VERTEX-BASED DELAUNAY TRIANGULATION OF MESHES OF ARBITRARY TOPOLOGICAL TYPE

A. KLEIN, A. CERTAIN, A. DEROSE, T. DUCHAMP, W. STUETZLE

1. Introduction

Triangular meshes have become a standard way of representing objects in computer graphics and geometric modeling. Unfortunately, highly complex triangular meshes, easily generated by laser scanning systems, can be both frustrating to edit and expensive to store, transmit, and render.

Multiresolution representations of meshes, developed by Lounsbery *et al.* [3, 4] and extended by others [2, 5], address these problems. A multiresolution representation of a mesh consists of a simple base mesh plus a series of local correction terms, wavelet coefficients, that capture the represented object's detail at increasing levels of resolution. Multiresolution mesh representations are therefore useful for applications such as compression and the progressive display and transmission of three-dimensional graphics [2].

The one noted drawback to the Lounsbery *et al.* multiresolution analysis is that this method can only be applied to meshes displaying subdivision connectivity, i.e. meshes obtained from a simpler base mesh by recursive 4-to-1 splitting. However, Eck *et al.* [2] have found a way to work around this obstacle and thereby convert arbitrary meshes into multiresolution form. The arbitrary mesh $M.$ is first transformed into a mesh M^J that exhibits subdivision connectivity, a process referred to as *remeshing*. Then, the Lounsbery *et al.* method is applied to M^J, yielding the multiresolution representation.

As part of the remeshing procedure employed by Eck *et al.*, M is partitioned into a number of triangular regions using a Delaunay triangulation. To obtain this Delaunay triangulation, a Voronoi tiling is first constructed. However, we found that in particular cases, discovered in meshes of relatively simple, real-world objects, the Voronoi tiling algorithm used in the Eck et al. paper fails. Hence, as part of our larger goal of implementing a working remeshing algorithm, it was necessary to modify this tiling algorithm.

Date: July 10, 1997.

2. Previous Tiling Algorithm

Before explaining our new algorithm, it is worthwhile to first give a brief overview of the Eck *et al. algorithm.* It relies on a discrete analogue of the notion of Voronoi tile. Given a collection of faces of M (called *seeds*), the *Voronoi tile* T_i, centered at the seed i, is comprised of all faces in a mesh which are closest in distance to i. Distance between adjacent faces is the Euclidean distance between their centroids, and distance between non-adjacent faces is the graph-theoretic distance between them in the dual graph to the original triangulation (see below).

The polyhedral dual of this tiling is called the *Delaunay triangulation.* The Delaunay vertices are the tiles, themselves; Delaunay edges are *cuts* (a cut is a contiguous set of edges of M along with a pair of tiles touch); and Delaunay faces are triple intersections of tiles (i.e. vertices of the original mesh that are common to three tiles). The Delaunay triangulation is guaranteed to be homeomorphic to the original mesh, provided that the tiling meets certain conditions. Specifically, the completed tiling must satisfy the following criteria:

1. Because the tiles will be mapped into a plane, every tile must be homeomorphic to a disk.
2. For the Delaunay triangulation to model the original surface, the Voronoi diagram should have the same number of boundary components as the original mesh. In order to guarantee this equivalence of boundary components, every triangle on the mesh boundary must belong to a tile whose seed is also on the mesh boundary.
3. No more than 3 tiles can meet at any vertex interior to the mesh, and no more than 2 tiles can meet at any vertex on the mesh boundary. If this criterion is not met, the dual polyhedron will not consist of triangles.
4. In order for the dual of the tiling to be a triangulation, no pair of tiles may share more than one cut. A corollary of this rule is any tile that abuts the mesh boundary may do so along only 1 cut. This rule guarantees that any pair of vertices in the dual triangulation will have at most one edge connecting them. Together with criterion (2), it also guarantees that the boundary of the dual triangulation is homeomorphic to the boundary of the original mesh.

Eck *et al.* add one additional criterion:

5. The sum of the lengths of any two adjacent cuts in a tile must be greater than 10% of the length of the tile perimeter.

