Improved Constructions of Delaunay Based Contour Surfaces

Siu-Wing Cheng*
Department of Computer Science
HKUST, Hong Kong

Tamal K. Dey†
Department of CSE
IIT Kharagpur
India 721302

Abstract

We revisit a method due to Boissonnat for surface reconstruction from parallel slices based on Delaunay triangulations. We eliminate the costly step of computing the three dimensional Delaunay triangulation, and instead compute the surface triangles directly. A non self-intersecting tiling is automatically guaranteed by these triangulations. Our experiment on some medical data shows that the method is effective.

CR Categories: I.4.5 [Image Processing and Computer Vision]: Scene Analysis—Surface fitting;

Keywords: Delaunay triangulation, Voronoi diagram, medical imaging, topography.

1 Introduction

We revisit the problem of fitting a triangulated surface between polygonal contours on parallel slices in 3D which finds applications in medical imaging and topography. The input is a series of parallel *slices* of the solid to be modeled. Each slice contains possibly several *contours*, whose boundaries consist of non-intersecting closed polygonal curves. The goal is to reconstruct a triangulated surface that bounds an object, called *solid connection*, whose geometry is more likely to resemble the original sampled object.

It is popular to reconstruct the solid connection between two consecutive slices and then concatenate them in series. However, a self-intersecting solid connection can be unavoidable unless new vertices are inserted on the contours [7]. Several approaches for connecting contours on two slices have been proposed and they usually add extra vertices in different ways. Boissonnat presented a method based on Delaunay triangulations [3] and this is refined further by Geiger [6]. Barequet and Sharir's algorithm [2] requires to discretize the contour boundary sufficiently fine and given an arbitrary data set, it is unclear how to determine a good discretization quantitatively. Bajaj *et al* [1] combines the approaches in [2, 6], but it requires a strong assumption on input sampling distance.

We study Boissonnat's method [3] further to generate the contour surface without the costly step of computing 3D Delaunay triangulation. We observe that the contour surfaces constructed by our implementation produce the expected branchings and make necessary correspondences.

*Email: scheng@cs.ust.hk

†Email: dey@cse.iitkgp.ernet.in. Research supported in part by a DST grant, 1997, Government of India.

2 The basic method

Let P_0 and P_1 be two parallel slices. A triangle/edge on the surface connecting P_0 and P_1 is called a *vertical* triangle/edge. Our method first constructs the 2D Delaunay triangulation D_i of vertices in P_i. When line segments in the contours intersect the triangulation, new vertices are repeatedly added at midpoints of them until every boundary edge in P_i is a union of Delaunay edges. Our experiments show that this increases the number of vertices by a small factor, typically 3 to 5.

Define *contour*(D_i) to be the set of vertices, edges, and triangles in D_i that lie inside or on the boundaries of the contours. Let V_i denote the dual Voronoi diagram of D_i. For each vertex/edge/triangle $\sigma \in D_i$, we use ν_σ to denote its dual in V_i. Define *contour*(V_i) = $\{\nu_\sigma |$ vertex/edge/triangle $\sigma \in$ *contour*$(D_i)\}$.

Let $\mathcal{D}$ denote the 3D Delaunay triangulation of vertices in P_0 and P_1. A vertical triangle is the convex hull of a vertex and an edge from the two slices. A surface connecting P_0 and P_1 is exposed by removing from $\mathcal{D}$ all tetrahedra that are incident with any edge outside *contour*(D_i). We call this the *basic_surface*. We describe how to compute *basic_surface* without constructing $\mathcal{D}$. It is based on the observation that the triangles on the *basic_surface* are either on the convex hull or are incident to two tetrahedra, exactly one of which is removed. We use the binary operator $*$ to denote the convex hull of the operands.

