Computer-Aided Design and Applications

An Automatic Hole-Filling Algorithm for Polygon Meshes

Xiao J. Wu[1], Michael Y. Wang[2] and B. Han[1]

[1]Harbin Institute of Technology Shenzhen Graduate School, {wuxj, bhan}@hitsz.edu.cn
[2]The Chinese University of Hong Kong, yuwang@mae.cuhk.edu.hk

ABSTRACT

This paper addresses the problem of automatic hole-filling on polygon meshes based on radial basis functions (RBFs). Firstly, we use the topology connectivity of a watertight triangle mesh to detect the undesired holes. Secondly, 2 or 3-ring vertexes of the boundary of the holes are sampled from the original mesh. Thirdly, a surface patch is reconstructed from the sampled interpolation points by using RBFs. In order to stitch the surface patch and the original mesh with holes, we project the surface patch and the boundary polygon of the holes onto a plane with maximum areas, which can reduce the complexity by converting the problem from 3D into 2D. After we blend the projected polygons and the projected surface patches, then they are remapped into 3D space to finish the stitch operation. Lots of experiments reveal the efficiency and accuracy of our proposed algorithm.

Keywords: geometry processing, hole filling algorithm, radial basis functions.
DOI: 10.3722/cadaps.2008.889-899

1. INTRODUCTION

Polygon meshes are extensively used in computer graphics, geometric processing, or even in engineering. One of the methods of creating polygon meshes for a real 3D object involves scanning it from a number of different view points and realigning the different point datasets into a coordinate system, and then constructing the surface model from point cloud. However, during this process of scanning, several factors such as occlusion, low reflectance coefficients of the surface, high grazing angles, or even missing pieces in the original object can lead to incomplete data. Fig. 1 gives an example to illustrate the holes caused by view occlusion.

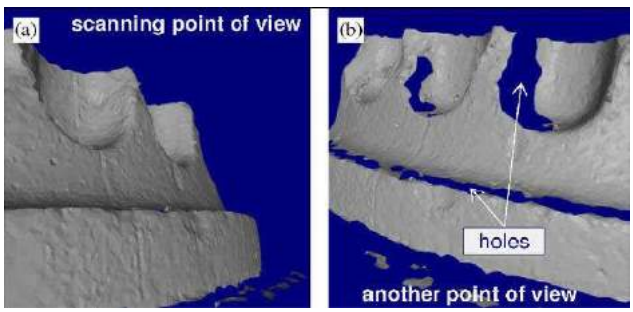


Fig. 1: Reconstructed surfaces with holes due to occlusion.

Although there are some interactive mesh-repairing tools, it could become very tedious if the input mesh model is huge and complex. A variety of methods have been developed to fill in the undesired holes on meshes. These methods can be distinguished into two main categories: the geometric and volumetric schemes [1]. Among the geometric approaches, there are methods of Liepa's minimum area triangulation of its 3D contour [2], Schneider and Kobbelt's

scheme based on solving a non-linear fourth order partial differential equation [3], Desburen's iterative implicit fairing [4], and other approaches based on the moving least-squares projection [5],[6].
Considering the volumetric methods approaches, Curless and Levoy [7] use a volumetric representation to detect the desired areas on meshes. Davis [8] applies a volumetric diffusion process to extend a signed distance function through this volumetric representation until its zero set bridges whatever holes may be present, which is also an iterative approach. Nooruddin and Turk [9] propose a simplified method to repair polygonal models using volumetric techniques. Clarenz et al. [10] use a finite element method to minimize the integral of the squared mean curvature of the filled holes. Casciola, et al. [11] suggest a method based on positive definite radial basis functions to reconstruct surface from point cloud and progressively fill the holes by using the neighboring information.
However, the main drawback of most of these methods concerns the use of iterative processes to find the shape of the mesh. Moreover, Podolak [12] suggests the following criteria to define a robust hole-filling algorithm.

Step1: Produce a non-self-intersecting watertight mesh;
Step2: Process arbitrary holes in complex meshes;
Step3: Avoid changing, approximating or re-sampling the original data away from the holes;
Step4: Incorporate user-provided constraints to allow the selection of multiple topologically differing solutions;
Step5: Process large scanned meshes with a running time proportional to the size of the holes, rather than that of the input mesh;

Radial basis functions are extensively used in neural network [13], PDE solving [14] and surface reconstruction [15],[16]. In this paper, we propose an approach to fill holes automatically based radial basis functions to meet the Podolak's standard. Firstly, the holes on the mesh are detected according to the topology of a watertight triangular mesh, then a set of centers are selected in the neighborhood area of the detected holes, thirdly, an implicit surface is reconstructed from these selected centers, and finally, the implicit surface is extracted and blended with the original mesh and the holes can be filled in.
The paper is organized as follows. In section 2 we present the basics of radial basis functions. In section 3 our new hole-filling method is described in detail, including holes detection, selection of interpolation centers, creation of the implicit surface patch and blending of the patch and the mesh. Some experimental results are performed in section 4 to validate our algorithm. The final section is conclusions.

