Automatic Hole-Filling of CAD Models with Feature-Preserving

Xiaochao Wang\textsuperscript{a}, Xiuping Liu\textsuperscript{a,b}, Linfa Lu\textsuperscript{c}, Baojun Li\textsuperscript{d}, Junjie Cao\textsuperscript{a,b}, Baocai Yin\textsuperscript{a,e}, Xiquan Shi\textsuperscript{a,f}\textsuperscript{*}

\textsuperscript{a}School of Mathematical Sciences, Dalian University of Technology, Dalian, China  
\textsuperscript{b}Department of Engineering Mechanics, State Key Laboratory of Structural Analysis for Industrial Equipment, Dalian University of Technology, Dalian, China  
\textsuperscript{c}School of Information Science & Technology, Sun Yat-sen University, Guangzhou, 510006, China  
\textsuperscript{d}National Engineering Research Center of Digital Life, Guangzhou, 510006, China  
\textsuperscript{e}School of Automotive Engineering, Faculty of Vehicle Engineering and Mechanics, Beijing Municipal Key Lab of Multimedia and Intelligent Software Technology, Beijing University of Technology, Beijing 100124, China  
\textsuperscript{f}Department of Mathematical Sciences, Delaware State University, Dover, DE 19901, USA

Abstract

In this paper, we propose an automatic hole-filling method, particularly for recovering missing feature curves and corners. We first extract the feature vertices around a hole of a CAD model and classify them into different feature sets. These feature sets are then automatically paired, using ordered double normals, Gaussian mapping and convex/concave analysis, to produce missing feature curves. Additionally, by minimizing a newly defined energy, the missing corners can be efficiently recovered as well. The hole is consequently divided into simple sub-holes according to the produced feature curves and recovered corners. Finally, each sub-hole is filled by a modified Advancing Front method individually. The experiments show that our approach is simple, efficient, and suitable for CAD systems.

Keywords: Hole-filling, Feature-preserving, Feature set matching, Corner recovery

1. Introduction

Triangular meshes are widely used to represent 3D models. However, for many reasons, such as occlusion, limitation of scanners and the damaging of original models, triangular meshes may contain holes. These holes make it difficult for many subsequent operations, such as model rebuilding, rapid prototyping and finite element analysis. So, it is necessary to fill holes in a reasonable manner. In addition, hole-filling also plays important roles in other applications, such as feature suppression [1], mesh merging [2] and mesh parametrization [3].

A large number of hole-filling methods have been proposed in current literatures. Most of them work well for small holes located on smooth regions, however, it is still a challenge to fill larger and complex holes with missing sharp features. Due to the diversity and complexity of the holes, none of the existing methods works for all holes. In this paper, we focus on the hole-filling problem of incomplete piecewise smooth meshes that may have holes locating on feature regions. In addition, holes may also contain missing corners. Inspired by the idea of divide-and-conquer, we propose an automatic feature-preserving hole-filling framework for restoring the missing corners and sharp edges. Besides recovering the feature structures, our results are of high quality in approximating the original meshes. On top of that, we also present a new energy function to recover the missing corners.

To recover the missing sharp features, three main steps are involved. First, feature vertices around the hole are extracted and classified into different feature sets. Then, we automatically match the feature sets to construct the missing feature curves to divide the original hole into some simple sub-holes. A simple sub-hole is a hole of containing no any feature boundary vertex except those of sharing as common boundary vertices with other sub-holes. Finally, the sub-holes are filled by the modified advancing front method (MAFM) [4]. In this process, the reconstructed feature curves are preserved unchanging once they are constructed. If there are missing corners in the hole, they can also be efficiently recovered by minimizing a newly defined energy.

The main contributions of this paper can be summarized as follows

- The extracted feature vertices are automatically paired to recover the missing sharp features, which can avoid tedious user interactions and thus improve the algorithm’s efficiency.

- The missing feature curves are explicitly reconstructed by cubic splines, which interpolate the corresponding feature vertices and guarantee the accuracy of the recovered feature structures.
• Significantly different from previous works, the missing corners can be recovered by minimizing a newly presented energy before the hole is filled.

The rest of the paper is organized as follows. The related work is briefly reviewed in Section 2. Section 3 gives the outline of our method and the details are described in Section 4. Experimental results are presented in Section 5, and Section 6 concludes the paper.

