ELSEVIER

Comput. Methods Appl. Mech. Engrg. 179 (1999) 31–52

Computer methods in applied mechanics and engineering

www.elsevier.com/locate/cma

Tetrahedral meshes from planar cross-sections ☆

Chandrajit L. Bajaj [a,*], Edward J. Coyle [b,1], Kwun-Nan Lin [b,2]

[a] *Department of Computer Science, University of Texas, Austin, TX 78733, USA*
[b] *School of Electrical Engineering, Purdue University, West Lafayette, IN 47907, USA*

Received 20 July 1998

Abstract

In biomedicine, many three-dimensional (3D) objects are sampled in terms of slices such as computed tomography (CT), magnetic resonance imaging (MRI), and ultrasound imaging. It is often necessary to construct surface meshes from the cross sections for visualization, and thereafter construct tetrahedra for the solid bounded by the surface meshes for the purpose of finite element analysis. In Ref. [1] (C. Bajaj, E. Coyle and K. Lin, *Graphical Models and Image Processing* 58 (6) (1996) 524–543), we provided a solution to the construction of a surface triangular mesh from planar -section contours. Here we provide an approach to tetrahedralize the solid region bounded by planar contours and the surface mesh. It is a difficult task because the solid can be of high genus (several through holes) as well as have complicated branching regions. We develop an algorithm to effectively reduce the solid into prismatoids, and provide an approach to tetrahedralize the prismatoids. Our tetrahedralization approach is similar to the advancing front technique (AFT) for its flexible control of mesh quality. The main criticism of AFT is that the remaining interior may be badly shaped or even untetrahedralizable. The emphasis of our prismatoid tetrahedralization approach is on the characterization and prevention of untetrahedralizable parts. Ruppert and Seidel (J. Ruppert, R. Seidel, On the difficulty of tetrahedralizing three-dimensional non-convex polyhedra, in: Proceedings 5th Annual ACM Symposium Comput. Geom., 1989, p. 380–392) have shown that the problem of deciding whether a polyhedron is tetrahedralizable without adding Steiner points is NP-complete. We characterize this problem under certain constraints, and design one rule to reduce the chance of generating untetrahedralizable shapes. The characterization also leads to the classification of two common untetrahedralizable categories which can be better processed if they do occur.

1. Introduction

The finite element method (FEM) has wide applications in simulating a domain subjected to external influences. Typical examples are mechanics, fluid mechanics, and heat transfer. FEM requires the problem domain to be divided into small simple elements. These are, for example, triangles or quadrilaterals in 2D, and tetrahedra, prisms, or hexahedra in 3D. Tetrahedra can approximate complicated 3D regions just as triangles can approximate complicated 2D regions.

Many 3D objects are sampled in terms of slices. For example, technologies such as magnetic resonance imaging (MRI), computed tomography (CT), and ultrasound imaging obtain measurements of internal properties of objects in a non-destructive 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.

☆ Research sponsored in part by NSF-CCR-9732306.

* Corresponding author. Tel.: +1-512-471-8870; fax: +1-512-492-0921; e-mail: bajaj@cs.utexas.edu
[1] E-mail: coyle@ecnpurdue.edu
[2] E-mail: klin@cs.purdue.edu

0045-7825/99/$ – see front matter
PII: S0045-7825(99)00030-4

Once these measurement slices have been obtained, the goals are to enable a human to easily visualize, in 3D, this large collection of data, as well as to simulate physical properties of the data. This first requires triangular surface mesh construction from the slice data, and the second utilizes tetrahedral element generation of the solid domain bounded by the surface mesh. In Ref. [1] we present an algorithm to achieve surface construction from slices. The images are automatically or manually segmented to obtain contours, and triangular surface meshes are constructed from planar contours. This contour-based approach generates a compact number of surface elements. This paper details our approach to tetrahedralize the polyhedron bounded by the constructed surfaces generated by our algorithm in Ref. [1].

The rest of this paper is as follows. We provide an overview of previous approaches in Section 2, then detail our algorithm to reduce the solid domain into prismatoids in Section 3. Our approach to tetrahedralize a prismatoid is presented in Section 4. We present several examples in Section 5, and conclude in Section 6.

2. Overview of previous approaches

The construction 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 as defined in Section 3.3), the goal is to tetrahedralize it with the additional constraint of pre-triangulated top and bottom facets. Except for an extreme contour pair, the top facet shall always be triangulated when it occurs as the bottom facet during the triangulation of the upper prismatoid. For a parallel algorithm, where each region within adjacent cross-sections can be dealt with independently, both facets must be pre-triangulated. The tetrahedralization is difficult because the prismatoid can be complicated by through holes (higher genus). Furthermore, good aspect ratio tetrahedra generation is complicated by having the contours in planar slices (i.e. multiple sets of points on a plane and hence not in general position for three dimensions).

Extensive research has been conducted on unstructured tetrahedral mesh generation. Refs. [2,3,9,11,19] provide a good coverage of different approaches toward automatic mesh generation from a polyhedron. These methods include octree decomposition, convex decomposition, Delaunay triangulation, and advancing front technique (AFT). Of these methods, the Delaunay-based approaches [6,5,12,26] and AFT [18,24,8,16,19,14,13,21,22] have received much attention in recent years.

Lo [19] discusses the difficulties of the Delaunay-based 3D mesh generations. The difficulties include degenerate tetrahedra and also tetrahedra intersecting the surface mesh. For example, Fig. 1 shows a case where the Delaunay tetrahedron *pqrs* cuts across the inner surface mesh. Recent research by Weatherill and Hassan [26] attempts 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 triangulation yields a large fragmentation with no polynomial upper bound on the required final number of subdivided tetrahedra.

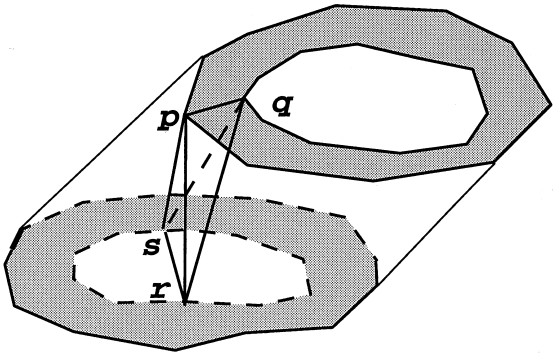

Fig. 1. A Delaunay tetrahedron *pqrs* cuts across the surface mesh.

We discuss prior 3D advancing front approaches in slightly greater detail because of the strong similiarity to our approach. The following steps illustrate a simplified advancing front approach.

Step 1: Form a control background.

Step 2: Form the initial front.

Step 3: Pick a triangular face from the front.

Step 4: Select a vertex of the front or create a point to form a tetrahedron.

Step 5: Update the front.

Step 6: If the remaining set of faces is not empty, go to Step 3.

The control background is an optional method to control the sizes and directions of formed mesh elements and/or to prevent problems caused by joining widely varying fronts. The control background can be a mesh [20,24], a regular grid [13], or a control line/surface scheme [14]. The desired properties are stored with the nodes of the background grid/mesh, or they are the size of the line/surface scheme. Linear interpolation is necessary to determine mesh parameters of the nodes of the background grid/mesh.

The initial front of Step 2 is simply the triangular faces of an input polyhedron surface mesh.

Step 3 has three variations. The first method is to choose a face sequentially from the data structure containing boundary faces. Dannelongue and Tanguy [10] pick the root of the tree structure of boundary faces as the next face. Lo [19] chooses the last item of the face set. The second method is to pick a face based on certain metrics of the tetrahedron to be formed. Chae and Bathe [8] pick the next face based on the type of operations applicable to the face. For example, the operation of removing a corner has higher priority than extracting a tetrahedron from the solid. If two or more faces have the same highest priority operation, other factors such as face area ratio are considered. Löhner and Parikh [20], Möller and Hansbo [22], and Peraire et al. [24] pick the smallest face on the front to avoid the problem of large elements crossing over regions of small elements. Jin and Tanner [14] also consider the positions of a face relative to its neighboring faces in addition to the size criterion. The drawback of the first two methods is the complexity of the remaining part described in Step 6. Although the second method alleviates this problem by considering certain metrics in choosing the next face, the remaining part still becomes complicated after digging operations. Digging operations form a tetrahedron using a triangular face and a vertex which is not on any adjacent face to the triangular face.

Instead of picking a facet, the third method is to construct an offset layer toward the interior of the object. The offset layer is tetrahedralized, and the remaining part becomes smaller. The remaining part also becomes simpler if the inner faces of the offset layer have no sharp corner. This process is repeated until there is no remaining part. This method is considered to be a special variation of AFT, and Step 4 of AFT is no longer needed. Forming a well conditioned offset layer in 3D is not a trivial problem. Johnston and Sullivan [16] offset points along vectors normal to faces. In order to avoid the problems of sharp corners, layer self intersection, and dense/sparse node concentrations, they adjust, add, merge, or delete nodes of the inner faces of the offset layer. The process is done by applying four checks, distance, angle, intersection and termination, to the layer.

Step 4 has two variations. One is that additional internal points (also called Steiner points) are created as needed by most techniques. The other is that Steiner points are created before Step 2 is applied [18,19]. Both variations perform tests to check that no part of a proposed tetrahedron is outside the front. In both approaches, the selection of a node is usually based on metric calculation. Lo [19] calculates the volume to surface area ratio of a proposed tetrahedron to choose an existing node on or inside the front for a selected face. In the first approach, the control background described in Step 1 can be used to control the size and direction of a mesh element. This approach has more freedom to form a well formed tetrahedron by creating a new node respective to a selected face. However, this approach faces the choice of using an existing node or creating a new node, and then the decision of the location of a created node. Chae and Bathe [8] attempt to form a tetrahedron using a face and an existing node subject to edge angle restrictions. If no existing node is qualified for any face, Chae and Bathe create a new node after the consideration of surrounding faces, surrounding edges, and a scaling factor. The approaches of Peraire et al. [24], Löhner and Parikh [20], Jin and Tanner [14], and George and Seveno [13] are basically similar. The ideal position of a node to form a tetrahedron with a face is calculated. The position usually has equal distances to the vertices of the face. The distance is determined from the face size and the control background, if applicable.

Then existing nodes in proximity to the ideal position are tested to satisfy some conditions. In the case of no qualified existing node, a new node is created as close to the ideal position as possible.

We believe that AFT has flexibility to form good tetrahedra. The flexibility comes from the freedom to choose or create a node to form a tetrahedron in Step 4. On the other hand, this freedom largely contributes to the complexity of the remaining shape. This problem does not apply to the advancing by a layer approach of Johnston and Sullivan [16]. The complex remaining shape leads to the main criticism of AFT: there is no proof that two fronts will join correctly [13,15]. In other words, the remaining part of Step 6 may be irregularly shaped or even untetrahedralizable. This problem has been acknowledged by [7,14,22]. The solution of Möller and Hansbo is to save the current state at regular intervals. Thus when AFT fails, this technique can recover the last state and try a new path of generation.

Our problem domain of tetrahedral mesh generation of prismatoids has been studied by Cavendish et al. [6]. They slice an arbitrary polyhedral object into a stack of prismatoids, and thereafter tetrahedralize each individual prismatoid by 3D Delaunay triangulation. They do not address the boundary conformation problem (Fig. 1) associated with 3D Delaunay triangulation. Furthermore a polyhedron of certain topology, such as branching, cannot be sliced only into prismatoids.

Our approach consists of two parts. The first part is to reduce a non-prismatoid, resulting from certain topology such as branching, into one or more prismatoids. The second part is to tetrahedralize a prismatoid.