2. RADIAL BASIS FUNCTIONS

The problem of scattered data interpolation can be stated as given a set of fixed points $\{x_i\}_{i=1}^N \in R^n$ on a surface S in R^3 and a set of function values $\{f_i\}_{i=1}^N \in R$, find an interpolant $\phi : R^3 \mapsto R$ such that

$$\phi(\boldsymbol{x}_i) = f_i, \quad i = 1,2,...,N \tag{1}$$

The interpolant can be chosen from $BL^{(2)}(R^3)$, the Beppo-Levi space of distribution on R^3 with square integrable second derivatives. This space is sufficiently large to have many solutions to the interpolation problem, and therefore we can define the affine space of interpolants:

$$\Phi = \left\{\phi \in BL^{(2)}(R^3) : \phi(\boldsymbol{x}_i) = f_i, \quad i = 1,2,...,N\right\} \tag{2}$$

The space $BL^{(2)}(R^3)$ is equipped with the rotation invariant seminorm defined by

$$\left\|\phi\right\|2 = \int_{R3}\left(\frac{\partial^2\phi(x)}{\partial x^2}\right)^2 + \left(\frac{\partial^2\phi(x)}{\partial y^2}\right)^2 + \left(\frac{\partial^2\phi(x)}{\partial z^2}\right)^2 + 2\left(\frac{\partial^2\phi(x)}{\partial x\partial y}\right)^2 + 2\left(\frac{\partial^2\phi(x)}{\partial x\partial z}\right)^2 + 2\left(\frac{\partial^2\phi(x)}{\partial y\partial z}\right)^2 dxdydz \tag{3}$$

This seminorm is a measure of the energy of "smoothness" of functions, Duchon[21] shows that the smoothest interpolant, i.e.,

$$\phi_0 = \arg\min_{\phi\in\Phi}\left\|\phi\right\| \tag{4}$$

has the simple form

$$\phi_0(\boldsymbol{x}) = \sum_{i=1}^{N}\alpha g(\left\|\boldsymbol{x} - \boldsymbol{x}_i\right\|) + P(\boldsymbol{x}) \tag{5}$$

This function is a particular example of a radial basis function. In general, a RBF is a function of the expression

$$\phi(\boldsymbol{x}) = \sum_{i=1}^{N} \alpha g(\left\|\boldsymbol{x} - \boldsymbol{x}_i\right\|) + P(\boldsymbol{x}) \tag{6}$$

Where $\boldsymbol{x}_i$ are called RBFs interpolation centers, and α_i are coefficients corresponding to each center, $g(r)$, $r = \left\|\boldsymbol{x} - \boldsymbol{x}_i\right\|$, is basis function with real value unbounded $[0,\infty)$.

There are lots of useful basis functions g which are either positive definite (PD) or conditionally positive definite (CPD) of order m, such as Gaussian, Multiquadric, Triharmonic, Thin-plate spline, etc. When PD basis function is used, the polynomial $P(\boldsymbol{x})$ can be eliminated. On the contrary, CPD basis functions of order m require a polynomial with degree of at least m [17]. Wendland [18] has developed a minimum degree polynomial solution for compact, locally-supported radial basis functions (CSRBF) that guarantee positive definiteness of the sparseness of RBFs matrix and continuity C^n of the radial basis function. For C^2 continuity, the form of CSRBF is $g(r) = (1-r)_+^4 + (4r+1)$. The 3D profiles of some basis functions with a single center are plotted in Fig. 2.

Gaussian	$g(r) = e^{\delta\|\|r\|2}$	PD	NULL
Multiquadric (MQ)	$g(r) = \sqrt{\|\| r \|\|^2 + c^2}$	CPD	order1
Inverse MQ	$g(r) = 1/\sqrt{\|\|r\|\|^2 + c^2}$	PD	NULL
Multivariate splines	$g(r) = \|\| r \|\|^{2m+1}$	CPD	order m
Multivariate splines	$g(r) = \|\| r \|\|^{2m} \ln \|\| r \|\|$	CPD	order m
CSRBF	$C^2 : g(r) = (1-r)_+^4 + (4r+1)$	PD	NULL

Tab. 1: Commonly used basis functions g.

In Eqn. (6), $P(\boldsymbol{x})$ is a polynomial of low degree. If P is a linear polynomial, the coefficients α_i mush satisfy the following side conditions or orthogonality.

$$\sum_{i=1}^{N} \alpha_i = \sum_{i=1}^{N} \alpha_i x_i = \sum_{i=1}^{N} \alpha_i y_i = \sum_{i=1}^{N} \alpha_i z_i = 0 \tag{7}$$

Combine the Eqn. (1) and (6) and the constraints (7) give rise to the following linear system:

$$\begin{bmatrix} A & P \\ P^T & \mathbf{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{\alpha} \\ \boldsymbol{b} \end{bmatrix} = \begin{bmatrix} \boldsymbol{f} \\ \mathbf{0} \end{bmatrix}, \tag{8}$$

where

$$\begin{aligned} &(A)_{i,j} = \phi(\left\|\boldsymbol{x}_i - \boldsymbol{x}_j\right\|),\ i = 1,...,N; j = 1,...,N \\ &(P)_{i,j} = pj(\boldsymbol{x}_i),\ i = 1,...,N; j = 1,...,N \\ &\boldsymbol{\alpha} = [\alpha_1,...,\alpha_N]^T \\ &\boldsymbol{b} = [\alpha_{N+1},\alpha_{N+2},\alpha_{N+3},\alpha_{N+4}]^T \\ &\boldsymbol{f} = [f_1,...,f_N]^T \end{aligned} \quad ,$$

and a 3-order polynomial is used. The solutions of the linear system (8) compose of the weight coefficients and the polynomial coefficients for the interpolation function $\phi(\boldsymbol{x})$.

