$. Considering about numerical stability, we need to keep $I'_{ij}(v)$ and $T_{ij}(v)$ approximately orthogonal. Thus, we choose

$$\begin{cases} T_{ij}(0) = \mathrm{Norm}(D_{ij} \times N_{ij}) \\ T_{ij}(1) = \mathrm{Norm}(V_{ij} \times N) \end{cases}, \tag{4}$$

where $N_{ij} = D_v^i(1) \times D_v^j(0)$.

Since we use ordinary B-spline surface to represent the triangular patch, we assume that $C_i(u)$ is the cross-boundary derivative curve of the regressive boundary at O, and that $B_i(u)$ is the cross-boundary derivative curve on the corresponding boundary $D_O^i(u)$. Then suppose that

$$\begin{aligned} C_i(u) &= -V_{ki}(1-u) - V_{ij}u, \\ B_i(u) &= \alpha_i(u)D_u^i(u) + \beta_i(u)D_v^i(u), \end{aligned} \tag{5}$$

where

$$\begin{cases} \alpha_i(u) = \alpha_i^{(0)}(1-u) + \alpha_i^{(1)}u \\ \beta_i(u) = \beta_i^{(0)}(1-u) + \beta_i^{(1)}u \end{cases},$$

$$\begin{cases} [\alpha_i^{(0)}, \beta_i^{(0)}]^{\mathrm{T}} = \tilde{\sigma}_{D_u^i(0), D_v^i(0)}(D_{ki}) \\ [\alpha_i^{(1)}, \beta_i^{(1)}]^{\mathrm{T}} = \tilde{\sigma}_{D_u^i(1), D_v^i(1)}(D_{ij}) \end{cases}.$$

$\tilde{\sigma}_{\mathbf{A},\mathbf{B}}(\mathbf{x})$ is an linear mapping operator, which returns the coordinate values of $\mathbf{x}$ in the space spanned by $\mathbf{A}$ and $\mathbf{B}$. That is $[a, b]^{\mathrm{T}} = \tilde{\sigma}_{\mathbf{A},\mathbf{B}}(\mathbf{x})$, satisfying that $a\mathbf{A} + b\mathbf{B} + c(\mathbf{A} \times \mathbf{B}) = \mathbf{x}$. The definition of $B_i(u)$ ensures that (a) the generated patch is G^1-continuous with the specified boundary (it can be proved analogously to the proof above of the two adjacent patches), and that (b) $B_i(0) = D_{ki}$ and $B_i(1) = D_{ij}$. Then we list all compatibility conditions below.

Positional compatibility requires that

$$\begin{cases} I_{ij}(1) = O \\ I_{ij}(0) = C_{ij} = D_O^i(1) = D_O^j(0) \end{cases}.$$

Tangential compatibility at the central point requires that

$$\begin{cases} C_i(0) = -I'_{ki}(1) = -V_{ki} \\ C_i(1) = -I'_{ij}(1) = -V_{ij} \\ L_{ij}(1) = p_{ij}(1)I'_{ij}(1) + q_{ij}(1)T_{ij}(1) = \mathbf{0} \\ R_{ij}(1) = r_{ij}(1)I'_{ij}(1) + s_{ij}(1)T_{ij}(1) = \mathbf{0} \end{cases}.$$

So, we have

$$p_{ij}(1) = q_{ij}(1) = r_{ij}(1) = s_{ij}(1) = 0. \tag{6}$$

Tangential compatibility at the corner point requires that

$$\begin{cases} B_i(0) = I'_{ki}(0) = D_{ki} \\ B_i(1) = I'_{ij}(0) = D_{ij} \\ L_{ij}(0) = p_{ij}(0)I'_{ij}(0) + q_{ij}(0)T_{ij}(0) = -D^i_u(1) \\ R_{ij}(0) = r_{ij}(0)I'_{ij}(0) + s_{ij}(0)T_{ij}(0) = D^j_u(0) \end{cases}.$$

And we assume that $I'_{ij}(0) = D_{ij}$. Thus, we have

$$\begin{cases} [p_{ij}(0), q_{ij}(0)]^{\mathrm{T}} = \tilde{\sigma}_{D_{ij},T_{ij}(0)}(-D^i_u(1)) \\ [r_{ij}(0), s_{ij}(0)]^{\mathrm{T}} = \tilde{\sigma}_{D_{ij},T_{ij}(0)}(D^j_u(0)) \end{cases}.$$

