Surface and 3D Triangular Meshes from Planar Cross Sections

Chandrajit L. Bajaj — Department of Computer Science, Purdue University, West Lafayette, IN 47907

Edward J. Coyle, Kwun-Nan Lin — School of Electrical Engineering, Purdue University, West Lafayette, IN 47907

email: {bajaj@cs, coyle@ecn klin@cs}.purdue.edu

Correspondent:
Chandrajit L. Bajaj, Computer Science Dept., Purdue University, West Lafayette, IN 47907, Tel: (317)494-6531, Fax: (317)496-2567, http://www.cs.purdue.edu/people/bajaj

Abstract. *This paper presents two unstructured mesh generation algorithms with a discussion of their implementation. One algorithm is for the generation of a surface triangular mesh from a parallel stack of planar cross-sections (polygons). The other algorithm is for the construction of a 3D triangular (tetrahedral) mesh of the solid region (polyhedron) bounded by the surface mesh and the planar cross-sections.*

Construction of a surface triangular mesh from planar contours is difficult because of "correspondence", "tiling" and "branching" problems. We provide a simultaneous solution to all three of these problems. This is accomplished by imposing a set of three constraints on the constructed surface mesh and then by deriving precise correspondence and tiling rules from these constraints. The constraints ensure that the regions tiled by these rules obey physical constructs and have a natural appearance. Regions which cannot be tiled by these rules without breaking one or more constraints are tiled with their medial axis (edge Voronoi diagram).

Construction of the tetrahedral mesh of the solid region bounded by planar contours and the surface mesh is difficult because the solid can be of high genus (several tunnels and holes) as well as have internal voids. We present a new algorithm to tetrahedralize the prismatoid bounded by two slices and the reconstructed tiling surfaces. Surface and tetrahedral meshing results are obtained with both synthetic and actual medical data.

keywords. tiling, mesh generation, tetrahedralization

1 Introduction

Technologies such as magnetic resonance imaging (MRI), computed tomography (CT), and ultrasound imaging allow measurements of internal properties of objects to be obtained in a nondestructive fashion. These measurements are usually obtained one slice at a time, where each slice is a 2D array of scalar values corresponding to measurements distributed over a plane passing through the object.

Once these measurement slices have been obtained, one goal is to reconstruct boundary and finite element models of substructure of the scanned data for quantification, geometric and physical reasoning, and three dimensional visualization. Important goals in the construction of both surface triangular meshes and tetrahedral meshes (3D finite elements) is to reduce the number of triangles and tetrahedra as well as assume that these finite elements have good aspect ratio [5, 30].

In this paper we present implementations of two algorithms. One is to construct surface elements, and the other is to construct tetrahedral finite elements from planar contour data. Our algorithms are not limited to the biomedical field. They are applicable to other areas where the input can be represented by a parallel stack of planar polygons (contours).

The overview of previous work is discussed in Section 2. We present our surface mesh construction algorithm in Section 3, our tetrahedral mesh generation algorithm in Section 4, and implementation results in Section 5.

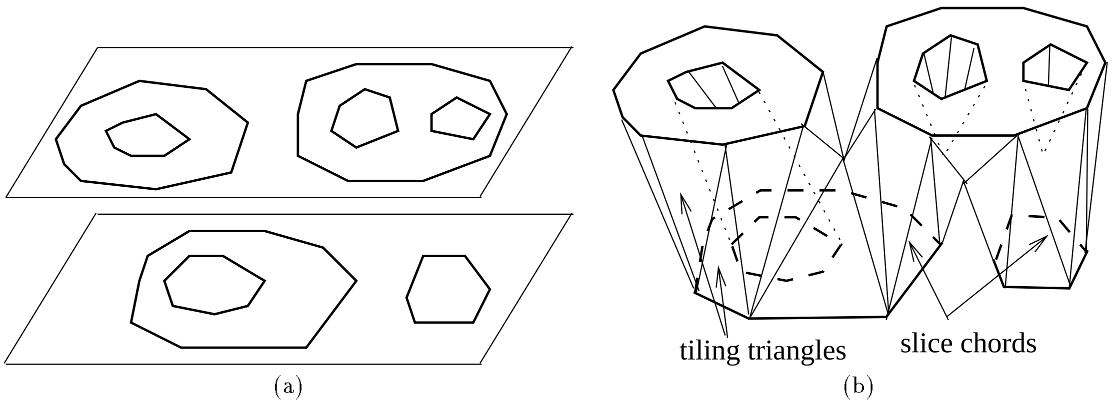


Figure 1: (a) two slices of contours, (b) one possible surface mesh construction.

2 Overview of Previous Approaches

2.1 Boundary element mesh construction

The task here is to construct surface meshes which interpolate the contours (polygons) on two adjacent slices. Each slice contains zero or more non-intersecting polygons which may be nested. Fig. 1 shows two slices of contours and one possible surface mesh construction. The boundary element mesh construction requires a solution to *correspondence*, *tiling*, and *branching* problems that we shall now address.

The *correspondence* problem involves finding the correct connections between the contours of adjacent slices. Bresler et al. [7] uses domain knowledge to constrain the problem. Meyers et al. [29] and Soroka [37] approximate the contours by ellipses and then assemble them into cylinders to determine the correspondence. Wang et al. [38] checks the overlapping area as the criterion for the correspondence.

