

Marking Vertices in EdgeBreaker traversal: substitute the marking of individual vertices by a walk around vertex checking for marked of triangles

1. Swing Wrapper

(<http://www.gvu.gatech.edu/~jarek/papers/SwingWrapper.pdf>)

Use **EdgeBreaker** to encode connectivity and use the *dihedral angle predictor* to encode geometry.

But first approximate (retile) a mesh M by a mesh M' of simply connected regions called *triangloids*.

1.1. How to encode/decode M' :

1. Starting in a “C” triangle (which is forced to be isosceles, and whose gate edge is its base)
2. Make an orthogonal circle to triangle and swing around gate until it hits surface. If

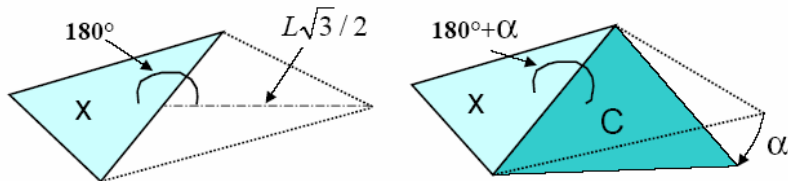
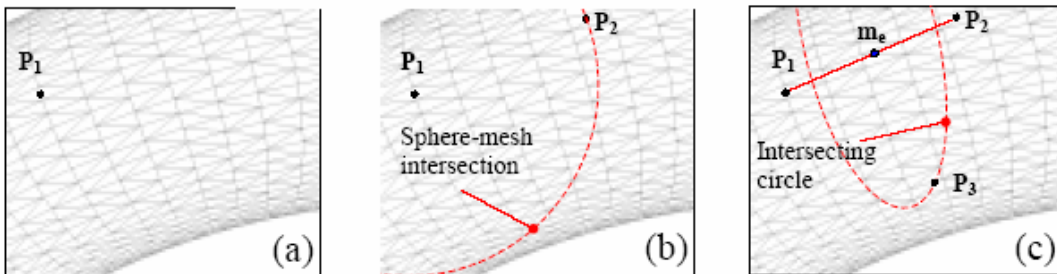


Figure 2: Construction of a new C triangle; prediction (left) and correction of α degrees (right).

Predict a 180° degree rotation (great for planar regions) or other angle (based on average model curvature), and encode the difference (using quantization).

3. If the radius is too big and the swinging circle misses the surface (or otherwise fails) choose a smaller radius L for circle

1.2. Retiling: making a mesh M' out of *triangloids* in M .



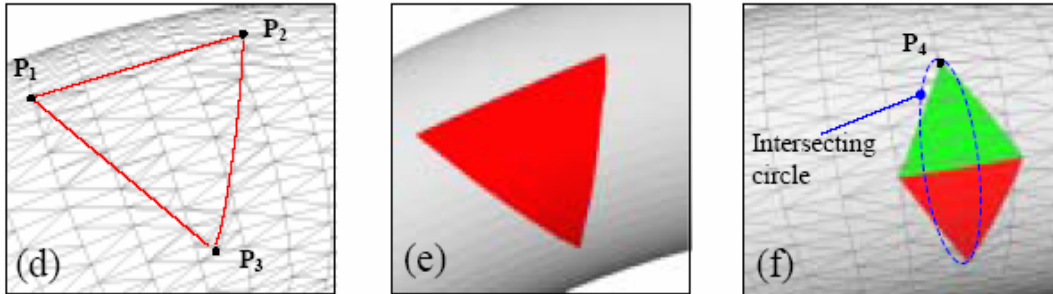


Figure 3: Construction of the initial triangles of M' .

Triangloids:

- Sets of triangles of M that are approximated by a triangle in M'
- A triangle on M may be shared by several triangloids
- Triangloid are approximated by triangles whose vertices sit on the surface

1. Pick starting point P_1
2. Point P_2 lies on the sphere-mesh intersection
3. Points P_3 and P_4 are on intersection of M with circle of radius L , centered at midpoint of P_1P_2 (if there are less than one, L is too large)
4. Find triangles on M that are included in the triangloid, following path on surface that most closely follows the sides of the approximating triangle. This may require insertion of vertices (edge or triangle splits). Paths are approximated Geodesics (SGP: Shortest Geodesic Paths, which minimize walk on a surface). This seems to be the hardest part.

1.3. Alternative: adaptive retiling

Shorten or elongate sampling step depending on Hausdorff distance between triangloid and approximating triangle.

- Make a sandwich – if path is in sandwich, h (height of sandwich) is the error

Pros: improves quality without using more triangles, and preserves sharp features.

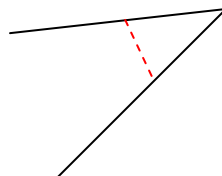
Cons:

- takes more room, to encode the changes in step-size (radius)
- no longer able to predict size of final compressed model
- uniform sampling may produce redundant triangles suitable for high entropy or run-length compression

1.4. Summary

If it works it yields great compression ratios (600-to-1, 4000-to-1, depending on allowed error). Otherwise, it is too costly, so use something else.