Although the concept of our prismatoid tetrahedralization is based on AFT, it is significantly different from other AFT approaches. Instead of picking a face from the front to form one tetrahedron at a time in Step 4, we pick a triangle from the top or bottom slice to form a group of tetrahedra. The construction of a group of tetrahedra is chosen from among all possible tetrahedralization permutations of that group. So we can alleviate the problem that forming a tetrahedron might later cause other faces: that is, to form ill-shaped tetrahedra. The major difference is that we do not allow adding Steiner points in Step 4 because forming a tetrahedron with a Steiner point usually complicates the remaining part. If Steiner points are required, we can add Steiner points before Step 2 just as in Lo's [19] approach. However, before tetrahedralization, we break a prismatoid with Steiner points into prismatoids without any Steiner point.

Our research emphasizes the characterization and prevention of an untetrahedralizable interior, the aspect which has been the main criticism of AFT. We characterize untetrahedralizable shapes in our problem domain, and then provide a protection rule to reduce the chance of generating untetrahedralizable remaining parts. The characterization also leads to the classification of two common untetrahedralizable categories so they can be better postprocessed if they do occur. For example, a Schönhardt prism (Fig. 2(a)) [3,25] cannot be tetrahedralized without adding any Steiner point. It can be tetrahedralized by adding a Steiner point which is visible to all faces. Traditional AFT cannot characterize an untetrahedralizable shape. AFT keeps generating smaller tetrahedra, and the remaining part (Fig. 2(b)) becomes too complicated to be processed. Our approach will recognize untetrahedralizable shapes and leave them for post-processing.

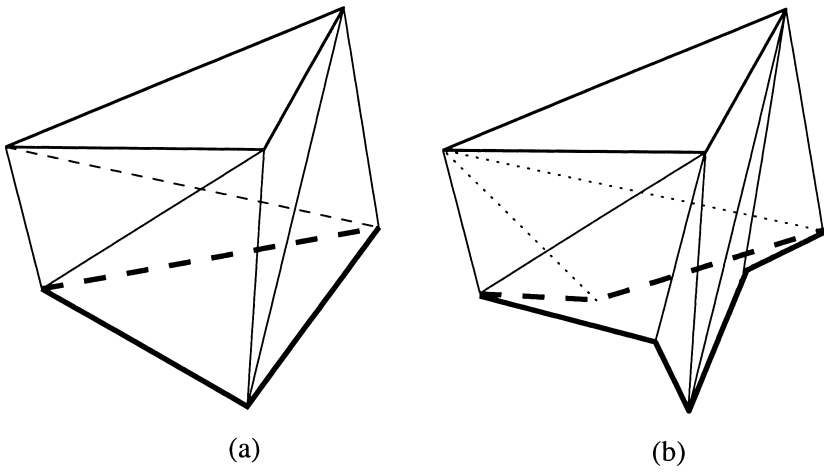

Fig. 2. (a) An untetrahedralizable Schönhardt prism; (b) AFT could result in a more irregular remaining part.

The description of our approach and its theoretical background are presented in Section 4. The next section details our algorithm to divide a non-prismatoid into one or more prismatoids.

3. Preprocessing

The polyhedron constructed from contours on two adjacent slices may not be a prismatoid if it develops any untiled region during our surface construction process. It is more difficult to tetrahedralize a non-prismatoid than a prismatoid because of the former's more complicated shape. This section discusses how preprocessing divides a non-prismatoid polyhedron into one or more prismatoids. We quickly review our surface construction algorithm in Section 3.1, so we can discuss the problem domain in Section 3.2. The reducing algorithm is presented in Ref. Section 3.3, Its appropriateness is discussed in Ref. [17]. Finally, Section 3.4 states the applicable conditions of the reducing algorithm.

A prismatoid is defined by Beyer [4] as the following: "A prismatoid is a polyhedron having for bases two polygons in parallel planes, and for lateral faces triangles or trapezoids with one side lying in one base, and the opposite vertex or side lying in the other base, of the polyhedron" (page 128). In our definition of a prismatoid, we limit the lateral faces to triangles. We also allow a polygon segment to exactly overlap another polygon segment, or allow a polygon to degenerate into a point. Fig. 3 shows some examples. We treat a degenerate polygon as a simple polygon by separating the overlapped polygon segments by an arbitrary small distance (see Fig. 4). Hence the prismatoids based on our definition are still considered to be manifold polyhedra, w hich require that any edge must be shared exactly by two polygons. In our implementation, we do not really separate a pair of overlapped contour segments by a small distance. We simply treat them as two different contour segments. One or more void prismatoids could be inside one prismatoid and form nested prismatoids such as in Fig. 5.

3.1. Sketch of our surface construction algorithm

We will quickly review our surface construction algorithm for completeness. The detailed procedure is described in Ref. [1]. Our algorithm constructs surfaces from contours of two adjacent slices. All contours are simple polygons which are oriented so the solid region is inside a counter-clockwise (CCW) contour and is outside a clockwise (CW) contour. We present some definitions before discussing the algorithm.

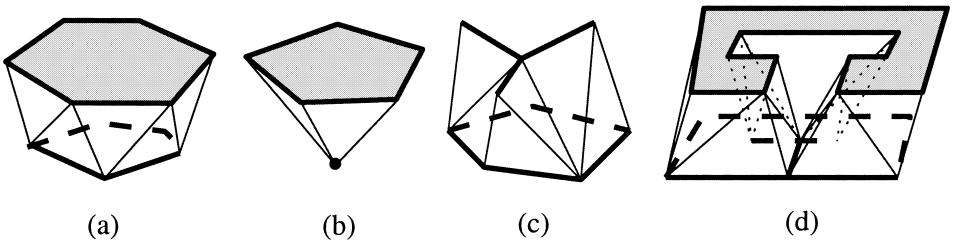

Fig. 3. Some examples of prismatoids based on our definition.

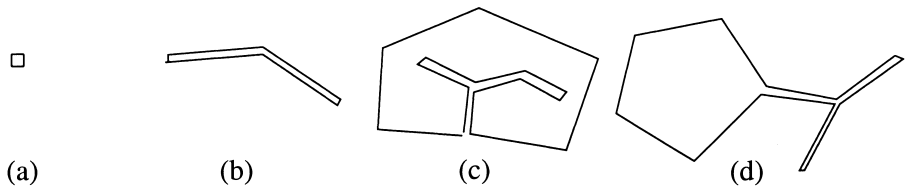

Fig. 4. The degenerate polygons are treated as simple polygons. (a) The entire polygon degenerates into a point. (b) The entire polygon degenerates into line segments. (c) and (d) A contour segment can exactly overlap another contour segment.

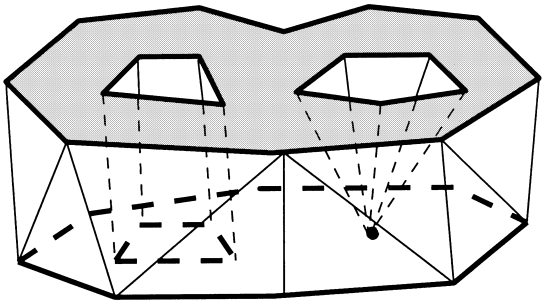

Fig. 5. Nested prismatoids.

Definition 1. "Planar" means that all vertices of an object are on the same slice.

Definition 2. Planar solid region: a region which is inside a CCW contour and is outside CW contours. The shadow regions of Fig. 6 are examples.

Definition 3. A top, middle, or bottom vertex is a vertex on the top, the imaginary middle, or the bottom slice, respectively.

Definition 4. Tiling triangle, middle triangle, and slice chord: A tiling triangle consists of exactly one contour segment and two slice chords (see Fig. 6). A slice chord is an edge of the constructed surface and its two end points are vertices of the top and the bottom slices. A middle triangle contains one or two middle vertices. Its orientation is defined as the orientation of its projection onto one slice.

Our algorithm uses a multi-pass tiling approach followed by the postprocessing of untiled regions. Tiling means using slice chords to triangulate the strip lying between contours of two adjacent slices into tiling triangles. The tiling algorithm is mainly based on the constraint of single-sheeted constructed surfaces between two slices. This constraint implies that any vertical line (perpendicular to a slice) can intersect the constructed surface at no point, one point, or one line segment. Please note that a planar solid region does not belong to the constructed surface, so a vertical line can have two intersections with planar solid regions on both slices.

After the tiling passes of our surface construction algorithm, some regions might not be covered by tiling triangles because covering them with tiling triangles would violate the single-sheeted surface criterion. An untiled region is an oriented 3D polygon which consists of slice chords and/or unused contour segments. It is postprocessed with tiling to its medial axis, which is placed at the imaginary middle slice. So the post-processing covers an untiled region with middle triangles. The projections of middle triangles have the same

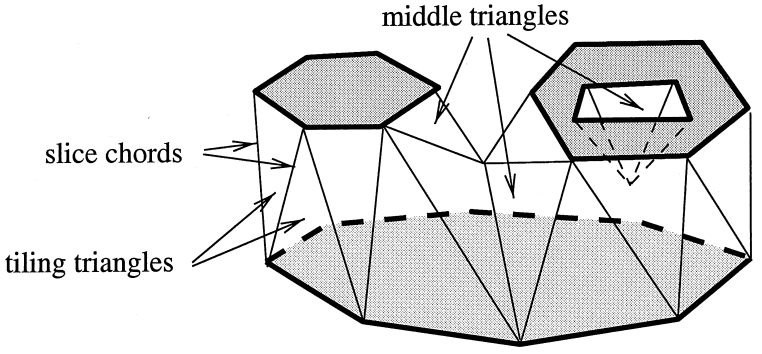

Fig. 6. The shaded regions are planar solid regions. The constructed surface consists of all non-planar triangles.

orientations (CCW or CW) as the projection of the untiled region. An untiled region occurs at a branching region, a dissimilar portion, or an appearing/disappearing vertical feature. Figs. 7–9 show three examples.

The constructed surface consists of non-planar triangles. All triangle are oriented so their surface normals are calculated in the CCW direction. The constructed surface is guaranteed to be single-sheeted. All vertices of the constructed surface belong to three planes: the top slice, the bottom slice, and the middle slice.

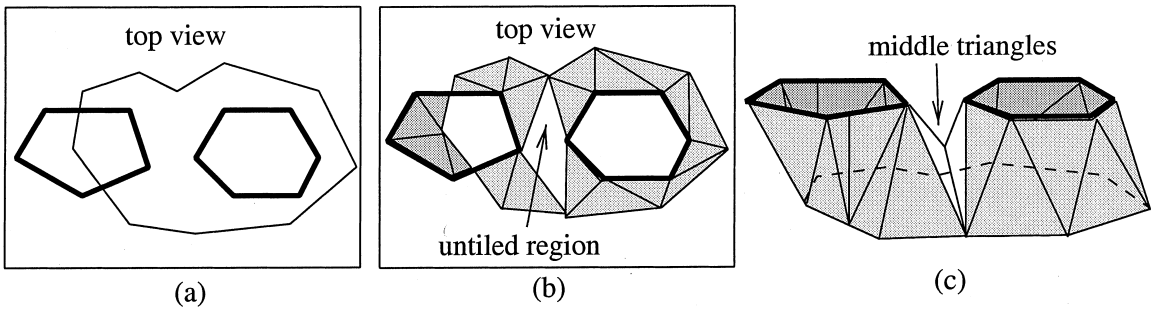

Fig. 7. An untiled region occurs at branching region. Shaded regions shoe the tiling triangles. (a) The thick contours are on the top slice. (b) The branching region is untiled. (c) Perspective view of the constructed surface. The untiled region is covered by middle triangles.

