GRAPHICAL MODELS AND IMAGE PROCESSING
Vol. 58, No. 6, November, pp. 524–543, 1996
ARTICLE NO. 0044

Arbitrary Topology Shape Reconstruction from Planar Cross Sections

CHANDRAJIT L. BAJAJ

Department of Computer Sciences, Purdue University, West Lafayette, Indiana 47907

AND

EDWARD J. COYLE AND KWUN-NAN LIN

School of Electrical Engineering, Purdue University, West Lafayette, Indiana 47907

Received October 23, 1995; revised April 30, 1996; accepted July 31, 1996

In computed tomography, magnetic resonance imaging and ultrasound imaging, reconstruction of the 3D object from the 2D scalar-valued slices obtained by the imaging system is difficult because of the large spacings between the 2D slices. The aliasing that results from this undersampling in the direction orthogonal to the slices leads to two problems, known as the correspondence problem and the tiling problem. A third problem, known as the branching problem, arises because of the structure of the objects being imaged in these applications. Existing reconstruction algorithms typically address only one or two of these problems. In this paper, we approach all three of these problems simultaneously. This is accomplished by imposing a set of three constraints on the reconstructed surface and then 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). Our implementation of the above approach generates triangles of 3D isosurfaces from input which is either a set of contour data or a volume of image slices. Results obtained with synthetic and actual medical data are presented. There are still specific cases in which our new approach can generate distorted results, but these cases are much less likely to occur than those which cause distortions in other tiling approaches. © 1996 Academic Press, Inc.

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. The set of planes generating the slices are usually parallel to each other and equispaced along some axis through the object.

Once these measurement slices have been obtained, the goal is to enable a human to easily visualize, in 3D, this large collection of data. Many algorithms have been developed for this purpose, but they can all be classified into two categories [8]: volume rendering methods and surface reconstruction methods.

This paper concentrates on surface reconstruction methods, all of which proceed by extracting the isosurfaces corresponding to a specified image intensity. Each isosurface is represented as an assembly of simple surface primitives, such as triangles or other polygons. Once these surface primitives are calculated, they can be quickly rendered from different viewpoints using widely available graphics hardware. This allows the user to quickly examine many different viewing spaces.

This paper presents a surface-based algorithm which achieves both faster rendering and lower likelihood of reconstruction error than previous surface reconstruction algorithms. These improvements are obtained by taking a unified approach to the three problems inherent in all surface-based approaches [20]: the correspondence problem, the tiling problem, and, the contour branching problem. These problems and their previous solutions are discussed in the following subsections.

2. OVERVIEW OF PREVIOUS APPROACHES

The three fundamental problems in surface-based reconstruction—the correspondence problem, the tiling prob-

E-mail: {bajaj@cs, coyle@ecn, klin@cs}.purdue.edu.

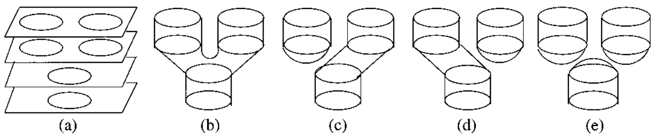

FIG. 1. The correspondence problem: (a) cross section contours; (b)-(e) different topologies with the same cross sections as in (a).

lem, and the branching problem—have motivated many research efforts.

2.1. *The Correspondence Problem*

The correspondence problem involves finding the correct connections between the contours of adjacent slices. Figure 1 shows an example with four different joint topologies (b)–(e) resulting from the same cross sections as in (a). If the distance between slices is large, *a priori* knowledge or global information is required to determine the correct correspondence. Bresler *et al.* [3] use domain knowledge to constrain the problem. Meyers *et al.* [20] and Soroka [26] approximate the contours by ellipses and then assemble them into cylinders to determine the correspondence. Wang *et al.* [28] check the overlapping area as the criterion for the correspondence.

2.2. *The Tiling Problem*

Tiling means using slice chords to triangulate the strip lying between contours of two adjacent slices into tiling triangles (Fig. 2). A slice chord connects a vertex of a given contour to a vertex of contour in an adjacent slice. Each tiling triangle consists of exactly two slice chords and one contour segment. There are two related issues. One is how to accomplish optimal tiling in terms of certain metrics such as surface area and enclosed volume. The other is the topological correctness of the tiling.