2. Related work

Numerous hole-filling methods have been proposed. Generally speaking, they can be classified into two categories: volume-based methods and mesh-based methods [5].

Volume-based methods. The input model is first converted into an intermediate volumetric grid. After hole-filling, the volume presentation is again converted to a triangular mesh by different iso-surface extraction methods [6–12]. Davis et al. [7] defined a signed distance function in the vicinity of observed surfaces. Then a diffusion process was applied to extend this function through the volume until its zero set bridged all holes. A voxel-based method [8] was proposed to simplify and repair polygonal models by adopting open and close morphological operators. Bischoff et al. removed typical mesh artifacts by using volumetric geometry representation in [10, 11]. Héthroy et al. [13] extracted a valid two-manifold surface from a voxel set in geometrical and topological defect regions in a user-friendly manner.

Volume-based methods excel in robustness in filling complex holes. However, as stated in [5], most volume-based methods use the Marching Cubes algorithm for reconstruction, which generates blobby surfaces to result in losing the sharp corners and edges of the input model. Although feature-preserving contouring algorithm [14] has been proposed and used in [9, 10], it still cannot efficiently reproduce all geometric features, especially the missing sharp features.

Mesh-based methods. Unlike volume-based method, mesh-based methods fill holes locally with the rest of the surface unchanged. Barequet and Sharir [15] combined matched border stitching and minimum area triangulation to fill holes. Instead of a simple step hole-filling, Li et al. [16] used refinement and fairing to refine the obtained patch mesh, so that the triangle density agrees to that of the surrounding mesh. This method is further used in [17]. Zhao et al. [18] first got an initial patch mesh by advancing front method (AFM), then optimized vertex’s position by solving the Poisson equation. Pernoit et al. [19] filled holes by minimizing the curvature variant between the surrounding and inserted meshes. However, for some complex cases, stiffer lines have to be created manually. This method was further improved in [20].

Jun [21] presented a piecewise hole-filling algorithm for complex holes. The algorithm incrementally splits a complex hole into simple sub-holes with respect to the 3D shape of the hole boundary, and then consecutively fills each sub-hole.

Different from filling holes directly on 3D meshes, [22, 23] triangulated holes on 2D parametric domain and embedded the triangulation into 3D meshes by minimizing some energy functions. The hole-filling patches can also be obtained by adapting scattered data fitting techniques, such as radial basis function (RBF) [24–26] and moving least squares [27].

Although the methods mentioned above work well for holes located on smooth regions, most of them cannot propagate feature structures therein. The features involved in hole-filling can be divided into two categories. One is geometric detail or texture structure, such as the geometric details of a golf or the hair region on a head. The other is sharp features, such as sharp edges or corners in CAD models, on which is what this work focusing.

Handling geometric details. For reconstructing geometric details of the missing holes, a variety of methodologies have been proposed. In the example-based-based methods [28–30] or context-based methods [31, 32], holes are iteratively filled by copying similar matching patches from the input model itself or from some existing models. Kraevoy et al. [33] first computed a mapping between the incomplete mesh and a template model, then employed this mapping to glue together the components of the input mesh and to fill the holes simultaneously. Nguyen et al. [34] reconstructed 3D geometric information of holes from synthesizing local gradient images on 2D parametric domain. Xiao et al. [35] presented a texture synthesis and context based method for appearance and geometric completion of point set surfaces. Recently, Li et al. [36] developed a skull completion method by combining symmetric detection and template-based method.

Handling sharp features. Comparing to handling geometric details, a few methods exist for processing sharp features. Chen et al. [25, 26] reconstructed sharp features via sharpness dependent filter [37]. Two processes are involved in this approach: producing an initial repaired model by using RBF based interpolation method and then recovering the sharp features by a sharpness dependent filter.

Although [25, 26] can recover sharp features, there are two potential problems. First, RBF cannot provide well-shaped initial patch meshes for sharpness dependent filtering when the hole is big or complex. Second, the sharpness dependent filter has an important parameter that should be carefully selected to prevent the positions of the features unexpectable. This two drawbacks may very easily prevent the method from recovering the missing sharp features of the input models.