Unlike the first 4 criteria, this additional criterion is not strictly necessary to obtain a valid dual triangulation. Its serves to avoid triangles in the dual triangulation that have poor aspect ratios, and thus poor rates of compression.

To obtain a tiling that satisfies the above criteria, the Eck *et al.* algorithm initially constructs a dual to the mesh M where nodes in the dual graph are faces in M, and edges in the

dual graph represent adjacency relationships among the faces of M. The edges in the dual graph are given a weight equivalent to the Euclidean distance between the centroids of the corresponding faces in M.

Given this dual graph, constructing a Voronoi tiling is then a multi-source shortest path problem, which is solved using a multi-source variant of Dijkstra's algorithm [1]. Algorithm execution begins by picking a single randomly chosen site face or seed, which functions as the center of the Voronoi tile. Using a priority queue in which the priority of a node in the dual graph (a face in M) is the distance to the nearest seed, all of the nodes can be assigned to tiles in $O(n \log n)$ time. The tile is grown by adding mesh triangles in order of proximity to the seed until all the triangles have been added or conditions (1) or (2) are violated. If either of these conditions is violated, a new tile seed is inserted at the mesh triangle where the violation was detected. Tile growth is then restarted.

When tile growth is complete, conditions (3), (4), and (5) are checked. Inserting a new site face adjacent to the offending vertex is the result of a violation of condition (3). Inserting a new site face near the mid-point of one of the offending cuts solves a violation of condition (4). If a pair of cuts failing condition (5) is detected, a new site is added on one of the faces the cuts share. If new faces have been added during these post checks, tile growth is restarted. The algorithm finishes when tiling is completed, and conditions (3), (4), and (5) have all been met.

As we mentioned above, in certain situations, this algorithm fails. Specifically, if condition (3) is violated and all the triangles adjacent to the offending vertex are already seeds, inserting a new seed at this point will not fix the situation, and so the algorithm cannot finish.

3. New Tiling Algorithm

Our solution is to make the tiling algorithm vertex-based instead of face-based. Our tiles are "grown" by adding, so-called, *derived neighborhoods* of vertices (see Figure 1(a)). Each tile is the union of such neighborhoods.

Voronoi tilings generated in this way automatically satisfy condition (3) because multiple intersections can only occur at the centroid of a triangle, where at most three tiles can touch (see Figure 1(b)). In the extreme case, where *all* vertices of the original mesh are seeds, the dual Delaunay triangulation is the original mesh.

Much of the algorithm coincides with the algorithm of [2]. We also use the multi-source Dijkstra algorithm and the same criteria for correctness of a tiling, although we do not need to perform as many checks. Our modified algorithm proceeds in the following manner:

1. Choose seed vertices and insert them into a priority queue. This queue sorts mesh vertices mesh based on their proximity to the closest seed. For a mesh with no boundaries,

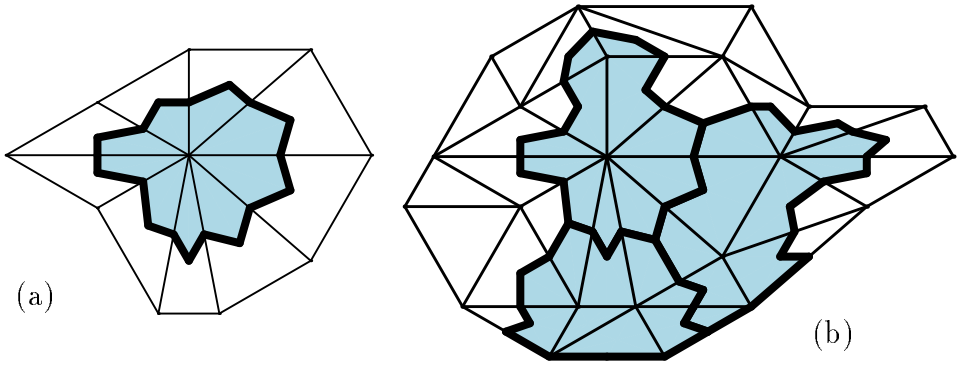

FIGURE 1. (a) The derived neighborhood surrounding a vertex. The corners on the boundary of the derived neighborhood are midpoints of edges and centroids of faces of the mesh. (b) At most three tiles can intersect at a point, and the point of triple intersection must be the centroid of a face of the mesh.

