Eurographics Symposium on Geometry Processing(2003)
L. Kobbelt, P. Schröder, H. Hoppe (Editors)

Filling Holes in Meshes

Peter Liepa

Alias | Wavefront

Abstract

We describe a method for filling holes in unstructured triangular meshes. The resulting patching meshes interpolate the shape and density of the surrounding mesh. Our methods work with arbitrary holes in oriented connected manifold meshes. The steps in filling a hole include boundary identification, hole triangulation, refinement, and fairing.

Categories and Subject Descriptors: I.3.5 [Computer Graphics] Computational Geometry and Object Modeling - surface and object representations.

1. Introduction

During the surface reconstruction process, a mesh is calculated from a cloud of points. This mesh may have holes corresponding to deficiencies in the original point data, or to regions in which for one reason or another the meshing algorithm did not create a mesh. It is often desirable to fill in these holes for reasons ranging from aesthetics to the need for a watertight mesh for STL (stereo lithography) prototyping.

A satisfactory hole filling method should:

1. be automatic. If the user is to select holes, the interaction should be as simple as possible.

2. run in reasonable time, preferably at interactive speeds

3. fill a hole with a patching mesh that is minimally distinguishable from the surrounding mesh. In particular, mesh density and shape should match across the original hole boundary.

4. be able to deal with arbitrary holes in arbitrary meshes. For example, arbitrary meshes may include non-manifold edges, and arbitrary holes may be extremely irregular and non-planar in their outline. Our philosophy is similar to that of Held[7], in that we will always attempt to generate a meaningful patch whose correctness will gracefully degrade as the hole becomes more pathological.

The approach described here aims to meet these goals. We will discuss how successful we are in Section 7.

Some existing non-geometric approaches to hole filling come from the surface reconstruction literature. Carr et al.[3] use radial basis functions to compute thin-plate interpolations of holes in scattered height data, a technique that should be extendable to variational implicit surfaces[15]. Szeliski and Tonnesen[13] use oriented particles to extend and interpolate sparse 3D data. These techniques can be extended to meshes. Geometric hole filling is avoided by Curless and Levoy[4] by using a volumetric method of filling holes in voxel space. Davis et al.[5] use volumetric diffusion to extend a clamped distance function into space, and then find the zero set of the resulting distribution.

Two existing geometric approaches to hole filling influenced the techniques used in this paper. Barequet and Sharir[1] use a dynamic programming method to find a minimum area triangulation of a 3D polygon in order to fill mesh holes. Pfeifle and Seidel[9] use mesh refinement and fairing techniques to fill holes in piecewise polynomial surfaces, after finding a spanning triangulation in 2D (parametric) space. A cousin of the hole filling problem is the Plateau problem, which seeks a minimal disk-like surface that spans a given closed 3D curve[10]. The main difference is that the discrete Plateau problem does not include the constraints given by a surrounding mesh. A method for fairing a mesh region so that it is discrete G1 continuous with its surrounds is given by Schneider and Kobbelt[11].

In the literature we have surveyed, existing approaches to discrete Plateau and fairing problems do not address the problem of finding an initial spanning mesh, but rather assume this as a given. In fact, to the best of our knowledge, there is only one description of general 3D polygon triangulation in the literature[1]. Other hole triangulation algorithms that have been proposed in the context of vertex removal, for example, have depended on

simplifying assumptions such as star-shape or projectability to a plane[12,14].

The main contribution of this paper is to give a complete account of a geometric method for filling holes in triangular meshes. The main stages of this method are: hole identification, hole triangulation, mesh refinement, and mesh fairing. An existing hole triangulation algorithm[1] is generalized and improved to overcome difficulties due to crenellations. 2D mesh refinement[9,16] is generalized here to 3D.

2. Preliminaries

A *triangular mesh* is defined as a set of vertices and a set of oriented triangles that join these vertices. Two triangles are *adjacent* if they share a common edge. Adjacent triangles are *consistently oriented* if their borders traverse their common edge in opposite directions. The *dihedral angle* between two consistently oriented adjacent triangles is the angle between the triangle normals. A mesh is *oriented* if all of its triangles are consistently oriented. An edge is *adjacent* to a triangle if it is part of the border of that triangle.