The problem of mating points between contours into triangles is formalized by Keppel [15] into a graph search problem. Fuchs *et al.* [10] provide an efficient algorithm based on an Euler tour of a toroidal graph to obtain an optimal solution. Their algorithm has a time complexity of $O(n^2 \log n)$, where n is the total number of vertices on the contours bounding the triangles. Sloan *et al.* [25] locate bottlenecks and improve the speed of Fuchs' algorithm by a constant factor. Shinagawa *et al.* [24] generalize the discrete toroidal graph into a continuous one. Homotopy is used for reconstructing smooth surfaces from the toroidal graph. Homotopy is similar to metamorphosis (morphing) in which one contour is gradually changed into another contour. Kehtarnavaz *et al.* [14] represent the search problem as a Levenshtein graph and use dynamic programming to find its minimum cost path. Wang *et al.* [28] present a method which first assigns an initial merit to each triangle. It then uses relaxation to iteratively refine these weights, and it finishes by utilizing the A^* search algorithm to find a triangulation with minimum weight.

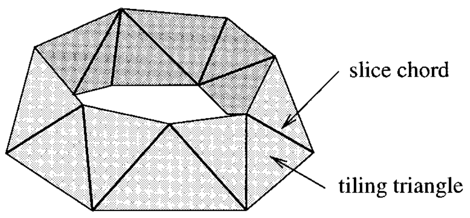

FIG. 2. An example of tiling.

Some fast heuristic methods have also been developed for tiling. The strategy of Christiansen *et al.* [4] is based on selection of shortest slice chords. Ganapathy *et al.* [11] use the concept of least tension as a heuristic guideline to tiling. These heuristic methods [4, 11] usually work quickly and work well when the contours being matched have similar shapes. Ekoule *et al.* [7] develop an approach to tile two dissimilar contours which have similar convex hulls. The two convex hulls of corresponding contours are first heuristically tiled using shortest slice chord metric. Thereafter, their method maps the concave portions of a contour onto its convex hull to look up the tiling pair from the convex hull tiling. This method avoids some abnormalities produced by Christiansen's algorithm. After comparing different algorithms, Meyers [19] points out that the minimum surface area optimization approach produces fewer abnormalities.

When two corresponding contours are very different, it is difficult to obtain a topologically correct and natural tiling. Gitlin *et al.* [13] show one example in which two polygons cannot be tiled to form a polyhedron. Their example is a pair of extremely different contours. Even in a moderately dissimilar contour pair in which a polyhedron can be formed, the tiling algorithm may result in a non-polyhedron. For the example in Fig. 17a, the minimum surface optimizing algorithm generates the non-polyhedral surfaces shown in Fig. 17c even though there exist many polyhedral solutions ((e) is one example). The arrow in Fig. 17c shows the self intersecting portions of the surface. Even when the tiling result is a polyhedron, it might be physically unlikely. For example, Fig. 3a shows two cross section contours. Figure 3c shows a vertical cross section of a reconstruction that tiles the interior of the top contour to the bottom contour (see Fig. 3b). The scalar data along the vertical line L of (c) flips its sign twice between two adjacent slices. This is an unlikely topology, especially when the distance between the two slices is small. Figure 3d shows another tiling in which the vertices of the dissimilar portion tile to the medial axis of the dissimilar portion. The medial axis is placed at between two slices. Figure 3e shows a cross section of (d). It shows that Fig. 3d is a highly

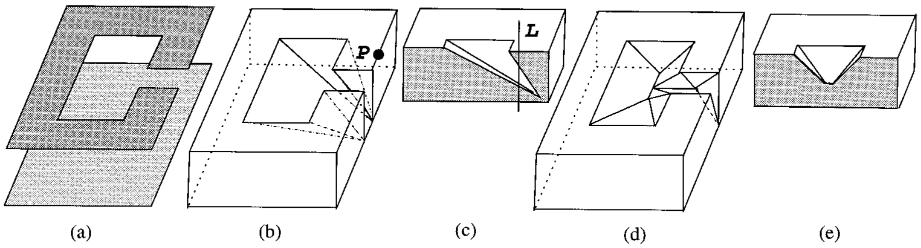

FIG. 3. Tiling of dissimilar contours: (a) the contours of two adjacent slices; (b) a tiling in which all vertices of the top contours tile to vertices of the bottom contour; (c) the vertical section passing through the point P of the solid in (b); (d) a tiling in which the dissimilar portion of the top contour tiles to its medial axis located at between two slices; (e) a vertical section through the solid shown in (d).