Following the piecewise scheme of [21], a feature-preserving hole-filling method was presented in [38] by using polynomial blending technique. Polynomial blending curves are constructed based on the detected feature vertices around the hole to complete the missing parts of the feature curves. Generally speaking, this method has two limitations. One is that the feature curve segments around the hole are matched interactively. The other is that the method can not recover missing corners. To overcome these limitations, an automatic hole-filling method is proposed in this paper, which can not only reconstruct the missing feature curves, but also recover the missing corners.
3. Method overview

Following most mesh-based methods [16, 18–20], we assume that all meshes are triangular, oriented, manifold and connected, with boundary is allowed. That is, two separate holes will have no vertices in common and any hole contains no islands (otherwise the mesh can not be connected). A triangular mesh is represented by \( M = (V, E, F) \), where \( V = \{v_1, v_2, \ldots, v_n\} \) denotes the set of vertices, \( E \) denotes the set of edges and \( F = \{f_1, f_2, \ldots, f_m\} \) denotes the set of faces.

The flowchart of our framework is shown in Fig. 1. At the beginning, the method described in [18] is adopted to detect all holes of the input mesh. To this end, we first identify all boundary vertices based on the property that the numbers of a boundary vertex's 1-ring triangles and 1-ring edges are not equal. Then, starting from any boundary vertex, a set of connected boundary edges are traced until getting a closed loop. This loop should be the boundary of a hole. All holes can be identified in the same manner.

![Flowchart of the hole-filling framework](image)

Fig. 1. Overall flowchart of the hole-filling framework.

If there are degenerated triangles along holes, an optional cleaning process will be applied. More details can be found in [19]. For each hole, a local mesh, usually up to five or six rings of vertices and triangles near to the hole boundary, is used in hole-filling (see Fig.2(a)).

To determine whether the feature recovery process is necessary or not, the feature vertices around the hole are detected based on normal tensor voting [39]. According to [39], all vertices can be classified into three types, that is, face type, sharp edge type and corner type. Both sharp edge type and corner type vertices are called feature vertices.

As mentioned above, a hole is called simple if it contains no feature vertices except those of sharing as common boundary vertices with other holes. A simple hole will be directly filled by AFM or MAFM. Otherwise, the missing feature curves and corners are explicitly recovered to divide the original hole into simple sub-holes. During the hole filling process, the reconstructed features are effectively preserved. We demonstrate a number of experiments to illustrate the applicability and effectiveness of our method. In the following sections, we focus on recovering the missing sharp features.

4. Feature recovery

Denote all detected feature vertices around a hole by \( F_h \). Some of them will be removed out and the rests are grouped into different feature sets to construct the missing features.

4.1. Feature sets

A boundary feature vertex is a feature vertex locating exactly on the hole’s boundary. For each hole, all boundary feature vertices \( F_h \) are ordered in the oriented way of agreeing with the orientation of the input mesh and are denoted by \( F_h = \{v_1, v_2, \ldots, v_n\} \) (see Fig.2(b)).

The feature sets can be constructed as follows. For each vertex \( v_i \) of \( F_h \), a feature set \( F_{v_i} \) is created by putting \( v_i \) as the first element of \( F_{v_i} \) and \( v_i \) is re-denoted by \( v^0_i \). The rest elements of \( F_{v_i} \) are recursively defined by: \( v^j_i \) will be the \( j \)-th element of \( F_{v_i} \) if \( v^j_i \) is the only feature vertex in \( F_h \setminus \{v^{j-1}_i\} \), such that \( \{v^{j-1}_i, v^j_i\} \) is an edge of \( M \). That is, \( v^j_i \) should be the end element of \( F_{v_i} \) if there is no feature vertices or there are more than one feature vertices in \( F_h \setminus \{v^{j-1}_i\} \) to co-edge with \( v^j_i \). The vertices of feature set \( F_{v_i} = \{v^0_i, v^1_i, v^2_i, \ldots, v^n_i\} \) are connected one by one and each set contains one and only one boundary feature vertex \( v^0_i \) as its representative element (see also Fig.2(b)). The feature vertices in \( F_h \) that are not selected into any feature set should be removed out.

4.2. Feature sets matching

In order to obtain pleasing results for big or complex holes with missing sharp features, a reasonable choice is to divide the original hole into some simple sub-holes. To this end, the feature sets should be correctly matched to construct the missing

![Feature recovery](image)

Fig. 2. Feature recovery. (a) Boundary vertices are marked in red and the vertices of the local mesh are marked in green. (b) Selected feature vertices are marked in colors and the boundary feature vertices are marked in green. (c) Reconstructed feature curves marked in red and the sampled vertices on feature curves located in the hole are marked in blue. (d) The obtained patch mesh is marked in green and the reconstructed feature lines are shown in red.
feature curves. In this section, we first consider the case that holes do not contain any missing corners and thus each missing feature curve has to intersect the hole to two boundary feature vertices. We postpone discussing holes with missing corners in Section 4.4. Hence only even number of feature sets will be handled in this section.