Twist compatibility at the central point requires that

$$\begin{cases} C'_i(1) + L'_{ij}(1) = 0 \\ C'_j(0) - R'_{ij}(1) = 0 \end{cases}.$$

Uniting (2), (3), and (6), we have

$$\begin{cases} p'_{ij}(1)I'_{ij}(1) + q'_{ij}(1)T_{ij}(1) + q_{ij}(1)T'_{ij}(1) = -C'_i(1) \\ r'_{ij}(1)I'_{ij}(1) + s'_{ij}(1)T_{ij}(1) + s_{ij}(1)T'_{ij}(1) = C'_j(0) \end{cases},$$

where $I'_{ij}(1) = V_{ij}$, and $T_{ij}(1)$ is defined by (4). Note that $I'_{ij}(1)$, $T_{ij}(1)$, $C'_i(1)$, and $C'_j(0)$ are coplanar. Thus, we can assume $T'_{ij}(1) = \mathbf{0}$. Accordingly, we have

$$\begin{cases} [p'_{ij}(1), q'_{ij}(1)]^{\mathrm{T}} = \tilde{\sigma}_{I'_{ij}(1),T_{ij}(1)}(V_{ki} - V_{ij}) \\ [r'_{ij}(1), s'_{ij}(1)]^{\mathrm{T}} = \tilde{\sigma}_{I'_{ij}(1),T_{ij}(1)}(V_{jt} - V_{ij}) \end{cases},$$

where $t = (j+1) \bmod n$.

Twist compatibility at the corner point requires that

$$\begin{cases} L'_{ij}(0) = -B'_i(1) \\ R'_{ij}(0) = B'_j(0) \end{cases}.$$

Expanding the expressions of $L'_{ij}(0)$ and $R'_{ij}(0)$, we have

$$\begin{cases} p'_{ij}(0)I'_{ij}(0) + q'_{ij}(0)T_{ij}(0) = X_i \\ r'_{ij}(0)I'_{ij}(0) + s'_{ij}(0)T_{ij}(0) = Y_i \end{cases}, \tag{7}$$

where

$$\begin{cases} X_i = -B'_i(1) - p_{ij}(0)I''_{ij}(0) - q_{ij}(0)T'_{ij}(0) \\ Y_i = B'_j(0) - r_{ij}(0)I''_{ij}(0) - s_{ij}(0)T'_{ij}(0) \end{cases}. \tag{8}$$

Equation (7) has solution if and only if $I'_{ij}(0)$, $T_{ij}(0)$, X_i, and Y_i are coplanar. To fulfill that condition, we assume that X_i is the projection of $-B'_i(1)$ to the plane spanned by $I'_{ij}(0)$ and $T_{ij}(0)$, and that Y_i is the projection of $B'_j(0)$ to this plane. That is,

$$\begin{cases} X_i = [I'_{ij}(0), T_{ij}(0)] \cdot \tilde{\sigma}_{I'_{ij}(0),T_{ij}(0)}(-B'_i(1)) \\ Y_i = [I'_{ij}(0), T_{ij}(0)] \cdot \tilde{\sigma}_{I'_{ij}(0),T_{ij}(0)}(B'_j(0)) \end{cases}.$$

Thus, (8) can be translated into a linear equation system concerning $I''_{ij}(0)$ and $T'_{ij}(0)$, with the following coefficient matrix:

$$M_i = \begin{bmatrix} p_{ij}(0) & q_{ij}(0) \\ r_{ij}(0) & s_{ij}(0) \end{bmatrix}.$$

$\operatorname{rank} M_i < 2$ if and only if $L_{ij}(0)$ and $R_{ij}(0)$ are parallel. That is, $D^i_u(1)$ and $D^j_u(0)$ are dependent. Under this condition, the two boundary curves $D^i_O(u)$ and $D^j_O(u)$ are G^1-continuously connected, and can be merged into one curve. Thus, we only need to handle the case that $D^i_u(1)$ and $D^j_u(0)$ are independent. That is $\operatorname{rank} M_i = 2$. Thus, the linear equation has only one solution:

$$\begin{bmatrix} I''_{ij}(0) \\ T'_{ij}(0) \end{bmatrix} = M_i^{-1} \begin{bmatrix} -B'_i(1) - X_i \\ B'_j(0) - Y_i \end{bmatrix}.$$