Tiling refers to the use of slice chords to triangulate the strip lying between contours of two adjacent slices into triangles (Fig. 1(b)). There are two related issues. One is how to accomplish optimal tiling in terms of certain metrics such as surface area, enclosed volume, etc. The other is the topological correctness of the tiling.

The problem of mating points between contours into triangles is formalized by Keppel [25] into a graph search problem. Fuchs et al. [17] provide an efficient algorithm to obtain metric optimal solutions based on an Euler tour of a toroidal graph. Sloan et al. [36], Shinagawa et al. [35], Kehtarnavaz et al. [24] and Wang et al. [38] either improve Fuchs' algorithm or develop new algorithms to find the minimum cost path. Some fast heuristic tiling methods are developed by Christiansen et al. [12], Ganapathy et al. [18] and Ekoule et al. [15].

When two corresponding contours are very different, it is difficult to obtain a topologically correct and natural tiling. Gitlin et al. [21] show one example in which two extremely different polygons cannot be tiled to form a polyhedron. Even in a moderately dissimilar contour pair in which a polyhedron can be formed, the tiling algorithm may result in the surface mesh self-intersecting and/or physically unlikely topologies. Algorithms [12, 13, 15, 17, 18, 24, 25, 36, 38] which attempt to tile all contour vertices to the adjacent slice might produce an unlikely topology in the cases of very different contours. Boissonnat [6] and Barequet et al. [4] produce horizontal triangles which lie on the slice to avoid this problem.

A *branching* problem occurs when a contour in one slice may correspond to more than one contour in an adjacent slice. Fig. 2(a) shows that contour $C3$ of slice $S2$ branches into $C1$ and $C2$ of slice $S1$. The branch processing approaches which do not generate many intermediate contours can be classified into the four methods shown in Fig. 2(b)-(e). Our branching handling uses the method in Fig. 2(b).

Christiansen et al. [12], Shantz [34] and Shinagawa et al. [35] use the method in Fig. 2(d). They dip down the middle of the bridge to model the saddle point of the branching region. Ekoule et al. [15] form an intermediate contour between two slices for the case of one-to-many branching. His method produces less distortion than the method of Fig. 2(d). Meyers et al. [29] use the scheme in Fig. 2(e). The branching processing result of Barequet's [4] approach is similar to Fig. 2(d) or (e) depending on whether bridges are added or not.

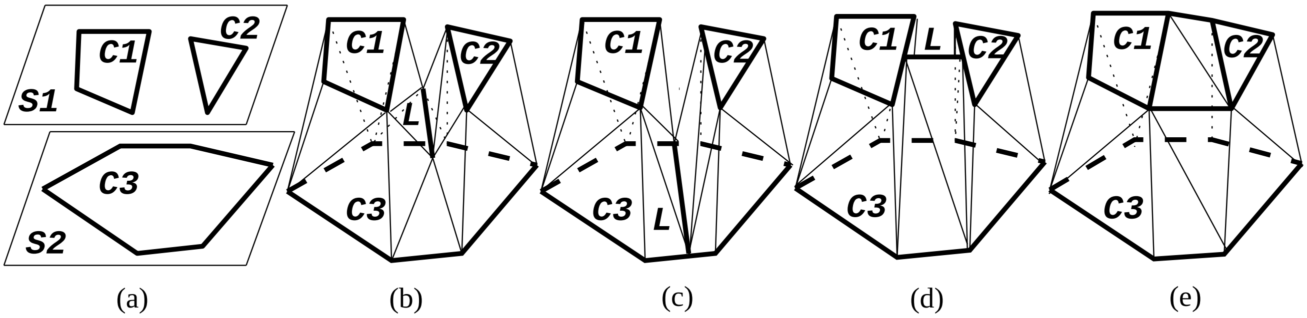


Figure 2: Branching contours: (a) branching contours on adjacent slices, (b)-(e) different surface mesh constructions.

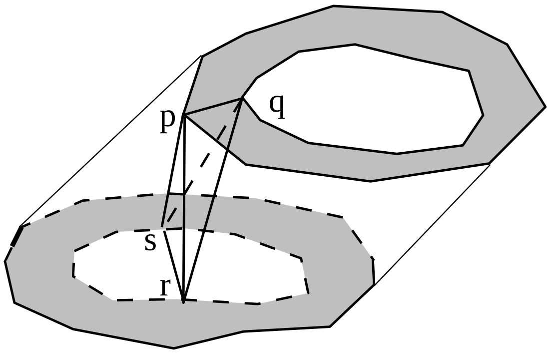


Figure 3: A Delaunay tetrahedron $pqrs$ cuts across the surface mesh.

Boissonnat [6] and Geiger [19] use a different approach than tiling. They applies 3D Delaunay triangulation to the contour vertices of two adjacent slices. The surface mesh of the polyhedron formed by the union of tetrahedra is the desired mesh. Geiger's [19] branching handling is as in Fig. 2(c). Their approach has the the advantage of producing both surface triangles and tetrahedra.

