A rational extension of Piegl's method for filling n-sided holes

Yi-Jun Yang, Jun-Hai Yong, Hui Zhang, Jean-Claude Paul, Jia-Guang Sun

A rational extension of Piegl's method for filling n-sided holes

Yi-Jun Yang[a,b,*], Jun-Hai Yong[a], Hui Zhang[a], Jean-Claude Paul[a], Jia-Guang Sun[a,b]

[a] *School of Software, Tsinghua University, Beijing 100084, China*
[b] *Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China*

Received 13 March 2006; accepted 15 July 2006

Abstract

N-sided hole filling plays an important role in vertex blending. To deal with the case that the corner is surrounded by rational surfaces (i.e. NURBS surfaces), an algorithm to fill n-sided holes with ε-G^1 continuous NURBS patches that interpolate the given boundary curves and approximate the given cross-boundary derivatives is presented based on Piegl's method. The NURBS surfaces joining along inner or boundary curves have normal vectors that do not deviate more than the user-specified angular tolerance ε. The boundaries as well as cross-boundary derivatives can all be NURBS curves. No restrictions are imposed on the number of boundary curves, and the cross-boundary derivatives can be specified independently.

Keywords: N-sided hole; NURBS; Coons surfaces; ε-G^1 continuity

1. Introduction

N-sided hole filling for vertex blending is a recurring operation in computer aided geometric design. N-sided hole filling aims to provide surface patches that interpolate the hole's boundary and cross-derivative curves. For industrial applications, NURBS has become the de facto standard representation for 3D models [13]. Since most design systems rely on quadrilateral surfaces, whenever the need arises to represent surfaces with an arbitrary number of boundary curves, a collection of NURBS patches satisfying some level of continuity is needed. The relevant papers on this subject are listed in [1,6,7,13,18]. The majority of these methods fill the hole with quadrilateral patches by first splitting the hole into a quadrilateral mesh and then fitting Coons patches in each quadrilateral subregion with parametric or geometric continuity. The approaches of this kind may incur the well-known twist compatibility problem [18], in which the twist vectors for patches around the centre are difficult to derive, especially when the number of incident edges is even. To avoid twist incompatibility, Chiyokura [2] used Gregory-type patches to fill each region which can then be converted to NURBS form to yield fairly high-degree surfaces. Piegl [13] presented an algorithm to fill n-sided B-spline holes (holes whose boundaries and cross-boundary derivatives are all B-spline curves) with ε-G^1 continuous B-spline patches. Given a set of B-spline boundary curves, a set of B-spline cross-boundary derivatives and an angular tolerance ε, the algorithm constructs a collection of B-spline surfaces that fill the hole, bounded by the given curves, with ε-G^1 continuity, i.e. the angle between the surface Normals of any two neighboring B-spline patches does not deviate more than the user-specified angular tolerance ε. Twist incompatibility is handled by knot insertion and derivative approximation. There are no restrictions on the number of boundary curves and the degree of the input entities is raised by only one.

Surfaces are said to be ε-G^1 continuous if the maximum tangent-plane discrepancy between any pair of adjacent patches is bounded by some tolerance $\varepsilon > 0$. Approximate continuity, first proposed by DeRose and Mann [3], has been applied in many CAD applications such as surface interpolation [3, 16], surface blending [4], terrain modelling [11] and n-sided hole filling [13]. DeRose [3] gave an approximate continuity scheme of interpolating the vertices of a polyhedron using cubic triangular Bézier patches and showed that the resulting surfaces have much better distribution of curvature than traditional G^1 schemes. Tookey [16] proposed a method of

* Corresponding address: Institute of CG & CAD, School of Software, Tsinghua University, Beijing 100084, China. Tel.: +86 010 62795442; fax: +86 010 62795460.
E-mail address: yangyijuny@gmail.com (Y.-J. Yang).

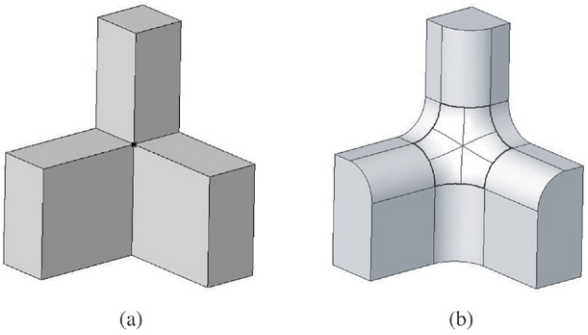


Fig. 1. An example: (a) a model (the black point denotes a vertex to be blended); (b) vertex blending result of (a) (the bold curves denote the boundaries of the n-sided hole which are conics).

interpolating a quadrilateral network of rational cubic curves using approximate continuous surfaces, the method virtually de-skills the surface interpolating activity without degrading the quality of the finished product. Elsas [4] presented a method of generating approximate continuous surface blending to support interactive sculptured feature design. The results show that the sculptured feature design is flexible and with high speed. To satisfy the requirements of 3D terrain models, Pferifer [11] gave an approximate continuity scheme of computing one patch for each triangle of the terrain mesh models. Piegl [13] presented a method of filling n-sided holes with approximate continuous patches, the method has no restrictions on the input entities. Summarily, approximate continuity schemes can introduce two benefits. First, lower degree surfaces are used in approximate continuity schemes. Second, as a consequence of the relaxation of continuity conditions, approximate continuity schemes are not obliged to solve the problems encountered by G^1 methods such as vertex enclosure. Traditional G^1 methods use higher degree surfaces to ensure the smooth connection of adjacent surfaces and leave more degrees of freedom to set than algorithms using approximate continuity. The setting of these extra degrees of freedom complicates the algorithms and may result in poor distribution of curvature. To date, the experience from presented algorithms [3,4,11,13,16] suggests that approximate continuity will virtually simplify the surface modelling activities without degrading the quality of the finished product.

In the past 20 years, B-spline hole filling has been extensively studied in many literatures such as [2,5,13,14, 17], but little attention has been paid to the case where the boundaries or/and cross-boundary derivatives are rational curves. The rational case exists extensively in vertex blending. As conics can be represented as rational Bézier curves in CAD/CAM, there exist rational boundaries in the generated n-sided holes even when blending a simple model with just plane surfaces (see Fig. 1). Moreover, the rational surfaces give the users the ability to model more complex parts such as cars and airplanes. So we should pay special attention to the case where the boundaries or/and cross-boundary derivatives are rational curves. Generally, there are three approaches for filling a rational hole with NURBS patches. The first approach is to increase the degree of filling surfaces as appropriate according to the propagation of continuity constraints, the second approach is to localize the propagation of continuity constraints by refining filling surfaces in order to obtain supplementary degrees of freedom [15], and the third approach is to fill the hole with ε-G^1 continuous NURBS patches. The first approach usually uses the patches of degree higher than biquartic. The second approach introduces more patches and more specific cases to handle than the other two approaches. The third approach fills the hole with ε-G^1 continuous NURBS patches. The first two approaches are not included in options because a large number of patches or high degree patches may cause some troubles in practice. The third approach using approximation with a manageable tolerance is preferable.

To handle the case that given boundaries or/and cross-boundary derivatives are all rational curves, an algorithm to fill an n-sided hole with ε-G^1 continuous NURBS patches that interpolate boundary curves and approximate given cross-boundary derivatives is presented based on Piegl's method. To fill NURBS holes, twist vectors and derivative fields must be embedded in homogeneous space to obtain four-dimensional compatibility while still maintaining the angular tolerance on the three-dimensional vector directions. Because most design systems do not allow surfaces with negative weights, boundary curves are reparameterized and Lin's method [9] to construct homogeneous Coons patches is introduced to guarantee that all the weights of the resultant patches are positive. The main contribution of our work can be summarized as follows:

- We extend Piegl's method to deal with the case where the boundaries or/and cross-boundary derivatives are rational curves.
- We utilize boundary reparameterization and Lin's Coons construction method to avoid negative weights of the resultant patches.

The organization of the paper is as follows. In Section 2, input and output handling are discussed. Section 3 describes boundary reparameterization. Section 4 contains details on how to compute a central point and a central Normal, and in Section 5, inner curves are computed. Section 6 describes the cross-derivative generation. The problems of boundary compatibility and twist compatibility are dealt with in Section 7. In Section 8, Coons patches are constructed. Section 9 concludes the paper.

2. Input and output handling

A rational curve in three-dimensional space is defined by

$$\mathbf{C}(t) = \frac{\sum_{i=0}^{n} N_{i,p}(t)\omega_i \mathbf{P}_i}{\sum_{i=0}^{n} N_{i,p}(t)\omega_i},$$

where the $\mathbf{P}_i$ are the control points, the ω_i are the weights, and the $N_{i,p}(t)$ are the pth-degree B-spline basis functions. Homogeneous coordinates offer an efficient method

to represent NURBS curves. The NURBS curve can be represented by a nonrational curve in four-dimensional space

$$\mathbf{C}^{\omega}(t) = \sum_{i=0}^{n} N_{i,p}(t) \begin{pmatrix} \omega_i \mathbf{P}_i \\ \omega_i \end{pmatrix}.$$

For the rest of the paper, we assume that points are represented using homogeneous coordinates. Assume that we have a set of boundary curves (see Fig. 2)

$$\mathbf{C}_k^{\omega}(t) = \sum_{i=0}^{n_k} N_{i,p_k}(t)\mathbf{P}_{k_i}^{\omega}, \quad k = 0, 1, \ldots, N,$$

and a set of corresponding cross-boundary derivatives

$$\mathbf{D}_k^{\omega}(t) = \sum_{i=0}^{n_k} N_{i,p_k}(t)\mathbf{V}_{k_i}^{\omega}, \quad k = 0, 1, \ldots, N,$$

where $\mathbf{P}_{k_i}^{\omega}$ and $\mathbf{V}_{k_i}^{\omega}$ denote the homogeneous control points of boundaries and cross-boundary derivatives. We want to construct a set of $N+1$ NURBS patches to fill the hole with ε-G^1 continuity so that, along each edge shared by two patches, the angle of the surface Normals does not deviate more than ε. We assume that the curves satisfy the following compatibility conditions:

1. The parameterizations of the homogeneous boundary curves are consistent, i.e.:

   $$\mathbf{C}_k^{\omega}(1) = \mathbf{C}_{k+1}^{\omega}(0), \quad k = 0, 1, \ldots, N, \tag{1}$$

   where all indexes are computed modulo $N+1$. If this condition is not satisfied (the common control points of two adjacent boundary curves may have different weights), the curves are reparameterized to make the parameter flow around the boundary (see Section 3).
2. The homogeneous boundary curves and the corresponding homogeneous cross-boundary derivatives are defined over the same knot vector and have the same degree. If the homogeneous boundary curves and the corresponding homogeneous cross-boundary derivatives are not defined over the same knot vector and/or have different degrees, knot insertion and/or degree elevation are applied.
3. The homogeneous cross-boundary derivatives point toward the inside of the hole. If the homogeneous cross-boundary derivatives point toward the outside of the hole, the control vectors of the offending derivative curves are flipped.