a vertex is randomly selected to be the seed. For a mesh with boundaries, 3 seeds, roughly equidistant, are placed on each boundary. Creating these boundary seeds is not a necessary component of the algorithm, but is there simply to expedite the process. (Any valid tiling requires at least 3 tiles along every boundary, otherwise condition (4) would be violated.)

2. While the priority queue is not empty, remove the vertex v at the top of the queue. Add the vertex to the tile T to whose seed is nearest to v, and such that adding v to T does not violate conditions (1) or (2) from above.
3. If v violates conditions (1) or (2), we make this vertex a seed, empty the queue, insert the new seeds into the queue and return to step 2.
4. Otherwise, we update the neighbors of v in the expected manner for Dijkstra's algorithm and return to step 2. That is, for each vertex u adjacent to v not already added to a tile and not already in the priority queue, we insert it, setting its priority equal to the sum of the priority of v and the Euclidean distance from v to u. If it is already in the priority queue, we update its priority to the minimum of its current priority and the sum of the priority of v and the Euclidean distance from v to u.
5. When the priority queue is empty, indicating that all vertices have been added to a tile, we check that criteria (4) and (5) from above have been met. If either condition (4) or (5) is not met, we find the appropriate vertex to be a seed, insert all of our seeds into a new priority queue, and return to step 2. Otherwise, the algorithm terminates.

Our check that condition (1) is satisfied is the following:

(*) Let $S = \{v_1, \ldots, v_n\}$ be the set of vertices of the mesh adjacent to v, cyclically ordered (with $v_{n+1} = v_1$). We require that the intersection $T \cap S$ be a contiguous set, i.e. that $T \cap S = \{v_i, v_{i+1}, \ldots, v_j\}$, for $i \leq j$. (see Figure 2).

If there is more than one such arc, then v will cause T to wrap around and touch itself, thus rendering it homeomorphically inequivalent to a disk.

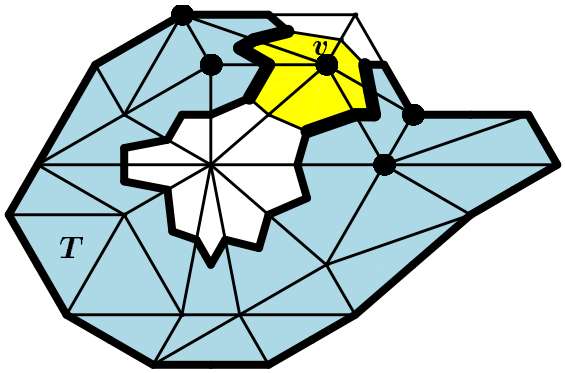

FIGURE 2. Adding the vertex v to T creates a tile that is homeomorphic to an annulus. Notice that the four vertices in $S \cap T$ do not form a contiguous set of vertices about v, violating criterion (*).

Checking for condition (2) is straightforward, and condition (3) is always satisfied.

To check for condition (4), notice that the number of *corners* (vertices of a tile that are adjacent to vertices contained in two tiles) of a tile T can never be less than the number of adjacent tiles, so if the number of adjacent tiles is less than the number of corners, the tile T intersects another tile along 2 or more cuts.

To check for condition (5) we simply walk along the tile perimeter to calculate the length of each cut and the total perimeter length, and from there we can perform the necessary calculations to determine if the lengths of any two adjacent cuts are less than 10

Observe that our algorithm is simpler and more streamlined than that of [2]. Furthermore, the computational complexity is the same. For both algorithms are based on the multi source Dijkstra algorithm whose complexity is $O(sn \log n)$ where s is the number of seeds and n is the number of nodes. In [2] n is the number of faces; in our algorithm. n is the number of vertices. Because the number of faces is roughly twice the number of vertices, this does not change the overall performance.