A *boundary edge* is an edge adjacent to exactly one triangle. A *boundary vertex* is a vertex that is adjacent to a boundary edge. A *hole* is a closed cycle of boundary edges. A *singular vertex* is a vertex that has more than two adjacent boundary edges. A *non-manifold edge* is an edge that has more than two adjacent triangles. An *interior edge* is an edge that is not a boundary edge. A *manifold* mesh has no non-manifold edges and no singular vertices, but may have boundaries.

In the hole filling process, we refer to the original mesh as the *surrounding mesh* and to the mesh that fills the hole as the *patching mesh*.

We will assume that all meshes, unless otherwise noted, are oriented, manifold, and connected. In particular, two separate holes will have no vertices in common (otherwise those would be singular vertices), and a given hole will not have islands (otherwise the mesh would not be connected).

Note that although the definition of a hole is entirely topological, there are examples that do not conform to our conventional notion of a hole. In Figure 1 each jagged edge is a boundary of its respective mesh and therefore a hole in the sense of this paper. However, in practice, the notion of what constitutes a hole (more specifically whether a hole needs to be filled) will be application or user dependent.

Geometric information can help in distinguishing between the different kind of holes in Figure 1. For simplicity we will assume that the hole boundary contains no singular vertices and has no adjacent non-manifold edges. Each vertex v on the hole has a set of adjacent edges $\{e_1,..,e_n\}$, in the order of walking the fan of triangles adjacent to v. The *surface angle* at v is $\sum_{0<i<n} \angle(e_i, e_{i+1})$. The *total surface angle* for a hole boundary is the sum of the vertex surface angles. In Figure 1 it is easy to compute that the jagged holes in shapes (a), (c) and (e) are $\pi(n+2)$, πn, and $\pi(n-2)$, where n is the number of boundary vertices. In general, the larger the total surface angle of its boundary relative to the number of boundary vertices, the more likely it is that a hole will be judged to be a hole in the conventional sense.

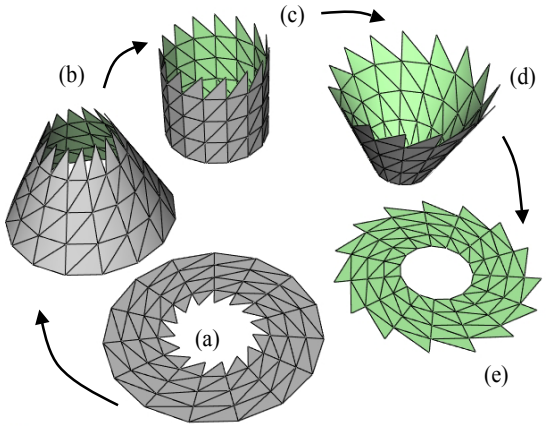

Figure 1: *The jagged hole in the annulus at (a) is continuously deformed into the jagged outside boundary of the annulus at (e). At some point between (c) and (e) the jagged boundary ceases to become a hole in the conventional sense of the word but remains a hole as defined in this paper.*

3. Hole Identification

The first step in filling a hole is identifying it. Holes can be identified automatically by looking for boundary vertices, or can be easily indicated by a user by clicking on a representative edge or vertex. Requiring a user to, for example, lasso-select a hole can be onerous for holes that cannot be viewed from a single position. Given a seed boundary vertex, an application can trace a series of boundary edges until it identifies a closed loop of boundary edges.

4. Hole Triangulation

Once a hole is identified, the first step in filling it is to find a triangulation of the three-dimensional polygon defined by that boundary. Barequet et al[2] give an interesting overview of the problem of triangulating three-dimensional polygons, and show that the problem of determining whether a 3D polygon has a triangulation that is not self-intersecting and that defines a simply-connected 2-manifold is NP-complete. In this paper we concern ourselves with the lesser problem of finding weight-minimizing (but possibly self-intersecting) triangulations.

An existing $O(n^3)$ algorithm[1] for triangulating a 3D polygon is extended here. A *weightset* L is an ordered set with an addition operation and a distinguished element 0_L that is both the minimal element and the additive identity. A *weight* is an element of a weightset.