And, therefore, we have

$$\begin{cases} [p'_{ij}(0), q'_{ij}(0)]^{\mathrm{T}} = \sigma_{I'_{ij}(0),T_{ij}(0)}(X_i) \\ [r'_{ij}(0), s'_{ij}(0)]^{\mathrm{T}} = \sigma_{I'_{ij}(0),T_{ij}(0)}(Y_i) \end{cases}.$$

In summary, Table 1 lists all derived derivative values. We can create Bézier or B-spline curves using these values as boundary conditions. Taking Bézier form as an example, we have

$$I_{ij}(v) = \sum_{l=0}^{4} B_{l,4}(v) I_{ij}[l], \tag{9}$$

Table 1 Computed derivative vector values

Curve	Derivatives
$I_{ij}(v)$	$I_{ij}(0)$, $I'_{ij}(0)$, $I''_{ij}(0)$, $I'_{ij}(1)$, and $I_{ij}(1)$
$T_{ij}(v)$	$T_{ij}(0)$, $T'_{ij}(0)$, $T'_{ij}(1)$, and $T_{ij}(1)$
$p_{ij}(v)$	$p_{ij}(0)$, $p'_{ij}(0)$, $p'_{ij}(1)$, and $p_{ij}(1)$
$q_{ij}(v)$	$q_{ij}(0)$, $q'_{ij}(0)$, $q'_{ij}(1)$, and $q_{ij}(1)$
$r_{ij}(v)$	$r_{ij}(0)$, $r'_{ij}(0)$, $r'_{ij}(1)$, and $r_{ij}(1)$
$s_{ij}(v)$	$s_{ij}(0)$, $s'_{ij}(0)$, $s'_{ij}(1)$, and $s_{ij}(1)$

where $B_{m,i}(v)$ $(m = 4)$ is the Bernstein basis function, and

$$\begin{cases} I_{ij}[0] = I_{ij}(0) \\ I_{ij}[1] = I_{ij}(0) + I'_{ij}(0)/4 \\ I_{ij}[2] = I_{ij}(0) + I'_{ij}(0)/2 + I''_{ij}(0)/12 \\ I_{ij}[3] = I_{ij}(1) - I'_{ij}(1)/4 \\ I_{ij}[4] = I_{ij}(1) \end{cases}.$$

Analogously, we have the Bézier form of the other curves, and those are

$$\Gamma_{ij}(v) = \sum_{l=0}^{3} B_{l,3}(v)\Gamma_{ij}[l],$$

where $\Gamma \in \{T, p, q, r, s\}$, and

$$\begin{cases} \Gamma_{ij}[0] = \Gamma_{ij}(0) \\ \Gamma_{ij}[1] = \Gamma_{ij}(0) + \Gamma'_{ij}(0)/3 \\ \Gamma_{ij}[2] = \Gamma_{ij}(1) - \Gamma'_{ij}(1)/3 \\ \Gamma_{ij}[3] = \Gamma_{ij}(1) \end{cases}.$$

Thus, $L_{ij}(v)$, $R_{ij}(v)$, and $B_i(u)$ can be computed using B-spline addition and multiplication operators [18], according to their definitions (2), (3), and (5). Since the degree of the boundary is only raised by one (see Formula (5)), and the maximum degree of the curves above is 6, the degree of the finally constructed surface is $((d_i + 1), 6)$[1], where d_i is the degree of the corresponding boundary curve. We can also construct these curves using two-degree *uniform B-spline* so that the degree of the constructed surface can be reduced to $((d_i + 1), 4)$. That is,

$$\Gamma_{ij}(v) = \sum_{l=0}^{3} N_{l,2}(v)\Gamma_{ij}[l],$$

where $\Gamma \in \{T, p, q, r, s\}$, the knot vector is $\{0, 0, 0, 1/2, 1, 1, 1\}$, and

$$\begin{cases} \Gamma_{ij}[0] = \Gamma_{ij}(0) \\ \Gamma_{ij}[1] = \Gamma_{ij}(0) + \Gamma'_{ij}(0)/4 \\ \Gamma_{ij}[2] = \Gamma_{ij}(1) - \Gamma'_{ij}(1)/4 \\ \Gamma_{ij}[3] = \Gamma_{ij}(1) \end{cases}.$$