Biggest errors are near sharp features, corners, where a chamfer may easily occur cutting the corner.



2. Edge Sharpener

(<http://www.gvu.gatech.edu/~jarek/papers/EdgeSharpener.pdf>)–

An algorithm to recover sharp features in triangle meshes. Motivated by the problem last stated above in SwingWrapper. Also applicable to models obtained from scanners, or isosurface extraction from volume datasets.

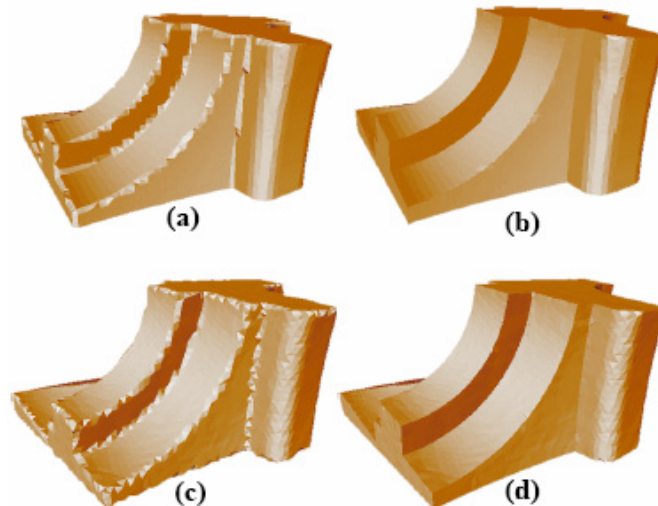


Figure 1. An aliased model (a) generated with Marching Cubes is improved automatically (b) by EdgeSharpener. A version (c) of the original model was generated through the lossy SwingWrapper compression⁹. An improved version (d) was obtained with no additional information by running EdgeSharpener after decompression.

2.1. EdgeSharpener algorithm outline:

- Identify chamfer triangles that must be processed
- Decide how to subdivide them
- For each newly inserted vertex, estimate the sharp edge or corner that needs to be restored and move the vertex onto it.

2.2. How to identify Chamfer edges?

- Use parallelogram prediction?
- Compute the dihedral angle to label edges as “flat” or “sharp” – threshold based on statistics
- Grow “flat” regions of triangles and build clusters/components
- Chamfer edges are those of vertices A and B, such that A and B are not in the same component
- For all Chamfer edges – break and add midpoint

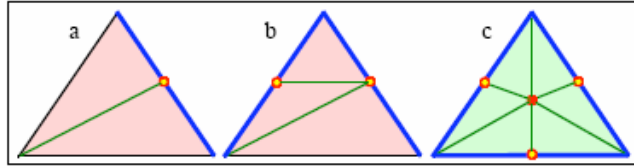


Figure 4: Subdivision of a chamfer triangle with one (a) two (b) or three (c) chamfer edges.

- Snap new vertices:

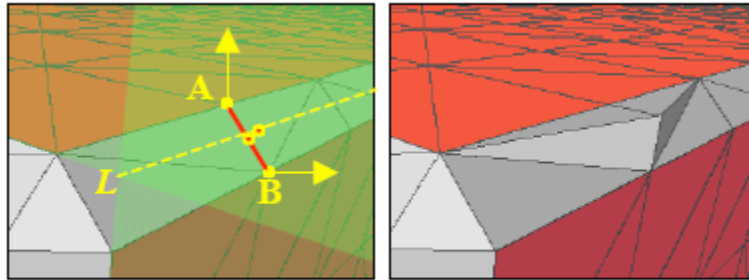


Figure 5: Insertion of a new vertex to split a chamfer edge.

- Find two planes P_1 and P_2 from the connected components separated by the chamfer triangles. $L = P_1 \cap P_2$: project new vertex on L
- Use local plane, say vertex normal

3. Sharpen+Bend

(<http://www.gvu.gatech.edu/~jarek/papers/Sharpen&Bend.pdf>)

Subdivision without smoothing sharp features

- Identify sharp edges with EdgeSharpener
- Then subdivide using a modified Butterfly subdivision scheme
 - Use points on sharp edges and subdivide curve
 - Rest of points subdivide surface

4. Resampling mesh as a distance field

(<http://www.gvu.gatech.edu/~jarek/papers/PRM.pdf>)

- Classify triangles based on their normal into +X, -X, +Y, -Y, +Z and -Z triangles.
 - There could be a different sampling densities
- Triangulate with new samples
- Eliminate original vertices by collapsing
 - Becomes “regular” – easy to predict

Results in very good prediction using EdgeBreaker. Two coordinates out of 3 usually require no correction. There are some issues in joints between regions