Mesh Repair with Winding Numbers

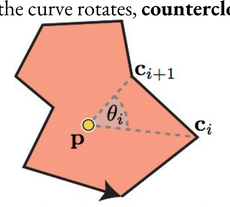
Joy Ding, Athina Panotopoulou, Emily Whiting
GROW, BU, Commonwealth School

Abstract

Complex meshes, used for representing 2D and 3D objects in computer graphics, often contain problematic anomalies like self-intersections and holes. These problems often prevent further processing of the mesh for rendering and other purposes. Given an input mesh description, this project repairs the mesh so that the final product is both consistent with the input shape and free of any topological anomalies. For the initial assessment of the input, the program uses winding numbers, which determine the inside or outside location of any point, to identify gaps and overlapping regions. The program then segments the mesh using the winding number as a threshold to generate a final, repaired mesh. Drawing from methods described in [1], this program is able to create accurate volume-based meshes from surface descriptions of objects with topological inconsistencies.

Introduction

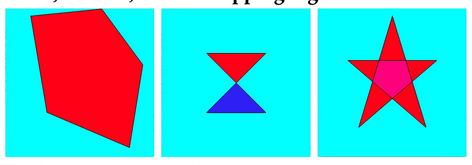
Given a curve and a point p, the winding number of p can be thought of as the number of times the curve rotates, **counterclockwise**, around p.



$$w(\mathbf{p}) = \frac{1}{2\pi}\sum_{i=1}^{n}\theta_i$$

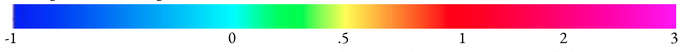
Fig. 1 (left) & Fig. 2 (right): Illustration of 2D winding number calculations. Retrieved from [1]

For closed 2D shapes, the winding number can be used as an exact threshold to detect inside, outside, and overlapping regions.



Inside (red): w(p) = 1
Outside (light blue): w(p) = 0
Overlapping (pink): w(p) = 2
Clockwise orientation (navy): w(p) = -1

Winding number color gradient:



For open 2D shapes, the winding number can be used as a confidence measure for inside-out detection. For a simple threshold, any point with w(p) > .5 is "inside."

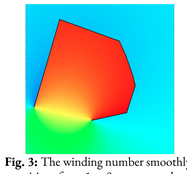


Fig. 3: The winding number smoothly transitions from 1 to 0 on an open boundary

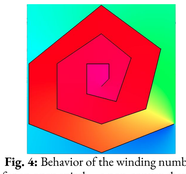


Fig. 4: Behavior of the winding number for an open spiral— a non-convex shape

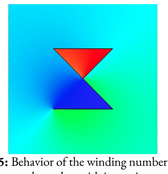
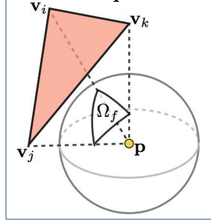


Fig. 5: Behavior of the winding number for an open hourglass with inconsistent orientation

Similar in 3D. The generalized winding number is the sum of the solid angles divided by 4π, where the solid angle is the area of a surface's projection onto a unit sphere centered at p.



$$w(\mathbf{p}) = \sum_{f=1}^{m}\frac{1}{4\pi}\Omega_f(\mathbf{p})$$

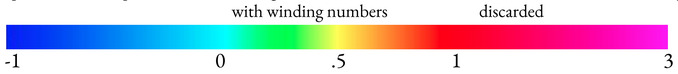
Fig. 6 (left) and Fig. 7 (right): Illustration of a winding number calculation in 3D.

Methods

2D Case:

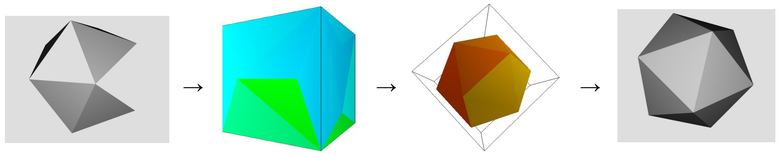
1. Input: mesh description (vertices, edges), oriented counterclockwise
2. Use Triangle [2] to triangulate the input vertices.
3. Evaluate the winding number at the centroid of each triangle in the mesh. If w(p) > .5, keep. If w(p) <= .5, discard.