5 Surface form

For each triangular patch (see Fig. 4), the boundary curves and the corresponding cross-boundary derivative curves are computed in the previous section. They are the boundary conditions of the final patches. We list these conditions in Table 2, where Inverse is a B-spline operator, satisfying

$$\text{Inverse}\ (\Gamma(t)) = \Gamma(1 - t).$$

Table 2 The boundary conditions of the ith patch (all positive parametric directions are anticlockwise)

Boundary No.	Boundary curve	Derivative curve
#0	$D^i_O(u)$	$B_i(u)$
#1	$I_{ij}(v)$	$L_{ij}(v)$
#2	O	Inverse$(C_i(u))$
#3	Inverse$(I_{ki}(v))$	Inverse$(R_{ki}(v))$

Since the construction method of these curves ensures that there is no compatibility problem in the corner, we can directly create Coons B-spline patches using the boundary and cross-boundary derivative conditions (see [8] and [19]). And these Coons surfaces are the final patches, interpolating all boundary conditions and G^1-continuity conditions. They are G^1 connected exactly, and has G^1-continuity with the n-sided boundary.

6 Examples

Example 1 is a classical example of filling a six-sided hole, which is often called *three-convex-three-concave hole* (see Fig. 5). This can be constructed by blending the common corner vertex of three cubes. The inner curves are in four-degree Bézier form. And all cross-boundary derivatives are seven-control-point four-degree B-spline curves. Since the degree of the boundary curves is 2, and the resulting surfaces are in the $(4, 4)$-degree B-spline form. Figure 5 (a) shows the rendered boundary surfaces and these triangular patches. There is no positional nor tangential error between adjacent surfaces. Figure 5 (b) presents the isoparametric meshes, and all these curves are continuous. The reflection lines (Fig. 5 (c)) show the G^1 continuity of the generated patches, since there is no G^0-discontinuous point anywhere in the figure.

Our algorithm also can be applied to irregular n-sided holes. Example 2 is an irregular three-sided hole filling instance. The specified boundary conditions are all in three-degree B-spline form. Figure 6 (a) shows the rendered result of the triangular blending patches and the boundary surfaces. And Fig. 6 (b) is the corresponding isoparametric curves. Figure 6 (c) shows the zebra stripes chart[2] of

[1] In this paper, (d_u, d_v)-degree means in u-direction, the degree of the parametric surface is d_u, and in v-direction, it is d_v.

[2] Like reflection lines, the *zebra stripes chart* is also a powerful tool to see small changes in a surface, which simulates the reflection of long strips of light on a very shiny surface. The figures are gener-

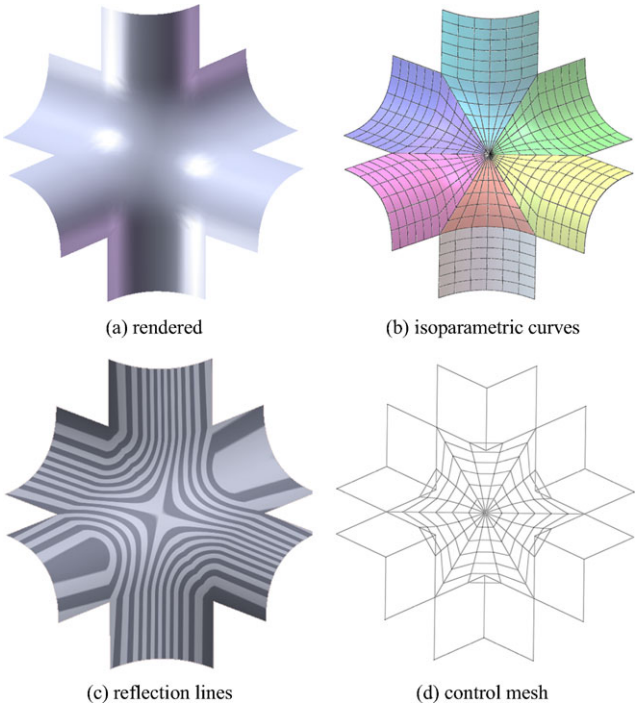

Fig. 5 Example 1: filling a six-sided hole. (**a**) Is the rendered patches, with three lights in different directions; these patches are colored differently in (**b**), with their isoparametric curves; (**c**) shows continuous reflection lines, which underline their G^1 continuity; and in (**d**), all control meshes of these B-spline surfaces are plotted