2.2 Tetrahedral Mesh Construction

The construct of a 3D triangular (tetrahedral) mesh of a stack of planar cross-sections can be reduced to the following subproblem. Given the solid bounded by two adjacent contours and surface triangular meshes (referred to as a prismatoid), the goal is to tetrahedralize it with the additional constraint of a pre-triangulated top facet. Note that except for an extreme contour pair, the top facet shall always be triangulated as it occurs as the bottom facet during the triangulation of the upper prismatoid. The tetrahedralization is difficult because the prismatoid can be complicated by holes (non-convexity and higher genus). Furthermore, nice aspect ratio tetrahedron generation is complicated by having the contours in planar slices (i.e. multiple sets of points on a plane and so not in general position for three dimensions).

Extensive research has been conducted in the unstructured tetrahedral mesh generation. Chazelle et. al [10], Field [16], Lo [27] and Bern et. al. [5] provide a good coverage of different approaches toward automatic mesh generation from a polyhedron. These methods include subdivision, octree, Delaunay-based tetrahedral decompositions, and advancing fronts.

The Delaunay-based approaches [6, 8, 19, 39] and the advancing front approaches [9, 20, 22, 23, 26, 27, 28, 31, 32] receive much attention. Lo [27] discusses the difficulties of the Delaunay-based 3D mesh generations. They includes degenerate tetrahedra and also tetrahedra intersecting the surface mesh. For example, Fig. 3 shows a case where the Delaunay tetrahedron $pqrs$ cuts across the inner surface mesh. Recent research by Weatherill et al. [39] attempt a solution to this problem. Their method subdivides the tetrahedra, which cut across the surface mesh, into sub-tetrahedra so the surface mesh is contained in the faces of new tetrahedra. This process of producing a 3D conforming Delaunay triangulations yields a large fragmentation with no polynomial upper bound.

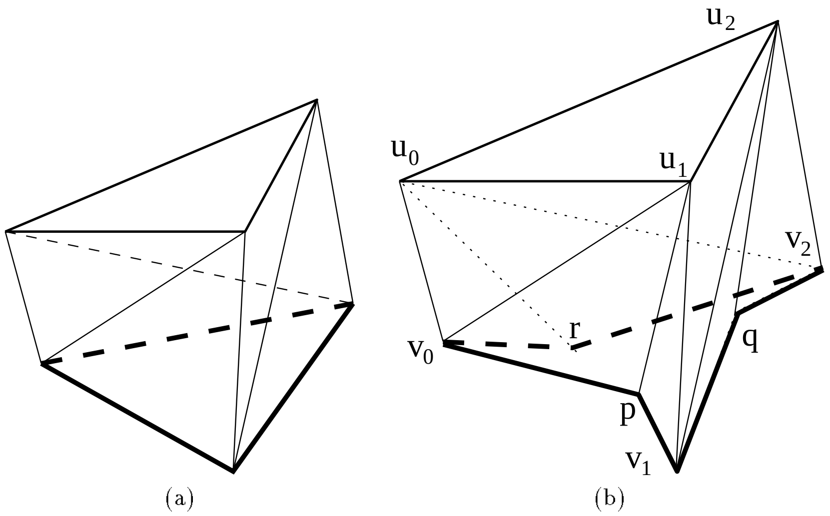


Figure 4: (a) An un-tetrahedralizable Schönhardt prism. (b) The advancing front approach could make it even more twisted.

We discuss prior advancing front approaches in slightly greater detail because our approach draws much from this approach. The following steps illustrate a simplified advancing front approach.

Step 1: form the initial front

Step 2: pick up a triangular face from the front.

Step 3: select a vertex of the front or create a point to form a tetrahedron.

Step 4: update the front

Step 5: if the remaining set of faces is not empty, go to step 2

In step 1, the initial front is simply the triangular faces of an input polyhedron surface mesh. Step 2 has two variations. One is in choosing a face sequentially from the set of boundary faces [26, 27]. The other is to pick up a face based on certain metrics of the tetrahedron to be formed [9].

Step 3 has several variations. One is that additional data points (also called Steiner points) are created on the fly as done by most researches. The other is that Steiner points are created before applying the advancing front approach [26, 27]. In the first approach, the location of created Steiner points affects the shape of the mesh. George et al. [20] use a control field to guide the Steiner point insertion to avoid generating badly shaped elements when two fronts of very different sized facets join.

Our problem domain of tetrahedral mesh generation of prismatoids spanning adjacent planar contours has been studied by Lo [27] and Cavendish et al. [8]. They slice an arbitrary polyhedral object into a stack of prismatoids, and thereafter tetrahedralize each individual prismatoid. Lo's advancing front approach does not guarantee that a slice have the same triangulation for both the upper and lower prismatoid. Cavendish et al. use 3D Delaunay triangulation to tetrahedralize the prismatoids. However, the boundary conformation problem (Fig. 3) is not addressed.

It is believed that the advancing front approaches have flexibility to form good tetrahedra. However, one criticism of advancing front approaches is that they lack a proof of correctness that the front will be ultimately joined correctly or that the remaining part is un-tetrahedralizable. For example, a Schönhardt prism (Fig. 4(a)) [5, 33] can be so twisted that it is un-tetrahedralizable using only vertices on the faces. However, it can be post-processed using Steiner points between two facets. The advancing front approach doesn't know that it is better to leave a Schönhardt prism for post-processing. It keeps generating smaller tetrahedra. For example, the remaining part (Fig. 4(b)) may become even more twisted in an advancing front scheme and become more difficult to be post-processed.

Our approach systematically studies the formation of un-tetrahedralizable shapes. It classifies two most common categories of un-tetrahedralizable shapes so they can be better post-processed. Our study also provides rules to reduce the chance of generating un-tetrahedralizable remaining parts.