3. DESCRIPTION OF HOLE-FILLING ALGORITHM

3.1 Detection of Undesired Holes

The first step of hole-filling algorithm is to detect the undesired holes in a mesh model. In most applications, the mesh should be watertight. So, all of the vertexes and edges establish well topological connectivity, illustrated in the left in Fig. 3. If there is a hole on the mesh surface, the topological connectivity can be degraded, shown in the middle and right

of Fig. 3. From the right figure in Fig. 3, it is clear that a hole consists of a loop of boundary edges which are defined as an edge only belonging to one triangle, as opposite to non-boundary edges which must be shared by two triangles. So we can use this property to detect the unwanted holes.

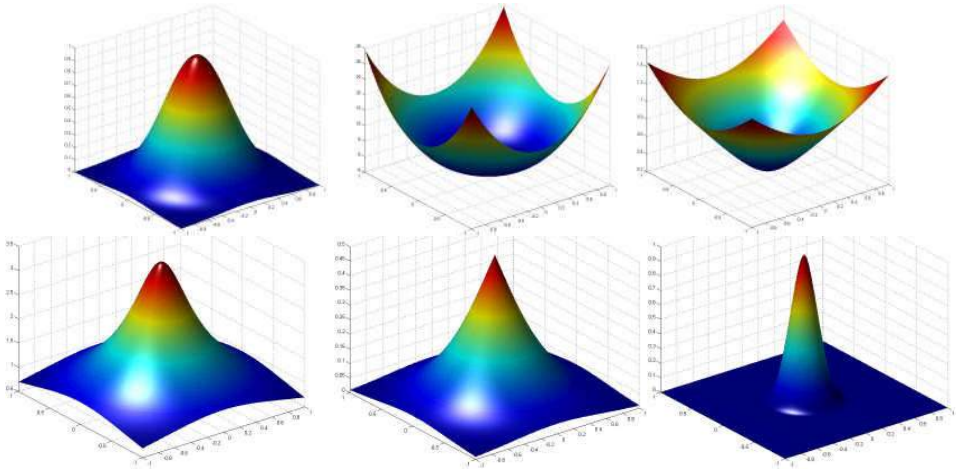


Fig. 2: 3D profiles of six different basis functions, from left to right there are Gaussian, Thinplate spline, Multiquadric, Inverse multiqudric, Triharmonic, Compactly supported, respectively.

The naive method to detect a hole is traversing all of the vertexes and the edges to check the topological connectivity, which is substantially time consuming. In our approach, we use a Kd-tree to accelerate the detection of undesired holes. The Kd-tree is a binary search tree in which each node represents a partition of the k-dimensional space. The root node represents the entire space, and the leaf nodes represent subspaces containing mutually exclusive small subsets of dataset. At any node, only one of the k dimensions is used as a discriminator, or key, to partition the space. The performance of the Kd-tree search is O(log n). Our hole-detection algorithm can be described as follows.

Step1: Compute the center of each triangle and the correspondences between the center and three vertexes of the triangle;

Step2: Set up the Kd-tree according the centers of triangles;

Step3: Search the Kd-tree and check up the topological connectivity of the corresponding edges, and find the boundary edges;

Step4: Distinguish the different boundary loops according to the topological connectivity of vertexes;

Now, we can detect all the holes from original meshes. Fig. 4 is an example of hole detection, the left figure is a mesh model with holes and the right is the detected boundaries.

Fig. 3: Hole on a mesh model and the detected hole boundary.

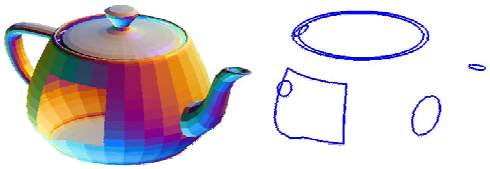


Fig.4: A mesh model and the detected holes.

3.2 RBFs Interpolation Centers

Since we want to fill the holes enclosed by the boundary edges, we only reconstruct the local implicit surface to cover the holes. From section 2, we have known that there should be some distinct points as interpolation centers. So some interpolation centers should be sampled from the original mesh. As illustrated in Fig. 5., the pink dots are the vertexes of the hole boundary loop. We can employ the topological connectivity of the boundary edge to find n-ring of its

neighborhood, demonstrated as the green and red dots in Fig. 5., from which the implicit surface can be reconstructed using radial basis functions interpolation. In our implementation, we select 2 or 3-ring of the neighborhood vertices of the boundary loop, which is sufficient to guarantee the accuracy.

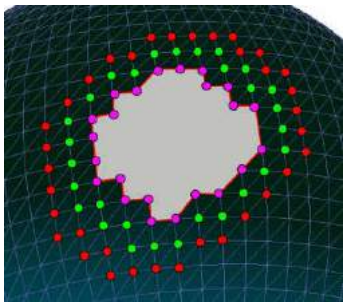


Fig. 5: The vertices on the hole boundary and the neighborhood vertices rings.

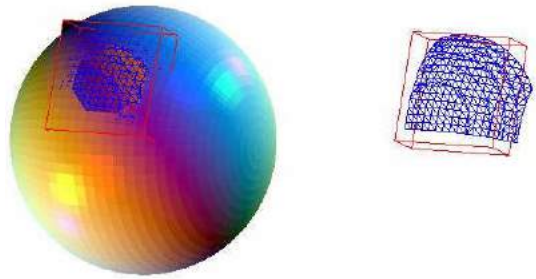


Fig. 6: The mesh model with a hole and reconstructed surface patch.