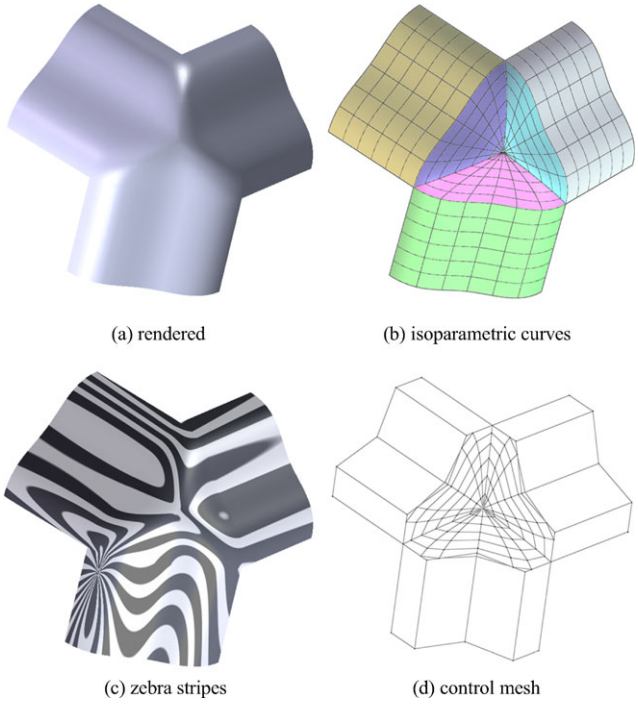

Fig. 6 Example 2: filling an irregular three-sided corner. (**c**) Shows the zebra stripes chart. It is also a powerful tool to verify the continuity among surface patches like reflection lines

these surfaces, which also show their G^1 continuity. The G^0-continuity of these lines represents the G^1 continuity of these surfaces. Figure 6 (d) is the mesh of control points. The resulting surfaces are in the $(4, 4)$-degree B-spline form. Figure 7 shows two alternative region-filling results of the same boundary condition, with various manually specified central point. Figure 8 shows comparison results of our method and other industrial modeling systems. The appearance of the generated surfaces by different methods are almost the same. But the corresponding computed errors listed in Table 3 show that the other methods introduce tangential errors on the boundary, but ours does not.

Figure 8 shows comparison results of our method and other industrial modeling systems. The appearance of the generated surfaces by different methods are almost the same. But the corresponding computed errors listed in Table 3 show that the other methods introduce tangential errors on the boundary, but ours does not.

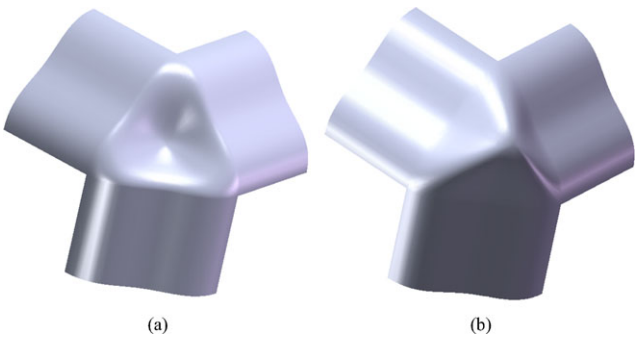

Fig. 7 Various region-filling results of the same boundary condition, generated by specifying different central point manually. (**a**) Shows a concave case and (**b**) shows an asymmetrical case where the central point is set to be closer to the right boundary surface

Example 3 is a five-sided hole filled with 5 triangular B-spline surfaces with incompatible boundary conditions. This case is often generated by blending three orthogonally intersecting edges using different radii (here, we choose 0.8, 1.0, and 0.6). The boundary surfaces are all in $(2, 1)$-degree B-spline form. And the degree of the generated surfaces is $(3, 6)$ (see Fig. 9 (d)). The reflection lines (Fig. 9 (c)) present the G^1 continuity of the generated patches, since there is no G^0-discontinuous point anywhere in the figure.

ated by SolidWorks 2007, using the *spherical map*. "The part appears to be inside a large sphere that is covered on the inside with strips of light. The zebra stripes are always curved (even on flat faces) and display singularities." For more information, please see the web page http://help.solidworks.com/2010/English/SolidWorks/sldworks/Display/HIDD_Zebra_Properties.htm.