3 Surface Mesh Construction Algorithm

This section briefly describe our construction algorithm. Details are discussed in [2].

We address *correspondence*, *tiling* and *branching* problems simultaneously by first defining a set of criteria for the desired surface meshes. The criteria are also chosen to let the produced surface meshes correspond well with expected physical models.

Criterion 1 *The constructed surface meshes and solid regions form piecewise closed surfaces of polyhedra.*

Criterion 2 *Any vertical line segment (perpendicular to the slice) between two slices intersects the constructed surface meshes at zero points, one point, or along line segment.*

Criterion 3 *Re-sampling of the constructed surface meshes on the slice should produce the original contours.*

Criterion 1 prohibits such incorrect structures as self-intersecting surface meshes. Criterion 2 is used to avoid the generation of unlikely topologies. The motivation behind Criterion 3 is obvious.

From these three criteria, we derive explicit tiling and correspondence rules. The correspondence rules determine the correspondences between contours on adjacent slices. The tiling rules prohibit those tilings which result in undesired or nonsensical boundaries, and allow detection of branching regions and dissimilar portions of contours. We develop a multipass tiling algorithm to achieve reasonably good tiling. The algorithm is illustrated in Fig. 5. It first constructs tilings for any regions not violating any of the tiling rules. The first pass constructs the optimal tiling triangles, and the latter passes lower the optimality requirement to build more tiling triangles. Regions that violate these rules correspond to holes, branching regions and dissimilar portions of contours. They are processed by tiling to their medial axes which is placed at the mid-section of two slices.

4 3D triangulation Algorithm

The following is the general sketch of our algorithm. Detail are discussed in [1].

We refer to a slice triangle as a triangle lying on either the top or the bottom slice of the prismatoid. A non-slice triangle is called a side triangle. If a slice triangle contains two contour segments, it is a boundary triangle. A side triangle is either a type 0 or type 1 triangle if it contains a top or bottom contour segment, respectively. A line segment is denoted as $\overline{pq}$. It is convex if the two triangles sharing it form a convex angle ($\leq \pi$). Otherwise, it is reflex.

The triangulation of the top facet is fixed. The branching region is preprocessed to reduce our problem domain to a prismatoid. We first apply a 2D Delaunay triangulation to the bottom slice. We do not add Steiner points to improve the triangulation quality [14, 11]. Ruppert et. al. [33] shows that the problem of deciding whether a given polyhedron can be tetrahedralized without adding Steiner points is NP-complete.

The metric of a tetrahedron is based on its volume/surface ratio as used by Lo [27]. Suppose a boundary triangle $\triangle u_1u_2u_3$ is on the top slice as shown in Fig. 6 (a), there can be zero or more type 1 triangles between the two type 0 triangles $\triangle u_1u_2v_1$ and $\triangle u_2u_3v_n$. Here, n is one plus the number of type 1 triangles. There are n different ways to form tetrahedra containing these side triangles. Fig. 6 (b) and (c) show two examples. The formed tetrahedra must be totally inside the prismatoid and they cannot violate the protection rule described in the following paragraphs. The chosen way is based on the average metric as well as the worst metric of the generated tetrahedra. The metric of a boundary triangle is the best metric of the n ways.

The prismatoid is broken into smaller prismatoids which has about 10-20 side triangles. This step is to dramatically decrease the tetrahedralization time as well as to eliminate through holes of the prismatoid. The cutting quadrilateral is chosen based on the minimum area criterion.

The following sketch provides an outline of the tetrahedralization of each small prismatoid:

Step 1: For each boundary triangle on both slices, calculate its metric.