For a sequence of vertices $V = \{v_0, v_1, ..., v_{n-1}\}$, $v_i \in \mathbf{R}^3$, let $P = (v_0, v_1, ..., v_{n-1})$ be the associated 3D polygon. Define a weight function $\Omega : V^3 \to L$, where L is a weightset and Ω assigns a weight to each triangle whose vertices are in V. For $0 \le i \le j < n$, let $W_{i,j}$ be the weight of the minimum-weight triangulation of the sub-polygon $(v_i, ..., v_j)$. We calculate minimum weights for polygons with increasing numbers of vertices, starting from polygons of 3 vertices (i.e. triangles) using the following

Triangulation Algorithm.

1. For $i = 0,1,...,n-2$, let $W_{i,i+1} \leftarrow 0_L$ and for $i = 0,1,...,n-3$, let $W_{i,i+2} \leftarrow \Omega(i, i+1, i+2)$. Set $j \leftarrow 2$.

2. Put $j \leftarrow j+1$. For $i = 0,1,...,n-j-1$ and $k \leftarrow i+j$, compute $W_{i,k} \leftarrow \min_{i<m<k} [W_{i,m} + W_{m,k} + \Omega(v_i, v_m, v_k)]$. Record the index m where the minimum is achieved as $\lambda_{i,k}$.

3. If $j < n-1$, go back to step 2. Otherwise the weight of the minimum-weight triangulation is $W_{0,n-1}$.

4. Recover the triangulation using the $\lambda_{i,k}$ values recorded in step 2 (see Barequet and Sharir[1] for details).

Define the weightset

$$L_{area} = [0, \infty), \text{ with } 0_{L_{area}} = 0 .$$

If we use the weight function

$$\Omega_{area}(v_i, v_m, v_k) = \text{Area of } (v_i, v_m, v_k) ,$$

the Triangulation Algorithm will yield a minimum area triangulation. (This weighting, modified to avoid skinny triangles, is used by Barequet and Sharir[1]).

Minimum area triangulations are acceptable when holes are relatively planar, but begin to go wrong when the plane of the triangulation is orthogonal to features in the boundary (see Figure 2(a), (b)). These features are called *crenellations*, after the teeth at the top of a castle wall (or of a chess rook). A minimum area triangulation will include triangles that duplicate crenellation triangles in the surrounding mesh, thus creating sharp folds and non-manifold edges (see Figure 2(c)).

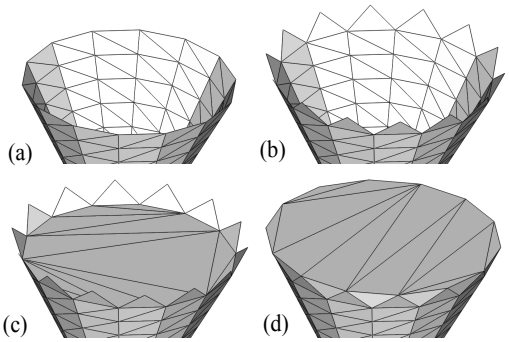

Figure 2: *(a) hole with no crenellations. (b) hole with crenellations. (c) minimum-area fill. The fill duplicates the crenellation triangles and creates non-manifold edges along the topmost horizontal edges. (d) min-max dihedral angle fill.*

To avoid this, we introduce a new weighting on triangles that takes into account their dihedral angles with adjacent triangles. The weights are ordered pairs (angle, area) belonging to a new weightset

$$L_{angle} = [0, \pi] \times [0, \infty), \text{ with } 0_{L_{angle}} = (0,0) .$$

The ordering in L_{angle} is designed to give precedence to dihedral angles over areas, and to penalize large dihedral angles:

$$(a,b) < (c,d) \text{ iff } (a < c \text{ or } (a = c \text{ and } b < d)).$$

The addition operator sums area but retains the "worst" (i.e., largest) dihedral angle:

$$(a,b)+(c,d) := (\max(a,c), b+d).$$

We can now define the weighting function $\Omega_{angle} : V^3 \rightarrow L_{angle}$

$$\Omega_{angle}(v_i,v_m,v_k) = (\mu(v_i,v_m,v_k), \Omega_{area}(v_i,v_m,v_k)),$$

where $\mu(v_i,v_m,v_k)$ is the maximum dihedral angle between (v_i,v_m,v_k) and existing adjacent triangles. The Triangulation Algorithm, using this weighting, will yield the triangulation that minimizes the maximum dihedral angle (see Figure 2(d)).