The second and the third conditions are the same as for filling B-spline holes [13]. Compared with the condition for filling B-spline holes, Condition (1) is different in that the weights are involved. The main algorithm flow is described as follows:

1. Reparameterize the boundary curves.
2. Compute a central point and a central Normal.
3. Generate the homogeneous inner curves running from the central point to the split points of each boundary curve.
4. Generate the homogeneous cross-boundary derivatives for each inner curve.
5. Deal with boundary and twist incompatibility in four-dimensional space while maintaining the prescribed angular tolerance between the three-dimensional vector directions.

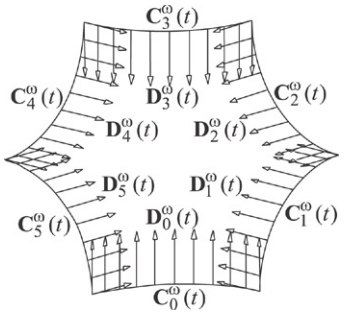


Fig. 2. Boundary curves and cross-boundary derivatives.

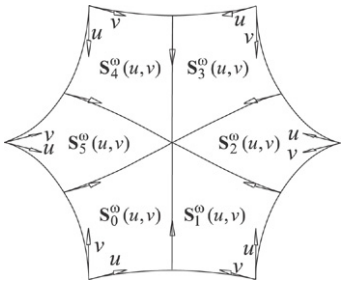


Fig. 3. Parameterizations of homogeneous surface patches.

6. Put constructed homogeneous Coons patches

   $$\mathbf{S}_k^{\omega}(u,v) = \sum_{i=0}^{n_k}\sum_{j=0}^{m_k} N_{i,p_k}(u)N_{j,q_k}(v)\mathbf{P}_{k_i,k_j}^{\omega},$$
   $$k = 0, 1, \ldots, N,$$

   in the hole in a consistent manner (see Fig. 3).

The following sections illustrate how to construct these homogeneous surfaces.

3. Boundary reparameterization

Let $\mathbf{CL}_k^{\omega'}(t)$ denote the derivative curve of $\mathbf{CL}_k^{\omega}(t)$ in homogenous space. To ensure that no negative weights are introduced to the resultant Coons patches, the boundaries must be reparameterized to satisfy the following condition:

$$E_4(\mathbf{CL}_k^{\omega}(1)) - \frac{E_4(\mathbf{CL}_k^{\omega'}(1))}{3} > 0, \tag{2}$$

(which is a sufficient condition required in Section 6) where E_4 is an operator that returns the fourth element of given vector and $\mathbf{CL}_k^{\omega}(t)$ is the left half curve of $\mathbf{C}_k^{\omega}(t)$. The geometric meaning of Condition (2) is that if we interpolate the end points and the end derivatives of $\mathbf{CL}_k^{\omega}(t)$ using a cubic Bézier curve, weight of the third control point is positive. Weight of the second control point is also guaranteed to be positive by twist adjustment (Section 8). Also to make the parameter flow around the boundary (see Condition (1)), boundary reparameterization is needed. We now outline the process of boundary reparameterization.

1. Split the boundary curves and derivatives at their parametric midpoints by computing

   $$\{\mathbf{CL}_k^{\omega}, \mathbf{CR}_k^{\omega}\} \longleftarrow SPLIT(\mathbf{C}_k^{\omega}, 0.5),$$

and

$$\{\mathbf{DL}_k^{\omega}, \mathbf{DR}_k^{\omega}\} \longleftarrow SPLIT(\mathbf{D}_k^{\omega}, 0.5).$$

Assuming the boundary curve $\mathbf{CL}_k^{\omega}(t)$ defined over the knot vector

$$\mathbf{U}_k = \{\underbrace{0, \ldots, 0}_{p_k+1}, u_{p_k+1}, \ldots, u_{n_k}, \underbrace{1, \ldots, 1}_{p_k+1}\},$$

Condition (2) can be rewritten as

$$\omega_{n_k} - \frac{(\omega_{n_k} - \omega_{n_k-1})p_k}{3(1-u_{n_k})} > 0, \tag{3}$$

where p_k is the degree of curve $\mathbf{CL}_k^{\omega}(t)$. If Conditions (1) and (2) hold for all boundary curves $\mathbf{CL}_k^{\omega}(t)$, the reparameterization process is complete; otherwise, the following steps are performed for every boundary that fails.

2. The general form of Möbius transformation $s = \psi(u)$ that maps the interval $\mathbf{I}[\tau_0, \tau_1]$ into $\mathbf{J}[\xi_0, \xi_1]$ with $\psi(\tau_0) = \xi_0$ and $\psi(\tau_1) = \xi_1$ is

$$s = \psi(u) = \frac{\alpha u + \beta}{\gamma u + \delta}. \tag{4}$$

The reparameterization theorem [8] states that, if we apply the transformation ψ to all the knot values u_i, the shape of the curve as well as its control points are unchanged, but its weights take the new values

$$\widetilde{\omega}_i = \omega_i \prod_{j=1}^{p_k} (\gamma s_{i+j} - \alpha).$$

To reparameterize the whole curve $\mathbf{CL}_k(t)$ with a Möbius transformation, the direction of the homogeneous derivative $\mathbf{CL}_k^{\omega'}(0)$ is inevitably altered though the direction of the three-dimensional vector $\mathbf{CL}_k'(0)$ is unchanged, which may cause violation of the original boundary compatibility (see Section 7 for details). At the same time, the last control point of $\mathbf{CL}_k^{\omega}(t)$ and the first control point of $\mathbf{CR}_k^{\omega}(t)$ should be the same. To maintain $\mathbf{CL}_k^{\omega'}(0)$ and $\mathbf{CL}_k^{\omega}(1)$ (the last control point) unchanged, the curve $\mathbf{CL}_k(t)$ is split into three curves and the three curves are then reparameterized respectively. The details are shown as follows. The curve $\mathbf{CL}_k(t)$ is split into three curves by inserting the first and last interior knot values u_{p_k+1} and u_{n_k} of the vector $\mathbf{U}_k$ $p_k - l_i$ and $p_k - l_j$ times, respectively, where l_i and l_j are the multiplicity of u_{p_k+1} and u_{n_k}. Let

$$\mathbf{P}_1 = \mathbf{CL}_k(u_{p_k+1}) \quad \text{and} \quad \mathbf{P}_2 = \mathbf{CL}_k(u_{n_k}).$$

Then the curve $\mathbf{CL}_k(t)$ is decomposed into three constituent NURBS curves $\mathbf{CL}_{k_1}(t)$, $\mathbf{CL}_{k_2}(t)$ and $\mathbf{CL}_{k_3}(t)$ (see Fig. 4) by the two points $\mathbf{P}_1$ and $\mathbf{P}_2$. The first curve $\mathbf{CL}_{k_1}(t)$ is reparameterized aiming to satisfy Condition (1) such that the parameter can flow around the boundary. The third curve $\mathbf{CL}_{k_3}(t)$ is reparameterized such that Condition (2) (to avoid introducing negative weights to the resultant patches) is satisfied for the current curve $\mathbf{CL}_k(t)$. After the first and third curves are reparameterized, the second curve $\mathbf{CL}_{k_2}(t)$ is then reparameterized such that it can connect the reparameterized two curves $\widetilde{\mathbf{CL}}_{k_1}(t)$ and $\widetilde{\mathbf{CL}}_{k_3}(t)$ consistently. The first curve is reparameterized as follows.

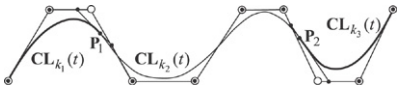


Fig. 4. Decomposition of $\mathbf{CL}_k(t)$. "○" stands for the control point of $\mathbf{CL}_k(t)$, and "•" stands for the control point of $\mathbf{CL}_{k_1}(t)$ (left thick curve), $\mathbf{CL}_{k_2}(t)$ (middle thin curve) and $\mathbf{CL}_{k_3}(t)$ (right thick curve), respectively.