likely topology. All solid surfaces should be single sheeted; that is, the scalar data along a vertical line changes its sign at most once, between two adjacent silices if the distance between slices is fine enough. The tiling method shown in Fig. 3b violates this claim.

Algorithms [4, 6, 7, 10, 11, 14, 15, 25, 28] which attempt to tile all contour vertices to the adjacent slice produce an unlikely topology, as shown in Fig. 3b. Boissonnat [2] and Barequet *et al.* [1] produce horizontal triangles which lie on the slice, thus avoiding tilings like those in Fig. 3b. This generates a result better than Fig. 3b without adding any intermediate vertices.

2.3. *The Branching Problem*

A branching problem occurs when a contour in one slice can correspond to more than one contour in an adjacent slice. Figure 4a shows that contour $C3$ of slice $S2$ branches into $C1$ and $C2$ of slice $S1$. A contour in one slice having no corresponding contour in an adjacent slice forms either a hole or the beginning/end of a vertical feature. The possibility of branching significantly complicates the task of tiling. It creates the problem of branching surface reconstruction. Lin *et al.* [17] model branching regions by interpolating many intermediate contours. This method generates a smooth surface at the cost of a large number of triangles. Other branch processing approaches can be classified into the four methods shown in Figs. 4b–4e. Figure 4b shows that a curve L or a point is added between two slices to model the valley or saddle point formed by the branching. The added curve L is placed at slice $S2$ in (c). One or more line segments are added to form a composite contour as in (d); thereafter the tiling between the composite contour and $C3$ becomes one-to-one. Figure 4e also forms a composite contour, which is the convex hull of the branching contours, in order to have one-to-one tiling. The branching region between $C1$ and $C2$ is filled up by horizontal triangles. In terms of topological correctness, the best branching handling is (b) because it corresponds to the expected physical object better than the others do.

Christiansen *et al.* [4], Shantz [23], and Shinagawa *et al.* [24] use the method in Fig. 4d. They dip down the middle of the bridge to model the saddle point of the branching region. This approach works well only in simple branching cases. Ekoule *et al.* [7] form an intermediate contour, similar to a composite contour, between two slices for the case of one-to-many branching. The intermediate contour is tiled to the merging contour as well as to the branching contours. This produces less distortion than the method of Fig. 4d. Meyers *et al.* [20] use the scheme in Fig. 4e. They improve the horizontal triangle problem associated with this approach by feeding the triangulation mesh into a surface fitting program to regenerate the surface. Barequet *et al.* [1] first match and tile similar portions between corresponding contours. Then, the clefts, which are the polygons formed by the untiled portions, are triangulated. If the XY projections of clefts are nested, bridges are added to break the nesting. Their bridge adding scheme solves the problem of the possible conflict between a bridge in one slice and the geometry of the other slice. In the case of Fig. 4a, the contour portions along the branching area

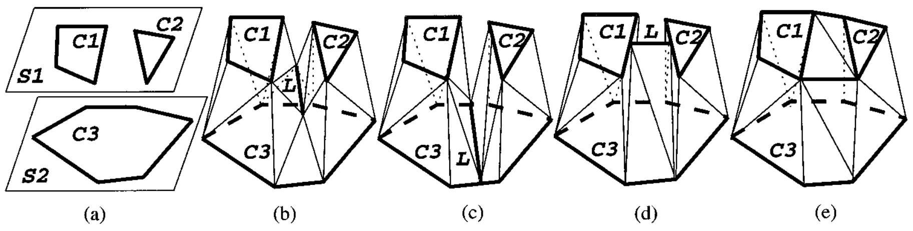

FIG. 4. Different reconstructions for branching contours: (a) branching contours on adjacent slices; (b)–(e) different surface reconstructions.

are unmatched, and the triangulation result is similar to Figs. 4d or 4e, depending on whether bridges are added or not.