In practice, to match two feature sets we only need to match their representative elements, i.e., the boundary feature vertices. Our matching method is based on the fact that there is at least one matching pair of $F_b$ which is comprised of two adjacent feature vertices. This matching pair determines all other matching pairs of $F_b$. As shown in Fig.2(b), if $[v_3, v_4]$ is a matching pair, then $[v_2, v_3]$ and $[v_1, v_6]$ should also be matching pairs.

In fact, if the number of boundary feature vertices is more than two, in $F_b$ there should be two pairs of comprised of two adjacent feature vertices. If one pair is $[v_1, v_m]$, then its corresponding pair should be $[v_{m/2}, v_{1+m/2}]$. For example, in Fig.2(b), two such pairs are $[v_1, v_6]$ and $[v_3, v_4]$. Therefore, we will focus on finding two matching pairs of adjacent boundary feature vertices.

**Ordered double normals.** Normal is one of the important differential geometric properties. The behavior of normal reflects local shape of the surface, which can be accurately expressed by Gaussian mapping. Usually, the normal of a vertex is computed by weighted average of its one ring triangles’ normals, not distinguishing it is a feature vertex or not. To fully use the properties of edge type feature vertices, we introduce ordered double normals for each boundary feature vertex. Namely, the ordered double normals $N_i = [N_1^i, N_2^i]$ of the boundary feature vertex $v_i$ are defined by the normals of two adjacent triangles associated to the oriented edge $[v_i, v_j]$ and $[v_j, v_i]$ (see Fig.3(a)).

At this point, to pair two adjacent boundary feature vertices, the following energy is constructed

$$E(i) = \frac{1}{2} \sum_{j=1}^{m} E(i, j),$$

where $1 \leq i \leq m/2$ and $j = i + 1 + m/2$. In fact, the term $|N_1^i - N_2^i|$ reflects the normal clustering of Gaussian mapping on the unit Gaussian sphere.

Then, two pairs $[v_1, v_{i+1}]$ and $[v_{i+m/2}, v_{i+1+m/2}]$ are what we want if

$$E(i) = \min_{1 \leq j \leq m/2} E(j).$$

Fig.3(b) shows a matching result obtained by minimizing $E$.

**Convex/Concave analysis.** Convex/concavity is another important geometric property of a vertex, but the energy $E$ doesn’t take it into account. This may lead to mismatching (see Fig.3(c)). In general, the matched boundary feature vertices should have the same convexity/concavity. Therefore, a pair, together with its corresponding pair, should be filtered out in advance, if the two adjacent boundary feature vertices contained in this pair have different convexity/concavity.

Convexity/concavity of a vertex can be determined by its normal $n_i$ and Laplacian coordinate $\delta_i$,

$$Booth(v_i) = \begin{cases} \text{convex} & \text{if } n_i \cdot \delta_i \leq 0, \\ \text{concave} & \text{if } n_i \cdot \delta_i > 0, \end{cases}$$

where $[40]$

$$\delta_i = \sum_{j \in N(v_i)} \omega_{ij} \cdot v_j - v_i,$$

with the default weights $\omega_{ij} = 1/d_i$ and $d_i$ is the valence of $v_i$. Combining with convexity/concavity, correct matching pairs can be obtained (see Fig.3(d)).

**Further checking.** To enhance the robust of feature sets matching, we further check the matching pairs according to the following observation. For a correct matching pair $[v_i, v_j]$, two vectors $\overset{\rightarrow}{w_i} \overset{\rightarrow}{w_j}$ and $\overset{\rightarrow}{v_i} \overset{\rightarrow}{v_j}$ should both point to the hole region. Otherwise, the pair is considered as false matched. This can be easily checked with the help of vectors $\overset{\rightarrow}{w_i} \overset{\rightarrow}{v_j}$ and $\overset{\rightarrow}{v_j} \overset{\rightarrow}{v_i}$, which both point to the hole region as well. For each correct matching pair $[v_i, v_j]$, it should hold both $\overset{\rightarrow}{w_i} \overset{\rightarrow}{v_j} \cdot \overset{\rightarrow}{v_i} > 0$ and $\overset{\rightarrow}{v_j} \overset{\rightarrow}{v_i} \cdot \overset{\rightarrow}{v_j} > 0$ (see Fig.4(a)).