Note that $\mu(v_i,v_m,v_k)$ depends on existing adjacent triangles. These are taken to be the adjacent triangles that exist at the time of evaluation. In Step 1, $\mu(i,i+1,i+2)$ is calculated against the triangles adjacent to edges $(i,i+1)$ and $(i+1,i+2)$ in the surrounding mesh. In Step 2, $\mu(v_i,v_m,v_k)$ is calculated against the triangles $(v_i,v_{\lambda_{i,m}},v_m)$ and $(v_m,v_{\lambda_{m,k}},v_k)$. When $(i,k)=(0,n-1)$ the surrounding mesh triangle adjacent to edge (v_0,v_{n-1}) is also taken into account.

The effect of the Ω_{angle}-triangulation is to compute a kind of convex hull of the surrounding mesh before attempting to find a minimum-area spanning surface.

5. Mesh Refinement

We wish to fill a hole with a mesh that approximates the density of the surrounding mesh. Given the spanning triangulation computed in Section 4, we can refine the triangulation into a mesh by adapting an algorithm given by Pfeifle and Seidel[9]. The basic idea is to compute edge length data for the vertices on the hole boundary and diffuse these values into the interior of the patching mesh, subdividing triangles to reduce edge lengths, and relaxing interior edges to maintain a Delaunay-like triangulation.

To *relax* an edge means, for the two triangles adjacent to the edge, to check that each of the two non-mutual vertices of these triangles lies outside of the circum-sphere of the opposing triangle (see Figure 3). If this test fails, the edge is swapped.

Given a density control factor α and a triangulation for the hole, we use the following

Refinement Algorithm:

1. For each vertex v_i on the hole boundary, compute the scale attribute $\sigma(v_i)$ as the average length of the edges that are adjacent to v_i in the surrounding mesh. Initialize the patching mesh as the given hole triangulation.
2. For each triangle (v_i,v_j,v_k) in the patching mesh, compute the centroid v_c and the corresponding scale attribute $\sigma(v_c) \leftarrow (\sigma(v_i)+\sigma(v_j)+\sigma(v_k))/3$. For $m=i,j,k$, if $\alpha\|v_c - v_m\| > \sigma(v_c)$ and $\alpha\|v_c - v_m\| > \sigma(v_m)$, then replace triangle (v_i,v_j,v_k) with triangles (v_c,v_j,v_k), (v_i,v_c,v_k), and (v_i,v_j,v_c) in the patching mesh and relax the edges (v_i,v_j), (v_i,v_k), and (v_j,v_k).
3. If no new triangles were created in Step 2, the patching mesh is complete.
4. Relax all interior edges of the patching mesh.
5. If no edges were swapped in Step 4, go to Step 2. Otherwise go to Step 4.

We found empirically that a density control factor $\alpha=\sqrt{2}$ yields a patching mesh that visually matches the density of the surrounding mesh.

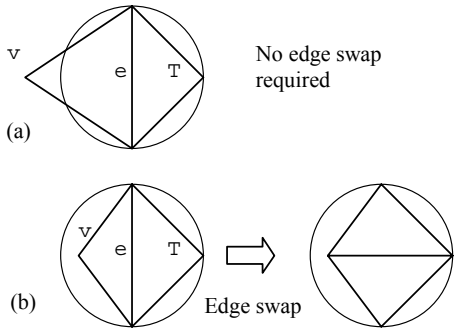

Figure 3: *Edge relaxation. If, for an edge e the vertex v is inside the circum-sphere of triangle T, the edge is swapped.*

6. Fairing

Fairing is the process of making a surface smooth, usually by minimizing a fairness functional. We generalize the

method described by Kobbelt et al.[8] to shape the patching mesh that was calculated in Section 5.

Let $\omega : V^2 \rightarrow \mathbf{R}$ be a weighting function that assigns a weight to every edge (v_i, v_j). For every vertex v, define the *weighted umbrella-operator*

$$\mathbf{U}_\omega(v) = -v + \frac{1}{\omega(v)} \sum_i \omega(v, v_i) v_i ,$$

where the v_i are the direct neighbors of v, and $\omega(v) = \sum_i \omega(v, v_i)$.