The only task remaining at this point, which was not an issue for the Eck *et al.* algorithm, is to retriangulate the mesh so that tile borders correspond to edges in the mesh. Because the circumference of the tiles borders is roughly proportional to the square-root of the number of vertices, this can be done rapidly and does not appreciably increase the size of the mesh.

4. Proof of Correctness for New Tiling Algorithm

A tiling that satisfies conditions (1–4) of the algorithm of Eck *et al.* [2] is guaranteed to have a dual triangulation that is homeomorphic to the original mesh. Therefore, it is only necessary to check that our new tiling algorithm is guaranteed to terminate with all four of these conditions satisfied.

We first show that each tile is homeomorphic to a disk (condition (1)). Let K be the underlying simplicial complex of our mesh, and let K' denote its first barycentric subdivision. Recall that each tile T is the union of derived neighborhoods of vertices of K. Because K is a finite simplicial surface this guarantees that T is homeomorphic to a compact surface (possibly with boundary. Therefore, we must only ensure that T is homeomorphic to a disk. We prove this by induction. When T is the derived neighborhood of a single vertex, the statement is clear. Now suppose that T is obtained from a tile T_0 by adding the derived neighborhood of a vertex v and that T_0 is homeomorphic to a disk. By virtue of our criterion (*), T is formed by gluing together two disks (T_0 and the neighborhood of v) along an arc, hence T is homeomorphic to a disk.

As we already mentioned, condition (3) is automatically satisfied.

Whenever either of conditions (2) and (4) are violated, we add another vertex seed and restart the tile growing step. Because K has a finite number of vertices, either (2) and (4) are eventually satisfied or every vertex of K is a seed. But in this case, the dual triangulation is K, itself, and conditions (2) and (4) are satisfied because K is assumed to be a simplicial surface.

5. Examples and Statistics

We ran our new algorithm on 4 meshes that were also utilized in [2]. With the exception of the dinosaur, we found the number of tiles we obtained was roughly equivalent to that of [2]. In the case of the dinosaur, we obtained a substantial reduction in the number of tiles (See Table 1).

mesh	#faces	#faces in smallest tile	#faces in largest tile	#faces in avg. tile	#tiles	#tiles in Eck *et al.*
cat	698	119	262	166.11	9	7
holes3	11,776	266	942	575.18	34	31
bunny	69,473	11	8482	1090.26	82	88
dino	103,713	380	6538	1874.97	70	117

Table 1. Comparison of our vertex-based tiling algorithm to the face-based algorithm of Eck *et al.* [2].

6. Conclusions.

We have designed a Delaunay triangulation algorithm for surfaces of arbitrary topological type. It has the same computational complexity as the algorithm described in [2]; but, whereas the algorithm of [2] may fail, ours is provably correct. As illustrated in Table 1 our tiling algorithm generates tilings that are similar to those of [2].

References

[1] A. Aho, J.E. Hopcroft, and J.D. Ullman. *Data structures and algorithms*. Addison-Wesley, Reading, Mass., 1983.

[2] Matthias Eck, Tony DeRose, Tom Duchamp, Hugues Hoppe, Michael Lounsbery, and Werner Stuetzle. Multiresolution analysis of arbitrary meshes. In *SIGGRAPH'95*, Computer Graphics Annual Conference Series, pages 173–182, August 1995.

[3] M. J. Lounsbery. *Multiresolution Analysis for Surfaces of Arbitrary Topological Type*. PhD thesis, University of Washington, Department of Computer Science and Engineering, September 1994.

[4] M. J. Lounsbery, T. DeRose, and J. Warren. Multiresolution surfaces of arbitrary topological type. *ACM Transactions on Graphics*, 16(3):34–73, 1997.

[5] P. Schröder and W. Sweldens. Spherical wavelets: Efficiently representing functions on the sphere. In *SIGGRAPH'95*, Computer Graphics Annual Conference Series, pages 161–172, Aug 1995.