If

$$f_s = \frac{E_4(\mathbf{C}_{k-1}^{\omega}(1))}{E_4(\mathbf{C}_k^{\omega}(0))} > 1,$$

the curve $\mathbf{CL}_{k_1}(t)$ is reparameterized as follows:

$$\omega_m = \omega_m f_s, \quad m = 0, 1, \ldots, p_k,$$

where ω_m denotes weight of the control point of the curve $\mathbf{CL}_{k_1}(t)$; otherwise, $\mathbf{CL}_{k_1}(t)$ is unchanged ($f_s \leq 1$). That is, we only enlarge weights of the control points of the curve with small corner point weight when two adjacent curves are not consistent at the mutual corner. The third curve $\mathbf{CL}_{k_3}(t)$ is reparameterized as follows. If $\mathbf{CL}_k^{\omega}(t)$ does not satisfy Condition (3), applying the Möbius transformation to the curve $\mathbf{CL}_{k_3}(t)$, letting the left expression of Condition (3) equal to $\omega_{n_{k_3-1}}$ and setting $\gamma - \alpha$ equal to 1, we have

$$a = \frac{3(1-s_{n_k})(\omega_{n_{k_3}-1} - \omega_{n_{k_3}}) + \omega_{n_{k_3}-1} s_{n_k} p_k - \omega_{n_{k_3}} p_k}{\omega_{n_{k_3}-1}(1-s_{n_k})p_k}, \tag{5}$$

where n_{k_3} denotes the index of the last control point of curve $\mathbf{CL}_{k_3}^{\omega}(t)$. After the Möbius transformation (4), the knot span $(u_{n_k}, 1)$ of curve $\mathbf{CL}_{k_3}^{\omega}(t)$ should be unchanged. Using $\gamma - \alpha = 1$ and Eq. (5), we have

$$\alpha = \frac{3(\omega_{n_{k_3}-1} - \omega_{n_{k_3}})(u_{n_k} - 1) + (u_{n_k}\omega_{n_{k_3}-1} - \omega_{n_{k_3}})p_k}{(1-u_{n_k})\omega_{n_{k_3}-1}p_k}.$$

Then β, γ and δ can be deduced similarly. After the above reparameterization, Condition (2) always holds for the curve $\mathbf{CL}_k^{\omega}(t)$. Last, the curve $\mathbf{CL}_{k_2}(t)$ is reparameterized such that it can connect the reparameterized curves $\widetilde{\mathbf{CL}}_{k_1}^{\omega}(t)$ and $\widetilde{\mathbf{CL}}_{k_3}^{\omega}(t)$ consistently at the points $\mathbf{P}_1$ and $\mathbf{P}_2$, respectively. Let

$$f_e = (\gamma u_{n_k} - \alpha)^{p_k},$$

if the curve $\mathbf{CL}_{k_3}(t)$ has been reparameterized; otherwise, set $f_e = 1$. We can demonstrate that $f_e \geqslant 1$ (the verification is omitted). Set $f_s = 1$ when the curve $\mathbf{CL}_{k_1}(t)$ is unchanged. Then the start and end derivatives of the curve $\mathbf{CL}_{k_2}(t)$ are scaled by f_s and f_e times respectively. Considering that the knot span is unchanged, we have:

$$\begin{cases} \alpha = \dfrac{u_{p_k+1}\sqrt[p_k]{f_e} - u_{n_k}\sqrt[p_k]{f_s}}{u_{n_k} - u_{p_k+1}} \\ \beta = \dfrac{u_{n_k}u_{p_k+1}(\sqrt[p_k]{f_s} - \sqrt[p_k]{f_e})}{u_{n_k} - u_{p_k+1}} \\ \gamma = \dfrac{\sqrt[p_k]{f_e} - \sqrt[p_k]{f_s}}{u_{n_k} - u_{p_k+1}} \\ \delta = \dfrac{u_{p_k+1}\sqrt[p_k]{f_s} - u_{n_k}\sqrt[p_k]{f_e}}{u_{n_k} - u_{p_k+1}}. \end{cases}$$

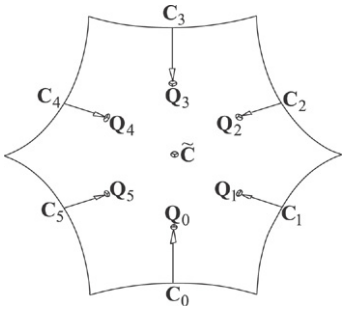


Fig. 5. Computing the central point.

The mutual point $\mathbf{P}_1$ has the same weight after the curves $\mathbf{CL}_{k_1}(t)$ and $\mathbf{CL}_{k_2}(t)$ are reparameterized while the mutual point $\mathbf{P}_2$ has the same weight after the curves $\mathbf{CL}_{k_2}(t)$ and $\mathbf{CL}_{k_3}(t)$ are reparameterized. After the above reparameterization process, Conditions (1) and (2) hold for the current curve $\mathbf{CL}_k^{\omega}(t)$. The weights of curve $\mathbf{CL}_k^{\omega}(t)$ are enlarged or unchanged during the reparameterization process, which helps to avoid introducing negative weights to the resultant Coons patches.

3. Reverse the parameterizations of $\mathbf{CR}_k$ and $\mathbf{DR}_k$. Similar reparameterization process is performed for all boundary curves $\mathbf{CR}_k(t)$.

4. Central point and Normal

The central point and Normal influence the shape of the n-sided patches greatly. To compute a satisfactory central point, we first derive an initial central point from the middle boundary points and the middle cross-boundary derivatives. Then the initial central point is projected to a least square plane of the middle boundary points, whose Normal is designated as the central Normal. Last the weights of the central point and Normal are determined. The details are shown as follows:

1. Compute

$$\mathbf{C}_k^{\omega} = \mathbf{C}_k^{\omega}(0.5) \quad \text{and} \quad \mathbf{D}_k^{\omega} = \mathbf{D}_k^{\omega}(0.5).$$

The central point and Normal are computed geometrically in three-dimensional space while the above derivatives $\mathbf{D}_k^{\omega}$ are homogenous control points. Thus we should determine the three-dimensional derivative $\mathbf{V}_k$ before computing the initial central point and Normal. The three-dimensional derivatives $\mathbf{V}_k$ are derived from $\mathbf{D}_k^{\omega}$ as follows. Let

$$\mathbf{C}_k^{\omega} = \begin{pmatrix} \omega_{k_0}\mathbf{C}_k \\ \omega_{k_0} \end{pmatrix} \quad \text{and} \quad \mathbf{D}_k^{\omega} = \begin{pmatrix} \mathbf{D}_k \\ \widetilde{\omega}_k \end{pmatrix},$$

where $\widetilde{\omega}_k$ that is the fourth element of the vector $\mathbf{D}_k^{\omega}$ may be zero. Considering that the point $\mathbf{C}_k$ and the first control point $\mathbf{P}_{k_0}$ of the kth inner curve are the same (see Fig. 5), we have:

$$\mathbf{P}_{k_1} = \frac{\omega_{k_0}\mathbf{P}_{k_0} + t\mathbf{D}_k}{\omega_{k_0} + t\widetilde{\omega}_k}, \tag{6}$$

where $\mathbf{P}_{k_1}$ is the second control point of the kth inner curve and t is a free factor. The three-dimensional derivative [12] can be represented as:

$$\mathbf{V}_k = 3\frac{\omega_{k_0} + t\widetilde{\omega}_k}{\omega_{k_0}}(\mathbf{P}_{k_1} - \mathbf{P}_{k_0}). \tag{7}$$

Substituting Eq. (6) into Eq. (7), we have the three-dimensional derivative

$$\mathbf{V}_k = \frac{3(\mathbf{D}_k - \widetilde{\omega}_k\mathbf{P}_{k_0})t}{\omega_{k_0}}.$$

2. Determine an approximate plane by solving a least square problem for the points $\mathbf{C}_k, k = 0, 1, \ldots, N$. Here we use $\mathbf{N}_f$ to denote the Normal of the approximate plane.
3. Scale the parameter t to let $\mathbf{V}_k$ have the following length [13]:

$$|\mathbf{V}_k| = \frac{1}{8}[Arc\_Length(\mathbf{C}_{k-1}) + Arc\_Length(\mathbf{C}_{k+1})].$$

The magnitude of $\mathbf{V}_k$ is one-quarter of the average of the arc lengths of the two neighboring boundaries (see Fig. 5).

4. Compute the point $\widetilde{\mathbf{C}}$ (see Fig. 5):

$$\mathbf{Q}_k = \mathbf{C}_k + \mathbf{V}_k,$$

and

$$\widetilde{\mathbf{C}} = \frac{1}{N+1}\sum_{k=0}^{N}\mathbf{Q}_k.$$

5. Get the point $\mathbf{C}$ by projecting $\widetilde{\mathbf{C}}$ to the approximation plane. Thus we get the central point $\mathbf{C}$ and Normal $\mathbf{N}_f$ in three-dimensional space.
6. Compute the weight of the central point:

$$\omega_c = \frac{1}{N+1}\sum_{k=0}^{N}\omega c_k,$$

where ωc_k is the weight of the control point $\mathbf{C}_k$. We set the weight of the Normal $\mathbf{N}_f$ to be zero. Then we get the homogeneous central point and Normal as follows:

$$\mathbf{C}^{\omega} = \begin{pmatrix} \omega_c\mathbf{C} \\ \omega_c \end{pmatrix} \quad \text{and} \quad \mathbf{N}^{\omega} = \begin{pmatrix} \mathbf{N}_f \\ 0 \end{pmatrix}.$$

5. Inner curves

The shape of the inner curves has a definite effect on the shape of the n-sided hole. Reasonable inner curves can be determined by interpolating the central point and Normal along with the middle boundary points $\mathbf{C}_k^{\omega}$ and the middle cross-boundary derivatives $\mathbf{D}_k^{\omega}$. To construct the inner curves, the control points of the inner curves are first determined followed by calculation of the weights. To represent the inner curves, cubic rational Bézier curves

$$\mathbf{CI}_k(t) = \frac{\sum_{i=0}^{3} B_{i,3}(t)\omega_{k_i}\mathbf{P}_{k_i}}{\sum_{i=0}^{3} B_{i,3}(t)\omega_{k_i}}, \quad k = 0, 1, \ldots, N,$$

are used, defined by their end points, end tangents and control point weights. The inner curves are computed as follows.