When $\omega(v_i, v_j) = 1$ for all v_i, v_j the corresponding $\mathbf{U}_\omega$ is called the *uniform umbrella-operator*. A popular method of smoothing meshes is to replace each vertex v with $v + \mathbf{U}_\omega(v)$, which has the effect of replacing each vertex with the average of its immediate neighbors. When $\omega(v_i, v_j) = 1/\|v_i - v_j\|$, we obtain the *scale-dependent umbrella-operator*, which introduces less global shape distortion when used to smooth irregular meshes[6]. For edges (v_i, v_j) that have exactly two adjacent triangles (v_i, v_{k1}, v_j) and (v_i, v_{k2}, v_j), we can define

$$\omega(v_i, v_j) = \cot(\angle(v_i, v_{k1}, v_j) + \cot \angle(v_i, v_{k2}, v_j) ,$$

which induces the *harmonic umbrella-operator*, which also tends to preserve triangle shape[6].

The umbrella-operator can be applied recursively to get a *second-order weighted umbrella operator*:

$$\mathbf{U}^2{}_\omega(v) = -\mathbf{U}_\omega(v) + \frac{1}{\omega(v)} \sum_i \omega(v, v_i) \mathbf{U}_\omega(v_i) .$$

For manifold surfaces, if $\mathbf{U}_\omega(v)$ measures the deviation of a vertex v from a taut surface bounded by its neighbors, then $\mathbf{U}^2{}_\omega(v) = 0$ implies that the deviation from tautness at a vertex v is equal to the average deviation from tautness of its neighbors. Given a patching mesh, we fair it by arranging the non-boundary vertices v_i of this mesh so that $\mathbf{U}^2{}_\omega(v_i) = 0$. This amounts to solving a linear system. The linear system is sparse, and although it is not strictly symmetric, the Conjugate Gradient method has given good results. These are shown in Figure 6 (see color section). In this case, scale-dependent fairing yields a more pleasing shape than uniform fairing. Schneider and Kobbelt's discrete curvature diffusion[11] gives even more pleasing results, but is not unconditionally stable. See Figure 4 through Figure 8 (some of these are in the color section) for more examples of the entire hole filling process, from triangulation to meshing and fairing.

7. Discussion

The exposition in this paper was confined to triangular meshes, but the techniques extend easily to meshes that contain arbitrary polygons.

Another interesting extension of the techniques in this paper would be to fill gaps between surfaces, or holes with islands. We envision the use of line segments (either automatically calculated or user defined) to create bridges between disconnected boundaries (a 2D version of this is described by Held[7]). These line segments are used in the hole identification and triangulation phases, but can be ignored for the refinement and fairing phases.

In Section 1 we outlined several goals for a good hole filling method. In practice, we have found the methods described here to be fast and robust in the context of filling holes in meshes reconstructed from clouds of points. But due to the $O(n^3)$ performance of the Triangulation Algorithm, holes with boundaries consisting of hundreds of edges can takes minutes as opposed to seconds. And although we set a goal of filling arbitrary holes in arbitrary meshes, we have confined our discussion to holes in oriented connected manifold meshes. Our software, however, deals with non-manifold meshes, where the main difference is that hole detection becomes more difficult. And it could deal with holes with islands by computing bridges. As mentioned earlier in a comparison with Held[7], these extensions lead to computed patches whose quality degrades with the pathology of the hole.

As for the goal of computing a patch that is compatible with the surrounding mesh, since our patch tends to be an isotropic triangular mesh, it does not blend well visually with anistropic meshes such as triangulated grids, or meshes with predominantly non-equilateral triangles.

Although the methods discussed here were developed for surface reconstruction applications, they can be generally useful in mesh processing objectives such as feature deletion and region replacement, and construction of surfaces from feature curve networks. We also note that Figure 8 (see color section) suggests that because hole filling is able to approximately reconstruct a mesh from partial information, it might be of use in mesh compression.

Acknowledgements

Thanks to Paul Besl for helpful discussions, to Jonathan Brown for help with an illustration, to Gerry Hawkins for allocating the time and resources for this work, to the Alias|Wavefront Spider team for providing the framework in which these methods were implemented, and to various anonymous reviewers for helpful comments. The bunny model is from the Stanford 3D Scanning Repository.