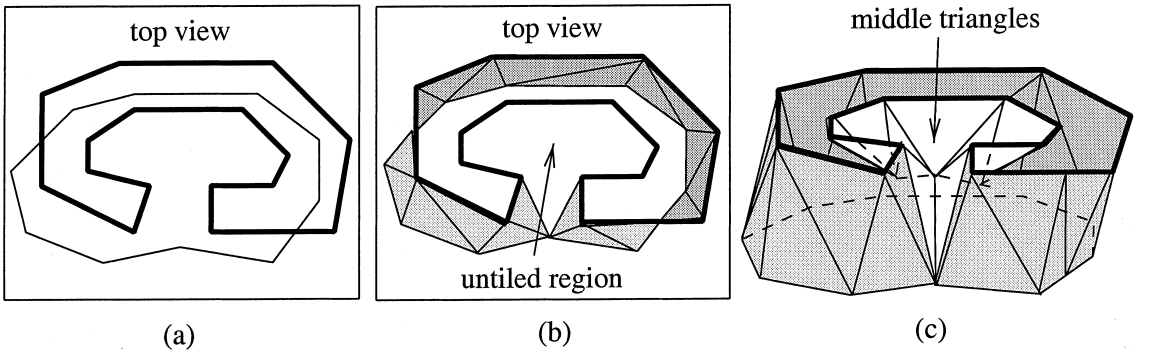

Fig. 8. An untiled region occurs at a dissimilar portion of a contour. Shaded regions show the tiling triangles. (a) The thick contour is on the top slice. (b) The dissimilar portion is untiled. (c) Perspective view of the constructed surface. The untiled region is covered by middle triangles.

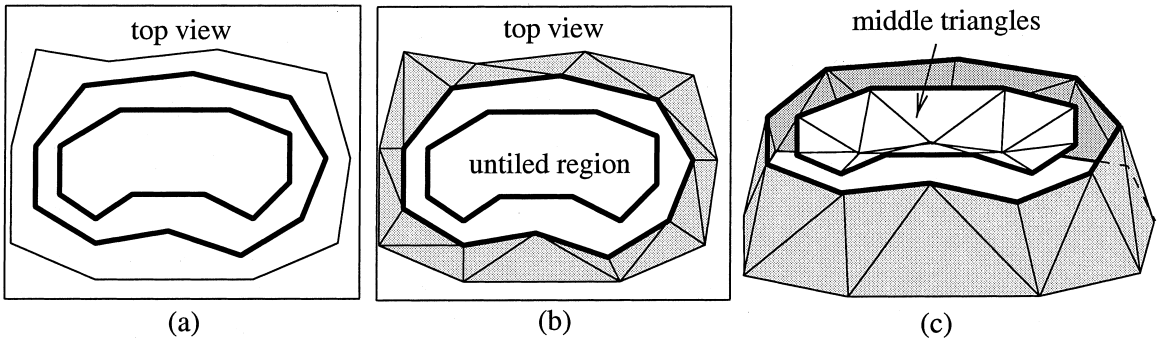

Fig. 9. An untiled region occurs at an appearing/disappearing vertical feature. Shaded regions show the tiling triangles. (a) The thick contours are on the top slice. (b) The inner top contour is untiled. (c) Perspective view of the constructed surface. The untiled region is covered by middle triangles.

3.2. Problem domain

The problem domain is the polyhedron bounded by the planar solid regions of both slices (the shaded regions of Fig. 6) and the constructed surface (the non-planar triangles of Fig. 6) which is generated by our surface construction algorithm. The constructed polyhedron can have through holes. It does not have any internal void because of the single-sheeted surface criterion.

If a constructed polyhedron has no middle triangle, it is either a prismatoid or nested prismatoids. If it contains any middle triangle, it is usually not a prismatoid because the vertices of the polyhedron are on three planes (the top, bottom, and middle slices). It could still be a prismatoid in the case of disappearing/appearing vertical features when there is no top or bottom contour so that all vertices are on two planes.

3.3. The reducing algorithm

If the constructed polyhedron contains at least one middle triangle, and the vertices of the polyhedron are on three planes, then it is not a prismatoid. This subsection will describe an algorithm to reduce the polyhedron into one or more prismatoids. The ideal is to cut tetrahedra from the polyhedron, so the remaining part is a prismatoid or can be divided into prismatoids. Of course, the dissected tetrahedra are part of the generated mesh to ensure that the union of the tetrahedral elements equals the original polyhedron.

The reducing algorithm starts with cutting tetrahedra from the constructed polyhedron. If a CCW middle triangle has any top vertex, we form a polyhedron using the middle triangle and the projections of its middle vertices onto the bottom slice. Fig. 10 shows all possible polyhedra. This polyhedron is a tetrahedron if the middle triangle has only one middle vertex (Fig. 10(a) and (b)). If the middle triangle has two middle vertices (Fig. 10(c)), the polyhedron can be divided into two tetrahedra by triangulating the rectangle $wvv'w'$. The rectangle $wvv'w'$ is shared twice by two polyhedra at both sides. So the triangulation of rectangle $wvv'w'$ must be consistent for both polyhedra. If a CW middle triangle has any bottom vertex, we form a polyhedron using the middle triangle and the projections of its middle vertices onto the top slice. We cut these tetrahedra from the constructed polyhedron. The remaining part is either a prismatoid or can be divided into prismatoids by tracing all non-planar triangles. The correctness of this algorithm is proved in Ref. [17].

3.4. The applicable conditions for use of the reducing algorithm

The algorithm for reducing the constructed polyhedron into prismatoids is based on the property that the projection of a middle vertex onto the appropriate slice falls into a planar solid region. The single-sheeted surface criterion of our surface construction algorithm is a sufficient, but not a necessary, condition

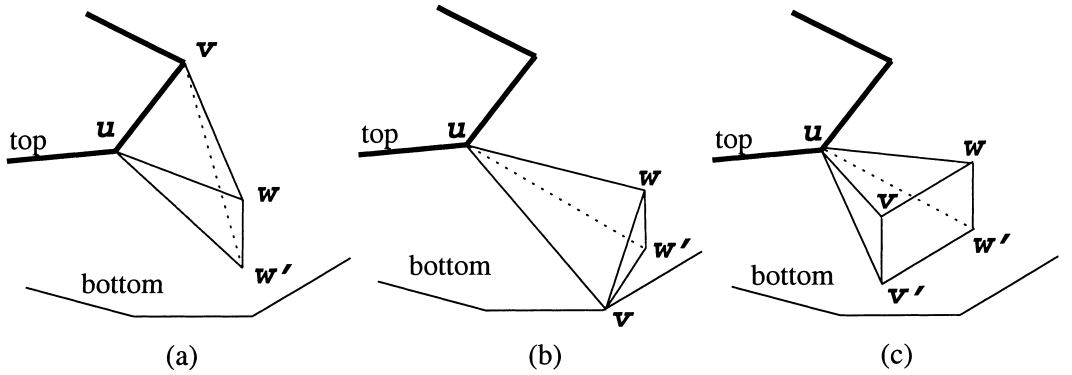

Fig. 10. If a CCW middle triangle contains any top vertex, there are three possible situations to cut a tetrahedron or a polyhedron from the constructed polyhedron. w' and v' are the projections of the middle vertices w and v onto the bottom slice, respectively. $\triangle uvw$ has (a) two top vertices and no bottom vertex, (b) one top vertex, one middle vertex w and one bottom vertex, or (c) one top vertex and two middle vertices w and v.

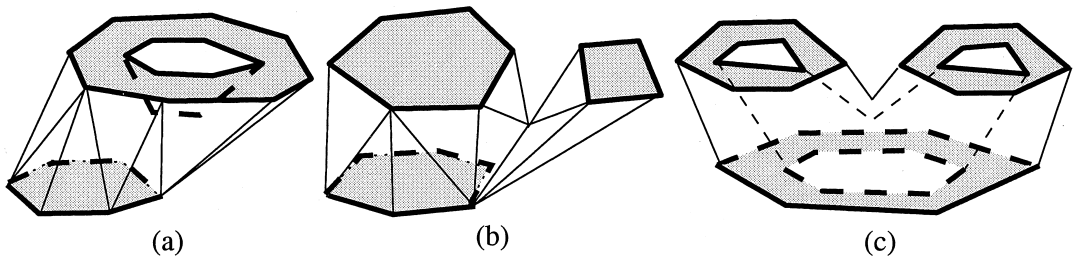

Fig. 11. Three examples which cannot be reduced into prismatoids by our algorithm because the projections of the middle vertices onto the bottom slice do not fall inside a planar solid region (the shaded region).

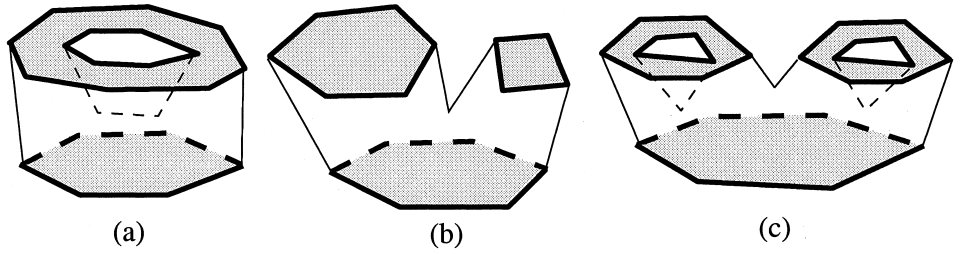

Fig. 12. Our tiling algorithm can generate these examples, and thus they can be reduced into prismatoids by our reducing algorithm.

for this property. Our reducing algorithm is guaranteed to work on the results of our surface construction algorithm. For any given surface, if the projection of a middle vertex does not fall into a planar solid region, our reducing algorithm does not work. Fig. 11 shows three examples of polyhedra which cannot be reduced by our reducing algorithm. However, our surface construction algorithm will not generate these types of polyhedra. Fig. 12 shows three other similar, yet plausible, examples where the projection of any middle vertex onto the proper slice falls inside a planar solid region.

4. Prismatoid tetrahedralization

Our problem domain is the polyhedron bounded by two planar solid regions and the surfaces constructed by our surface construction algorithm. If the polyhedron is not a prismatoid, the algorithm described in the previous section divides it into one or more prismatoids. The triangulation of both the top and bottom facets has been predefined (which are 2D constrained Delaunay triangulated). A prismatoid with Steiner points is broken into prismatoids without Steiner points. All prismatoids are further broken into small prismatoids of ten to twenty faces each. This section first presents a brief sketch of our prismatoid tetrahedralization approach and then discusses the effort to characterize untetrahedralizable prismatoids and one protection rule. Finally, the classification of untetrahedralizable shapes is described. The prismatoid breakdown and detailed implementation of our approach are presented in Ref. [17]. The time complexity is $O(n^2)$. Here, n is the number of vertices.

4.1. Tetrahedralization approach

We define more terms here in addition to those defined in Section 3.3.

Definition 5. Boundary triangle: a planar triangle which contains two contour segments.

Definition 6. Type 0 or type 1 triangle: A non-planar face of a prismatoid is either a type 0 or type 1 triangle if it contains a top or bottom contour segment, respectively.