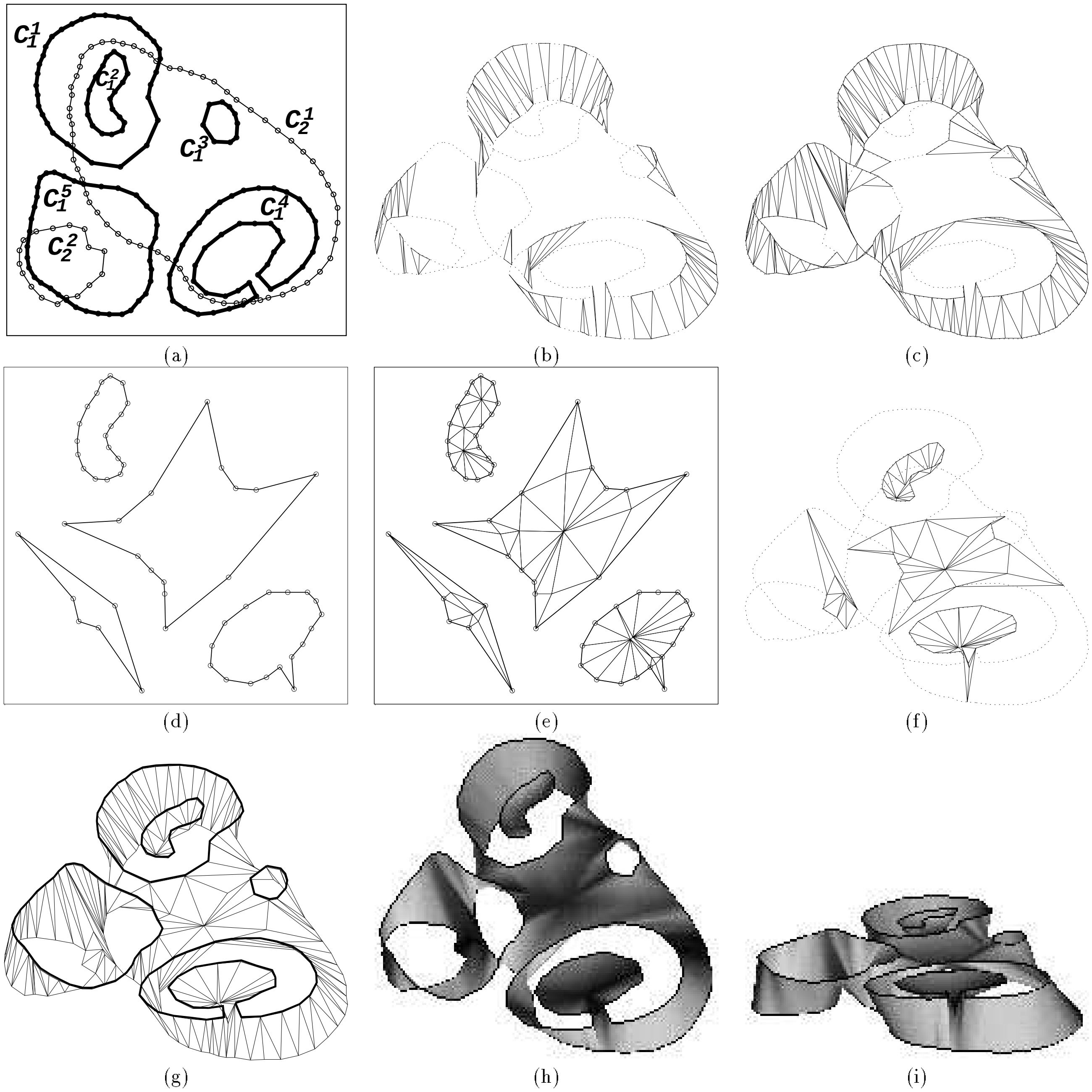


Figure 5: (a) Two slices of contours. Thicker contours are from the top slice. The small circles denote vertices. (b) result of the first tiling pass. Only good tiling triangles are formed. (c) the result of all tiling passes. (d) top view of untiled regions. (e) top view of untiled region triangulation by edge Voronoi diagram. (f) the perspective view of (e) with hidden lines removed. (g) the final result. (h) & (i) two different shaded views of the constructed surface meshes shown in (g).

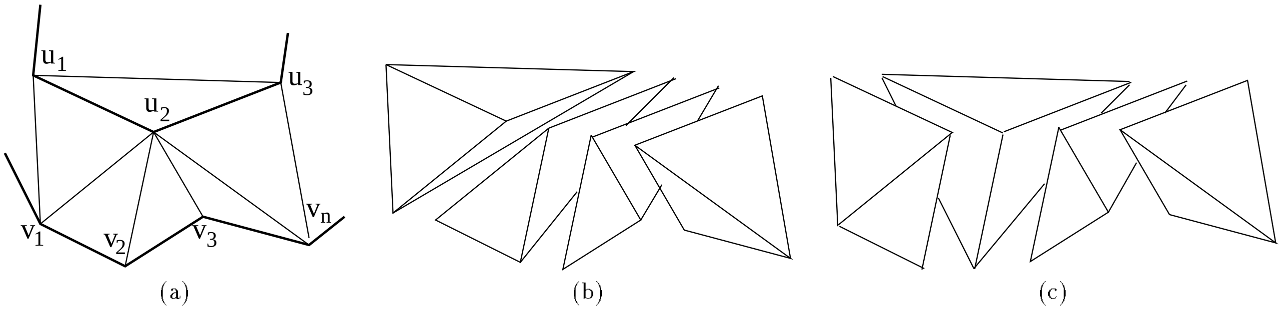


Figure 6: There are different ways to form tetrahedra involving a boundary triangle $\triangle u_1u_2u_3$.

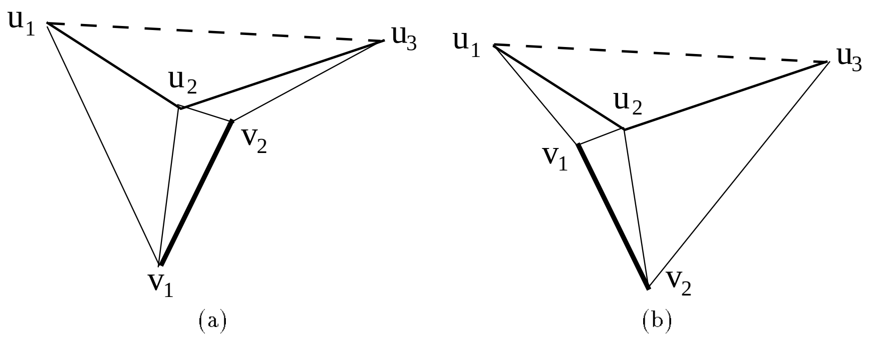


Figure 7: Un-tetrahedralizable shapes defined in Theorem 1. (a) reflex $\overline{u_2v_2}$ (b) reflex $\overline{u_2v_1}$

Step 2: Pick up the boundary triangle with the best metric and form one or more tetrahedra.