3.3 Surface Patch Creation and Blending

After we solve the coefficients of RBFs of Eqn., (8), we can extract the iso-surface from the reconstructed implicit surface. As illustrated in Fig. 6., the left figure is the original mesh with a hole and the bounding box of interpolation centers, and the reconstructed surface patch; and the right figure only shows the interpolation centers' bounding box and the extracted iso-surface. The next step of hole-filling algorithm is to blend the reconstructed surface and original mesh. As the reconstructed surface patch is larger than the holes need to be repaired. The procedure can be revealed in Fig. 7. In order to blend the surface patch and the original mesh, we must find the vertices of the surface patch locating inside of the hole. However, it is difficult to determine a point inside or outside of a polygon in 3D space. To tackle this problem, we project the surface patch onto a 2D plane, which converts the inside or outside test problem from 3D to 2D. Firstly, we use the Principal Component Analysis (PCA) method to estimate a normal to express the orientation of the surface patch, see Fig. 8. Then, the surface patch is projected onto a medium coordinate plane, $x'o'y'$, demonstrated in Fig. 8. The relationship between coordinate system $oxyz$ and $o'x'y'z'$ is can be expressed as follows.

$$P' = MP\,, \tag{8}$$

where P is a point in $oxyz$ space; P' is a point in $o'x'y'z'$ space and M is a transformation matrix, $M = [R \mid \mathbf{t}]$. R and $\mathbf{t}$ are the rotation matrix and the translation vector, respectively. Thirdly, the polygon of the hole is also projected onto the same plane, shown in Fig. 8. Then, all of the vertexes of the reconstructed surface patch inside the projected hole can be maintained as the final mesh vertexes and the triangles outside the projected polygon are eliminated. To blend the surface patch, we only deal with the triangles intersecting with the projected hole, illustrated in Fig. 9. In Fig. 9., (a) is the projected surface patch and the projected polygon (blue polygon), in (b), the red triangles can be considered as the triangles belonging to the original mesh; the two pink triangles are outside the projected polygon and can be ignored, and (c) is the fused patch. Finally, we remap these vertexes into 3D space to recover 3D information, and then the points in $o'x'y'z'$ can be transformed into $oxyz$ by using Eqn. (9). Finally, we can blend the reconstructed surface patch and the hole, and accomplish the hole-filling. In this procedure, there are lots of computations to check whether a point is inside or outside a polygon, called point inclusion test algorithm.

$$P = M^{-1}P' \tag{9}$$

3.3.1 Point Inclusion Test for Planar Polygon

Point inclusion test is a classic problem in computational geometry [19],[20]. A polygon is a simple closed curve by a finite collection of line segments. Traditionally, we can use the ray-polygon intersection algorithm to check a point is inside or outside of a polygon according to the number of the intersection points are odd or even. If an even number of intersections reveals that the point is outside the polygon, and vice versa. In our implementation, we use point in-out of polygon test method where introduces the monotonic and correlative edges to reduce the cross product operation and offer an efficient point inclusion test approach.

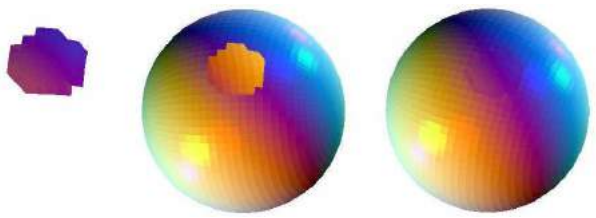


Fig. 7: The process of surface patch blending, reconstructed patch, the mesh with a hole, and the filled mesh model

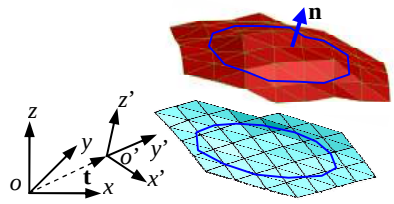


Fig. 8: The reconstructed patch, the mesh with hole, and the filled mesh model

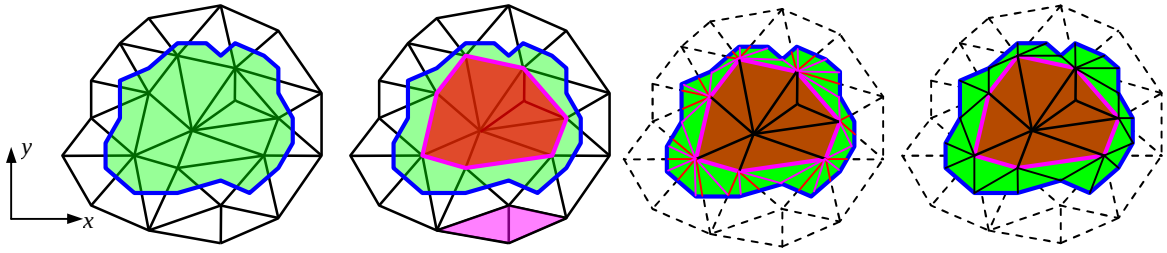


Fig. 9: Fusion of surface patch and the hole in 2D.