Boissonnat [2] uses a different approach than tiling. He applies 3D Delaunay triangulation to contour vertices of two adjacent slices. The surface of the polyhedron formed by the union of tetrahedra is the desired surface. He reduces the problem of tetrahedralization of the object delimited by the two slices into building tetrahedra from two 2D Delaunay triangulations of two adjacent slices. Geiger [12] improves Boissonnat's approach so it can handle complicated branching and dissimilar contours. He projects the external Voronoi skeleton (*EVS*) from one slice to the adjacent slice and adds the projection in the 2D Delaunay triangulation. Thus, the triangles of a merging contour are split into several regions corresponding to each of the branching contours. Tetrahedra can be constructed between these corresponding regions. His branching handling is as in Fig. 4c. In the case of dissimilar contours such as in Fig. 3a, his algorithm tiles the dissimilar area (i.e., the interior) of the top contour to its *EVS* projection on the adjacent slice. This results in a topology similar to Fig. 3d.

An analysis of the algorithms summarized above shows that they each violate at least one of the following guidelines:

1. As explained in Fig. 3, one should not enforce connecting every vertex of one contour to another contour if these two contours are very dissimilar.
2. The re-sampling of the reconstructed surface should yield the original contours. The branching methods shown in Figs. 4c, 4d, and 4e violate this guideline.
3. Composite contours should not be formed (e.g., as shown on the top slice of Figs. 4d and 4e) because they do not correspond well to the actual physical object.

From the above discussion, it is clear that the problem of shape reconstruction is underconstrained, which implies that there are many feasible solutions. Our goal is to define surface crieria which, by imposing reasonable constraints on the problem, lead to an algorithm that generates the most likely object. These criteria must therefore lead through derivation to explicit correspondence and tiling rules. Holes, branching regions and dissimilar portions of contours will be detected because they cannot tile to other vertices based on the derived rules. Unlike many other algorithms which treat holes, one-to-many branching, many-to-many branching and dissimilar contours as different special cases, our postprocessing algorithm treats them all in the same manner.

The algorithm we derive from a set of surface criteria does not violate any of the three guidelines listed above. The overall procedure followed by our algorithm is similar to that of Barequet *et al.* [1]—we will discuss the similarities and differences in the implementation section. Our postprocessing of the untiled region is similar to adding the *EVS* from one slice to the adjacent slice, as in Geiger's approach [12].

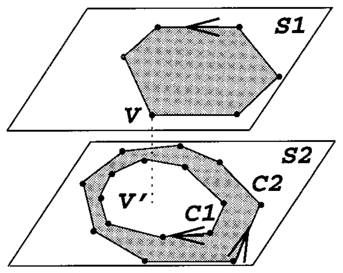

FIG. 5. Examples of oriented contours on parallel slices. The shaded regions are the solid regions.

We present the theory behind our algorithm in Section 3, the implementation in Section 4, and the results in Section 5. We discuss our contributions and their limitations in Section 6, and conclude in Section 7. Finally, the proofs of Section 3 are listed in the Appendix.

3. CORRESPONDENCE AND TILING RULES

As discussed in Section 2, any surface-based approach must address the correspondence problem, the tiling problem and the branching problem. The tiling problem itself has two aspects: (1) obtaining an optimal tiling in terms of certain metrics and (2) the detection and tiling of dissimilar portions of contours.

In this section, we address all of these problems simultaneously by first defining a set of criteria for the desired reconstructed surface. The criteria chosen can constrain the shape reconstruction problem so that the surfaces produced correspond well with expected physical models.

These criteria are then used to derive correspondence and tiling rules. The correspondence rules that are derived are local; they rely only on data in adjacent slices to determine the correspondence between contours. The tiling rules prohibit those tilings which result in undesired or nonsensical surfaces, and allow detection of branching regions and dissimilar portions of contours.

Section 4.1 will discuss how to use these rules to obtain a near-optimal tiling and to detect and process branching and dissimilar regions.