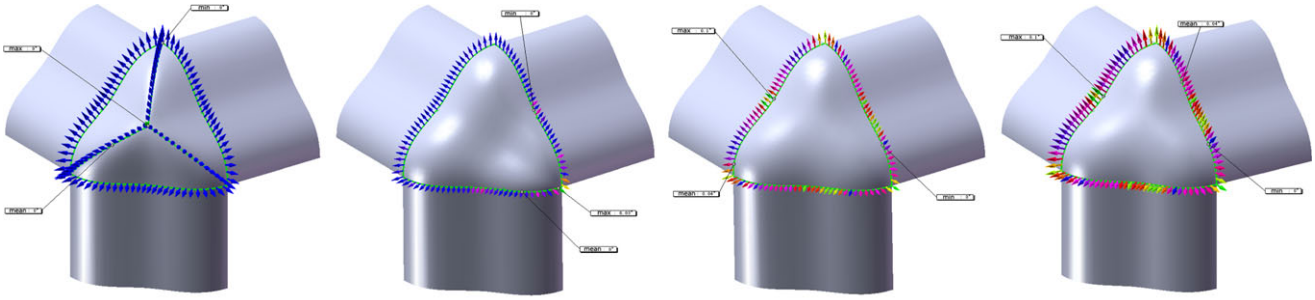

Fig. 8 Error comparison with the surfaces generated by other industrial modeling systems. The vectors in the figures are the normal vectors on the edges. The color gradient of these vectors from *blue* (for zero) to *red* (for the maximum error) shows the angular error value of two corresponding normal vectors of adjacent surfaces at the same point

Table 3 Angular error comparison of our method with SolidWorks 2007, Pro ENGINEER 2002 (Wildfire 3.0), and CATIA V5. These data were generated by SolidWorks 2007

Angular error	Maximum	Mean
SolidWorks	0.10°	0.04°
CATIA	0.10°	0.04°
Pro ENGINEER	0.03°	$< 0.01°$
Our method	0.00°	0.00°

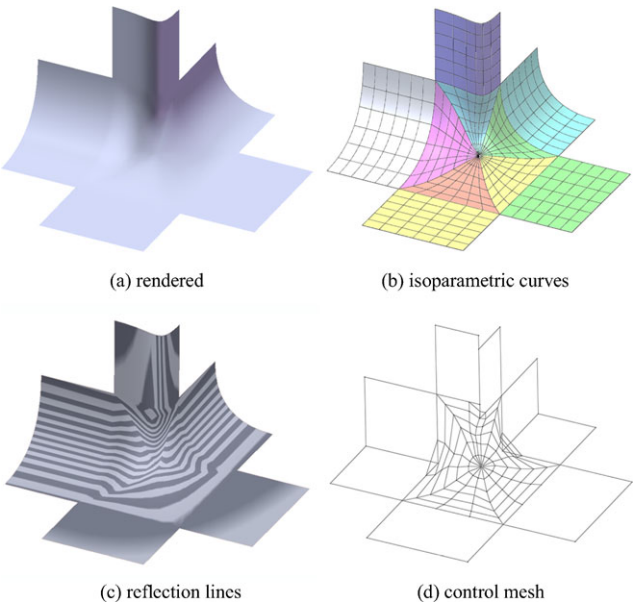

Fig. 9 Example 3: filling a five-sided hole

Example 4 shows a filling result of a randomly generated five-sided hole (and of course, the boundary conditions are incompatible in the corner) (see Fig. 10).