References

1. Gill Barequet and Micha Sharir. Filling Gaps in the Boundary of a Polyhedron. Computer-Aided Geometric Design, 12(2):207-229, March 1995.
2. G. Barequet, M. Dickerson, and D. Eppstein. On triangulating three-dimensional polygons. *Computational Geometry: Theory and Applications*, 10(3):155-170, June 1998.
3. J. C. Carr, W. R. Fright and R. K. Beatson. Surface Interpolation with Radial Basis Functions for Medical Imaging. *IEEE Transactions on Medical Imaging*, 16(1):96-107, February 1997.
4. Brian Curless and Marc Levoy. A Volumetric Method for Building Complex Models from Range Images**.** *SIGGRAPH 96 Conference Proceedings*, August 1996.
5. James Davis, Stephen R. Marschner, Matt Garr, and Marc Levoy. Filling holes in complex surfaces using volumetric diffusion. First International Symposium on 3D Data Processing, Visualization, and Transmission
   Padua, Italy, June 19-21, 2002.
6. Mathieu Desbrun, Mark Meyer, Peter Schröder and Alan H. Barr. Implicit Fairing of Irregular Meshes using Diffusion and Curvature Flow. *SIGGRAPH 99 Conference Proceedings*, pages 317-324.
7. M. Held. FIST: Fast Industrial-Strength Triangulation of Polygons. *Proc. Comput. Graphics Int. '98*, pages 633-643; Hannover, Germany, June 1998.
8. Leif Kobbelt, Swen Campagna, Jens Vorsatz, and Hans-Peter Seidel. Interactive Multi-Resolution Modeling on Arbitrary Meshes. *SIGGRAPH 98 Conference Proceedings*.
9. R. Pfeifle and H.-P. Seidel. Triangular B-Splines for Blending and Filling of Polygonal Holes. *Proc. Graphics Interface '96*, pages 186-193, Morgan Kaufmann Publishers, 1996.
10. Ulrich Pinkall and Konrad Polthier. Computing Discrete Minimal Surfaces and Their Conjugates. *Experimental Mathematics* 2(1):15-36, 1993
11. Schneider, R., and Kobbelt, L. Geometric fairing of irregular meshes for free-form surface design. *Computer-Aided Geometric Design*, 18(4):359-379, May 2001.
12. William J. Schroeder, Jonathan A. Zarge, William E. Lorenson. Decimation of Triangle Meshes. *SIGGRAPH 92 Conference Proceedings*.
13. R. Szeliski and D. Tonnesen. Surface Modeling with Oriented Particle Systems. *SIGGRAPH 92 Conference Proceedings*.
14. Greg Turk. Re-Tiling Polygonal Surfaces. *SIGGRAPH 92 Conference Proceedings.*
15. Greg Turk and James F. O'Brien. Variational Implicit Surfaces. *Tech Report GIT-GVU-99-15*, Georgia Institute of Technology, May 1999.
16. N.P.Weatherill and O.Hassan. Efficient Three-Dimensional Delauney Triangulation with Automatic Point Creation and Imposed Boundary Constraints. *International Journal for Numerical Methods in Engineering*, 37(12):2005-2039 (1994).

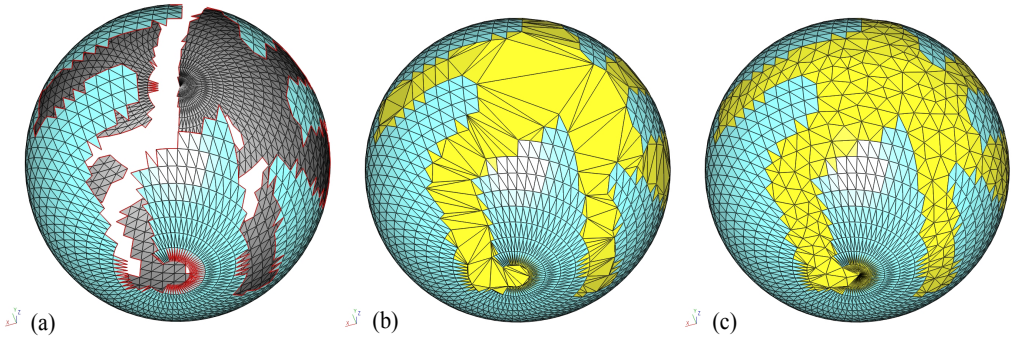

Figure 4: *(a) An example of a complex hole. Note that the hole almost "circumnavigates" the sphere. (b) Min-max dihedral angle triangulation. (c) Refined and faired hole patch.*