It is worth to point out that it could exist several potential matching results for a hole whose feature boundary vertices are all on a same feature curve, as shown in Fig.4(b). The matching strategy for Fig.4(b) may give two potential matching results. One is $[v_1, v_2]$ and $[v_3, v_4]$ (see Fig.4(a)), which generate two small feature segments to fill the missing part of the feature curve. The other is $[v_1, v_4]$ and $[v_2, v_3]$, which will generate a long and a short segments that overlap with each other. The latter case will cause serious problems in hole-filling, such as self-intersection. Fortunately, this mismatching can be eliminated by the fact that the correct matching should satisfy the conditions $\overset{\rightarrow}{w_i} \overset{\rightarrow}{v_j} \cdot \overset{\rightarrow}{v_i} > 0$ and $\overset{\rightarrow}{v_j} \overset{\rightarrow}{v_i} \cdot \overset{\rightarrow}{v_j} > 0$, as mentioned.
4.3. Feature curves reconstruction

For each matching pair \( \{v_i, v_j\} \), the vertices of \( F_{v_i} \) and \( F_{v_j} \) will be put together as the interpolation knots to reconstruct the potential feature curve. For flexibility and efficiency, a parametric cubic spline of cumulative chord length is adopted to fit the potential feature curve.

Given \( n + 1 \) vector values \( P_i(x_i, y_i, z_i), i = 0, 1, \ldots, n \), denote the chord length of two adjacent vectors by

\[
l_i = \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2 + (z_i - z_{i-1})^2},
\]

where \( i = 1, 2, \ldots, n \). Then, \( \Delta : 0 = t_0 < t_1 < \cdots < t_n \) is a partition of the parameter interval \( [t_0, t_n] \), where \( t_i = \sum_{j=1}^{i} l_j, i = 1, 2, \ldots, n \). Based on the partition \( \Delta \), we construct \( C^1 \) cubic spline functions \( x(t), y(t) \) and \( z(t) \) to interpolate data \( \{x_i\}, \{y_i\} \) and \( \{z_i\} \), respectively. \( P(t) = (x(t), y(t), z(t)) \) is called the parametric cubic splines of cumulative chord length with respect to data \( P_i(x_i, y_i, z_i), i = 0, 1, \ldots, n \).

The constructed feature curves and the new sampled vertices are shown in Fig.2(c) and Fig.5. These feature curves divide the original hole into several simple sub-holes. The boundary vertices of each sub-hole are composed of the sampled vertices obtained from feature curves and the original boundary vertices.

Due to the diversity and complexity of holes, some special cases may be encountered. For instance, if a feature curve intersects with a hole in only one vertex, there is nothing to reconstruct with this feature there. In this case, the separate feature vertex will be treated as an outlier in the set of the detected feature vertices.

Fig. 4. Further checking. (a) The boundary feature vertices are marked in green and the edge type vertices directly connected to them are marked in red. The dash red vectors composed by \( v_i \) and \( v_j \) point to hole region. For a matching pair \( \{v_i, v_j\} \), two vectors pointing to each other are shown in green. (b) A hole with boundary marked in green. (c) Filling result of (b).

Fig. 5. More results of feature curves reconstruction. (a) Octa-flower. (b) Twirl. (c) Twist. (d) Part of the Fandisk.

4.4. Corner recovery

In this section, we propose a novel energy to recover the potential missing corners. By minimizing the energy, it generates a vertex of the potential missing corner. Furthermore, if the minimized energy is relative small, the potential missing corner obtained should be a missing corner. It is also clear that the vertex of a missing corner should be the intersection of multiple (at least three) smooth surfaces. For a missing corner, its vertex is used, together with the boundary feature vertices, to reconstruct the missing feature curves by using cubic spline curves. In the following, we take Fig.6(a) as an example to illustrate the process of recovering the missing corner.

Fig. 6. Corner recovery. (a) Part of a cube model with a missing corner. (b) A virtual filled patch mesh. (c) The reconstructed corner and feature structures. (d) Filling result of (a).