Assume P_iP_{i+1} is an edge of polygon P, if y coordinate of P_i is greater than that of P_{i+1}, called P_iP_{i+1} decreasing monotonic edge; if y coordinate of P_i is smaller than that of P_{i+1}, called P_iP_{i+1} increasing monotonic edge, else called P_iP_{i+1} horizontal edge. For one point p_0 which will be determined, if the y coordinate of p_0 is greater or equal to the y coordinate of P_i and smaller to the y coordinate of P_{i+1}, called increasing correlative edge about p_0, shown in Fig. 10. (a), (b); if the y coordinate of p_0 is greater or equal to the y coordinate of P_{i+1} and smaller to the y coordinate of P_i, we call P_iP_{i+1} decreasing correlative edge about p_0, shown in Fig. 10.(c), (d), there is not correlative edge between P_iP_{i+1} and p_0 shown in Fig. 10. (f), (g).

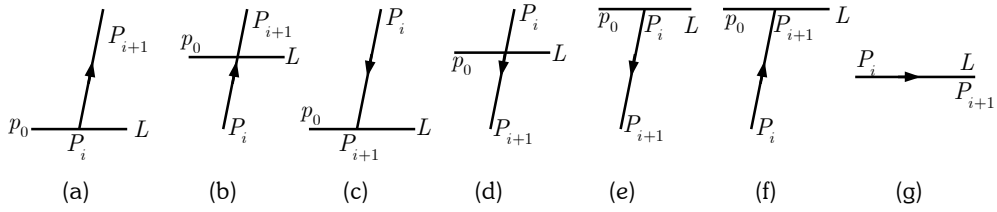


Fig. 10: Line L and edge P_iP_{i+1} intersection.

We denote R_i describing the position relation between point p_0 and its correlative edge, and suppose

$$R_i = \begin{cases} 1 \\ 0 \\ -1 \end{cases}. \tag{9}$$

In equation (9), 1 represents p_0 being the left of the ith correlative edge, -1 means p_0 being the right side of the ith correlative edge, 0 stands for p_0 being on the ith correlative edge. Then we calculate the summation of R_i on a series of correlative edges, denoted as s :

$$s = \sum_{i=1}^{n} R_i \,, \tag{10}$$

where n is the number of correlative edges. The value of s could be

$$s = \begin{cases} 0 \\ 2 \\ others \end{cases} . \tag{11}$$

In equation (11), if s is equal to 0, p_0 is out of polygon P ; if s is equal to 2, p_0 is in polygon P ; if s is equal to other values, p_0 is on the boundary of P .

3.3.2 Blending of Surface Patch and Hole

In subsection 3.3, we project the original boundary of a hole and the reconstructed surface patch onto one of the coordinate planes, which converts the problem of point inclusion test from 3D to 2D. Then we check every point of 2D polygon through the above stated algorithm of point inclusion test. There are three relations between the projected hole and the triangles of the projected mesh, shown in Fig. 11. If three vertexes of a triangle are inside or one of the vertexes are on the polygon, this triangle belongs to the hole-patch we require to fill in the hole (Fig. 11. (a) and the red triangles in Fig. 9. (a)). If all the vertexes of the triangle are out, this triangle can be eliminated (Fig. 11. (b) and the pink triangles in Fig. 9. (b)). If one or two points of a triangle are in and the others are out, this triangle can be considered as on the boundary of the hole (Fig. 11. (c) and the white triangles in Fig. 9. (a)(b)).

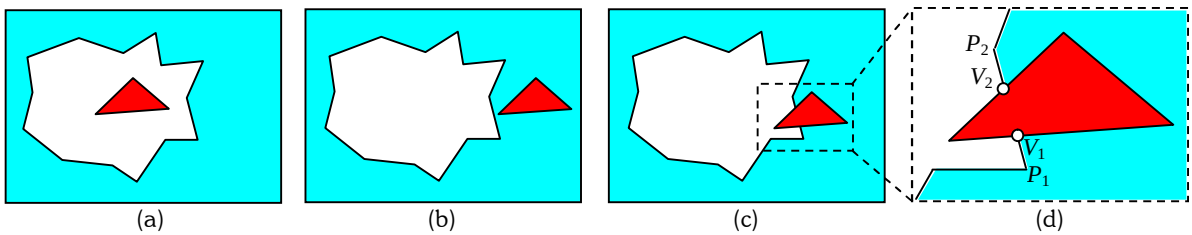


Fig. 11: Three simple relationships between a polygon and a triangle.

To accomplish the blending operation, we only deal with the boundary triangles. For example, in Fig. 11 (c)., P_1P_2 is an edge of the projected hole, and the intersection points with the boundary triangle are V_1 and V_2 , also see the red segments in Fig. 9 (c). Then using the vertexes of the projected hole and the boundary polygon of the inner triangles, we can create the topology of the projected surface patch and the projected holes, demonstrated in Fig. 9. (c). However, as one edge of the projected hole could be broken into one or two segments, it is very complex to find the connectivity between the vertexes of the inner triangles' boundary polygon, the vertexes of the projected hole and the intersection points. In practice, we just utilize the vertexes of the projected hole and the boundary triangles to polygonize the area between of them, illustrated in Fig. 9. (c). When accomplished the above steps, we remap the 2D information into 3D space to stitch the hole patch with the hole boundary of the original mesh.