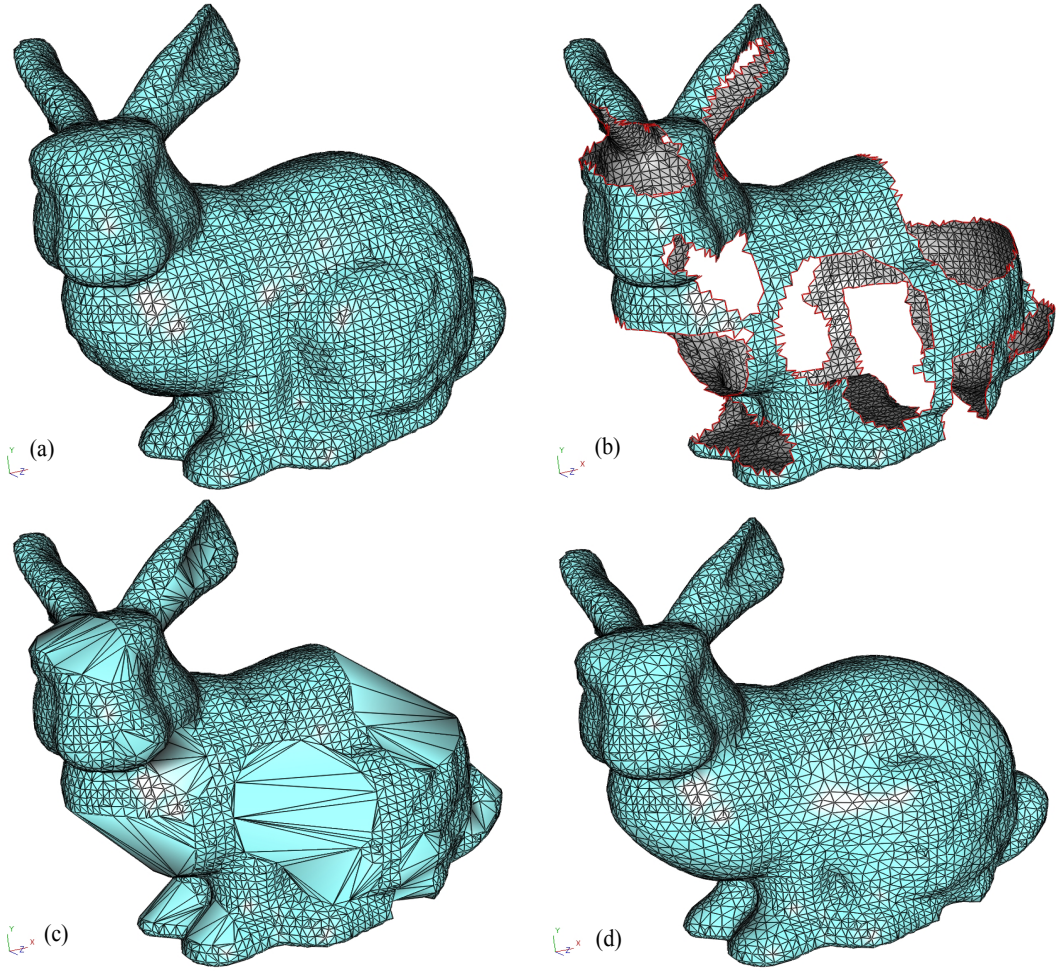

Figure 5: *(a) Stanford bunny. (b) Mutilated Stanford bunny. (c) After hole triangulation. (d) After meshing and fairing. (b) and (d) are reproduced in the color section in Figure 8.*