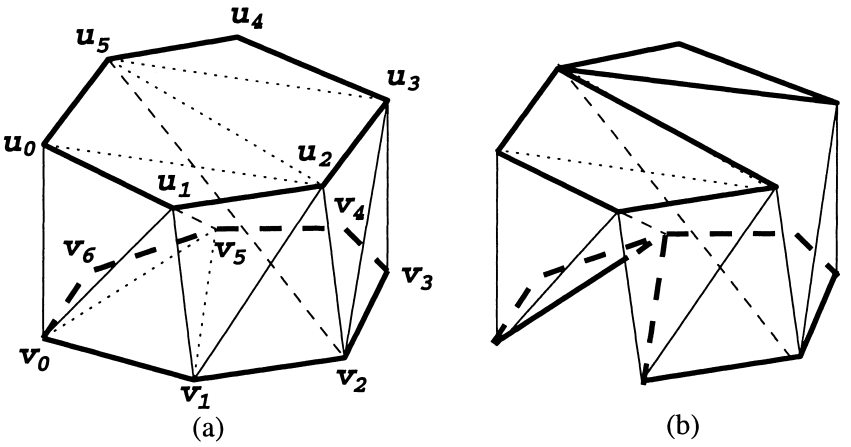

Fig. 13. Forming tetrahedra using non-boundary triangles often makes the remaining part untetrahedralizable. (a) Two tetrahedra formed using non-boundary triangles $u_2u_3u_5$ and $v_0v_1v_5$. (b) The remaining part.

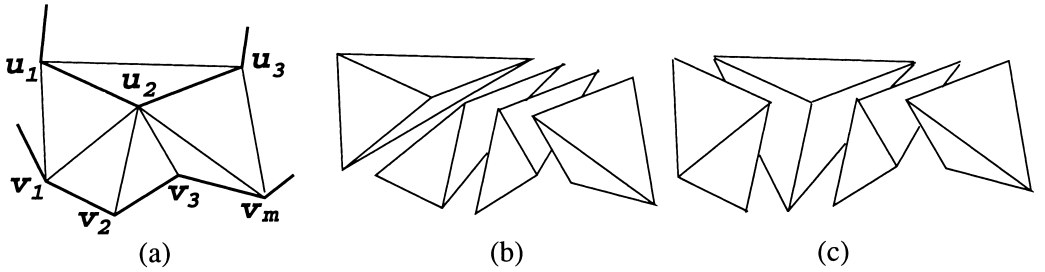

Fig. 14. There are different ways to form tetrahedra associated with a boundary triangle $\triangle u_1u_2u_3$.

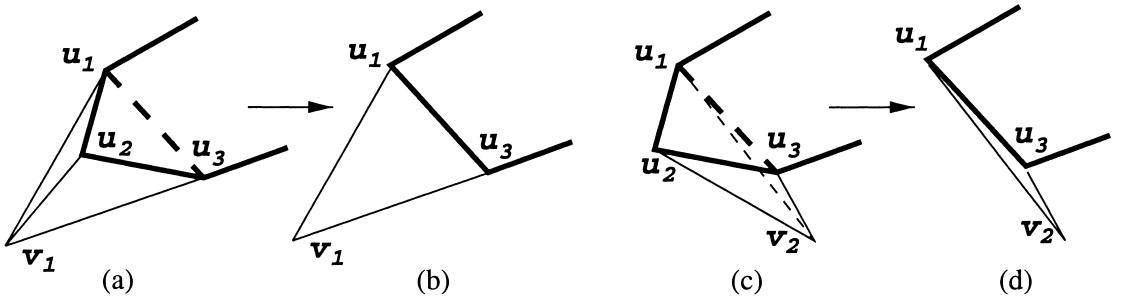

Fig. 15. The weight factor. (a) and (c) show forming a tetrahedron, and (b) and (d) show the their remaining part, respectively. The weight factor of (a) is smaller than 1 and that of (c) is larger than 1.

Definition 7. Convex or reflex edge: A line segment is convex if the two triangles sharing it form a convex angle ($\leqslant \pi$). Otherwise, it is reflex.

The following outlines our algorithm to tetrahedralize a non-nested prismatoid without Steiner points. The triangulation of the top and bottom faces has been defined.

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

Step 2: We pick up the boundary triangle with the best metric and form one group of tetrahedra.

Step 3: We update the front and go to Step 1.

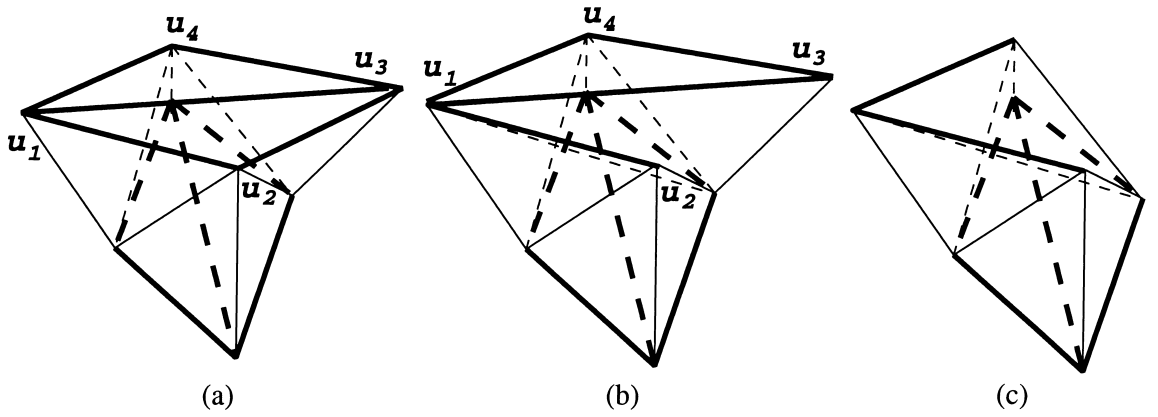

Fig. 16. Our group operator can avoid generating an irregular remaining part. (a) It is untetrahedralizable by our group operator. (b) The remaining part after forming one tetrahedron. (c) The remaining part after forming two tetrahedra.

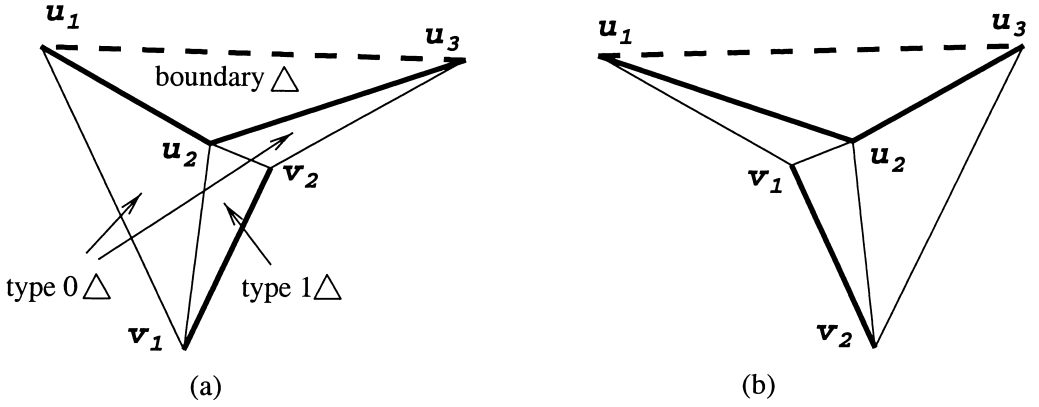

Fig. 17. Untetrahedralizable shapes defined in Lemma 5: (a) reflex $\overline{u_2v_2}$; (b) reflex $\overline{u_2v_1}$.

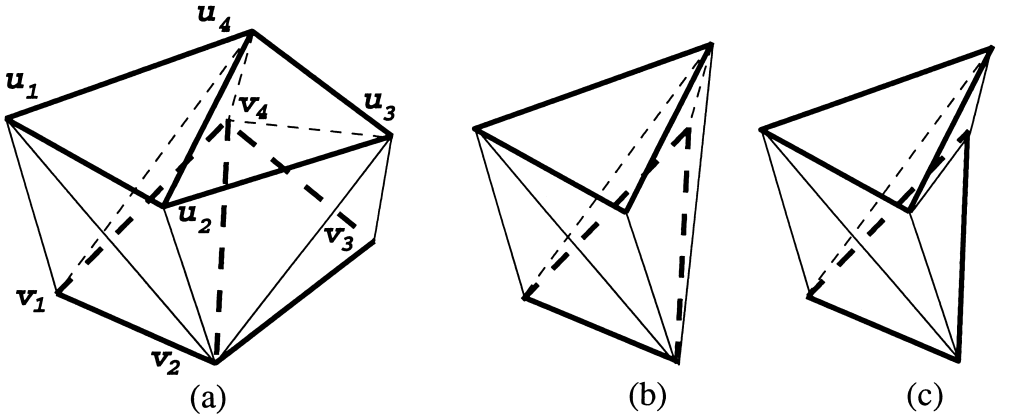

Fig. 18. (a) A prismatoid in which $\overline{u_1v_2}$ and $\overline{u_4v_1}$ are reflex. $\triangle u_4u_1v_2$ is protected. (b) No tetrahedra can be formed via our operator to cut across $\triangle u_4u_1v_2$. (c) The protection rule prohibits the generation of this untetrahedralizable remaining part because $\triangle u_4u_1v_2$ is cut across by $\overline{u_2v_4}$.

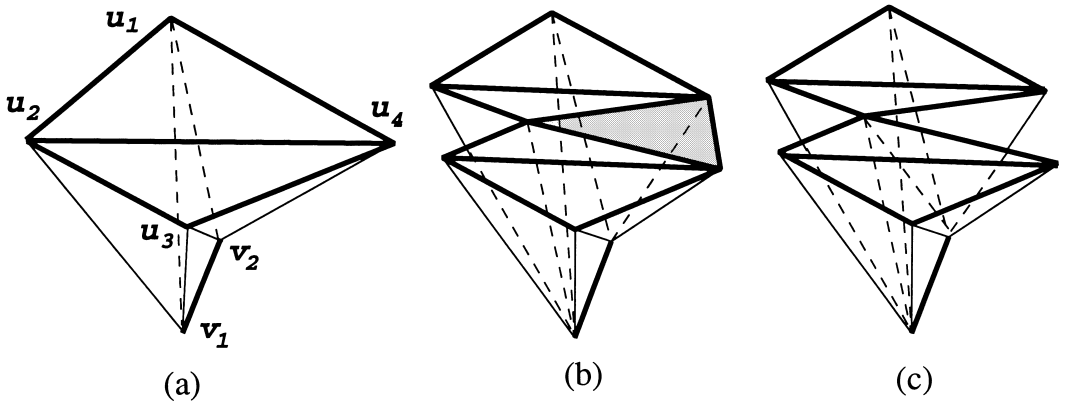

Fig. 19. Untetrahedralizable prismatoids which have two top boundary triangles and no bottom triangle: (a) the simplest case; (b) one variation which has non-boundary triangles between two top boundary triangles; (c) if a tetrahedron is formed using a non-boundary triangle (e.g. the shaded triangle in (b)), the remaining part becomes very irregular.