Step 3: Update the front and go to Step 1.

Step 4: If no boundary triangle is qualified in Step 2, we swap the bottom edge to have new boundary triangle to try to make it tetrahedralizable. Goto Step 1.

Step 5: If it is un-tetrahedralizable even after edge swapping, we post-process it.

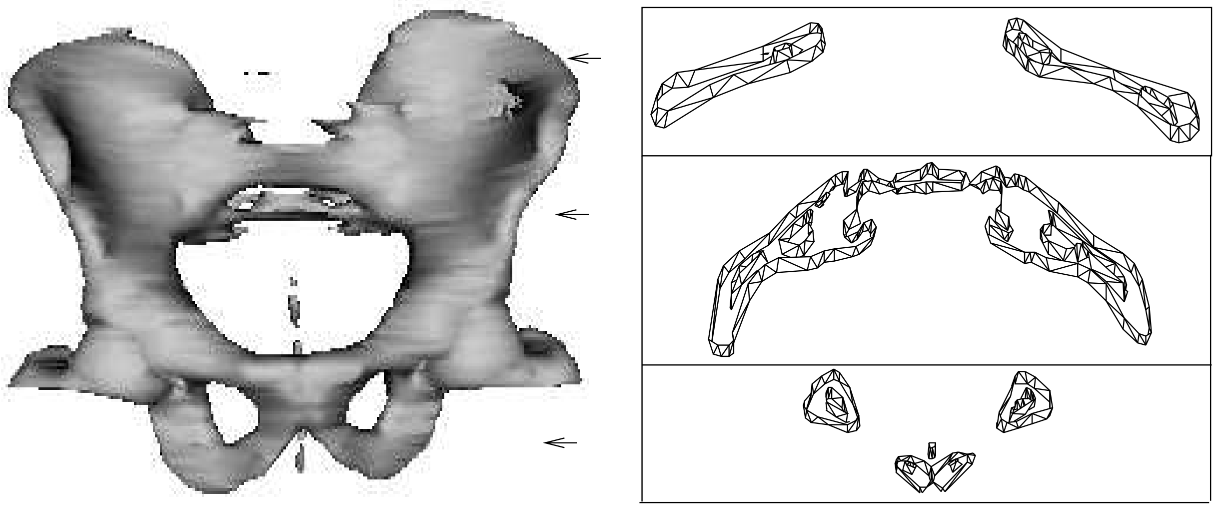
Our algorithm prohibits the formation of a tetrahedron not involving a boundary triangle. The reason is that using a non-boundary triangle to form a tetrahedron complicates the remaining part. Based on our operators, we present the following theorem.

Theorem 1 *Let a top boundary triangle contain two contour segments $\overline{u_1u_2}$ and $\overline{u_2u_3}$ (see Fig. 7), and no more than one type 1 triangle are between the two type 0 triangles containing $\overline{u_1u_2}$ and $\overline{u_2u_3}$. Further, let the bottom vertices of the two type 0 triangles be v_1 and v_2. No tetrahedron containing $\overline{u_1u_2}$, $\overline{u_2u_3}$ or vertex u_2 can be further formed if and only if all the following conditions are satisfied.*

1. *$\overline{v_1v_2}$ is exactly one contour segment.*
2. *One of the slice chords $\overline{u_2v_1}$ and $\overline{u_2v_2}$ is reflex and the other is convex.*
3. *Both $\overline{u_1v_2}$ and $\overline{u_3v_1}$ are not inside the prismatoid.*

Condition 2 of Theorem 1 can be derived from Condition 3. However we state it so it is easier to visualize the un-tetrahedralizable shapes. Fig. 7 shows the only two possible shapes which satisfy Theorem 1.

We define one protection rule based on Theorem 1 to reduce the chance of generating un-tetrahedralizable parts. The rule is that a new tetrahedron cannot satisfy condition 3 of Theorem 1 with respect to any boundary triangle which satisfies conditions 1 and 2. For example, if one boundary triangle has the shape of Fig. 7(a) and $\overline{u_1v_2}$ is totally inside the prismatoid, it is tetrahedralizable. The rule states that any proposed tetrahedron cannot cut across $\triangle u_1u_2v_2$.



(a) (b)

Figure 8: (a) Gouraud shading of the surface meshes. (b) The tiling of some slices.

The bottom face of an un-tetrahedralizable remaining part usually has fewer vertices than the top facet does because the bottom face has the freedom to take any 2D triangulation. From Theorem 1, we classify two categories of un-tetrahedralizable prismatoids. The two categories occur when the bottom facet is a line segment or a single triangle. There are other cases where the bottom facet contains more than one single triangle. The classification of un-tetrahedralizable prismatoids help their post-processing. If an un-tetrahedralizable part is not classified, we post-process it by convex decomposition [3, ?].

5 Results

The algorithms have been implemented in C and C++, and run on Sun Sparc and Silicon Graphics Indigo2 workstations.