We define the input before presenting the surface reconstruction criteria. The input consists of two sets of contours, one set on each of two adjacent slices (Fig. 5). Each set contains zero or more contours which are simple polygons. A single contour divides the slicing plane into a *solid region* (shown as the shaded region of Fig. 5) and a *void region.* Like isocontours, contours are simple polygons and they

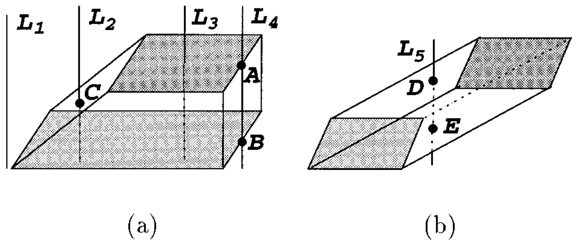

FIG. 6. Criterion 2: The shaded regions are the solid regions of two slices, and the reconstructed surface is the surface covering the space between two solid regions. (a) A vertical line between two slices intersects the reconstructed surface at zero points (e.g., L_1 and L_3), one point (e.g., L_2), or one line segment (e.g., L_4). (b) The reconstructed surface shown is not allowed because the vertical line L_5 intersects the reconstructed surface at two points, D and E.

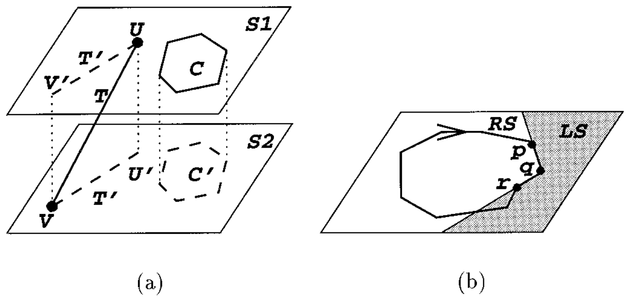

FIG. 7. (a) Examples of projection. (b) A shadow region $\mathcal{LS}(q)$ of a vertex q.

can be inside other contours. They cannot, however, intersect each other on the same slice. A contour does not contain its interior. It is oriented so the solid region is on its left side. The solid region is thus inside a *CCW* (counterclockwise) contour and is outside a *CW* (clockwise) contour. For the example in Fig. 5, contour $C1$ is *CW* and contour $C2$ is *CCW*. In the presence of looped contours, the solid region is inside a *CCW* contour and is outside zero or more *CW* contours. For simplicity, we assume that a solid region does not intersect the image boundary. A vertex is one endpoint of a linear contour segment. We define it to have the same *CCW* or *CW* direction as the contour.

We now define the surface reconstruction criteria:

Criterion 1. The reconstructed surface and solid regions form piecewise closed surfaces of polyhedra.

Criterion 2. Any vertical line (a line perpendicular to the slice) between two slices intersects the reconstructed surface at zero points, one point, or along one line segment (Fig. 6a).

Criterion 3. Resampling of the reconstructed surface on the slice should produce the original contours.

Criterion 1, which requires that surfaces be composed of polyhedra, prohibits such incorrect structures as self-intersecting surfaces. Criterion 2 is used to avoid the generation of unlikely topologies. This criterion is inspired from the unlikely topology of Fig. 3c, in which a vertical line L intersects the reconstructed surface twice between two slices. This criterion may not be enforceable if the distance between two adjacent slices is large, or if the sampling plane is nearly tangent to the surface of a long, thin object. For example, the desired surface in Fig. 6b is not allowed because there exists a vertical line L_5 intersecting the reconstructed surface at two points. Suggestions to handle these cases are presented in Section 6.2. The motivation behind Criterion 3 is obvious.

From these three criteria, we derive explicit tiling and correspondence rules. The theorems stated in this section are directly used in our reconstruction algorithm; the lemmas are only used to prove these theorems. Theorems 1–5 are related to the tiling rules, while Theorems 6 and 7 describe the correspondence rules. Based on Theorems 6 and 7, the correspondence relationship is unique if the reconstructed surface satisfies Criteria 1–3.

The following lemmas and theorems hold for any pair of adjacent slices. Their proofs are given in the Appendix.

LEMMA 1. *If a vertical line L not intersecting any contour does intersect the reconstructed surface, then L intersects exactly one solid region.*

Please recall that a contour does not contain its interior. For the example of Fig. 6a, all vertical lines except L_4 do not intersect any contour.