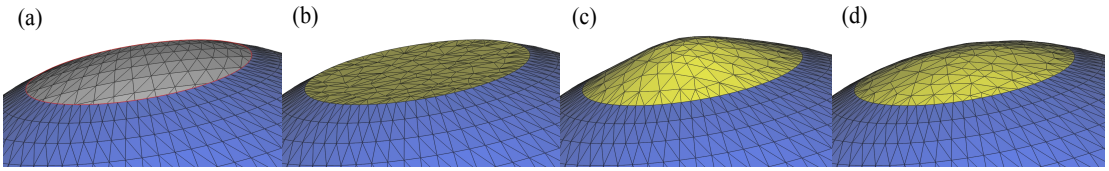

Figure 6: *(a) Hole at the top of a sphere (inside of sphere is gray). (b) Patching mesh (yellow), before fairing. (c) Patching mesh (yellow) after uniform fairing. (d) Patching mesh (yellow) after scale-dependent fairing.*

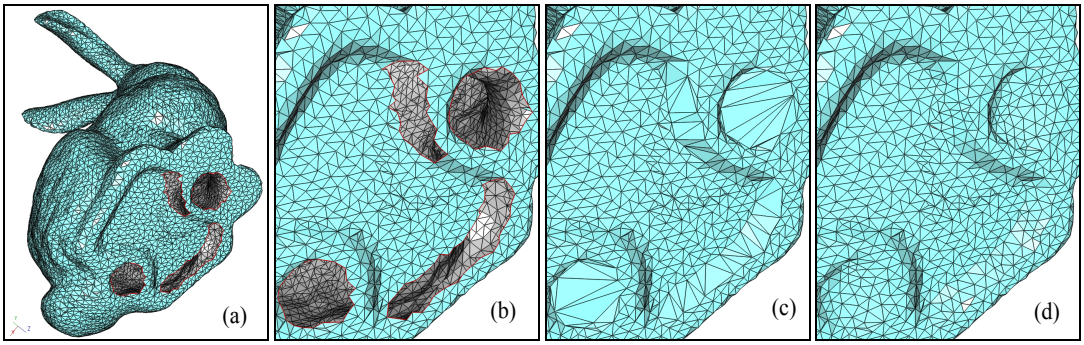

Figure 7: *(a) Four of the five holes in the Stanford bunny. (b) Close-up of holes. (c) Triangulated. (d) Meshed and faired.*

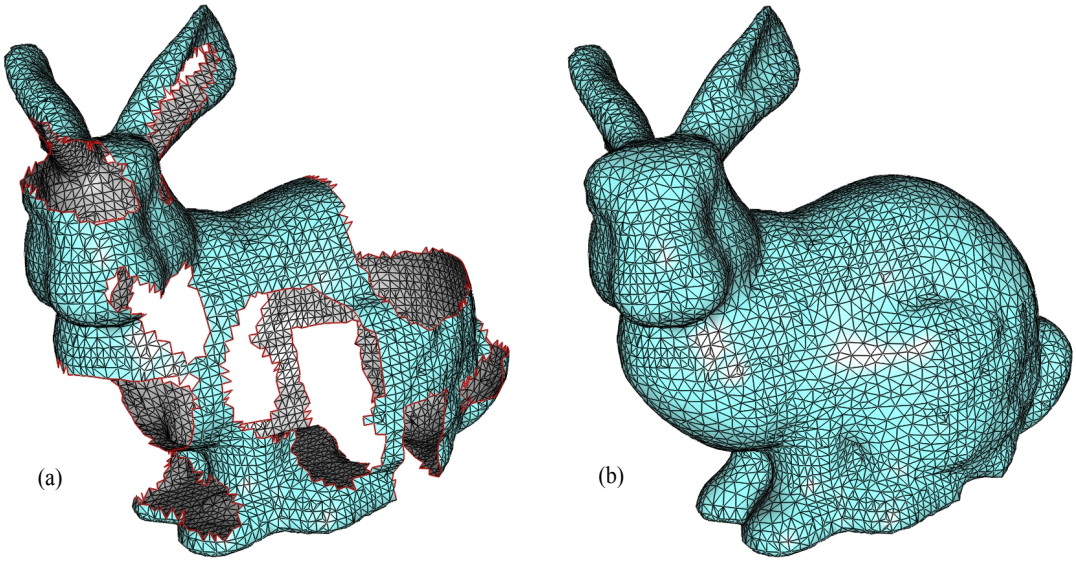

Figure 8: *(a) Mutilated Stanford bunny. (b) After hole triangulation, meshing and fairing.*