1. The start and end control points are

$$\mathbf{P}_{k_0} = \mathbf{C}_k \quad \text{and} \quad \mathbf{P}_{k_3} = \mathbf{C}, \text{ respectively.}$$

The weights of the start and end control points are

$$\omega_{k_0} = \omega c_k \quad \text{and} \quad \omega_{k_3} = \omega_c, \text{ respectively.}$$

2. The tangents at the end points are computed as

$$\mathbf{T}_s = \mathbf{V}_k(1) \quad \text{and} \quad \mathbf{T}_e = \mathbf{C} - \mathbf{C}_k^{\pi},$$

where $\mathbf{C}_k^{\pi}$ is the projection of $\mathbf{C}_k$ onto the approximation plane.
3. Make the end tangents have unit length (denote them as $\mathbf{T}_s^u$ and $\mathbf{T}_e^u$).
4. The inner control points are computed as

$$\mathbf{P}_{k_1} = \mathbf{P}_{k_0} + \frac{\omega_{k_0}\alpha\mathbf{T}_s^u}{3\omega_{k_1}} \quad \text{and} \quad \mathbf{P}_{k_2} = \mathbf{P}_{k_3} - \frac{\omega_{k_3}\alpha\mathbf{T}_e^u}{3\omega_{k_2}}, \tag{8}$$

where

$$\alpha = |\mathbf{CI}_k'(0)| = |\mathbf{CI}_k'(0.5)| = |\mathbf{CI}_k'(1)|, \tag{9}$$

that is, the speed is equal at the start point, midpoint and end point of the curve. Deduced from Eqs. (8) and (9) (the details are omitted), α is computed as the solution of the following equation:

$$a\alpha^2 + b\alpha + c = 0, \tag{10}$$

where

$$\begin{cases} a = \dfrac{\omega_{k_1}^2\omega_{k_2}^2\omega_4^2 - 16|\omega_{k_0}\omega_{k_2}\omega_1\mathbf{T}_s^u - \omega_{k_1}\omega_{k_3}\omega_2\mathbf{T}_e^u|^2}{\omega_{k_1}^2\omega_{k_2}^2\omega_4^2} \\ b = \dfrac{96((\omega_0+\omega_1)\mathbf{P}_{k_0} + (\omega_2-\omega_3)\mathbf{P}_{k_3})\cdot(\omega_{k_3}\omega_2\mathbf{T}_e^u - \omega_{k_0}\omega_1\mathbf{T}_s^u)}{\omega_{k_1}\omega_{k_2}\omega_4^2} \\ c = \dfrac{-144|(\omega_0+\omega_1)\mathbf{P}_{k_0} + (\omega_2-\omega_3)\mathbf{P}_{k_3}|^2}{\omega_4^2}. \end{cases}$$

Because weight of the second control point is represented as a linear function of α:

$$\omega_{k_1} = \omega_{k_0} + \alpha\widetilde{\omega}_k, \tag{11}$$

it cannot be determined separately. Directly substituting Eq. (11) into Eq. (10) will require to solve a sextic degree equation. An iterative method is preferred. The procedure is shown as follows:

1. Set $\omega_{k_1} = \omega_{k_0}$ and $\omega_{k_2} = \frac{\omega_{k_0}+2\omega_{k_3}}{3}$.
2. Get the positive α by solving Eq. (10) using the current ω_{k_1}. Eq. (10) has two real solutions for α, one positive and one negative. The positive value represents the magnitude of the derivatives at the ends.
3. Compute $\widetilde{\omega}_{k_1} = \omega_{k_0} + \alpha\widetilde{\omega}_k$. If $|(\widetilde{\omega}_{k_1} - \omega_{k_1})| > TOL$, set $\omega_{k_1} = \widetilde{\omega}_{k_1}$ and goto Step 2; otherwise, set $\omega_{k_1} = \widetilde{\omega}_{k_1}$.

When the above procedure exits from Step 3, we get the length of the end tangents α. Control points along with their weights are computed by the way during the procedure of determining α. Fig. 6 shows the inner curves of a hexagonal hole.

6. Inner cross-boundary derivatives

To ensure smooth joining of surface patches along the inner curves $\mathbf{CI}_k^{\omega}$, cross-boundary derivatives across these curves must be computed in four-dimensional space, one to the left, $\mathbf{BL}_k^{\omega}$, and one to the right, $\mathbf{BR}_k^{\omega}$ (see Fig. 7). To achieve surface tangent continuity, these derivatives, the derivative of the homogeneous inner curve and the homogeneous inner curve

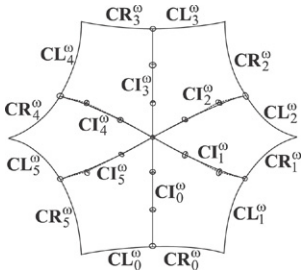


Fig. 6. Inner curves of a hexagonal hole.

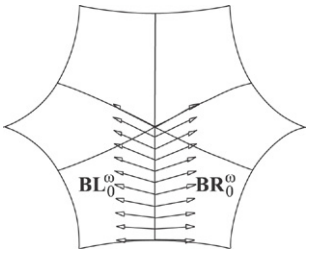


Fig. 7. Inner cross-boundary derivatives.

must define the same four-dimensional hyperplane at any point along $\mathbf{CI}_k^{\omega}$. For a fixed parameter t_f, determinant of the matrix

$$|\mathbf{BL}_k^{\omega}(t_f) \quad \mathbf{BR}_k^{\omega}(t_f) \quad \mathbf{CI}_k^{\omega'}(t_f) \quad \mathbf{CI}_k^{\omega}(t_f)| \tag{12}$$

must be zero to guarantee the smooth connection between adjacent patches where $\mathbf{CI}_k^{\omega'}(t)$ denotes the derivative curve of $\mathbf{CI}_k^{\omega}(t)$ [10]. We now outline how to compute $\mathbf{BR}_k^{\omega}$.

1. In three-dimensional space, two vectors can determine the normal vector direction while two vectors define a two-dimensional normal vector space in four-dimensional space. Here we use "$\times_4$" to denote a four-dimensional cross operator while "$\times_3$" is utilized to represent a three-dimensional cross operator. Compute an orthogonal base of the normal vector space defined by $\mathbf{CL}_k^{\omega'}(1)$ and $\mathbf{CI}_k^{\omega'}(0)$:

$$\begin{cases} \mathbf{N}_k^z = \begin{pmatrix} \mathbf{N}_k \\ 0 \end{pmatrix} \\ \mathbf{F}_k = \times_4(\mathbf{N}_k^z, \mathbf{CL}_k^{\omega'}(1), \mathbf{CI}_k^{\omega'}(0)), \end{cases}$$

where

$$\mathbf{N}_k = \mathbf{CL}_k'(1) \times_3 \mathbf{CI}_k'(0).$$

$\mathbf{N}_k^z$, $\mathbf{CL}_k^{\omega'}(1)$ and $\mathbf{CI}_k^{\omega'}(0)$ determine a four-dimensional hyperplane while $\mathbf{F}_k$ is a Normal vector of the hyperplane. Also compute

$$\mathbf{G}_k = \times_4(\mathbf{N}^{\omega}, \mathbf{CI}_k^{\omega'}(1), \mathbf{CI}_{k-1}^{\omega'}(1)),$$

and let

$$\mathbf{D}_s = \mathbf{CI}_k^{\omega'}(0) \quad \text{and} \quad \mathbf{D}_e = \mathbf{CI}_k^{\omega'}(1).$$

2. Compute

$$\mathbf{T}_s = \times_4(\mathbf{F}_k, \mathbf{N}_k^z, \mathbf{D}_s) \quad \text{and} \quad \mathbf{T}_e = \times_4(\mathbf{G}_k, \mathbf{N}^{\omega}, \mathbf{D}_e).$$

3. Get

$$\mathbf{B}_s = \mathbf{CL}_k^{\omega'}(1) \quad \text{and} \quad \mathbf{B}_e = \mathbf{CI}_{k-1}^{\omega'}(1),$$

where $\mathbf{B}_s$ and $\mathbf{B}_e$ are the start and end points of the derivative curve $\mathbf{BR}_k^\omega$, respectively.

4. Get p_s, q_s, p_e, q_e to satisfy

$$\mathbf{B}_s = p_s\mathbf{D}_s + (q_s l_s)\left(\frac{\mathbf{T}_s}{l_s}\right) \quad \text{and}$$

$$\mathbf{B}_e = p_e\mathbf{D}_e + (q_e l_e)\left(\frac{\mathbf{T}_e}{l_e}\right),$$

and form the functions

$$p(t) = (1-t)p_s + tp_e \quad \text{and} \quad q(t) = (1-t)q_s l_s + tq_e l_e,$$

where l_s and l_e are scaling factors that are introduced to avoid negative weights in the resultant patches.

5. Compute a tangent field along $\mathbf{CI}_k^w$

$$\mathbf{T}(t) = (1-t)\frac{\mathbf{T}_s}{l_s} + t\frac{\mathbf{T}_e}{l_e}.$$

6. With symbolic operators, compute the cross-boundary derivative as

$$\mathbf{BR}_k^\omega(t) = p(t)\mathbf{CI}_k^{\omega'}(t) + q(t)\mathbf{T}(t),$$

where $\mathbf{BR}_k^\omega(t)$ is a cubic Bézier curve.

7. Scale l_s and l_e to avoid negative weights in the resultant patches. If the fourth elements of $\mathbf{CI}_{k_0}^\omega$ and $\mathbf{CI}_{k_3}^\omega$ (weights for the first and third control points of the ith inner curve) are

   (a) different, compute l_s and l_e from the following equation system

$$\begin{cases} f_s E_4(\mathbf{BR}_{k_0}^\omega) + f_e E_4(\mathbf{BR}_{k_3}^\omega) = E_4(\mathbf{BR}_{k_1}^\omega) \\ \dfrac{E_4(\mathbf{B}_s)}{3} + \dfrac{2E_4(\mathbf{B}_e)}{3} = E_4(\mathbf{BR}_{k_2}^\omega), \end{cases}$$

   where

$$f_s = \frac{E_4(\mathbf{CI}_{k_3}^\omega - \mathbf{CI}_{k_1}^\omega)}{E_4(\mathbf{CI}_{k_3}^\omega - \mathbf{CI}_{k_0}^\omega)} \quad \text{and} \quad f_e = \frac{E_4(\mathbf{CI}_{k_1}^\omega - \mathbf{CI}_{k_0}^\omega)}{E_4(\mathbf{CI}_{k_3}^\omega - \mathbf{CI}_{k_0}^\omega)},$$

   and $\mathbf{BR}_{k_i}^\omega (i = 0, 1, 2, 3)$ is the ith control point of the derivative curve $\mathbf{BR}_k^\omega(t)$.