4. EXPERIMENTAL RESULTS

Based on the aforementioned hole-filling method, we perform lots of experiments to validate the proposed algorithm. From Fig. 12 to Fig. 14, (a) is the mesh with holes, (b) is the repaired result, (c) is the error map between the original mesh and the filled mesh, and (d) is the error analysis between the original mesh and the mesh repaired by one commercial software package. Tab. 2 gives the running time of our hole-filling algorithm, which includes the number of the mesh triangles, the timings of hole-checking, implicit surface reconstruction, surface patch triangulation, and surface blending. The experiments are carried out in the hardware configuration of PIV 2.8GHz and 256RAM. In order to show the accuracy of our scheme, we compute the distance error between the repaired mesh and the original mesh using 3D Comparison of the commercial software package.

Tab. 3 indicates the errors analysis between two meshes. In Tab. 3 *Max_pos dist.* means maximal positive distance; *Min_pos dist.* is minimal positive distance; *Min_neg dist.* stands for minimal negative distance; *Max_neg dist.* is

maximal negative distance; *Standard dev.* Means standard deviation. Tab. 2 shows that the error is relatively small. In our experiments, we utilized the commercial software to fill in the holes of meshes, and then the error distances are computed between the corresponding two meshes.

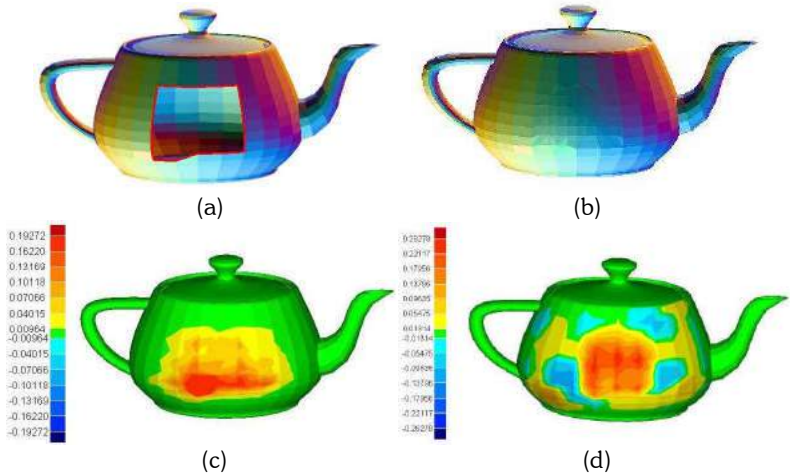


Fig. 12: Hole-filling example I—Teapot. (a) The mesh with a hole; (b) The repaired mesh; (c) The error map between the original mesh and the repaired mesh; (d) The error map between the original mesh and the mesh repaired by commercial software.

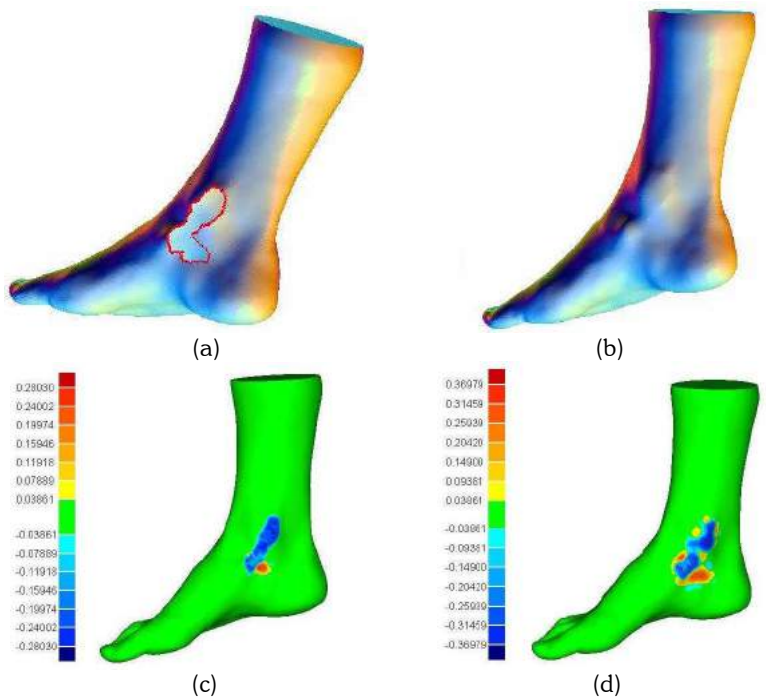


Fig. 13: Hole-filling example II—Foot. (a) The mesh with a hole; (b) The repaired mesh; (c) The error map between the original mesh and the repaired mesh, (d) The error map between the original mesh and the mesh repaired by commercial software.