If \( \mathbf{c} = (x, y, z) \) is a potential missing corner, we can obtain a virtual filling patch by connecting it to all boundary vertices (see Fig.6(b)). To determine the position of the missing corner, we construct an energy function as follows:

\[
E(c) = \sum_{v \in V} E_d(c) + \sum_{e \in E} E_f(c),
\]

where \( E_d \) and \( E_f \) are data and feature energy functions, respectively. The data energy function measures the smoothness of the corner and the feature energy function measures the consistency of the corner with the feature curves.
corner, the following energy is constructed. For each boundary edge \([v_i, v_j]\), there are two faces. One is the existing face \(f_{jk} = [v_i, v_j, v_k]\) and the other is the virtual face \(f_{jk} = [v_j, v_i, e]\). Consider the volume \(T_{ij}\) of the tetrahedron composed of vertices \(v_i, v_j, v_k, e\),

\[
T_{ij} = \frac{1}{6} \begin{vmatrix} v_{ix} & v_{ix} & v_{ix} & 1 \\ v_{iy} & v_{iy} & v_{iy} & 1 \\ v_{iz} & v_{iz} & v_{iz} & 1 \\ x & y & z & 1 \end{vmatrix},
\]

(6)

where \((v_{ix}, v_{iy}, v_{iz})\) is the coordinates of \(v_i\). Summing the squares of \(T_{ij}\) over all boundary edges, we get

\[
T_B = \sum_{(v_i, v_j) \in E_B} T_{ij}^2,
\]

(7)

where \(E_B\) is the set of all boundary edges. If \(e = (x, y, z)\) is not a potential missing corner, i.e., does not lie on the intersection of multiple piecewise surfaces, the minimum of \(T_B\) will be relevant large. If it does, the minimum of \(T_B\) must be a very small number. Thus, we can recover the missing corner by minimizing \(T_B\), which is the intersection of multiple piecewise surfaces.

Additionally, to obtain a more evenly distributed virtual patch mesh, the tetrahedron’s volume associating to each inner edge of the virtual patch mesh is also considered. That is,

\[
T_I = \sum_{(v_i, e) \in E_B} T_{ie}^2,
\]

(8)

where \([v_i, e]\) is the inner edge of the virtual patch mesh with the exception that \(v_i\) belongs to the boundary feature vertex set \(F_b\).

Then, the total energy for recovering the missing corner can be written as

\[
T = T_B + T_I.
\]

(9)

After restoring the missing corner by minimizing \(T\), the missing feature curves are reconstructed, as stated above. These reconstructed missing feature curves divide the original hole into simple sub-holes (see Fig.6(c)). Fig.6(d) shows the final filling result.

4.5. Sub-hole filling

At this point, all feature structures are recovered and the original hole has been divided into simple sub-holes. Several methods can be used to fill the sub-holes, such as the method used in [16], planar triangulation used in [21, 23, 41] or AFM used in [18].

It is well known that AFM works well for planar holes. However, due to each new vertex is created on the plane determined by the adjacent boundary edges, AFM does not work well for curved holes. To address this issue, a modified advancing front method [4] is adopted. The main idea of MAFM is to compute an optimal inserting direction to improve filling results. More details are referred to [4].

Additionally, an optional mean filter can be applied to the inner part of each newly created patch mesh. By applying the mean filter, the newly created vertices will be distributed more evenly and the quality of the patch mesh can be further improved.

4.6. Corner detection

After sub-hole filling, we turn back to discuss two problems. One is how to judge whether there are missing corners in a hole. The other is how to decide which parts of the boundary edges and boundary feature vertices should be used to recover each missing corner.

For each hole with the detected boundary feature vertices \(F_b\), we define \(K = \#(F_b)\), the number of elements of \(F_b\). If \(K = 0\) or 1, fill the hole directly by MAFM. If \(K = 2\), maybe there is a missing feature curve and the feature set matching is applied.

If \(K \geq 3\), we first assume that there are missing corners and use all boundary feature vertices and all boundary edges to compute the energy \(T\) according to Eq.9. If \(T\) is smaller than a specified threshold \(\varepsilon\), we say that there is a potential missing corner. Otherwise, we have to consider all the combinations of the vertices of \(F_b\) to determine if there are missing corners contained in the hole or not. Thus, the computational complexity could be the exponential in \(K\).