   (b) Equal, compute l_s and l_e from the following equation system:

$$\begin{cases} E_4(\mathbf{BR}_{k_1}^\omega) = f_s E_4(\mathbf{BR}_{k_0}^\omega) \\ \dfrac{E_4(\mathbf{B}_s)}{3} + \dfrac{2E_4(\mathbf{B}_e)}{3} = E_4(\mathbf{BR}_{k_2}^\omega), \end{cases}$$

   where

$$f_s = \frac{E_4(\mathbf{CI}_{k_1}^\omega)}{E_4(\mathbf{CI}_{k_0}^\omega)}.$$

Here we can verify that the fourth elements of control points of the curve:

$$\mathbf{SV}_k^\omega(t) = \sum_{i=0}^{3} B_{i,3}(t)\mathbf{P}_{k_i}^\omega = \mathbf{CI}_{k_i}^\omega(t) - \frac{\mathbf{BR}_{k_i}^\omega(t)}{3},$$

are all larger than zero (using Condition (2)). The geometric meaning of the assertion is that if we interpolate the boundaries and the cross-boundary derivatives of a patch (see Fig. 8) in one direction (for example, in the u direction) using a cubic Bézier surface, weights of the control points adjacent to the inner curve are all positive. This assertion helps to show that no negative weights are involved in the final Coons patches in Section 8. The computation of $\mathbf{BL}_k^\omega$ is the same except the computation of $\mathbf{B}_s$ and $\mathbf{B}_e$ is different:

$$\mathbf{B}_s = \mathbf{CR}_k^{\omega'}(1) \quad \text{and} \quad \mathbf{B}_e = \mathbf{CI}_{k+1}^{\omega'}(1).$$

Since the vectors $\mathbf{BL}_k^\omega(t)$ and $\mathbf{BR}_k^\omega(t)$ are in the hyperplane spanned by the vectors $\mathbf{CI}_k^{\omega'}(t), \mathbf{CI}_k^\omega(t)$ and $\mathbf{T}(t)$, the determinant of Matrix (12) is always equal to zero at any point of the homogeneous curve $\mathbf{CI}_k^\omega(t)$. Thus the adjacent patches are smoothly joined along the common inner curve.

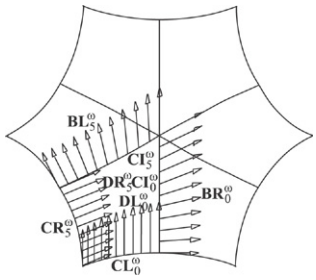


Fig. 8. Boundaries and cross-boundary derivatives of one patch.

7. Adjusting homogeneous derivatives

The surface patches that our method will place in each quadrilateral region are bicubic Coons patches in NURBS form. A Coons patch requires four boundary curves and four cross-boundary derivatives. A famous flaw of the Coons formulation is that the method works only if the cross-boundary derivatives are twist compatible as well as boundary compatible. To satisfy twist compatibility, at each corner, the derivatives of the corresponding two homogeneous cross-boundary derivatives must be the same. To satisfy boundary compatibility, at each corner, the derivative of the homogeneous boundary and the adjacent homogeneous cross-boundary derivative must be the same.

7.1. Boundary compatibility

To place homogeneous Coons patches in each quadrilateral region, the homogeneous cross-derivatives must be boundary compatible. Referring to Fig. 8, to satisfy boundary compatibility across the boundary $\mathbf{CL}_k^\omega$, the following conditions must hold:

$$\mathbf{DL}_k^\omega(0) = \mathbf{CR}_{k-1}^{\omega'}(0) \quad \text{and} \quad \mathbf{DL}_k^\omega(1) = \mathbf{CI}_k^{\omega'}(0).$$

The direction of given $\mathbf{DL}_k^\omega(0)$ should be defined to be the same as $\mathbf{CR}_{k-1}^{\omega'}(0)$. Also, from Section 5, we know that the directions of $\mathbf{DL}_k^\omega(1)$ and $\mathbf{CI}_k^{\omega'}(0)$ are the same. Now to satisfy boundary compatibility, the magnitudes of derivatives of boundaries must agree with those of the cross-boundary derivatives along the adjacent boundaries as follows:

$$|\mathbf{DL}_k^\omega(0)| = |\mathbf{CR}_{k-1}^{\omega'}(0)| \quad \text{and} \quad |\mathbf{DL}_k^\omega(1)| = |\mathbf{CI}_k^{\omega'}(0)|.$$

To bring these magnitudes into compliance, the following procedure is applied:

1. Calculate

$$f_0 = \frac{|\mathbf{CR}_{k-1}^{\omega'}(0)|}{|\mathbf{DL}_k^{\omega}(0)|} \quad \text{and} \quad f_1 = \frac{|\mathbf{CI}_k^{\omega'}(0)|}{|\mathbf{DL}_k^{\omega}(1)|}.$$

2. With symbolic operators, compute a new cross-derivative

$$\widetilde{\mathbf{DL}}_k^{\omega}(t) = f(t)\mathbf{DL}_k^{\omega}(t) \quad \text{and} \quad f(t) = (1-t)f_0 + tf_1.$$

This process will adjust the magnitudes while maintaining directions of the homogeneous cross-boundary derivatives along the boundary $\mathbf{CL}_k^{\omega}$. We should be on guard that the above adjustment may introduce negative weights when

$$E_4(\mathbf{C}_{k_i}^{\omega}) - \frac{E_4(\mathbf{D}_{k_i}^{\omega})}{3} < 0,$$

for some i. The negative weights can be avoided by scaling the control points $\mathbf{D}_{k_i}^{\omega}$ satisfying the above condition. Degree of the cross-boundary derivatives is raised by one, due to the multiplication by a linear function. A similar procedure is applied to the other three boundaries.

7.2. Twist compatibility

To place homogeneous Coons patches in each quadrilateral region, the homogeneous cross-derivatives must be twist compatible. Referring to Fig. 8, twist compatibility means:

$$\begin{cases} \mathbf{DR}_{k-1}^{\omega'}(0) = \mathbf{DL}_k^{\omega'}(0) \\ \mathbf{DL}_k^{\omega'}(1) = \mathbf{BR}_k^{\omega'}(0) \\ \mathbf{BR}_k^{\omega'}(1) = \mathbf{BL}_{k-1}^{\omega'}(1) \\ \mathbf{BL}_{k-1}^{\omega'}(0) = \mathbf{DR}_{k-1}^{\omega'}(1). \end{cases}$$

The derivatives of the homogeneous cross-boundary derivative curves meeting at the four corners must be the same. In practice, this is almost not the case. Degree elevation is not included in options for high degree surfaces may make the successive operations (such as intersecting and offsetting) difficult to carry out. Using approximation with a manageable tolerance is preferable. If any one of the boundary curves and cross-boundary derivatives is rational, the twist vectors and derivative fields must be embedded in homogeneous space in such a way as to obtain four-dimensional compatibility while maintaining the angular tolerance on the three-dimensional vector directions. The method is shown as follows:

1. Average the homogeneous twists at each corner.
2. Modify the fourth element of the averaged homogeneous twist.
3. Reapproximate each homogeneous derivative curve with the new end derivatives, i.e. the average of twists, while maintaining the angular tolerance on the three-dimensional vector directions.

Assume a cross-boundary derivative curve

$$\mathbf{D}(t) = \sum_{i=0}^{n} N_{i,p}(t)\mathbf{V}_i,$$

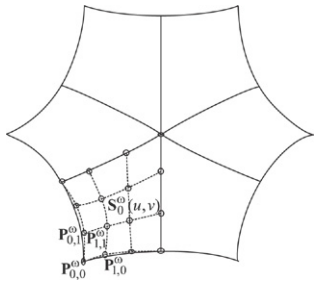


Fig. 9. Twist compatibility.

as well as its homogeneous form

$$\mathbf{D}^{\omega}(t) = \sum_{i=0}^{n} N_{i,p}(t)\mathbf{V}_i^{\omega},$$

defined over the knot vector

$$\mathbf{U} = \{\underbrace{0,\ldots,0}_{p+1}, u_{p+1},\ldots,u_n, \underbrace{1,\ldots,1}_{p+1}\}$$

(let $n+p+1 = m$). As the Coons patches must be constructed in homogeneous space, twist compatibility should be considered in homogeneous space while the normal deviation between original curve and approximating curve must be measured in three-dimensional space. The derivatives at the ends are

$$\mathbf{D}^{\omega'}(0) = \frac{p}{u_{p+1} - u_1}(\mathbf{V}_1^{\omega} - \mathbf{V}_0^{\omega}), \tag{13}$$

and

$$\mathbf{D}^{\omega'}(1) = \frac{p}{u_{m-1} - u_n}(\mathbf{V}_n^{\omega} - \mathbf{V}_{n-1}^{\omega}).$$

The derivatives depend on the positions of the knots u_{p+1} and u_n and on the homogeneous control points $\mathbf{V}_0^{\omega}$, $\mathbf{V}_1^{\omega}$, $\mathbf{V}_{n-1}^{\omega}$ and $\mathbf{V}_n^{\omega}$. Since the positions of the knots, and the end control points $\mathbf{V}_0^{\omega}$ and $\mathbf{V}_n^{\omega}$ must be fixed. The only way to alter the derivative is

1. to modify $\mathbf{V}_1^{\omega}$ and $\mathbf{V}_{n-1}^{\omega}$.
2. To insert new knots to restrict the deviation between the derivatives no more than the user-specified tolerance ε.