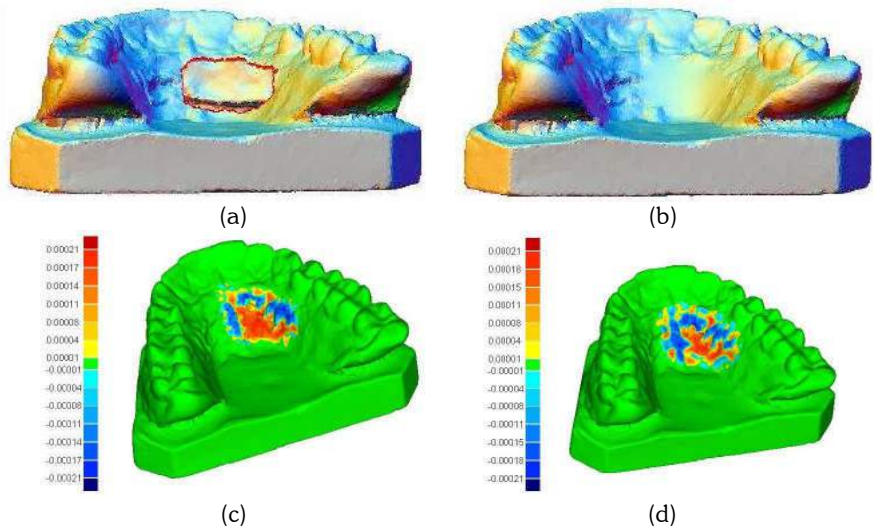


Fig. 14: Hole-filling example III—Teeth. (a) The mesh with a hole; (b) The repaired mesh; (c) The error map between the original mesh and the repaired mesh; (d) The error map between the original mesh and the mesh repaired by commercial software.

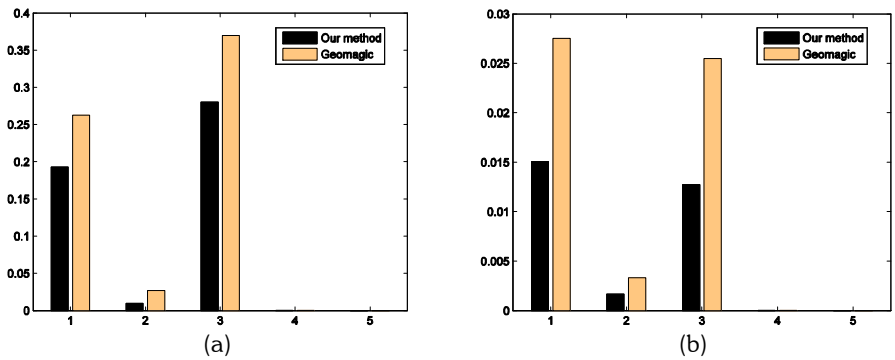


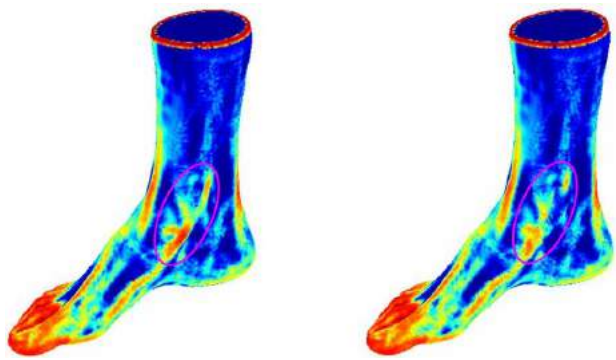
Fig. 15: Error comparison between our algorithm and the commercial software; (a) is the maximal positive distance error comparison; (b) is the standard deviation comparison.

The errors are given in Tab. 4, and it shows that the errors are greater than ours. Furthermore, we draw a bar chart of the errors produced by our algorithm and by the commercial software, see Fig. 15. Fig. 15(a). shows the maximal positive distance errors of our scheme and the commercial software; Fig. 15(b) stands for the standard deviation. From these figures, it is clear that our scheme outweighs the hole-filling algorithm of the commercial software in terms of accuracy. We also use the mean curvature of mesh to demonstrate the deviations of the repaired mesh and the original mesh. Fig. 16 illustrates the mean curvature maps of the original mesh and the repaired mesh. The curvature deviations are in the ellipsoid area where the curvature distribution is almost same with that of the original mesh, which indicates the error is small.

5. CONCLUSIONS

In this paper, we propose a hole-filling algorithm to repair meshes with undesired holes. We employ the interpolation property of RBFs to construct an implicit surface patch. Then, the triangulated surface patch is blended with the hole mesh, which includes the following steps: identification of holes, selection of interpolation centers, reconstruction of the implicit surface patch, blending of the implicit surface patch with the original mesh. In particular, in the blending operation, we convert the point inclusion test from 3D into 2D by projecting the surface patch onto a coordinate plane,

which mitigate the computational complexity. Experimental results validate the efficiency since it operates only in the vicinity of holes and the numerical comparisons with commercial software indicate the accuracy of our method.



(a) (b)

Fig. 16: Mean curvature comparison, the mean curvature distribution is almost same in the ellipsoid area. (a) Mean curvature distribution of the original mesh; (b) Mean curvature of the filled mesh.

Model	*Number of Triangle*	*Hole checking (sec.)*	*Implicit surface (sec.)*	*Polygonization (sec.)*	*Blending (sec.)*	*Ttotal (sec.)*
Teapot	3029	0.27	0.16	0.05	0.47	0.95
Cylinder	3188	0.14	3.34	0.09	3.59	7.13
Foot	50718	1.09	27.22	0.22	30.18	58.71
Teeth	56312	24.06	34.45	2.8	61.34	122.65
Santa	150509	45.39	83.88	105.83	235.63	470.73

Tab. 2: Running time of the hole-filling algorithm.

Model	*Max_pos dist. (mm)*	*Min_pos dist. (mm)*	*Min_neg dist. (mm)*	*Max_net dist. (mm)*	*Standard dev. (mm)*
Teapot	0.1927	0.0096	-0.0096	-0.1927	0.01505
Cylinder	0.0097	0.0005	-0.0005	-0.0097	0.00165
Foot	0.2803	0.0386	-0.0386	-0.2803	0.01274
Teeth	0.0002	1e-5	-1e-5	-0.0002	1e-5
Santa	0.0001	1e-5	-1e-5	-0.0001	3e-6

Tab. 3: Error analysis between the repaired mesh and the original mesh.

