Converting Sets of Polygons to Manifold Surfaces by Cutting and Stitching

Andre Gueziec, Gabriel Taubin, Francis Lazarus, William Horn
IBM T.J. Watson Research Center
P.O. Box 704
Yorktown Heights, New York 10598 USA
{gueziec,taubin,francis,hornwp}@watson.ibm.com

A manifold polygonal surface is such that the neighborhood of every vertex can be continuously deformed to a disk (to a half disk at the boundary). This corresponds to an intuitive "surface" definition, as opposed to an arbitrary collection of polygons. Topological degeneracies can occur by design choice (e.g., vertex merging to avoid duplicating coordinates, or polygon reduction tools), or they can be produced by incorrect algorithms for building surfaces (e.g., iso-surfaces, triangulation of scattered points) or by correct algorithms containing software bugs, etc.

Concrete examples of algorithms that can fail on input containing topological singularities are: surface subdivision, surface simplification, surface compression, progressive transmission of geometry, etc. Possible approaches include: modifying algorithms to handle non-manifold input; locating the source of errors in modeling or CAD packages, and lobbying (and hoping) for such errors to be corrected; and developing methods to correct the input. The first approach is application-dependent and probably requires re-defining objectives (beyond just accepting non-manifold input: e.g., a number of surface-simplification methods accept non-manifolds, but often they introduce many more degeneracies than those originally present.) The second approach has little short-term impact and may not be a complete solution in the long term as well (there will always be software bugs).

We have chosen the third approach and developed an automated method for correcting topological singularities. The algorithm manipulates the polygon vertex indices (surface topology) and essentially ignores vertex coordinates (surface geometry). This differentiates our method from most of the previous work. Also, we do not assume that the method works with solids. We assume that the topology of the surface is already built for the most part. However, we provide a stitching method ("edge snapping") to help build the topology from disconnected polygons. Except for the optional stitching, the algorithm has a linear complexity in the number of vertices, edges, and faces, and requires no floating point operation.

Our algorithm comprises two steps: cutting and stitching. Cutting consists of disconnecting the surface along a collection of marked edges or vertices. Multiple copies of vertices are created and assigned new indices. Vertex indices in faces are modified to refer to the proper copy. We mark singular edges and vertices (an edge is singular if more than two faces are incident to it; a singular vertex is such that several edge-connected fans of triangles are incident to it. We propose two different methods for cutting: a local method and a global method. The local method operates only on marked vertices and endpoints of marked edges, by counting the number of (unmarked) edge-connected sets of faces incident to a vertex and by duplicating the vertex and assigning a different copy of that vertex to each connected set. The global method operates on all the faces and vertices of the surface at once, by first breaking all connections between faces and later joining adjacent faces that share an unmarked edge. The global method is more appropriate when there is a large number of topological singularities to correct. The local method is more efficient when there are only few singular elements in a generally correct topology.

Several manifolds can be mapped to the original non-manifold by identifying vertices. To reduce the number of vertices, holes, or components, the cutting operation is followed by stitching. Stitching consists of taking two boundary edges and identifying them. We propose two different stitching strategies called "edge pinching" and "edge snapping." Edge pinching consists of stitching adjacent edges on one boundary where a cut was previously made. Edge snapping consists of stitching edges whose endpoints are within an epsilon distance. Stitching can be a delicate operation: singular edges can be created when stitched multiple times. We avoid this by testing that after each proposed edge stitch, none of the affected vertices becomes singular.

To provide a real-world example, we consider a polygonal model of the space ship Enterprise with 12,539 triangles and 15,011 vertices. Figure1 shows a global view, where disconnected surface components are painted with different colors. We discovered 594 singular edges and 1,878 singular vertices. Figure 2 shows a detail with singular edges painted in red, boundary edges in green, and regular edges in yellow. After removing degenerate triangles (Figure 3), there are 435 singular edges and 1,689 singular vertices left. After cutting (Figure 4) and edge snapping (Figure 5), there are 12,552 triangles and 7,429 vertices. This example is a good advocate for automated correction methods: asking a user to decide on how to correct the surface locally 1,800 times seems impractical.

References
Murali, T.M. and Funkhouser, T.A. Consistent solid and boundary representations from arbitrary polygonal data. 1997 ACM Symposium on Interactive 3D Graphics, 155-161.

Barequet, G. and Kumar, S. Repairing CAD models. IEEE Visualization 1997.

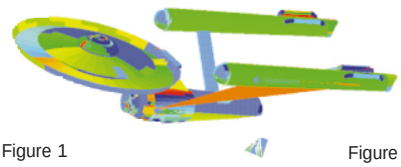

Figure 1

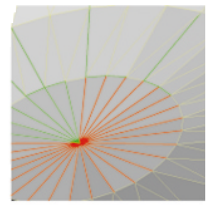

Figure 2

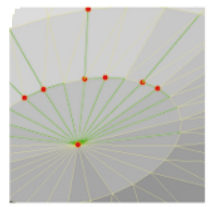

Figure 3

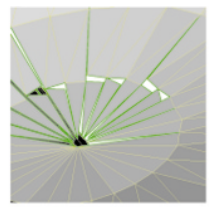

Figure 4

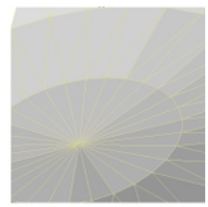

Figure 5