**Algorithm 1 Corner detection**

1. \(F_b = \{v_1, v_2, \ldots, v_r\}, K \geq 3\) // boundary feature vertices
2. \(\#(F_b)\) the number of element in \(F_b\)
3. \(\varepsilon\) // threshold
4. \(IF_b\) // involved boundary feature vertices
5. \(T\) // the energy for \(IF_b\)
6. for \(i = \#(F_b) - 1: 3\) do
7. \(i = \#(F_b)\) then
8. \(IF_b = F_b\)
9. compute \(T\) for \(IF_b\)
10. if \(T < \varepsilon\) then
11. compute corner
12. BREAK
13. end if
14. else
15. \(j = 1: \#(F_b)\) do
16. \(IF_b = 1\) adjacent boundary feature vertices of \(F_b\)
17. compute \(T\) for \(IF_b\)
18. if \(T < \varepsilon\) then
19. compute corner
20. \(F_b = F_b - IF_b\)
21. go to Step 6
22. end if
23. end for
24. end if
25. end for

To reduce the computational complexity, two important assumptions are taken into account. First, only the combinations composed of consecutive boundary feature vertices are considered, and second, the number of the feature vertices contained in each combination is not too large, at most \(B\) (a number to define later), for example. In fact, in geometry, it is reasonable only to use consecutive feature vertices to obtain a missing corner. It is also reasonable to further assume that \(B\) is not too large, since it is not common that a large number of feature curves meeting to one point. In our experiment, we set \(B\) equals five.

After first assumption, we only need to check a total number of not more than \((\#(F_b) - 3)\#(F_b)\) combinations whose sizes...
run from 3 to \#(F_b) − 1. After second assumption, there are only total number of not more than \((B − 2)\#(F_b)\) combinations to check.

Fig. 7. For recovering each corner, the information of the used boundary vertices are shown in red.

For each combination, boundary edges lying among the adjacent boundary feature vertices are also used in determining the missing corner. Additionally, to enhance the robust of this scheme, a few boundary edges exceeding the scope of the combination could be used as well. As shown in Fig.7, for each missing corner, the vertices of the used boundary edges are marked in red. If there is a case with the energy smaller than the specified threshold, then a potential missing corner is detected. After one corner is found, the involved boundary feature vertices of this corner are removed from \(F_b\). If the number of the remaining boundary feature vertices is not smaller than three, repeating the above process until all three adjacent boundary feature vertices are considered. In this way, all potential missing corners can be searched. The details of the method are shown in Algorithm 1.

If the hole contains missing corners, then there exists at least one combination such that its energy is small enough, smaller than a carefully selected threshold. Otherwise, we believe that there is no missing corner. In our implementation, the threshold \(\varepsilon\) is taken as 0.001 times of the regular tetrahedron’s volume (the edge length of this regular tetrahedron equals to the average of all edges in the mesh), which always results ideal results. For a hole with \(n\) boundary edges and \(K (K \geq 3)\) boundary feature vertices, there are at most \(O(n + K^2)\) operations. In practice, the number of boundary feature vertices of each hole is usually small. Therefore, we can find all potential missing corners in \(O(n)\) operations.

5. Experiments and results

In this section, we first demonstrate a number of experiments and comparisons with other state-of-art methods to illustrate the effectiveness of our approach. The limitations of our method are discussed at the end of this section.

5.1. Feature curves recovery

Let us first test our method on a box model (similar to the model used in [41]) and the Fandisk model (the same one used in [38]) to recover the missing sharp features. From Fig.8(a-d) we can see that the methods [38] and [41] work well for recovering the missing sharp features. In fact, for these simple cases we can also accomplish perfect reconstructions, which can be easily observed from Fig.8(f) and (h).

To further show the advantages of our method, we compare our method with that of Ju [9], Attene [17], and Chen [26] in four aspects: the shape recovery, the quality of the recovered meshes, the ability to handle noisy data, and the efficiency of filling big holes.

The first advantage of our method is that the missing shape can be effectively recovered. In Fig.9, for the small hole locates on a relative smooth area, all methods can achieve acceptable results, and the results of Attene and ours are slightly better. But for the hole with missing sharp features, both Ju and Attene’s methods could not nicely recover the missing sharp features. On the contrary, our method achieves an almost perfect result (see Fig.9(d)).

Another comparison between our approach and Chen et al. [26] is carried out in Fig.10. Due to the sharpness de-
Fandisk with two holes. (b-d) The filling results obtained by the methods of Ju [9], Attene [17] and ours, respectively.