Let ν_e be a Voronoi edge in *contour*(V_0). Let ν_s and ν_t be the two endpoints of ν_e such that $\nu_s \in$ *contour*(V_0). So the tetrahedron $p_0 * s$ in $\mathcal{D}$ is not removed. Let ν_{p_i}, $0 \leq i \leq k$, be the Voronoi faces in V_1 intersected by ν_e in order from ν_s to ν_t. The vertical triangles on *basic_surface* incident to e are:

1. $p_0 * e$ *if* $p_0 = p_k$ *and* $\nu_t \notin$ *contour*(V_0). Since $\nu_t \in \nu_{p_0}$ and $\nu_t \notin$ *contour*(V_0), $p_0 * t$ is a tetrahedron in $\mathcal{D}$ removed. So the triangle $p_0 * e$ stays on the *basic_surface*.

2. $p_0 * e$ *if* $p_0 \neq p_k$ *and the Voronoi edge* $\nu_{p_0} \cap \nu_{p_1} \notin$ *contour*(V_1). Since $\nu_e \cap (\nu_{p_0} \cap \nu_{p_1}) \neq \emptyset$ and $\nu_{p_0} \cap \nu_{p_1} \notin$ *contour*(V_1), $p_0 p_1 * e$ is a tetrahedron in $\mathcal{D}$ removed. So $p_0 * e$ is exposed. Similar reasoning yields the next three rules.

3. $p_k * e$ *if* $\nu_{p_{k-1}} \cap \nu_{p_k} \in$ *contour*(V_1) *and* $\nu_t \notin$ *contour*(V_0).

4. $p_k * e$ *if* $\nu_{p_{k-1}} \cap \nu_{p_k} \notin$ *contour*(V_1) *and* $\nu_t \in$ *contour*(V_0).

5. $p_j * e$ *if exactly one of* $\nu_{p_{j-1}} \cap \nu_{p_j}$ *and* $\nu_{p_j} \cap \nu_{p_{j+1}}$ *lies in contour*(V_1), *for* $1 \leq j \leq k-1$.

To apply rules 1–4, we carry out a binary search on the boundaries of ν_{p_0} and ν_{p_k} to determine if ν_e crosses any boundary edge of ν_{p_0} and ν_{p_k} in the projection. To apply rule 5, observe that the bounding edges in *contour*(V_1) of a Voronoi face ν_{p_j} form a connected chain. We draw the chord connecting the endpoints of this chain. Exactly one of $\nu_{p_{j-1}} \cap \nu_{p_j}$ and $\nu_{p_j} \cap \nu_{p_{j+1}}$ is in *contour*(V_1) if and only if ν_e intersects the chord. Hence, this becomes reporting the intersections between *contour*(V_0) and the chords defined for *contour*(V_1) which can be solved in $O(n \log n + k)$ time, where k is the output size, using a plane-sweep based algorithm [4]. A symmetric method works for edges in *contour*(D_1).

3 Handling correspondence and branching

The *basic_surface* may touch itself generating *pinchings* as shown in Figures 1 and 2. Let P_i^+ be the solid obtained after thickening P_i slightly. Let U be the union of P_0^+, P_1^+, and the region bounded by the contour surface. Then there is a pinching if and only if there is a point in the boundary of U whose neighborhood is not a topological disk.

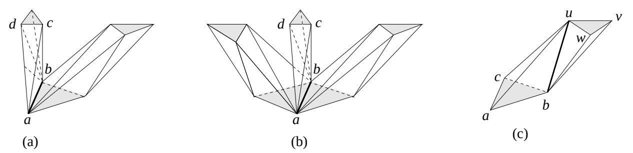

Figure 1: Pinchings at edges.

Figure 1 shows three types of pinchings at edges caused by tetrahedra that are connected to the rest of the solid connection via a single edge. We call such tetrahedra *loose tetrahedra* and removing them eliminates the pinchings at edges. (We ignore the type of pinching in Figure 1(c) as it is always associated with a cone-pinch shown in Figure 2(a).) For each edge $\nu_e \in V_i$, we compute its *left critical segment* and *right critical segment* as follows. The left critical segment is empty if the left endpoint of ν_e is outside *contour*(V_0). Otherwise, it extends from this endpoint to the first intersection with $V_{i+1} \setminus$ *contour*(V_{i+1}) or the right endpoint of ν_e. The right critical segment of ν_e is initialized symmetrically. Subtracting the left and right critical segments from ν_e yields the *non-critical segment* of ν_e. Then we shorten the critical segments in V_0 by cutting at extreme intersections with non-critical segments in V_1. The non-critical segments in V_0 are lengthened correspondingly. Then we shorten critical segments in V_1 similarly. The shortening of critical segments in V_0 and V_1 are repeated alternatively until there is no further change.