5.1 Results of surface mesh construction

Fig. 5 illustrates the capabilities of our algorithm when many-to-many branching, dissimilar contours and holes are present. As can be seen from Fig. 5, C_1^2 is constructed as a shallow hole since it has no corresponding contour on the bottom slice. The dissimilar portion of C_1^4 tiles to its medial axis and forms a shallow hole with a link to C_2^1. This is a highly likely topology. Fig. 8(a) shows the Gouraud shading of the surface mesh construction of a pelvis. It also contains the top portion of femurs. The tiling of three cross sections pointed by the arrows of (a) is shown in Fig. 8(b). The image volume contains 105 256*256 MRI slices.

5.2 Results of 3D triangulation

We have tried our algorithm on several test cases. The protection rule works very well in preventing the generation of un-tetrahedralizable remaining parts. The only encountered un-tetrahedralizable case is a twisted cube which was manually designed. The un-tetrahedralizable remaining part is a Schönhardt prism (Fig. 4(a)) which falls in the second group of our un-tetrahedralizable shape classification. When the protection rule is disabled, some un-tetrahedralizable parts do happen. Edge swapping on the bottom slices is not required in our experiments when the protection rule is enabled. It implies that this algorithm is robust enough to work on prismatoids with both pre-triangulated slices. This feature is required for the parallel processing of a set of prismatoids. Fig. 9(a) shows two faces of looped contours. Fig. 9(b) shows the tiling result, and (c) shows the tetrahedral meshes. This example fails the 3D Delaunay triangulation approaches [6, 19] and those that do not address the boundary conformation problem [8].

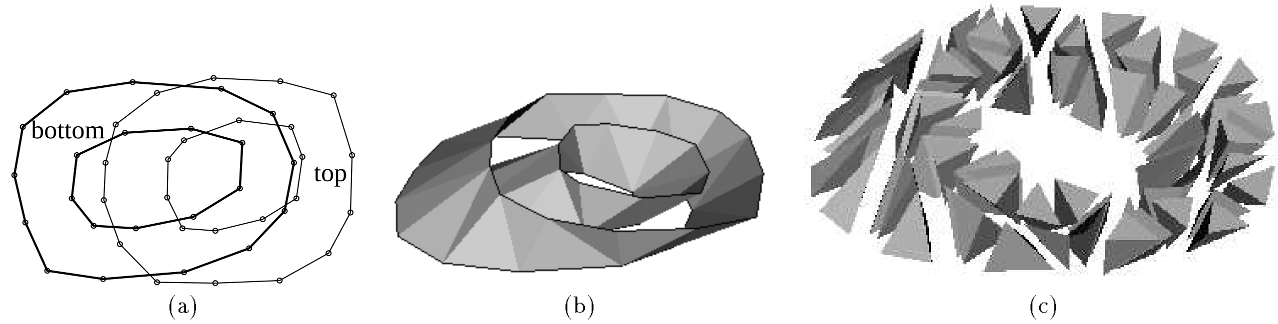


Figure 9: (a) two slices of nested contours. The thicker contours are on the bottom slice. (b) the surface mesh construction. (c) the tetrahedral mesh. The tetrahedra are separated for easy viewing.

6 Conclusion

This paper present mesh generation algorithms for reconstructing surface triangular meshes and tetrahedral meshes from a set of planar contours. Given any input data, the theoretical derivation of the correspondence and tiling rules allowed our algorithm to generate a unique topology satisfying the desired surface mesh criteria. This new approach led to reconstructed triangular meshes which correspond well with the physical surface.

As to the 3D triangulation, our approach systematically studies the formation of un-tetrahedralizable part and classified two most common categories of the un-tetrahedralizable shapes which can be further tetrahedralized by post-processing. This study also provides rules to reduce the chance of generating un-tetrahedralizable polyhedral shapes when Steiner point inclusion is not allowed.

References

[1] C. Bajaj, E. Coyle, and K. Lin. 3D mesh generation for sliced polyhedra. *Draft*.

[2] C. Bajaj, E. Coyle, and K. Lin. Arbitrary topology shape reconstruction from planar cross sections. *To appear in Graphical Models and Image Processing*.

[3] C. Bajaj and T. K. Dey. Convex decomposition of polyhedra and robustness. *SIAM J. Comput.*, 21(2):339–364, 1992.

[4] G. Barequet and M. Sharir. Piecewise-linear interpolation between polygonal slices. In *Proc. 10th Annu. ACM Sympos. Comput. Geom.*, pages 93–102, 1994.

[5] M. Bern and D. Eppstein. *Mesh Generation and Optimal Triangulation, Computing in Euclidean Geometry, edited by D.-Z. Du and F. K. Hwang*, pages 23–90. World Scientific, 1992.

[6] J. D. Boissonnat. Shape reconstruction from planar cross sections. (44):1–29, 1988.

[7] Y. Bresler, J. A. Fessler, and A. Macovski. A Bayesian approach to reconstruction from incomplete projections of a multiple object 3D domain. *IEEE Trans. on Patt. Anal. Mach. Intell.*, 11(8):840–858, Aug. 1989.

[8] J. C. Cavendish, D. A. Field, and W. H. Frey. An approach to automatic three-dimensional finite element mesh generation. *International Journal for Numerical Methods in Engineering*, 21:329–37, 1985.

[9] S. Chae and K. Bathe. On automatic mesh construction and mesh refinement in finite element analysis. *Computers & Structures*, 32(34):911–936, 1989.

[10] B. Chazelle and L. Palios. Triangulating a non-convex polytope. *Discrete Comput. Geom.*, 5:505–526, 1990.