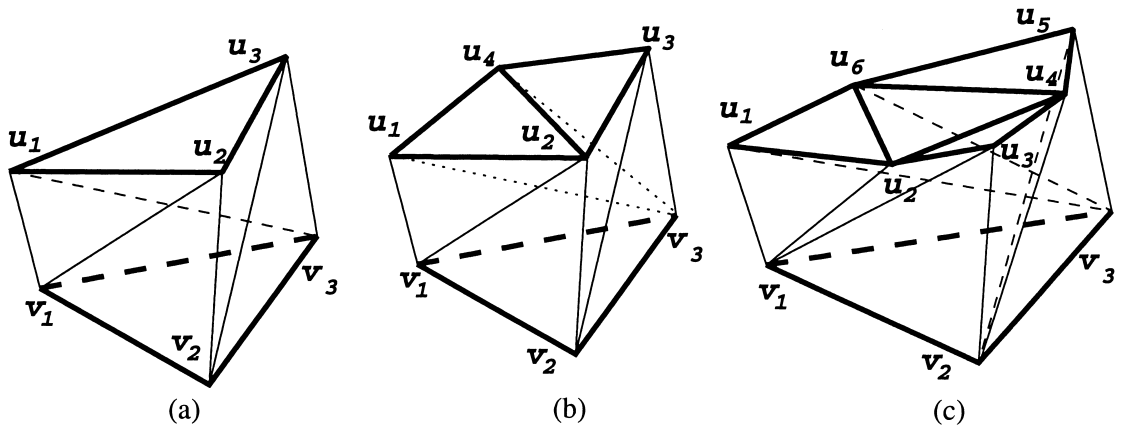

Fig. 20. The untetrahedralizable prismatoids have one bottom triangle. (a) the simplest form which is a Schönhardt prism; (b) and (c) are two variations. (b) To be untetrahedralizable, the prismatoid must be irregular enough so that both $\overline{u_4v_2}$ and $\overline{u_3v_1}$ are partly outside it. (c) None of $\overline{u_2v_3}$, $\overline{u_4v_1}$,$\overline{u_6v_2}$, $\overline{u_3v_3}$, $\overline{u_1v_2}$, and $\overline{u_5v_1}$ is totally inside the prismatoid.

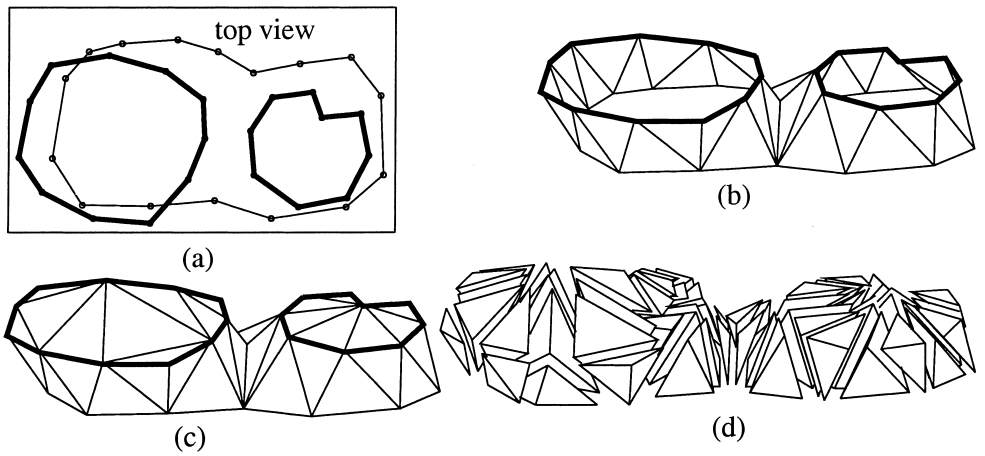

Fig. 21. This shows the tetrahedralization of a solid which contains a one-to-many branching. (a) The thick contours are on the top slice. (b) The reconstructed surface mesh. (c) Tetrahedralization. (d) The tetrahedra are separated for better visualization.

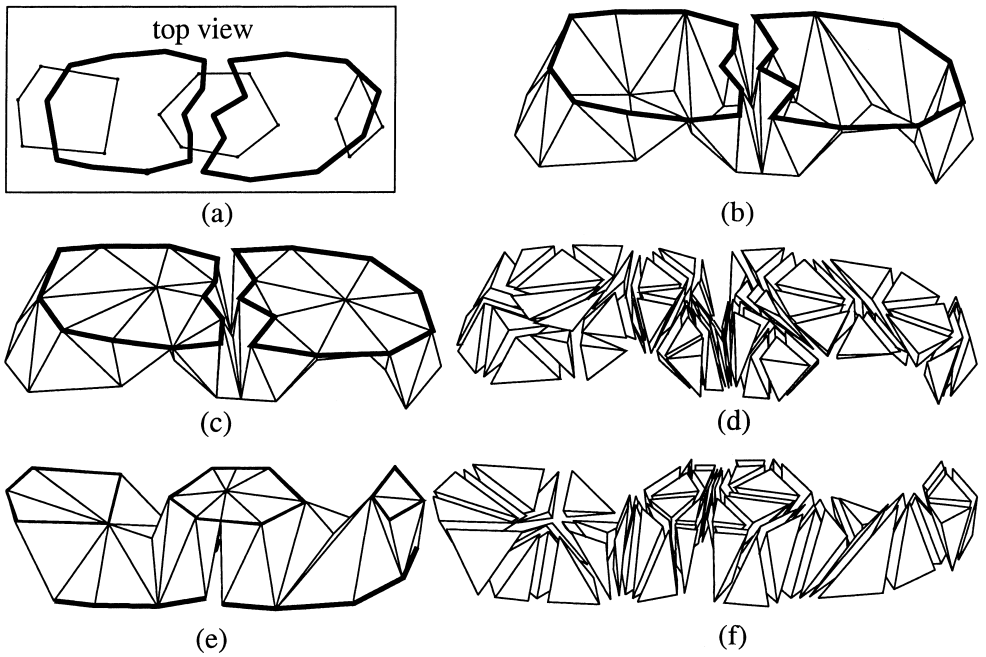

Fig. 22. This shows the tetrahedralization of a solid which contains a many-to-many branching. (a) The thick contours are on the top slice. (b) The reconstructed surface mesh. (c) Tetrahedralization. (d) The tetrahedra are separated for better visualization. (e) and (f) are viewed from the bottom.

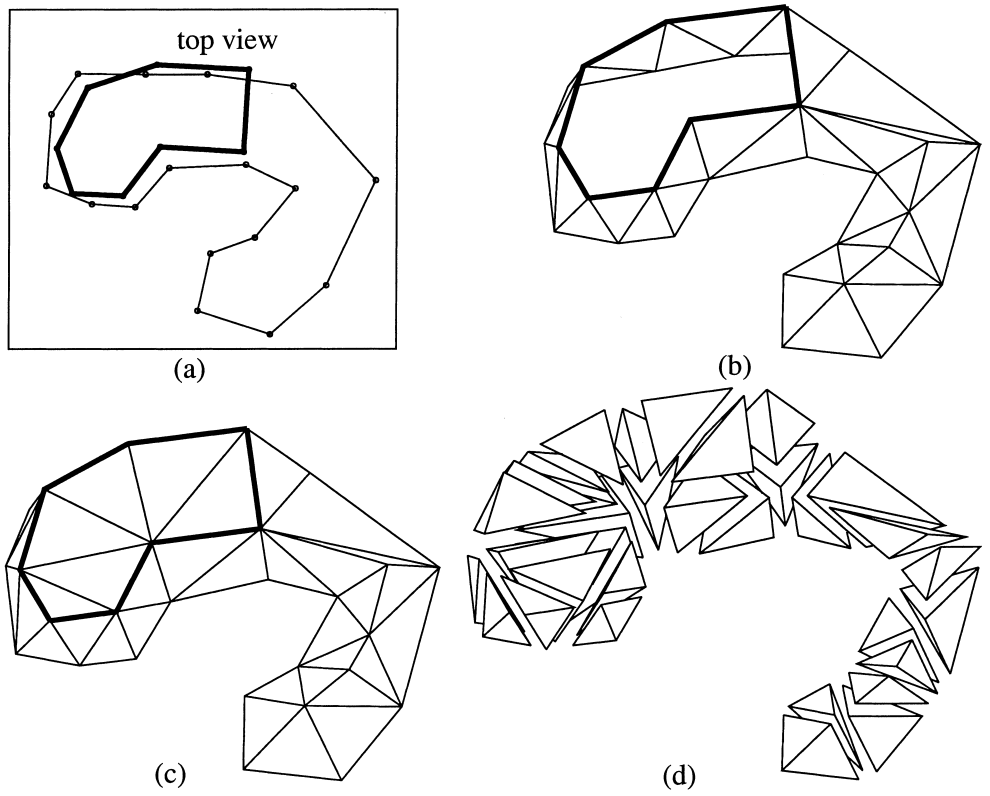

Fig. 23. This shows the tetrahedralization of a solid which contains a dissimilar region (the right bottom portion of the bottom contour). The interior of the dissimilar region is solid. (a) The thick contour is on the top slice. (b) The reconstructed surface mesh. (c) Tetrahedralization. (d) The tetrahedra are separated for better visualization.

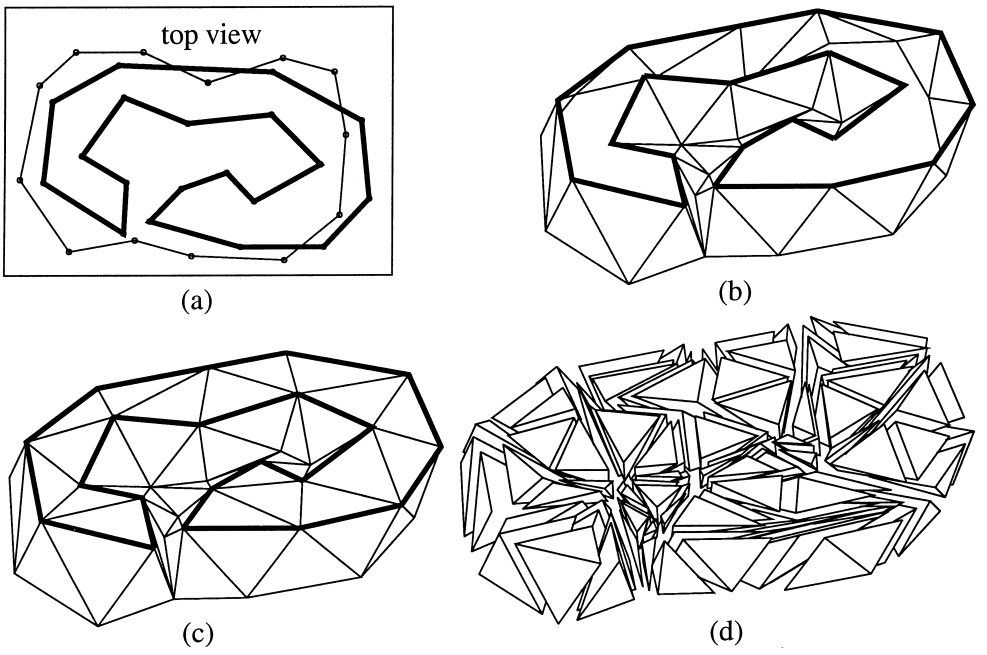

Fig. 24. This shows the tetrahedralization of a solid containing a dissimilar region (the inner portion of the top contour). The interior of the dissimilar region is void. (a) The thick contour is on the top slice. (b) The reconstructed surface mesh. (c) Tetrahedralization. (d) The tetrahedra are separated for better visualization.

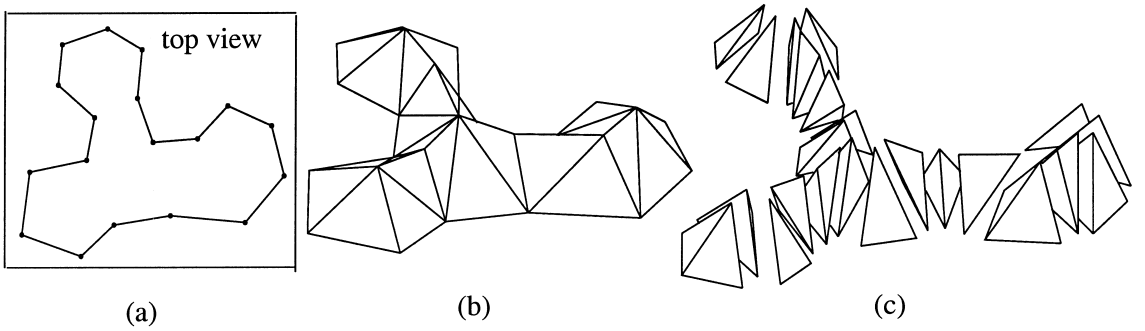

Fig. 25. This shows the tetrahedralization of a solid which contains an appearing/disappearing vertical feature of a solid interior. (a) The contour is on the bottom slice, and it has no corresponding contour on the top slice. (b) The reconstructed surface mesh. (c) The tetrahedra are separated for better visualization.