Fig. 10. Octa-flower. (a) Result of [26]. (b) Result of our method.

The second advantage of our method is that our results can well match the original feature parts. In Fig.11, it can be seen that both methods are able to reconstruct the missing sharp features. However, from the zoom-in views, we can see that the feature curve constructed by [26] could not smoothly match to its surrounding feature curve, while the feature curve constructed by our method is much better due to the nice fitting property of the cubic spline curves.

The third advantage of our method is that it can handle noisy models. Once the feature vertices of the noisy model are correctly detected, our method can generate reasonable results with features recovered. As shown in the zoom-in views of Fig.12, our method obtains a more natural filling result than that of [26]. In addition, the vertex distribution of Chen and Cheng’s result is over dense, while our vertices’ distribution matches to the surrounding meshes.

The fourth advantage of our method is that our method can deal with big hole with missing sharp features. Applying our method to the big hole that almost crosses the whole Fandisk model as shown in Fig.13, the missing part, in particular, the seven sharp feature curves are completely restored (see Fig.13(c) and (d)). Two more experiments are shown in Fig.17 and Fig.18.

5.2. Corners recovery
Different from the recovered missing corner shown in Fig.6(d)), another recovered missing corner is shown in Fig.14(b). One difference is that the missing part in Fig.14(a) is much bigger. Another notable difference is that there is a curved missing feature line in Fig.14(a), which makes the missing corner recovery much harder. Fig.14(b) shows that our method still recovers the missing corner and other missing features efficiently to obtain a visually pleasing result.

If there are more corners missed in a hole, we can restore them one by one. In Fig.15(a), after restoring the two missing corners, one missing feature line in the middle of the hole can be easily reconstructed. These recovered corners and feature curves divide the original hole into simple sub-holes, which are effectively filled by AFM (see Fig.15(b)).

5.3. Limitations
Our method mainly focuses on feature-preserving hole-filling for piecewise smooth meshes. The complex case considered in [7] or the geometry details considered in [31] are not
considered and they will be our future works. In addition, if there are no sufficient information available, it is difficult to restore the missing features automatically (see Fig. 16(a) and (b)).

It should be also noted that we only focus on the holes without islands. In practice, this problem could be solved if we consider the solution proposed in [16, 38], where line segments (either automatically or manually) are used to create bridges between disconnected boundaries. Following this scheme, if there are feature vertices detected in island, we can heuristically match feature vertices between island and the hole’s boundary to construct sharp features. Since dealing the hole with island is not the main contribution of the current work, it will also be considered in our future work.

6. Conclusions

In this paper, we propose a feature-preserving hole-filling method to automatically recover the missing sharp features and corners from piecewise smooth surfaces. Based on a sequence of processes including feature detection, feature sets matching, corner recovery, feature curves reconstruction and sub-hole filling, we can restore the potential sharp features hidden in incomplete meshes, not only the missing feature curves, but also the missing corners, in an automatic way. Experimental results demonstrated that our results are much closer to reality and outperform that of obtained by previous methods.

Acknowledgements

The authors are grateful to the anonymous reviewers for their extensive help in improving this work. We also express our thanks to Prof. Kuo-Yong Cheng and his assistant Sean, Prof. Kobbelt and Marcel Campen for providing the comparison data. We thank Prof. Ligang Liu and Prof. Zhong Li for their helpful discussions. In addition, we would like to thank Prof. Tao Ju and Prof. Marco Attene for their softwares Polymender and Meshfix, respectively. This work is partially

---

Fig. 13. Fandisk with a big hole. (a) and (c) The hole viewed from two sides. (b) and (d) The result of our method.

Fig. 14. (a) Fandisk model with a missing corner. (b) Filling result.

Fig. 15. (a) A complex hole with two missing corners in a cube model. (b) The resulting patch mesh shown in red with feature structures preserved.

Fig. 16. Difficult to recover missing corners without enough information.

Fig. 17. Twirl model. (a) Twirl model with a hole. (c) Filling result. (b) and (d) are the zoom-in views of the result from different viewpoints.
References


Highlights
- A feature preserved hole-filling method is proposed for triangular meshes. - Missing feature curves can be automatically restored from the detected features. - Missing corners can be effectively recovered by minimizing a new presented energy. - Cube spline interpolation guarantees the accuracy of the recovered feature curves.