Intersections of critical segments correspond exactly to the tetrahedra that are not loose. For each critical segment, only the two extreme intersections are needed to generate the vertical triangles on the modified solid connection (after loose tetrahedra are removed). This produces a contour surface of complexity $O(n)$ but may contain pinchings at vertices.

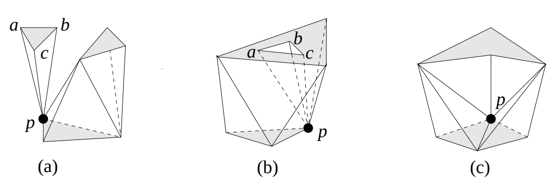

Figure 2: Pinchings at vertices.

Figure 2 shows the *cone-pinch*, *hole-pinch* and the *gorge-pinch* from left to right. For each cone-pinch, we remove it by removing the corresponding vertical triangles. A hole-pinch at a vertex p can be removed by adding tetrahedra incident to p to fill the “hole”. If not for respecting the boundaries of contours, this can be simulated by adding the Delaunay triangles at the bases of these triangles to the slice P_i opposite to p, and removing the cycle of vertical triangles. Instead, we need to introduce a copy of P_i between the two original slices, shift the contour surface to connect to this copy of P_i, and work with this copy of P_i instead. A gorge-pinch is transformed to a hole-pinch and then removed. For example, in Figure 2(c), adding the tetrahedra incident to exposed contour edge incident to p (simulated by adding vertical triangles incident to endpoints of this edge) converts the gorge-pinch at p to a hole-pinch. Details of pinching removal can be found in [5].

Figure 3 shows the heart model reconstructed by our implementation, where branchings for arteries have been well detected after removing pinchings.

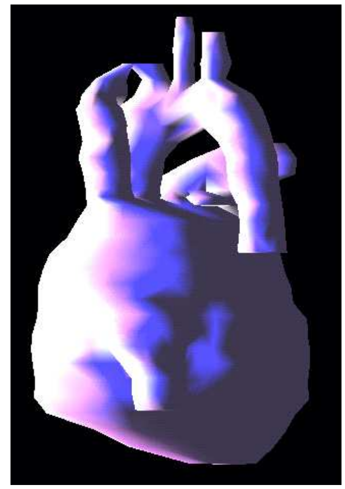

Figure 3: Human heart.

References

[1] C. Bajaj, K. Lin, and E. Coyle. Arbitrary topology shape reconstruction from planar cross sections. *Graphical Models and Image Processing* **58** (1996), 524–543.

[2] G. Barequet and M. Sharir. Piecewise-linear interpolation between polygonal slices. *Computer Vision and Image Understanding* **63** (1996), 251–272.

[3] J. D. Boissonnat. Shape reconstruction from planar cross-sections. *Computer Vision, Graphics, and Image Processing* **44** (1988), 1–29.

[4] T. Chan. A simple trapezoid sweep algorithm for reporting red/blue segment intersections. *In* “Proc. 6th Canad. Conf. Comput. Geom. 1994”, 263–268.

[5] S. W. Cheng and T. K. Dey. Improved constructions of Delaunay based contour surfaces. Manuscript 1999.

[6] B. Geiger. Three-dimensional modeling of human organs and its application to diagnosis and surgical planning. Technical report, 2105, INRIA, France, 1993.

[7] C. Gitlin, J. O'Rourke and V. Subramanian. On reconstructing polyhedra from parallel slices. *Intern. J. Comput. Geom. Appl.* **6** (1996), 103–122.