Step 4: If the remaining part is untetrahedralizable, we postprocess it.

The metric of a tetrahedron is based on its volume/(edge)^3 ratio as defined by Lo [19]. A higher metric value implies a better tetrahedron. We do not form tetrahedra by use of non-boundary triangles because the remaining part might become very complicated as shown in Fig. 13.

Traditional AFT forms one tetrahedron at a time. The drawback is that the formation of a tetrahedron could later cause the neighboring faces to form ill-shaped tetrahedra. Our approach alleviates this drawback. Our *group operator* forms one group of tetrahedra at a time and achieves optimal tetrahedralization within the group.

Definition 8. Group operator: an operator applies to a boundary triangle to form one group of tetrahedra at a time. The requirements are described in the following paragraph.

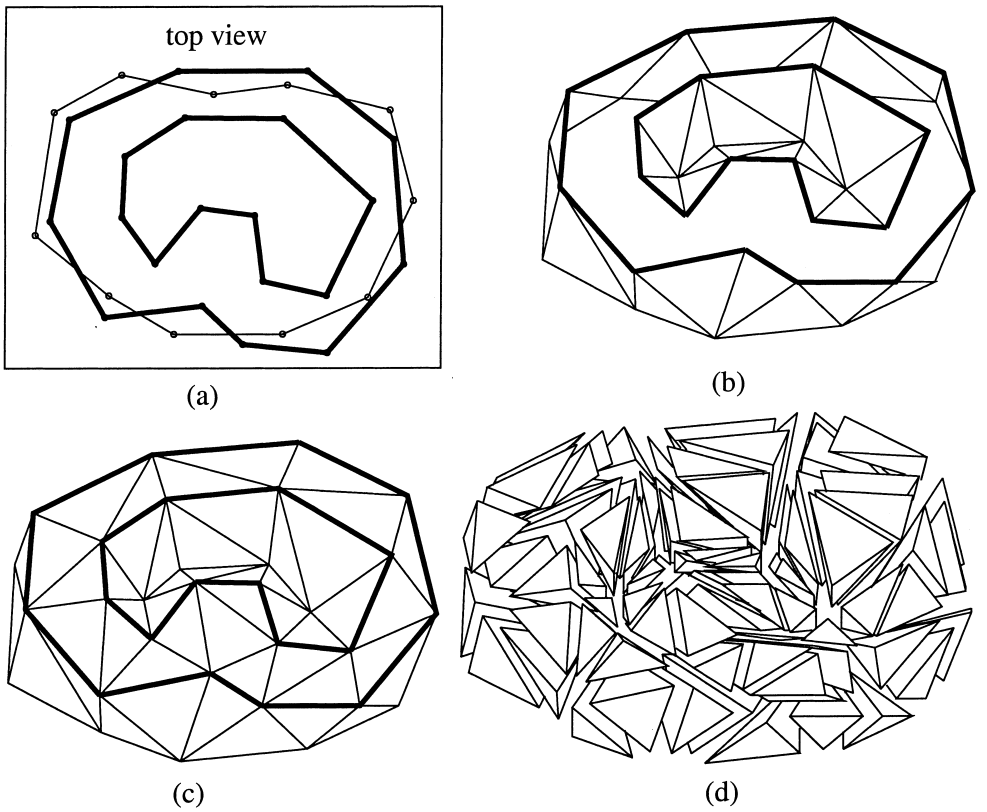

Fig. 26. This shows the tetrahedralization of a solid which contains an appearing/disappearing vertical feature (the top inner contour) of a void interior. (a) The thick contour is on the top slice. The top inner contour has no corresponding contour on the bottom slice. (b) The reconstructed surface mesh. (c) Tetrahedralization. (d) The tetrahedra are separated for better visualization.

The group operator applies to boundary triangles on both slices. We illustrate it using a top boundary triangle $\triangle u_1u_2u_3$ as shown in Fig. 14(a). There can be zero or more type 1 triangles between the two type 0 triangles $\triangle u_1u_2v_1$ and $\triangle u_2u_3v_m$. Here, m is one plus the number of type 1 triangles. There are m permutations to form tetrahedra containing these triangles. Fig. 14(b)(c) show two examples. The formed tetrahedra must neither cut across nor be outside the prismatoid, and they cannot violate the protection rule described in Section 4.2. The decision of selecting which permutation is based on the average metric as well as the worst metric of the generated tetrahedra. The metric of a boundary triangle is the average metric of the chosen permutation.

The metric of a boundary triangle is further multiplied by a weight factor. The weight factor intends to increase/decrease the metric of the boundary triangle which makes the remaining shape more regular/irregular, respectively. Let u_1, u_2 and u_3 be a boundary triangle as shown in Fig. 14. Let the signed distance h between u_2 and line u_1u_3 be positive. We find the maximum distance d among the signed distances from v_i, $i = 1$ to m, to line u_1u_3. The weight factor w is computed as the following:

$$w = \begin{cases} 2(1 - d/h) & \text{if } d \leqslant 0.5h, \\ 1 & \text{if } 0.5h < d < h, \\ h/d & \text{if } d \geqslant h. \end{cases}$$

The effect of the weight factor is illustrated in Fig. 15. Forming a tetrahedron shown in: (a) makes $\triangle u_1v_1u_3$ of the remaining part (b) to form a sharp angle between the bottom slice. So the metric of boundary $\triangle u_1u_2u_3$ in (a) is decreased by the weight factor which is smaller than 1. On the other hand, the remaining shape (d) is more regular than (c) because $\triangle u_1u_3v_2$ of (d) forms a less sharp angle with the top slice than that of $\triangle u_2u_3v_2$ or $\triangle u_1u_2v_2$ in (c). So the weight factor is designed to be larger than unity.

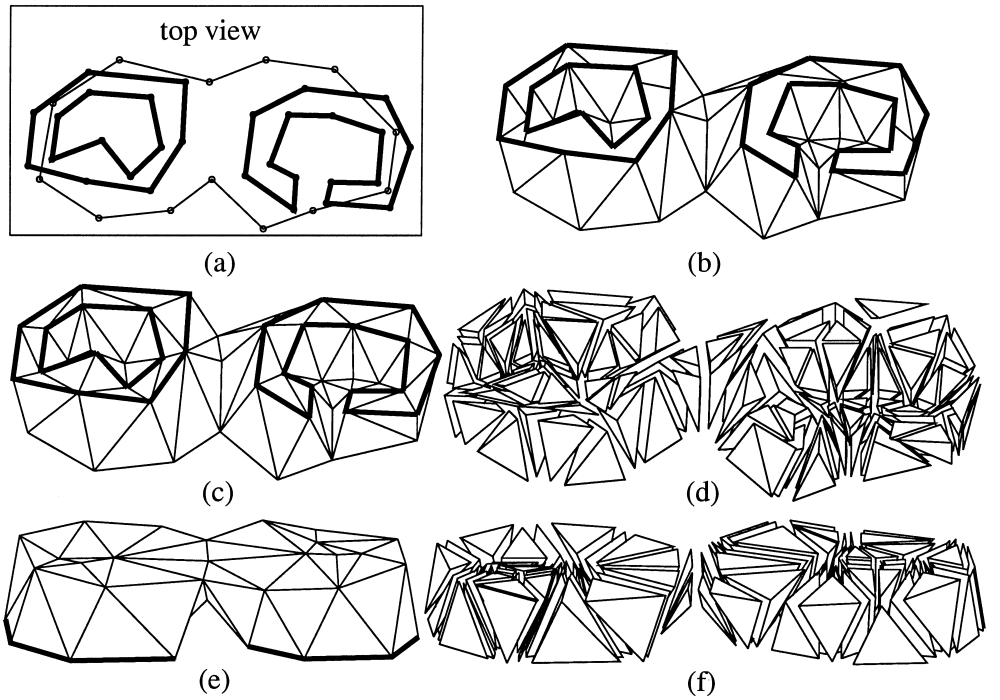

Fig. 27. This shows the tetrahedralization of a solid which contains a branching, a disimilar portion (the inner portion of the top right contour), and an appearing/disappearing vertical feature (the inner contour at the left of the top slice). (a) The thick contours are on the top slice. (b) The reconstructed surface mesh. (c) Tetrahedralization. (d) The tetrahedra are separated for better visualization. (e) and (f) are viewed from the bottom.

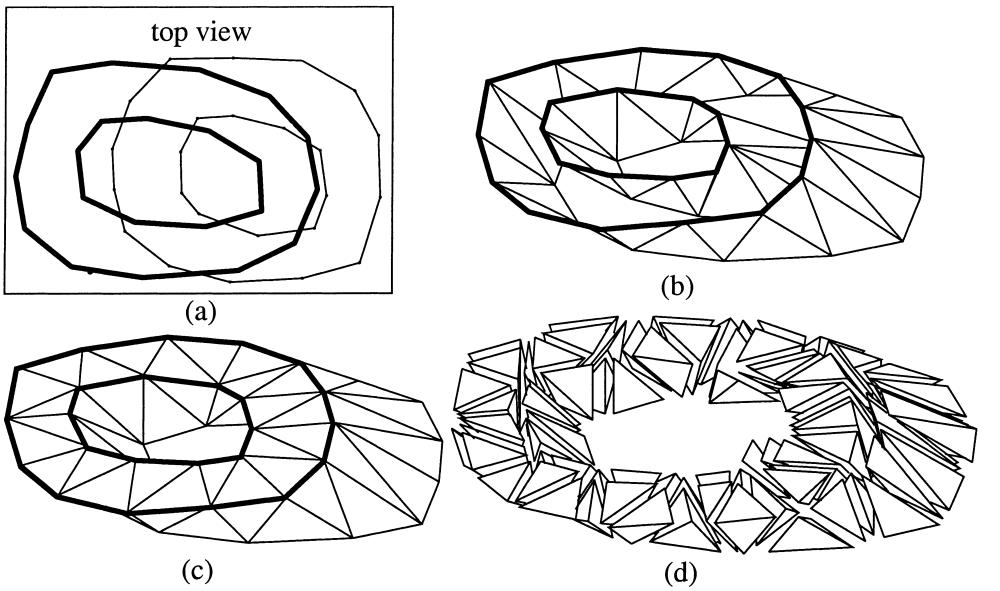

Fig. 28. This shows the processing of nested prismatoids. (a) The thick contours are on the top slice. (b) The reconstructed surface mesh. (c) Tetrahedralization. (d) The tetrahedra are separated for better visualization.