Model	*Max_pos dist. (mm)*	*Min_pos dist. (mm)*	*Min_neg dist. (mm)*	*Max_net dist. (mm)*	*Standard dev. (mm)*
Teapot	0.2628	0.0131	-0.0131	-0.2628	0.02754
Cylinder	0.0269	0.0013	-0.0013	-0.0269	0.00333
Foot	0.3697	0.0386	-0.0386	-0.3697	0.02549
Teeth	0.0002	1e-5	-1e-5	-0.000	1e-5
Santa	6e-5	0	0	-6e-5	2e-6

Tab. 4: Errors between the meshes filled by the commercial software and the original mesh.

However, there is also a limitation of the algorithm. That is the projected implicit surface patch should not contain any folds in the direction of the projection. It means that the reconstructed surface patch is monotony in the projection direction. To find more general method to deal with holes of arbitrary shapes is our future work.

6. REFERENCES

[1] Pernot, J. P.; Moraru, G.; Veron, P.: Filling holes in meshes using a mechanical model to simulate the curvature variation minimization, Computers & Graphics, 30(6), 2006, 892-902.

[2] Liepa, P.: Filling holes in meshes, Eurographics Symposium on Geometry Processing, 2003, 200-205.

[3] Schneider, R.; Kobbelt, L.: Geometric fairing of irregular meshes for free-form surface design, Computer-Aided Geometric Design, 18(4), 2001, 359-374.

[4] Desbrun, M.; Meyer, M.; Schröder, P.; Barr, A. H.: Implicit fairing of irregular meshes using diffusion and curvature flow, In Proceedings of ACM SIGGRAPH, 33, 1999, 317-324.

[5] Wang, J. N.; Oliveira, M. M.: A hole-filling strategy for reconstruction of smooth surfaces in range images, XVI Brazilian Symposium on Computer Graphics and Image Processing, 2003, 11-18.

[6] Tekumalla, L. S.; Cohen, E.: A hole-filling algorithm for triangular meshes, Technical Report UUCS-04-019, School of Computing, University of Utah, USA, http://www.cs.utah.edu/techreports/, 2004.

[7] Levoy, M.; Curless, B.: A volumetric method for building complex models from range images, In Processing of SIGGRAPH 1996, 1996, 303–312.

[8] Davis, J.; Marschner, S. R.; Garr, M.; Levoy, M.: Filling holes in complex surfaces using volumetric diffusion, In Processing of the first international symposium on 3D data, visualization and transmission, 2002.

[9] Nooruddin, F. S.; Turk, G.: Simplification and repair of polygonal models using volumetric techniques, IEEE Transactions on Visualization and Computer Graphics, 9(2), 2003, 191-205.

[10] Clarenz, U.; Diewald, U.; Dziuk, G.; Rumpf, M.; Rusu, R.: A finite element method for surface restoration with smooth boundary conditions, Computer Aided Geometric Design, 21(5), 2004, 427-445.

[11] Casciola, G.; Lazzaro, D.; Montefusco, L. B.; Morigi, S.: Fast surface reconstruction and hole filling using radial basis functions, Numerical Algorithms, 39(1-3), 2005, 289-305.

[12] Podolak, J.; Rusinkiewicz, S.: Atomic volumes for mesh completion, Eurographics Symposium on Geometry Processing, 2005, 33-41.

[13] Bors, A. G.: Introduction of the radial basis function (RBF) networks, http://axiom.anu.edu.au/~daa/courses/GSAC6017/rbf.pdf.

[14] Kansa, E. J.: Motivation for using radial basis functions to solve PDEs, http://rbf-pde.uah.edu/kansaweb.pdf.

[15] Carr, J. C.; Beatson, R. K.; Cherrie, J. B.; Mitchell, T. J.; Fright, W. R.; McCallum, B. C. and Evans, T. R.: Reconstruction and representation of 3d objects with radial basis functions, Computer Graphics Proceedings, Annual Conference Series, 2001, 67-76.

[16] Turk, G.; O'Brien, J. F.: Modelling with implicit surfaces that interpolate, ACM Transactions on Graphics, (SIGGRAPH 2002), 21(4), 2002, 855-873.

[17] Reuter, P.: Reconstruction and rendering of implicit surfaces from large unorganized point sets, PhD Thesis, LaBRI-Université Bordeaux 1, 2003.

[18] Wendland, H.: Piecewise polynomial, positive definite and compactly supported radial basis functions of minimal degree, Advances in Computational Mathematics, 1995, 389-396.

[19] Li, W. S.; Ong, E. T.; Xu, S. H.; Hung, T.: A point inclusion test algorithm for simple polygons, Lecture Notes in Computer Science, International Conference on Computational Science and Its Applications, ICCSA 2005, 3480, 2005, 769-775.

[20] Wu, H. Y.; Gong J. Y.; Li D. R; Shi W.: An algebraic algorithm for point inclusion query, Computers & Graphics, 2000, 24(4), 2000, 517-522.

[21] Duchon, J.: Splines minimizing rotation-invariatn semi-norms in Sobolev spaces, In Schempp, W. and Zeller, K., editors, Constructive Theory of Functions of Several Variables, number 571 in Lecture Notes in Mathematics, Springer-Verlag, 1977, 85-100.