LEMMA 2. *Let L be a vertical line not intersecting any contour. Suppose it has M* ($M = 0, 1,$ *or* 2) *intersections with the solid regions on the two adjacent slices, and it has N* ($N = 0$ *or* 1) *intersections with the reconstructed surface. Then, M and N have the same parity, that is,*

1. $M = 1 \Leftrightarrow N = 1$
2. $M = 0$ *or* $M = 2 \Leftrightarrow N = 0$.

DEFINITION 1. Cross: Two line segments that intersect, but not at their endpoints.

DEFINITION 2. Projection: The projection of an object is denoted by appending a prime sign ($'$). For the example shown in Fig. 7a, C', U', V', and T's are the projections of contour C, vertex U, vertex V and line segment T (which is $\overline{UV}$), respectively. Because T lies between two slices, it has projections, the two T's, on two slices. Unless stated otherwise, any projection is onto an adjacent slice.

LEMMA 3. *If the projection of one contour segment crosses any other contour segment, then these two contour segments cannot be tiled.*

DEFINITION 3. Augmented contours: New vertices are embedded in contours at those points where the projection of that contour would cross another contour. This breaking of contour segments ensures that any intersection between a contour and a contour projection is either a contour vertex or a contour segment.

Because of Lemma 3, *augmented contours* are formed to allow the tiling of contour segments whose projections *cross* each other. They are achieved by adding the intersection to these two contour segments as a new vertex. From this point on, all contours are assumed to be augmented contours.

DEFINITION 4. $\mathcal{NEC}$ is the nearest enclosing contour of a point or a contour. Note that $\mathcal{NEC}(C)$, where C denotes a point or a contour, cannot intersect C. In Fig. 5, $C1 = \mathcal{NEC}(V')$.

DEFINITION 5. Positive/negative vertex and overlapping vertex: Let V be a vertex (see Fig. 5). If its projection V' lies on a contour, then V is an overlapping vertex. An overlapping vertex pair represents the vertex pair (V_1, V_2) where $V_1 = V_2'$. If V is not an overlapping vertex, then V is said to be a positive or negative vertex if V and $\mathcal{NEC}(V')$ have opposite or identical orientations, respectively. For example, the vertex V of Fig. 5 is *CCW* and $\mathcal{NEC}(V')$ (contour $C1$) is *CW*, so V is a positive vertex.

THEOREM 1. *Any overlapping vertex must tile to its projection.*

(Note: This theorem does not mean an overlapping vertex cannot tile to any vertex other than its projection. A vertex can have more than one slice chord.)

DEFINITION 6. $\mathcal{LS}(q)$ and $\mathcal{RS}(q)$ are the left side and the right side of a vertex q, respectively. Suppose $\overline{pq}$ and $\overline{qr}$ of Fig. 7b are two ordered contour segments. The two half lines $\overrightarrow{qp}$ and $\overrightarrow{qr}$ divide the slice into two regions. Then $\mathcal{LS}(q)$ is the shadow region which contains the left side of $\widehat{pqr}$. Neither $\mathcal{LS}(q)$ nor $\mathcal{RS}(q)$ contains $\overrightarrow{qp}$ or $\overrightarrow{qr}$.

THEOREM 2. *Let T be a slice chord that is incident with a contour vertex V, and let $\mathcal{S}$ be the slice.*

1. *Suppose V is not an overlapping vertex:*

$$\text{if } \mathcal{NEC}(V') \text{ is CW} \Rightarrow T' \subset (\mathcal{S} - \mathcal{RS}(V)), \text{ or}$$
$$\text{if } \mathcal{NEC}(V') \text{ is CCW} \Rightarrow T' \subset (\mathcal{S} - \mathcal{LS}(V)).$$

2. *If V is an overlapping vertex, its projection V' is also a vertex because all contours are augmented contours. Then $T' \subset (\mathcal{S} - ((\mathcal{LS}(V) \cap \mathcal{LS}(V')) \cup (\mathcal{RS}(V) \cap \mathcal{RS}(V'))))$.*

Theorem 2 defines the region in which the projection of a slice chord must be located.