Let $\mathbf{T}_s^{\omega}$ and $\mathbf{T}_e^{\omega}$ be the averaged twists at the two ends of the derivative curve $\mathbf{DL}_k^{\omega}$. Before the Coons patches are constructed, we should be certain that no negative weights will appear in the resultant patches [12]. To avoid introducing negative weights, some restrictions must be imposed on the twist vectors. Considering the lower left corner of $\mathbf{S}_0(u, v)$ (see Fig. 9), the fourth coordinate c_s of the lower left twist vector $\mathbf{T}_s^{\omega}$ is restricted by the following equations:

$$\begin{cases} \omega_{1,1} > 0 \\ \omega_{1,1} = \omega_{1,0} + \dfrac{\omega_1 v_{q+1}}{q} \\ \omega_1 = \omega_0 + \dfrac{c_s u_{p+1}}{p}. \end{cases} \tag{14}$$

That is, weight of the control point $\mathbf{P}_{1,1}$ is positive. From Eq. (14), we obtain:

$$c_s > -\frac{\omega_{10} pq + \omega_0 p v_{q+1}}{u_{p+1} v_{q+1}}. \tag{15}$$

$$\triangle u_e = \frac{p\omega_n\omega_{n-1,0}\sin\varepsilon|\mathbf{D}(1)|}{\omega_n|(\mathbf{A}'(1)-\mathbf{B}_e)+(c_e-\omega'(1))\mathbf{P}_{n-1,0}|+\omega_{n-1}\omega_{n-1,0}\sin\varepsilon|\mathbf{D}'(1)|},$$

Box I.

If precomputed $\mathbf{T}_s^\omega$ does not satisfy Condition (15), $\mathbf{T}_s^\omega$ is modified as follows. We set

$$\omega_{1,1} = \frac{\omega_{01}+\omega_{10}}{2}. \tag{16}$$

Substituting Eq. (16) into Eq. (14), we have

$$c_s = \frac{pq(\omega_{01}-\omega_{10})-2p\omega_0 v_{q+1}}{2u_{p+1}v_{q+1}} \quad \text{and} \quad \mathbf{T}_s^\omega = \frac{\mathbf{T}_s^\omega c_s}{|\mathbf{T}_s^\omega|}.$$

After the new twists are determined, the new homogeneous control points are computed as:

$$\widetilde{\mathbf{V}}_1^\omega = \mathbf{V}_0^\omega + \frac{\triangle u_s}{p}\mathbf{T}_s^\omega, \quad \triangle u_s = u_{p+1}-u_1,$$
$$\widetilde{\mathbf{V}}_{n-1}^\omega = \mathbf{V}_n^\omega - \frac{\triangle u_e}{p}\mathbf{T}_e^\omega, \quad \triangle u_e = u_{m-1}-u_n,$$
$$\widetilde{\mathbf{V}}_i^\omega = \mathbf{V}_i^\omega, \quad i = 0,2,3,\ldots,n-2,n.$$

Given the homogeneous approximation curve

$$\widetilde{\mathbf{D}}^\omega(t) = \sum_{i=0}^{n} N_{i,p}(t)\widetilde{\mathbf{V}}_i^\omega,$$

the corresponding approximation curve

$$\widetilde{\mathbf{D}}(t) = \frac{\sum_{i=0}^{n} N_{i,p}(t)\widetilde{\mathbf{V}}_i\widetilde{\omega}_i}{\sum_{i=0}^{n} N_{i,p}(t)\widetilde{\omega}_i},$$

deviates from the original one, that is, the angle between them is

$$\alpha(t) = \angle(\mathbf{D}(t),\widetilde{\mathbf{D}}(t)) \le \max(\angle(\mathbf{V}_i,\widetilde{\mathbf{V}}_i)) = \max(\alpha_s,\alpha_e),$$

where $\alpha_s = \angle(\mathbf{V}_1,\widetilde{\mathbf{V}}_1)$ and $\alpha_e = \angle(\mathbf{V}_{n-1},\widetilde{\mathbf{V}}_{n-1})$. If $\alpha(t) < \varepsilon$, the angular deviation between the curves should be within the user-specified tolerance. Otherwise, knots can be inserted to make α_s and/or α_e as small as required. Here we must consider the deviation in three-dimensional space while the Coons patches are constructed in homogeneous space. Let

$$\mathbf{A}(t) = \mathbf{D}(t)\omega(t),$$

where $\mathbf{A}(t)$ is a vector-valued function whose coordinates are the first three coordinates of $\mathbf{D}^\omega(t)$. The second control point of curve $\mathbf{D}(t)$ as well as the second control point of homogeneous curve $\mathbf{D}^\omega(t)$ are

$$\mathbf{V}_1 = \frac{q}{\triangle v_s}\frac{\omega_{1,1}}{\omega_{1,0}}(\mathbf{P}_{1,1}-\mathbf{P}_{1,0}), \tag{17}$$

and

$$\mathbf{V}_1^\omega = \frac{q}{\triangle v_s}\begin{pmatrix}\omega_{1,1}\mathbf{P}_{1,1}-\omega_{1,0}\mathbf{P}_{1,0}\\ \omega_{1,1}-\omega_{1,0}\end{pmatrix}, \tag{18}$$

where $\mathbf{P}_{i,j}$ and $\omega_{i,j}$ denote the control points and the weights of constructed Coons patches, (see Fig. 9). From Eqs. (17) and (18), we get

$$\mathbf{V}_1 = \frac{\mathbf{A}_1\triangle v_s + q(\omega_{1,0}-\omega_{1,1})\mathbf{P}_{1,0}}{\triangle v_s\omega_{1,0}}, \tag{19}$$

where

$$\omega_{1,1} = \omega_{1,0} + \frac{\triangle v_s}{q}\omega_{0,0} + \frac{\triangle u_s\ \triangle\ v_s}{pq}\omega'(0),$$

and $\mathbf{A}_1$ denotes the second control point of the curve $\mathbf{A}(t)$. Substituting Eq. (13) into Eq. (19), we have:

$$\mathbf{V}_1 = \frac{\mathbf{A}_0 p\triangle v_s + \mathbf{A}'(0)\triangle u_s\triangle v_s - (p\triangle v_s\omega_{0,0}+\triangle u_s\triangle v_s\omega'(0))\mathbf{P}_{1,0}}{p\triangle v_s\omega_{1,0}}.$$

The second control points of $\widetilde{\mathbf{D}}(t)$ can be computed similarly as follows:

$$\widetilde{\mathbf{V}}_1 = \frac{\mathbf{A}_0 p\triangle v_s + \mathbf{B}_s\triangle u_s\triangle v_s - (p\triangle v_s\omega_{0,0}+\triangle u_s\triangle v_s c_s)\mathbf{P}_{1,0}}{p\triangle v_s\omega_{1,0}},$$

where $\mathbf{B}_s$ denotes the vector whose coordinates are the first three coordinates of $\mathbf{T}_s^\omega$ and c_s denotes the fourth coordinate of $\mathbf{T}_s^\omega$. α_s can be controlled as follows:

$$\begin{aligned}\sin(\mathbf{V}_1,\widetilde{\mathbf{V}}_1) &\le \frac{|\mathbf{V}_1-\widetilde{\mathbf{V}}_1|}{|\mathbf{V}_1|} = \frac{\triangle u_s|(\mathbf{A}'(0)-\mathbf{B}_s)+(c_s-\omega'(0))\mathbf{P}_{1,0}|}{p\omega_{1,0}|\mathbf{D}(0)+\frac{\triangle u_s\omega_1}{p\omega_0}\mathbf{D}'(0)|}\\ &\le \frac{\triangle u_s|(\mathbf{A}'(0)-\mathbf{B}_s)+(c_s-\omega'(0))\mathbf{P}_{1,0}|}{\omega_{1,0}(p|\mathbf{D}(0)|-\frac{\triangle u_s\omega_1}{\omega_0}|\mathbf{D}'(0)|)},\end{aligned}$$

where ω_1 and ω_2 are the weights of the first two control points of curve $\mathbf{D}(t)$. Then

$$\sin\varepsilon = \frac{\triangle u_s|(\mathbf{A}'(0)-\mathbf{B}_s)+(c_s-\omega'(0))\mathbf{P}_{1,0}|}{\omega_{1,0}(p|\mathbf{D}(0)|-\frac{\triangle u_s\omega_1}{\omega_0}|\mathbf{D}'(0)|)}.$$

From the above one, we get:

$$\triangle u_s = \frac{p\omega_0\omega_{1,0}\sin\varepsilon|\mathbf{D}(0)|}{\omega_0|(\mathbf{A}'(0)-\mathbf{B}_s)+(c_s-\omega'(0))\mathbf{P}_{1,0}|+\omega_1\omega_{1,0}\sin\varepsilon|\mathbf{D}'(0)|}.$$

The new knot to be inserted is

$$\widetilde{u}_s = u_1 + \triangle u_s.$$

Similar deduction yields (see the equation given in Box I) where $\mathbf{B}_e$ denotes the vector whose coordinates are the first three coordinates of $\mathbf{T}_e^\omega$ and c_e denotes the fourth coordinate of $\mathbf{T}_e^\omega$. The new knot to be inserted is

$$\widetilde{u}_e = u_{m-1} - \triangle u_e.$$

Based on these deductions, we can approximate the cross-boundary derivatives up to the user-specified angular tolerance ε as follows.