Table 4 shows the storage space comparison among SolidWorks, Pro ENGINEER, CATIA, and our method. The data in the last row shows the numbers of required control points for the blending surfaces by our method. The first row is for the single trimmed blending surface constructed by SolidWorks 2007 using positional and tangential constraints. CATIA V5 yields analogous results in the second row. The third row is for n untrimmed B-spline patches generated by Pro ENGINEER 2002. Our algorithm needs less than 50% storage space of SolidWorks and CATIA, and this rate is even lower compared with the results by Pro ENGINEER. Further analysis shows that the rate reduces if the model is more complicated. Our algorithm generates $(4, 4)$-degree surfaces, SolidWorks 2007 and CATIA V5 yield $(5, 5)$-degree trimmed surfaces, Pro ENGINEER 2002 can generate $(3, 3)$-degree untrimmed surfaces in IGES format.

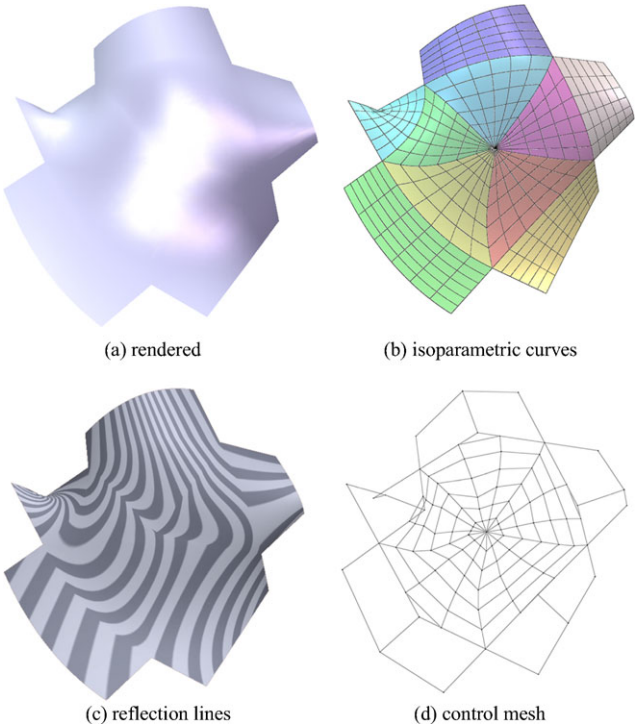

Fig. 10 Example 4: filling a five-sided random hole

Table 4 Storage space (control points) comparison among SolidWorks 2007, Pro ENGINEER 2002 (Wildfire 3.0), CATIA V5 and our method

Algorithm	Ex. 1	Ex. 2	Ex. 3	Ex. 4
SolidWorks	324	540	504	594
CATIA	324	630	504	648
Pro ENGINEER	1750	324	600	1680
Ours	140	147	168	140

Table 5 Elapsed-time comparison between a constrained-optimization method and our algorithm

Algorithm	Ex. 1	Ex. 2	Ex. 3	Ex. 4
SolidWorks	120 ms	90 ms	200 ms	160 ms
Ours	45 ms	43 ms	47 ms	46 ms
Rate	38%	47%	24%	29%

Table 5 is a comparison between the constrained-optimization method of an industrial modeling system[3] and our algorithm. Since our algorithm is implemented in C# 2.0, Microsoft Visual Studio 2005, it should be much faster if we re-implement this in C++. The testing environment is Intel E5300 @ 2.60 GHz, 4 GB Memory. The results demonstrate that our algorithm is much more efficient.

7 Conclusions

This paper presents a method for filling n-sided regions with triangular Coons B-spline surfaces, which achieves G^1 continuity exactly. Our method solves the compatibility problem in the corner using patched standard B-spline surfaces without introducing positional nor tangential error. It requires less storage space and the B-spline form benefits data exchange. The algorithm is simple, containing no time-consuming iteration or large matrix calculation. We also provide geometric parameters to control the shape of the constructed surfaces. The patches usually requires less storage space than popular constrained-optimization methods, and the surface degree is only one larger than the degree of corresponding boundary curve. The continuity of this method can be improved to G^2 while more compatibility conditions at the corner points are solved. And this future work is more challenging and significant for high-accuracy modeling.

[3]We only compared the efficiency of our algorithm with SolidWorks 2007, since it can provide a statistical table of the elapsed time of each complicated operation, rather than kernel/extended programming of other platforms.

Acknowledgements The authors would like to thank the anonymous reviewers for their helpful comments, and also thank Jia-Ting Chen of Tsinghua University for his valuable suggestions. The research was supported by Chinese 973 Program (2010CB328001), the National Science Foundation of China (60625202, 90715043), and Chinese 863 Program (2007AA040401). The second author was supported by the Fok Ying Tung Education Foundation (111070). The fourth author was supported by ANR-NSFC (60911130368).