Input mesh with open areas → Triangulated mesh color coded with winding numbers → Finalized mesh, with outside parts discarded



3D Case:

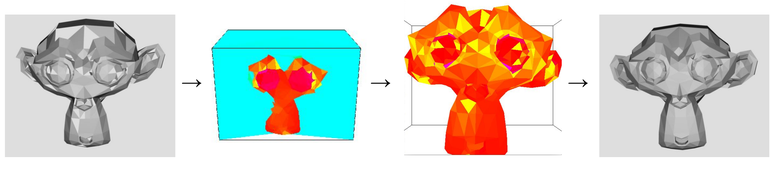
1. Input: mesh description (vertices, faces)
2. Use TetGen [3] to mesh the bounding box of the input vertices. Instead of meshing the convex hull as suggested in [Jacobson et al.], the program meshes the bounding box for better thresholding.
3. Evaluate the winding number at each tetrahedra.
4. Find the segmentation threshold by averaging the winding numbers of all tetrahedra.
   a. If the average > .5, the program defaults to using .5 as the threshold.
   b. Else, the program uses the average value as a threshold (anything less than the average is considered an "outside" point).
5. Outside tetrahedra are removed in the final mesh



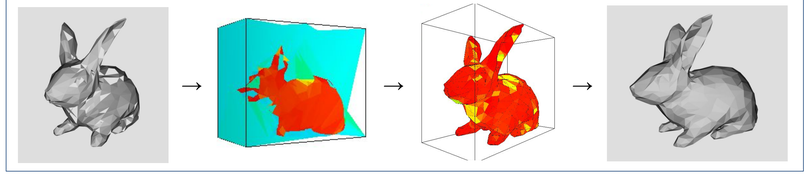
Input mesh with 5 holes → Tetrahedral Mesh (using TetGen) of the bounding box → After removing outside portions → Final result with threshold = .5

Results

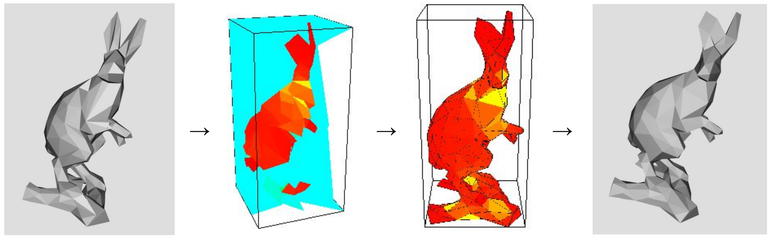
- Evaluating the winding number not only reveals open areas (orange/yellow), but also reveals self-intersections (pink) in the original mesh:



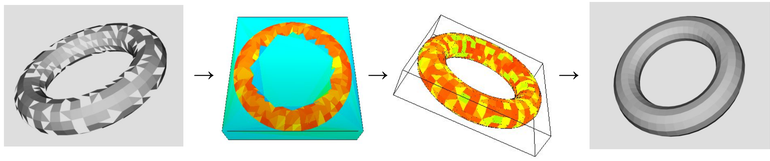
- Repaired mesh of an input bunny object with holes (threshold = .40):



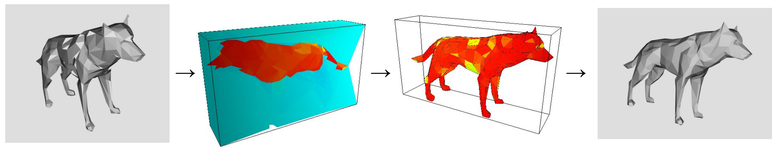
- Repair of a rabbit mesh. The repaired mesh has some differences from the original surface mesh (note the ears) due to the low resolution of the input and meshing with TetGen [3]:



- Repair work done on a torus mesh object. Threshold = .38 (threshold decreases with more holes):



- Wolf mesh. Threshold = .50:



In general, the quality of repaired mesh and its consistency with the initial mesh improve as the resolution (polygon count) of the input mesh increases.

Future directions

1. The algorithm used for determining the threshold isn't perfect for cases where the resolution of the mesh is relatively low. Further experimentation and exploration of is needed to establish a more effective method for dealing with such cases.
2. For low resolution meshes or highly intricate meshes, using Tetgen to generate a tetrahedral mesh might delete faces originally described in the surface mesh. Further work can be done to implement surface constraints in order to preserve all input facets.
3. The current program has difficulty handling high resolution meshes. Because of the amount of processing that must be done, higher resolution meshes take longer to repair.. Improvements on speed can be made by simplifying the winding number evaluation process and using a faster language (currently, the program is written in Python).

Citations

1. Jacobson, A., Kavan, L., & Sorkine-Hornung, O. (2013). Robust Inside-Outside Segmentation Using Generalized Winding Numbers. *ACM Transactions on Graphics, 32* (4), 1. doi:10.1145/2461912.2461916
2. Triangle Software—generates constrained delaunay triangulation meshes in 2D
3. Tetgen Software—generates constrained delaunay tetrahedralization meshes in 3D

Acknowledgements

I'd like to thank Dr. Emily Whiting for hosting me over the summer and introducing me to an interesting project topic. I'd also like to thank both my mentor, Athina Panotopoulou, and Dr. Whiting for guiding me through the project and helping me brainstorm, troubleshoot, and so much more. Finally, I'd like to thank GROW and BU for giving me the opportunity for a memorable internship experience.