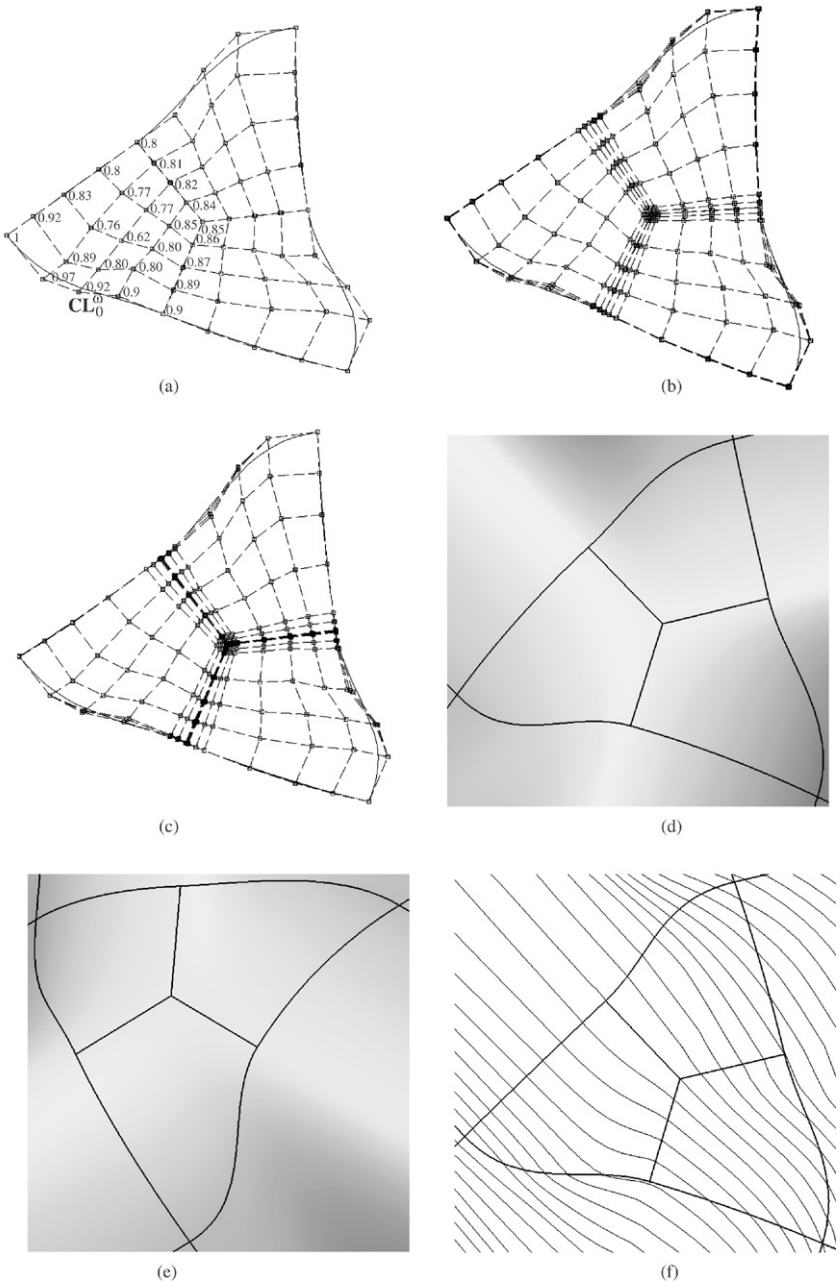


Fig. 10. Control points of the trigonal hole for (a) $\varepsilon = 10°$, (b) $\varepsilon = 1°$, (c) $\varepsilon = 0.1°$; (d) the trigonal patch and its adjacent surfaces when $\varepsilon = 1°$; (e) another view of surfaces in (d); (f) highlight lines of NURBS surfaces in (d).

1. Compute α_s and α_e.
2. If $\alpha_s < \varepsilon$ and $\alpha_e < \varepsilon$, goto Step 6.
3. If $\alpha_s \geq \varepsilon$, compute $\widetilde{u}_s$.
4. If $\alpha_e \geq \varepsilon$, compute $\widetilde{u}_e$.
5. Insert the new knot(s).
6. Compute $\widetilde{\mathbf{V}}_1$ and $\widetilde{\mathbf{V}}_{n-1}$.
7. Replace $\mathbf{V}_1$ with $\widetilde{\mathbf{V}}_1$, and $\mathbf{V}_{n-1}$ with $\widetilde{\mathbf{V}}_{n-1}$.

If there is compatibility at both ends, no new knots need be computed. Once all four cross-boundary derivatives are approximated to assume given twists, we are ready to place Coons patches in each region.

8. Coons patches generation

Given all inner boundary curves and approximations of cross-boundary derivatives, bicubically blended Coons patches are placed in each quadrilateral subregion. A simple four-dimensional extension of Coons construction may result in an unexpected shape or introduce negative weights [9]. To overcome these two problems, Lin's method [9] to construct homogeneous Coons patches is introduced. Referring to Fig. 8, the process is outlined as follows.

1. Compute

$$\mathbf{S}_u^{\omega}(u, v) = \mathit{CUBIC\_BLEND}\{\mathbf{CR}_{k-1}^{\omega}, \widetilde{\mathbf{DR}}_{k-1}^{\omega}, \mathbf{CI}_k^{\omega}, \widetilde{\mathbf{BR}}_k^{\omega}\},$$

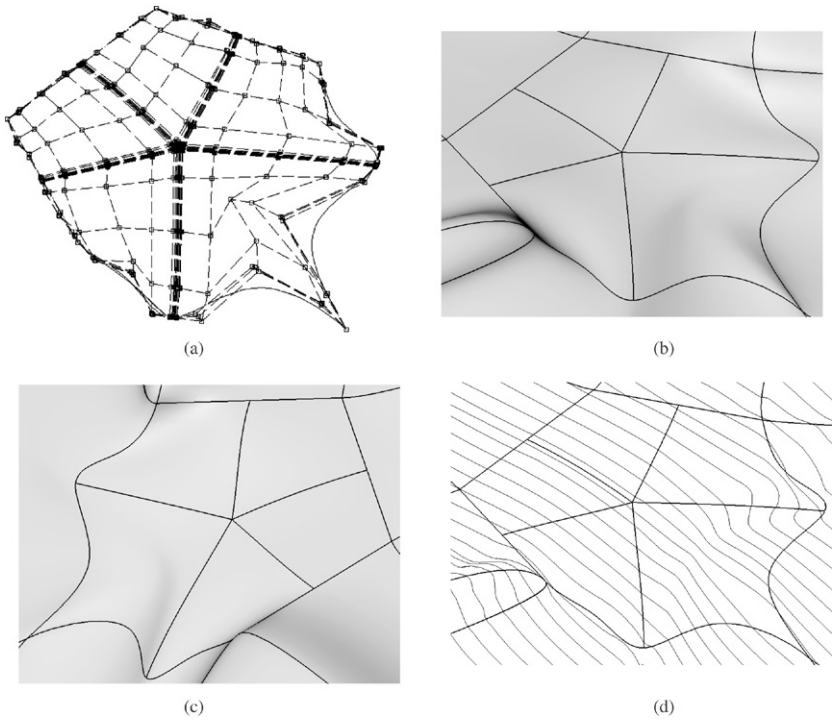


Fig. 11. (a) Control points of the pentagonal hole for $\varepsilon = 0.1°$; (b) the pentagonal hole and its adjacent surfaces when $\varepsilon = 0.1°$; (c) another view of surfaces in (b); (d) highlight lines of NURBS surfaces in (b).

where *CUBIC_BLEND* is an operator that creates a cubically blended surface that interpolates the boundary curves $\mathbf{CR}^{\omega}_{k-1}$ and $\mathbf{CI}^{\omega}_{k}$, as well as the derivatives $\widetilde{\mathbf{DR}}^{\omega}_{k-1}$ and $\widetilde{\mathbf{BR}}^{\omega}_{k}$.

2. Compute

$$\mathbf{S}^{\omega}_{v}(u,v) = \mathit{CUBIC\_BLEND}\{\mathbf{CL}^{\omega}_{k}, \widetilde{\mathbf{DL}}^{\omega}_{k}, \mathbf{CI}^{\omega}_{k-1}, \widetilde{\mathbf{BL}}^{\omega}_{k-1}\}.$$

This operator creates a cubic blended surface in the other direction.

3. Compute

$$\mathbf{S}^{\omega}_{t}(u,v) = \mathit{TENSOR\_PRODUCT}\{\mathbf{CR}^{\omega}_{k-1}, \mathbf{CL}^{\omega}_{k}, \mathbf{CI}^{\omega}_{k}, \mathbf{CI}^{\omega}_{k-1}, \mathbf{T}^{\omega}_{00}, \mathbf{T}^{\omega}_{10}, \mathbf{T}^{\omega}_{11}, \mathbf{T}^{\omega}_{01}\},$$

where the operator *TENSOR_PRODUCT* creates a bicubic tensor-product surface to the four boundaries $\mathbf{CR}^{\omega}_{k-1}$, $\mathbf{CL}^{\omega}_{k}$, $\mathbf{CI}^{\omega}_{k}$ and $\mathbf{CI}^{\omega}_{k-1}$ and to the four twists $\mathbf{T}^{\omega}_{00}$, $\mathbf{T}^{\omega}_{10}$, $\mathbf{T}^{\omega}_{11}$ and $\mathbf{T}^{\omega}_{01}$.

4. Make $\mathbf{S}^{\omega}_{u}(u,v)$, $\mathbf{S}^{\omega}_{v}(u,v)$ and $\mathbf{S}^{\omega}_{t}(u,v)$ compatible.
5. Compute the control points of $\mathbf{S}^{\omega}(u,v)$ as follows. Use

$$\begin{bmatrix}\omega\mathbf{P}\\ \omega\end{bmatrix} = \begin{bmatrix}\omega_u\mathbf{P}_u + \omega_v\mathbf{P}_v - \omega_t\mathbf{P}_t\\ \omega_u + \omega_v - \omega_t\end{bmatrix}$$

to evaluate the boundary control points and their adjacent control points and use:

$$\begin{bmatrix}\omega\mathbf{P}\\ \omega\end{bmatrix} = \begin{bmatrix}\omega_u\omega_v(\mathbf{P}_u + \mathbf{P}_v - \mathbf{P}_t)\\ \omega_u\omega_v\end{bmatrix}$$

to evaluate all the other internal control points.

From the construction and adjustment of derivatives in Sections 6–8, we can demonstrate that no negative weights are involved in the surfaces $\mathbf{S}^{\omega}(u,v)$. Figs. 10–12 show several examples. Fig. 10(a)–(c) shows the control points of a trigonal hole for $\varepsilon = \{10°, 1°, 0.1°\}$. The weights of the control points of the left patch of Fig. 10(a) are labelled beside the control points. The weights of the control points of $\mathbf{DL}^{\omega}_{0}(t)$ are $\{-0.4, -0.41, -0.38, -0.20, -0.05\}$. The boundaries as well as the derivatives of Fig. 10 are rational curves. Two views of the trigonal hole and its adjacent surfaces are given in Fig. 10(d)–(e). To show how well the filling result is, highlight lines of the resultant NURBS surfaces are given in Fig. 10(f). Fig. 11 shows an example of a pentagonal hole for $\varepsilon = 0.1°$ and an example of a hexagonal hole for $\varepsilon = 1°$ is shown in Fig. 12. From the smoothness of the highlight lines, we can see that our method can achieve reasonable surface tangent continuity. To compare the computational cost of B-spline based method and NURBS based method, we convert the boundaries and derivatives of the holes to B-spline curves by setting the weights of the control points of the boundaries and cross-boundary derivatives to be 1. Piegl's method is then implemented to fill these B-spline holes. Time comparison between Piegl's method and our algorithm is given in Table 1.