References

1. Gregory, J.A., Lau, V.K.H., Zhou, J.W.: Smooth parametric surfaces and N-sided patches. In: Computation of Curves and Surfaces (1989)
2. Plowman, D., Charrot, P.: A practical implementation of vertex blend surfaces using an n-sided patch. In: Proceedings of the 6th IMA Conference on the Mathematics of Surfaces, pp. 67–78 (1994)
3. Shi, K.L., Yong, J.H., Sun, J.G., Paul, J.C.: G^n blending multiple surfaces in polar coordinates. Comput. Aided Des. (2010). doi:10.1016/j.cad.2009.11.009
4. Hosaka, M., Kimura, F.: Non-four-sided patch expression with control points. Comput. Aided Geom. Des. **1**, 75–86 (1984)
5. Sabin, M.A.: Non-rectangular surface patches suitable for inclusion in a b-spline surface. In: Proceedings of Eurographics, pp. 57–69 (1983)
6. Gregory, J.A., Zhou, J.W.: Filling polygonal holes with bicubic patches. Comput. Aided Geom. Des. **11**, 391–410 (1994)
7. Hahn, J.: Theory and Practice of Geometric Modeling. Springer, Berlin (1989)
8. Piegl, L.A., Tiller, W.: Filling n-sided regions with NURBS patches. Vis. Comput. **15**, 77–89 (1999)
9. Yang, Y.J., Yong, J.H., Zhang, H., Paul, J.C., Sun, J.G.: A rational extension of Piegl's method for filling n-sided holes. Comput. Aided Des. **38**, 1166–1178 (2006)
10. Karčiauskas, K., Peters, J.: Bicubic polar subdivision. ACM Trans. Graph. **26**, 14:1–14:6 (2007)
11. Karčiauskas, K., Myles, A., Peters, J.: A C^2 polar jet subdivision. In: Proceedings of the fourth Eurographics symposium on Geometry processing, pp. 173–180 (2006)
12. Karčiauskas, K., Peters, J.: Bi-3 C^2 polar subdivision. ACM Trans. Graph. **28**, 48:1–48:12 (2009)
13. Hwang, W.C., Chuang, J.H.: n-sided hole filling and vertex blending using subdivision surfaces. J. Inf. Sci. Eng. **19**, 857–879 (2003)
14. Li, G.Q., Li, H.: Blending parametric patches with subdivision surfaces. J. Comput. Sci. Technol. **17**, 498–506 (2002)
15. Liang, X.Z., Che, X.J., Li, Q.: G^2 continuity conditions for two adjacent B-spline surfaces. In: Proceedings of the Geometric Modeling and Processing (2004)
16. Ye, X.Z.: The Gaussian and mean curvature criteria for curvature continuity between surfaces. Comput. Aided Geom. Des. **13**, 549–567 (1996)
17. Ye, X.Z., Liang, Y.D., Nowacki, H.: Geometric continuity between adjacent Bézier patches and their constructions. Comput. Aided Geom. Des. **13**, 521–548 (1996)
18. Piegl, L.A., Tiller, W.: Symbolic operators for NURBS. Comput. Aided Des. **29**, 261–268 (1997)
19. Piegl, L.A., Tiller, W.: The NURBS book, 2nd edn. Springer, Berlin (1997)

Kan-Le Shi received his B.Sc. degree from School of Software, Tsinghua University in 2007. Currently, he is a Ph.D. candidate in the Department of Computer Science and Technology, Tsinghua University, and also in the Institute of Computer Graphics and Computer-Aided Design, School of Software, Tsinghua University. His research interests are geometric modeling, computer-aided design, and computer graphics.

Jun-Hai Yong is currently a professor at the School of Software at Tsinghua University, China. He received his B.S. and Ph.D. in computer science from the 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.

Jia-Guang Sun is a Member of the Chinese Academy of Engineering, and 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.

Jean-Claude Paul is a senior researcher at CNRS and INRIA (France), and is currently the 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 modeling, and computer-aided design.

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 the USA from 2005 to 2006. He held a visiting professor position at the 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.