[11] L. P. Chew. Guaranteed-quality mesh generation for curved surfaces. In *Proc. 9th Annu. ACM Sympos. Comput. Geom.*, pages 274–280, 1993.

[12] H. N. Christiansen and T. W. Sederberg. Conversion of complex contour line definitions into polygonal element mosaics. *Computer Grapics*, 12:187–192, Aug. 1978.

[13] L. T. Cook, P. N. Cook, K. R. Lee, S. Batnitzky, B.Y.S. Wong, S. L. Fritz, J. Ophir, S. J. Dwyer III, L. R. Bigongiari, and A. W. Templeton. An algorithm for volume estimation based on polyhedral approximation. *IEEE Trans. on Biomedical Engineering*, BME-27(9):493–499, Sep. 1980.

[14] T. K. Dey, C. L. Bajaj, and K. Sugihara. On good triangulations in three dimensions. *Internat. J. Comput. Geom. Appl.*, 2(1):75–95, 1992.

[15] A. B. Ekoule, F. C. Peyrin, and C. L. Odet. A triangulation algorithm from arbitrary shaped multiple planar contours. *ACM Trans. Graphics*, 10(2):182–199, Apr. 1991.

[16] D. A. Field. The legacy of automatic mesh generation from solid modeling. *Computer Aided Geometric Design*, 12:651–673, 1995.

[17] H. Fuchs, Z. M. Kedem, and S. P. Uselton. Optimal surface reconstruction from planar contours. *Communications of the ACM*, 20(10):693–702, Oct. 1977.

[18] S. Ganapathy and T. G. Dennehy. A new general triangulation method for planar contours. *Computer Grapics*, 16:69–75, 1982.

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

[20] P. L. George and E. Seveno. The advancing-front mesh generation method revisited. *International Journal for Numerical Methods in Engineering*, 37:3605–3619, 1994.

[21] C. Gitlin, J. O'Rourke, and V. Subramanian. On reconstructing polyhedra from parallel slices. Technical Report 25, Dept. Comput. Sci., Smith College, Northampton, MA. 1993. Appeared/to appear in IJCGA.

[22] H. Jin and R. I. Tanner. Generation of unstructured tetrahedral meshes by advancing front technique. *International Journal for Numerical Methods in Engineering*, 36:1805–1823, 1993.

[23] B. P. Johnston and J. M. Sullivan. A normal offsetting technique for automatic mesh generation in three dimensions. *International Journal for Numerical Methods in Engineering*, 36:1717–1734, 1993.

[24] N. Kehtarnavaz, L. R. Simar, and R.J.P. De Figueiredo. A syntactic/semantic technique for surface reconstruction from cross-sectional contours. (42):399–409, 1988.

[25] E. Keppel. Approximating complex surfaces by triangulation of contour lines. *IBM J. Res. Develop.*, (19):2–11, Jan. 1975.

[26] S. H. Lo. A new mesh generation scheme for arbitrary planar domains. *International Journal for Numerical Methods in Engineering*, 21:1403–1426, 1985.

[27] S. H. Lo. Volume discretization into tetrahedra - II. 3D triangulation by advancing front approach. *Computers & Structures*, 39(5):501–511, 1991.

[28] D. L. Marcum and N. P. Weatherill. Unstructured grid generation using iterative point insertion and local reconnection. *AIAA Journal*, 33(9):1619–1625, Sep. 1995.

[29] D. Meyers, S. Skinner, and K. Sloan. Surfaces from contours. *ACM Trans. Graphics*, 11(3):228–258, Jul. 1992.

[30] S. A. Mitchell and S. A. Vavasis. Quality mesh generation in three dimensions. In *Proc. 8th Annu. ACM Sympos. Comput. Geom.*, pages 212–221, 1992.

[31] P. Moller. On advancing front mesh generation in three dimensions. *International Journal for Numerical Methods in Engineering*, 38:3551–3569, 1995.

[32] J. Peraire, J. Peiro, L. Formaggia, K. Morgan, and O. C. Zienkiewicz. Finite element euler computations in three dimensions. *International Journal for Numerical Methods in Engineering*, 26:2135–2159, 1988.

[33] J. Ruppert and R. Seidel. On the difficulty of tetrahedralizing 3-dimensional non-convex polyhedra. In *Proc. 5th Annu. ACM Sympos. Comput. Geom.*, pages 380–392, 1989.

[34] M. Shantz. Surface definition for branching contour-defined objects. *Computer Grapics*, 15(2):242–270, Jul. 1981.

[35] Y. Shinagawa and T. L. Kunii. The homotopy model: a generalized model for smooth surface generation from cross sectional data. *The Visual Computer*, 7:72–86, 1991.

[36] K. R. Sloan and J. Painter. Pessimal guesses may be optimal: a counterintuitive search result. *IEEE Trans. on Patt. Anal. Mach. Intell.*, 10(6):949–955, Nov. 1988.

[37] B. I. Soroka. Generalized cones from serial sections. (15):154–166, 1981.

[38] Y. F. Wang and J. K. Aggarwal. Surface reconstruction and representation of 3-D scenes. *Pattern Recognition*, 19(3):197–207, 1986.

[39] N. P. Weatherill and O. Hassan. Efficient three-dimensional Delaunay triangulation with automatic point creation and imposed boundary constraints. *International Journal for Numerical Methods in Engineering*, 37:2005–2039, 1994.