Table 1
Time (ms) comparison between our algorithm and Piegl's method for filling n-sided holes

n-side hole	Fig. 10(a)	Fig. 10(b)	Fig. 11(a)	Fig. 12(a)
Piegl's method	13	15	14	15
Our algorithm	15	17	16	16

From Table 1, we can see that we extend Piegl's method to handle the rational case with little extra computational cost. For the example shown in Fig. 12, the corresponding B-spline hole (setting all the weights of the control points of the boundaries

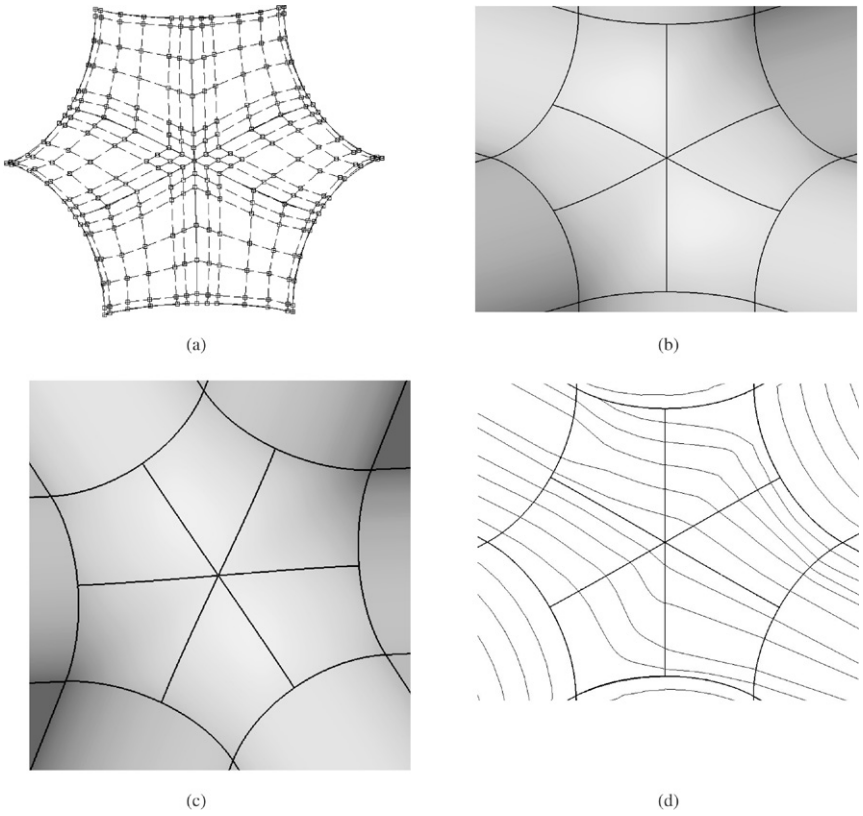


Fig. 12. (a) Control points of hexagonal hole for $\varepsilon = 1°$; (b) the hexagonal hole and its adjacent surfaces when $\varepsilon = 1°$; (c) another view of surfaces in (b); (d) highlight lines of NURBS surfaces in (b).

and cross-boundary derivatives to be 1) is shown in Fig. 13. From Fig. 13, we can see that Piegl's method cannot fill the rational hole shown in Fig. 12 (there are gaps between the boundaries and the filling patches). If we construct the B-spline boundary curves by interpolating a few points on the rational boundaries, the gaps may become smaller. But the gaps would not vanish. Thus our algorithm makes great sense for filling NURBS holes when the boundaries or/and cross-boundary derivatives are rational curves.

The examples shown in Figs. 10–12 are all implemented in the environment with Intel Pentium IV CPU 2.0 GHz, 1 G, Microsoft Windows XP, and Microsoft Visual C++ 6.0.

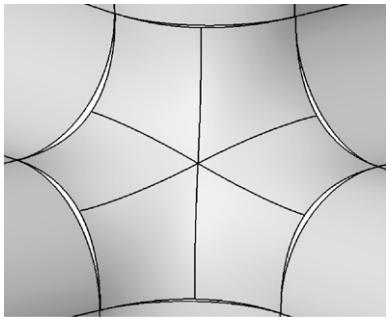


Fig. 13. Filling the 6-sided NURBS hole in Fig. 12 with B-spline patches.

9. Conclusions

To fill a hole with rational boundaries and/or rational cross-boundary derivatives, we have presented an algorithm to generate a collection of ε-G^1 continuous NURBS surfaces that fill the hole based on Piegl's method. Reasonable inner curves are computed through geometric interpolating. Homogeneous inner cross-boundary derivatives are generated to guarantee the smooth connection of adjacent surface patches along the inner curves. Twist compatibility is satisfied in four-dimensional space while the angular tolerance on vector directions is maintained in three-dimensional space. To avoid negative weights, Lin's method to construct four-dimensional Coons patches (in NURBS form) as well as boundary reparameterization are introduced. The method is general in that it can fill an arbitrary hole independently of the number of boundary curves, and it can take any cross-boundary derivatives.

The presented algorithm can be applied to fill a n-sided hole on a NURBS surface or to blend a corner surrounded by NURBS surfaces. The boundaries and cross-boundary derivatives of the hole can be obtained directly from the located surface or surrounded surfaces. The examples given demonstrate that the presented algorithm can simplify n-sided hole filling without visibly degrading the surface quality. For many applications such as NC-machining and terrain modelling, it is sufficient for surfaces to be ε-G^1 continuous. Engineering practice suggests that there is no ambiguity within acceptable working tolerance. The geometric discrepancies in the ε-G^1 surface models are very small and may be blended out in the subsequent manufacturing processes.

Acknowledgements

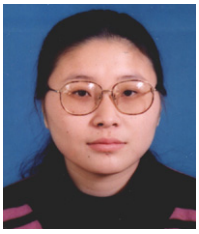
The authors appreciate the comments and suggestions of the anonymous reviewers. The research was supported by Chinese 973 Program(2004CB719400), and the National Science Foundation of China (60403047, 60533070). The second author was supported by the project sponsored by a Foundation for the Author of National Excellent Doctoral Dissertation of PR China (200342), and a Program for New Century Excellent Talents in University(NCET-04-0088).

References

[1] Braid IC. Non-local blending of boundary models. Computer-Aided Design 1997;29(2):89–100.
[2] Chiyokura H. Solid modelling with DESIGNBASE: Theory and implementation. New York: Addison-Wesley; 1988.
[3] DeRose T, Mann S. An approximately G^1 cubic surface interpolant. In: Lyche T, Schumaker L, editors. Mathematical methods in computer aided geometric design II. Academic Press; 1992. p. 185–96.
[4] Elsas PAV, Vergeest JSM. Fast approximate G^1 surface blending to support interactive sculptured feature design. In: Pratt MJ, Sriram RD, Wozny MJ, editors. Product modelling for computer-integrated design and manufacture. London: Chapman & Hall; 1997. p. 149–61.
[5] Hahn J. Filling polygonal holes with rectangular patches. In: Strasser W, Seidel H-P, editors. Theory and practice of geometric modelling. New York: Springer; 1989. p. 81–91.
[6] HsuandD KL, Tsay M. Corner blending of free-form n-sided holes. IEEE Computer Graphics and Applications 1998;18(1):72–8.
[7] Hwang WC, Chuang JH. N-sided hole filling and vertex blending using subdivision surfaces. Journal of Information Science and Engineering 2003;19(5):857–79.
[8] Lee ETY, Lucian ML. Möbius reparametrizations of rational B-spline. Computer-Aided Geometric Design 1991;8(3):213–5.
[9] Lin F, Hewitt WT. Expressing Coons–Gordon surfaces as NURBS. Computer-Aided Design 1993;26(2):145–55.
[10] Liu D. GC^1 continuity conditions between two adjacent rational Bézier surface patches. Computer Aided Geometric Design 1990;7(1–4):151–63.
[11] Pferifer N. 3D terrain models on the basis of a triangulation. Geowissenschaftliche Mitteilungen; 2002. heft.
[12] Piegl LA, Tiller W. The NURBS book. 2nd ed. New York: Springer; 1997.
[13] Piegl LA, Tiller W. Filling n-sided regions with NURBS patches. The Visual Computer 1999;15(2):77–89.
[14] Sarraga RF. G^1 interpolation of generally unrestricted cubic Bézier curves. Computer Aided Geometric Design 1986;4(1–2):23–39.
[15] Shi X, Wang T, Yu P. A practical construction of G^1 smooth biquintic B-spline surfaces over arbitrary topology. Computer-Aided Design 2004; 36(5):414–24.
[16] Tookey R, Ball A. Approximate G^1 continuous interpolation of a rectangular network of rational cubic curves. Computer-Aided Design 1996;28(12):933–1022.
[17] Varady T. Survey and new results in n-sided patch generation. In: Martin RR, editor. The mathematics of surfaces II. Oxford: Clarendon Press; 1987. p. 203–35.
[18] Varady T, Rockwood AP. Geometric construction for setback vertex blending. Computer-Aided Design 1997;29(6):413–25.

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

Jun-Hai Yong is a faculty member in School of Software at Tsinghua University, China, since 2002. 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.



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

Jean-Claude Paul is a senior researcher at CNRS and INRIA (France), and currently visiting professor at Tsinghua University. He received his Ph.D. in Mathematics from Paris University in 1976. His research interests include Numerical Analysis, Physics-Based Modeling and Computer-Aided Design.

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