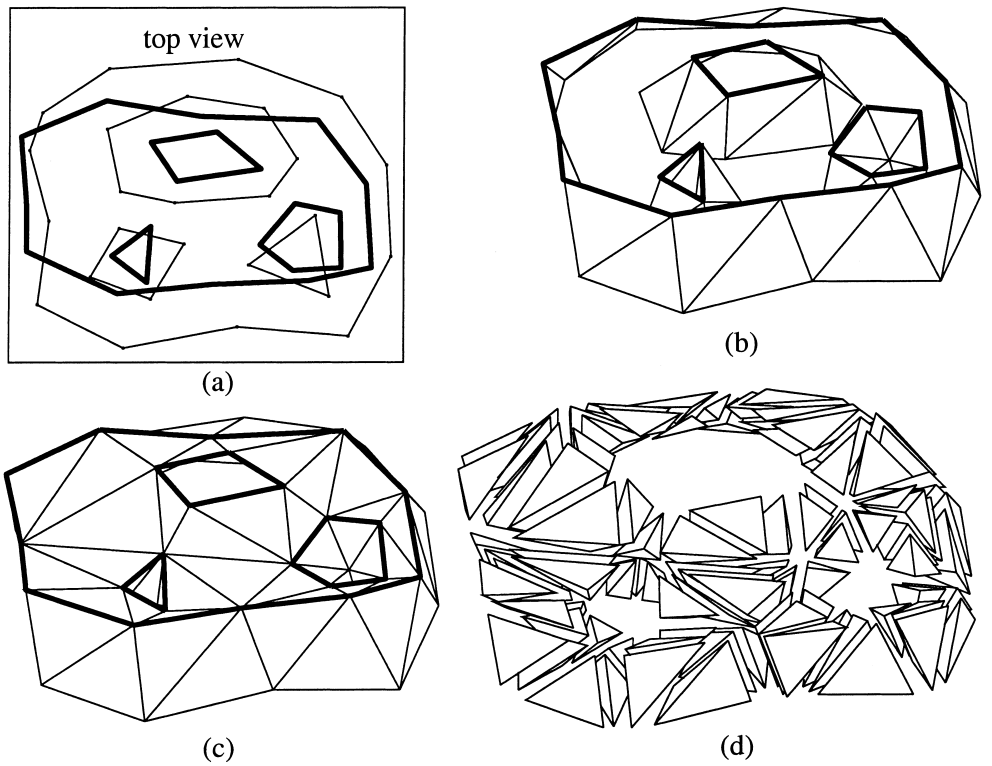

Fig. 29. This shows the processing of multiply nested prismatoid. (a) The thick contours are on the top slice. (b) The reconstructed surface mesh. (c) Tetrahedralization. (d) The tetrahedra are separated for better visualization.

The group operator can avoid generating an irregular remaining part. For the example in Fig. 16(a), other methods can form tetrahedra, and the remaining parts are shown in Figs. 16(b) and (c) after forming one and two tetrahedra, respectively. The remaining parts are very irregular. Our group operator will not form additional tetrahedra, and it will leave Fig. 16(a) for postprocessing.

Step 2 is similar to the auxiliary test of Jin and Tanner [14]. It picks up the best tetrahedra group from among the qualified groups.

4.2. Protection rule

O'Rourke [23] had posed the problem of determining the complexity of deciding whether a polyhedron is tetrahedralizable without adding any Steiner points. Ruppert and Seidel [25] showed that this problem is NP-complete. Although a prismatoid is a simplified polyhedron, determining what factors contribute to an untetrahedralizable prismatoid is still difficult. Our effort is to characterize the prismatoids, which are untetrahedralizable by our group operator and under certain constraints. The constraints should be strict enough to let our analysis be tractable, and loose enough to contain a large set of cases to be useful. Although we are unable to prove that a prismatoid, which cannot be tetrahedralized by our group operator, cannot be completely tetrahedralized by other operators, we never meet or construct a counter example. We present the following lemma to characterize a condition in which our group operator cannot form any additional tetrahedra. The lemma applies to boundary triangles on both slices, although it is illustrated using a top boundary triangle. The proof details are in Ref. [17].

Lemma 1. *Suppose a top boundary triangle* $\triangle u_1u_2u_3$ *is under the constraint that no more than one type* 1 *triangle is between the two type* 0 *triangles containing the contour segments* $\overline{u_1u_2}$ *and* $\overline{u_2u_3}$ *as shown in Fig.* 17. *Furthermore, let the bottom vertices of the two type* 0 *triangles be* v_1 *and* v_2. *Our group operator cannot apply to* $\triangle u_1u_2u_3$ *to form a tetrahedra group if and only if all the following conditions are satisfied.*

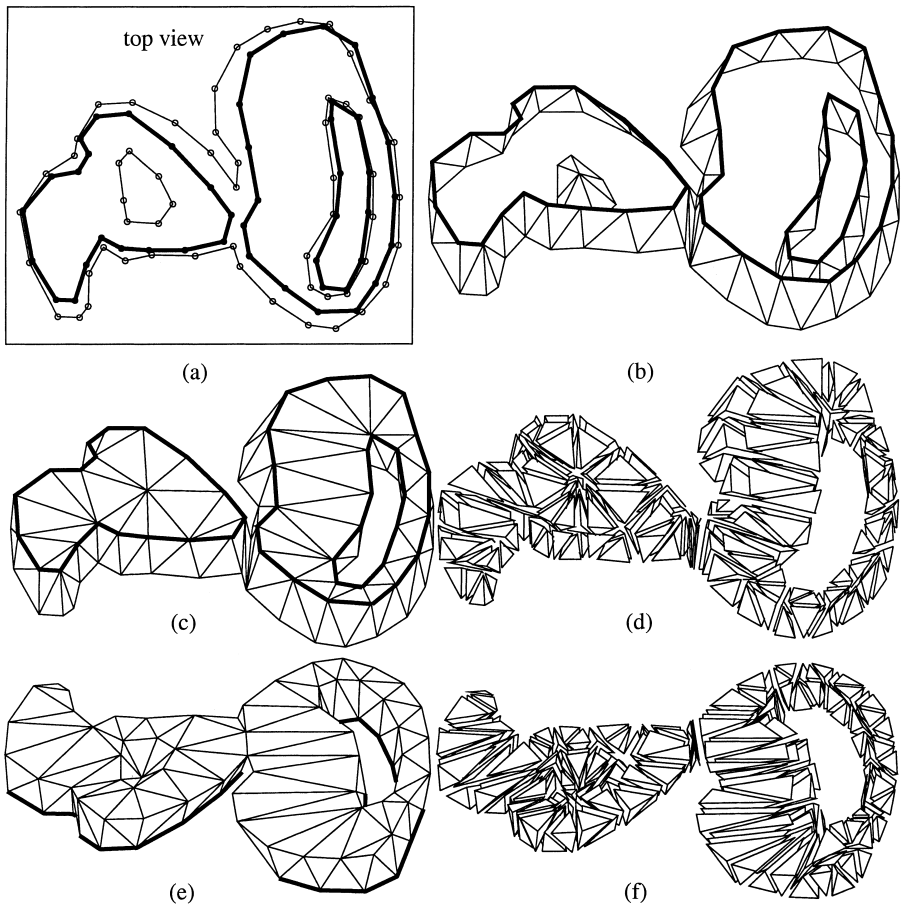

Fig. 30. This shows the tetrahedralization of the solid region between two slices of a human tibia. The distance between slices is enlarged 5 times. The reconstructed shape includes a branching and a appearing/disappearing vertical feature. (a) The thick contours are on the top slice. (b) The reconstructed surface mesh. (c) Tetrahedralization. (d) The tetrahedra are separated for better visualization; (e) and (f) are viewed from the bottom.

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 Lemma 1 can be derived from Condition 3. However, we state Condition 2 so it is easier to visualize the untetrahedralizable shapes. Fig. 17 shows the only two possible shapes which satisfy Lemma 1. They are mirror images of each other. Note that this lemma only applies to our group operator which does not form a tetrahedron using $\triangle u_1u_2u_3$ and a vertex other than v_1 and v_2.

We define one protection rule based on Lemma 1 to reduce the chance of generating untetrahedralizable remaining parts. The rule is that a new tetrahedron cannot satisfy condition 3 of Lemma 1 with respect to any boundary triangle which satisfies conditions 1 and 2. For example, if one boundary triangle has the shape of Fig. 17(a) and $\overline{u_1v_2}$ is totally inside the prismatoid, it is tetrahedralizable. The rule states that any proposed tetrahedron cannot cut across $\triangle u_1u_2v_2$.

Fig. 18 illustrates the usage of the protection rule. We asume both $\overline{u_1v_2}$ and $\overline{u_4v_1}$ are reflex. Then $\triangle u_4u_1u_2$ and $\triangle v_4v_1v_2$ satisfy Conditions 1 and 2, but not 3, of Lemma 1. The protection rule states that $\triangle u_4u_1v_2$ and $\triangle v_1v_2u_4$ cannot be cut across by any proposed tetrahedron. Thus, after forming tetrahedra associated with boundary triangles $\triangle u_2u_3u_4$ and $\triangle v_2v_3v_4$, the remaining part is shown in Fig. 18(b). The untetrahedralizable region, shown in Fig. 18(c) is prohibited because at least one of $\triangle u_4u_1v_2$ and $\triangle v_1v_2u_4$ is cut across by $\overline{u_2v_4}$.

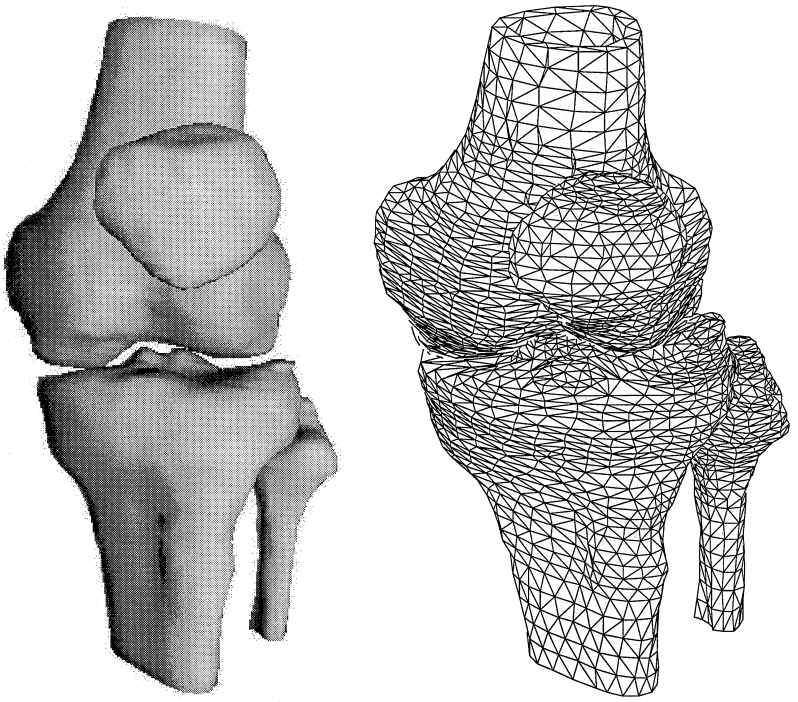

Fig. 31. A knee joint consisting of the lower femur, the upper tibia and fibula and the patella: (a) Gouraud shaded; (b) the tetrahedralization.

4.3. Classification of untetrahedralizable prismatoids

From Lemma 1, we can classify two categories of untetrahedralizable prismatoids. There are other categories. However, these two categories are common enough to contain all our experimental untetrahedralizable prismatoids.

Lemma 1 characterizes the untetrahedralizable prismatoid in terms of a boundary triangle. If the top or bottom face of a prismatoid is a single triangle, we treat it as three boundary triangles because any two of the single triangle's three edges satisfy the boundary triangle definition. Therefore, if the top or bottom face of a prismatoid is not of zero area, the face must contain at least two boundary triangles. We will classify the simplest cases and their variations.

The first untetrahedralizable category has two boundary triangles on the top face and one line segment on the bottom face. Based on Lemma 1, each boundary triangle of an untetrahedralizable prismatoid requires a type 1 triangle. So the bottom face of the simplest shape with two top boundary triangles is a line segment which can provide two type 1 triangles. The simplest shape is illustrated in Fig. 19(a). Fig. 19(b) shows one variation in which there are non-boundary triangles between two boundary triangles. No additional tetrahedra can be formed by our operator. Although some tetrahedra could be formed using non-boundary triangles, the remaining part (Fig. 19(c)) becomes very irregular. Our algorithm leaves Fig. 19(b), rather than Fig. 19(c), for postprocessing.

The second category has one bottom triangle which is treated as three boundary triangles. Thus it requires three type 0 triangles which can be provided by one top triangle. The simplest case, which is in Fig. 20(a), is a Schönhardt prism. Figs. 20(b) and (c) show two variations.

Although Figs. 19(a) and 20(a) are isomorphic ($\triangle v_1u_3u_2$ and $\triangle v_2u_4u_1$ of Fig. 19(a) map to $\triangle u_1u_2u_3$ and $\triangle v_1v_2v_3$ of Fig. 20(a), respectively), we consider them to be different categories because their variations are different. Some other categories can be derived from Lemma 1; however, these seldom occur.

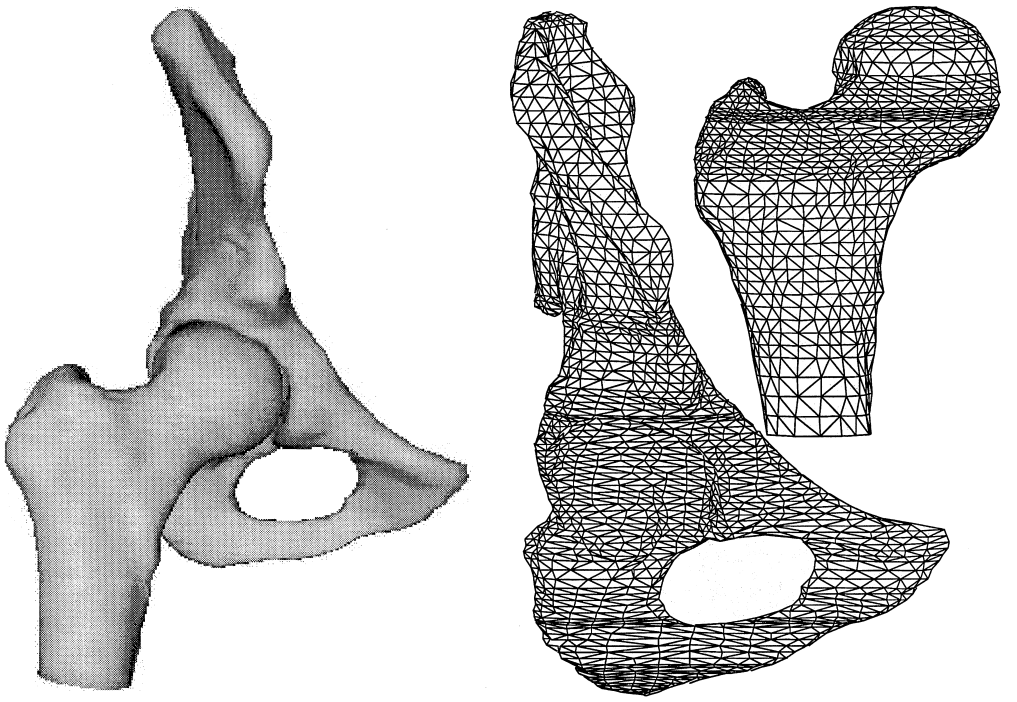

Fig. 32. A hip joint consisting of the upper femur and the pelvic joint: (a) Gouraud shaded; (b) the tetrahedralization.

The untetrahedralizable part is postprocessed by forming tetrahedra using a Steiner point between two slices as the apex and each face as the base. The Steiner point must be visible to all vertices. The classification of untetrahedralizable parts helps in locating a qualified Steiner point. Suggestions to postprocess the untetrahedralization parts are described in Ref. [17].

5. Results

From the analysis in Section 3.1, if the reconstructed polyhedron is not a prismatoid, it develops untiled regions during the tiling process. Our reducing algorithm treats the constructed polyhedra in the same manner. We do not classify the untiled regions as different cases in our reducing algorithm. We present the results of applying our unified reducing algorithm to those polyhedra which develop different kinds of untiled regions.

An untiled region corresponds to a branching region, a dissimilar portion, or an appearing/disappearing vertical feature as described in Section 3.1. The cases of dissimilar portion and appearing/disappearing vertical features have two situations: a void interior or a solid interior. The branching region does not form an interior because it is similar to a saddle surface.

Figs. 21 and 22 show the cases of one-to-many branching and many-to-many branching. Figs. 23 and 24 show two examples of dissimilar contours with a solid interior and a void interior, respectively. Figs. 25 and 26 show two examples of appearing/disappearing vertical features with a solid interior and a void interior, respectively. Fig. 27 shows a solid which contains all three cases. These figures verify the correctness of our reducing algorithm.

As the results of our prismatoid tetrahedralization approach, Figs. 28 and 29 show single-nested and multiply nested prismatoids. The 3D Delaunay triangulation without boundary conformation cannot process Fig. 28. Our approach works fine in this situation. Fig. 31 shows a knee joint and its tetrahedralization. Fig. 30 shows the tetrahedralization of two sample slices of the tibia. Fig. 32 shows a hip joint and its tetrahedralization. Tables 1 and 2 detail the results. The CPU time is based on a SGI Indigo2 workstation with IMPACT graphics.

Table 1
Knee joint tetrahedralization

	# of slices	# of tetrahedra	CPU time (s)
Femur	49	6804	22.1
Patella	27	1066	2.8
Tibia	38	5796	25.7
Fibula	37	2104	5.3

Table 2
Hip joint tetrahedralization

	# of slices	# of tetrahedra	CPU time (s)
Femur	45	7090	30.1
Pelvis	92	8660	31.2

The protection rule described in 4.2 works very well to reduce the chance of generating untetrahedralizable parts. When this rule is disabled, 15 untetrahedralizable remaining parts do occur during the process of the knee and hip joints. We find nine different classes of Schönhardt prisms. Three are as shown in 19(a). The others belong to the two classified categories, but are not simple shapes. With the protection rule enabled, we do not encounter any untetrahedralizable remaining parts in the tetrahedra decomposition of the knee and hip joints.

6. Conclusion

We have presented a new approach to build a 3D triangular mesh from planar contours. There are two main contributions in this paper. The first contribution is the characterization, prevention, and postprocessing of untetrahedralizable parts. The tetrahedralizability characterization of a general polyhedron is shown to be NP-complete [25]. Under reasonable constraints on our degenerated problem domain, we characterize this problem based on our group operator and develop one rule to reduce the chance of generating untetrahedralizable remaining parts. The characterization also helps the postprocessing of untetrahedralizable parts if they do occur. The second contribution is the algorithm to reduce a non-prismatoid into one or more prismatoids. This algorithm is guaranteed to work on the non-prismatoids generated by our surface construction program.

References

[1] C. Bajaj, E. Coyle, K. Lin, Arbitrary topology shape reconstruction from planar cross sections, Graphical Models and Image Processing 58 (6) (1996) 524–543.
[2] C. Bajaj, T.K. Dey, Convex decomposition of polyhedra and robustness, SIAM J. Comput. 21 (2) (1992) 339–364.
[3] M. Bern, D. Eppstein, in: D.-Z. Du, F.K. Hwang (Eds.), Mesh Generation and Optimal Triangulation, Computing in Euclidean Geometry, World Scientific, Singapore, 1992, p. 23–90.
[4] W.H. Beyer, CRC Standard Mathematical Tables, 28th ed., CRC Press, Boca Raton, Florida, 1987.
[5] J.D. Boissonnat, Shape reconstruction from planar cross sections, Comput. Vision and Image Proc. 44 (1988) 1–29.
[6] J.C. Cavendish, D.A. Field, W.H. Frey, An approach to automatic three-dimensional finite element mesh generation, Int. J. for Numer. Meth. in Eng. 21 (1985) 329–337.
[7] CG Impact Task Froce Report. Application challenges to computational geometry. Technical report, Technical Report TR-521-96, Princeton University, http://www.cs.princeton.edu/ chazelle/taskforce/CGreport.ps, April 1996.

[8] S. Chae, K. Bathe, On automatic mesh construction and mesh refinement in finite element analysis, Comput. and Struct. 32 (34) (1989) 911–936.
[9] B. Chazelle, L. Palios, Triangulating a non-convex polytope, Discrete Comput. Geom. 5 (1990) 505–526.
[10] H.H. Dannelongue, P.A. Tanguy, Three-dimensional adaptive finite element computations and applications to non-newtonian fluids, Int. J. for Numer. Meth. in Fluids 13 (1991) 145–165.
[11] D.A. Field, The legacy of automatic mesh generation from solid modeling, Comput. Aided Geometric Design 12 (1995) 651–673.
[12] B. Geiger, Three-dimensional modeling of human organs and its application to diagnosis and surgical planning, Technical Report, 2105, INRIA, France, 1993.
[13] P.L. George, E. Seveno, The advancing-front mesh generation method revisited, Int. J. for Numer. Meth. in Eng. 37 (1994) 3605–3619.
[14] H. Jin, R.I. Tanner, Generation of unstructured tetrahedral meshes by advancing front technique, Int. J. for Numer. Meth. in Eng. 36 (1993) 1805–1823.
[15] B. Joe, Tetrahedral mesh generation in polyhedral regions based on convex polyhedron decompositions, Int. J. for Numer. Meth. in Eng. 37 (1994) 693–713.
[16] B.P. Johnston, J.M. Sullivan, A normal offsetting technique for automatic mesh generation in three dimensions, Int. J. for Numer. Meth. in Eng. 36 (1993) 1717–1734.
[17] K. Lin, Surface and 3D triangular meshes from planar cross sections, Ph.D. Thesis, Purdue University, 1997.
[18] S.H. Lo, A new mesh generation scheme for arbitrary planar domains, Int. J. for Numer. Meth. in Eng. 21 (1985) 1403–1426.
[19] S.H. Lo, Volume discretization into tetrahedra - II. 3D triangulation by advancing front approach, Comput. and Struct. 39 (5) (1991) 501–511.
[20] R. Lohner, P. Parikh, Generation of three-dimensional unstructured grids by the advancing-front method, Int. J. for Numer. Meth. in Eng. 8 (1988) 1135–1149.
[21] D.L. Marcum, N.P. Weatherill, Unstructured grid generation using iterative point insertion and local reconnection, AIAA J. 33 (9) (1995) 1619–1625.
[22] P. Moller, P. Hansbo, On advancing front mesh generation in three dimensions, Int. J. for Numer. Meth. in Eng. 38 (1995) 3551–3569.
[23] J. O'Rourke, Art Gallery Theorems and Algorithms, Oxford University Press, Oxford, 1987.
[24] J. Peraire, J. Peiro, L. Formaggia, K. Morgan, O.C. Zienkiewicz, Finite element euler computations in three dimensions, Int. J. for Numer. Meth. in Eng. 26 (1988) 2135–2159.
[25] J. Ruppert, R. Seidel, On the difficulty of tetrahedralizing 3-dimensional non-convex polyhedra, in: Proceedings 5th Annual ACM Symposium Comput. Geom., 1989, pp. 380–392.
[26] N.P. Weatherill, O. Hassan, Efficient three-dimensional Delaunay triangulation with automatic point creation and imposed boundary constraints, Int. J. for Numer. Meth. in Eng. 37 (1994) 2005–2039.