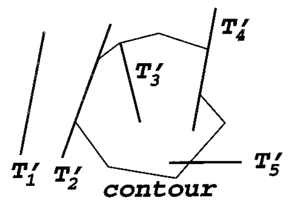

FIG. 8. Theorem 4: The thick line segments denoted by T''s are slice chord projections. T_1', T_2', and T_3' satisfy Theorem 4, while T_4' and T_5' violate Theorem 4.

THEOREM 3. *Suppose $\triangle V_1V_2V_3$ is a tiling triangle defined by vertices V_1, V_2, and V_3. Let V_1 and V_2 be two ordered adjacent contour vertices. Let $\mathcal{LS}(\overline{V_1V_2})$ and $\mathcal{RS}(\overline{V_1V_2})$, respectively, denote the left side and right side of the line passing through V_1 and V_2.*

1. *If either V_1 or V_2 is not an overlapping vertex and its projection has a CW $\mathcal{NEC}$, then $V_3' \notin \mathcal{RS}(\overline{V_1V_2})$.*

2. *If either V_1 or V_2 is not an overlapping vertex and its projection has a CCW $\mathcal{NEC}$, then $V_3' \notin \mathcal{LS}(\overline{V_1V_2})$.*

(Because all contours are augmented contours, it is impossible for one of V_1 or V_2 to satisfy Case 1 while the other satisfies Case 2.)

Theorem 3 is required in addition to Theorem 2 because some invalid slice chords can satisfy Theorem 2.

DEFINITION 7. $\mathcal{I}(C)$ and $\mathcal{O}(C)$ denote the inside and outside regions of a simple polygon C, respectively. Neither $\mathcal{I}(C)$ nor $\mathcal{O}(C)$ contains C.

THEOREM 4. *Let T be a slice chord, and C be any contour. Then T' cannot have intersections with both $\mathcal{I}(C)$ and $\mathcal{O}(C)$.*

Theorem 4 does not imply that T' cannot have any intersection with contours. Figure 8 shows some valid and invalid slice chord projections.

THEOREM 5. *Let T_2 be any existing slice chord, and T_1 be the newly proposed slice chord.*

1. *If T_1' crosses T_2', then T_1 cannot be a slice chord.*

2. *If T_1' intersects, but does not cross, T_2', and T_1 crosses T_2 (in 3D), then T_1 cannot be a slice chord.*

LEMMA 4. *Suppose $\overline{U_iU_j}$ and $\overline{V_kV_l}$ are two contour segments of two corresponding contours, and there exist two slice chords $\overline{U_iV_k}$ and $\overline{U_jV_l}$ as shown in Fig. 9, then $i - j = k - l = 1$ or -1. Here i, j, k and l are the sequence indices of vertices.*

This lemma implies that the tiling sequence cannot be increasing for one contour and be decreasing for its corresponding contour.

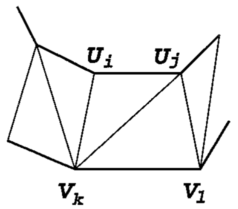

FIG. 9. If $\overline{U_iV_k}$ and $\overline{U_jV_l}$ are two slice chords, then $i - j = k - l$. Here i, j, k, and l are the sequence indices of vertices.

LEMMA 5. *Let T be a slice chord incident with a vertex V on a contour C. If V is a positive vertex, then $T' \subset (\mathcal{I}(C) \cup C)$. If V is a negative vertex, then $T' \subset (\mathcal{O}(C) \cup C)$.*

Lemma 5 simply restates Theorem 2 in a different way.

THEOREM 6. *If a tiling triangle can be placed between two contours $C1$ and $C2$, then one of the following is true* (*see Fig.* 10).

1. *$C1'$ intersects $C2$.*

2. *If $C1'$ and $C2$ are disjoint, then they have different orientations, each has at least one negative vertex, and their $\mathcal{NEC}$s do not insulate $C1'$ and $C2$. In other words, there exist non-overlapping vertices $V_1 \in C1$ and $V_2 \in C2$ such that $\mathcal{NEC}(V_1') = \mathcal{NEC}(C2)$, and $\mathcal{NEC}(V_2') = \mathcal{NEC}(C1)$.*

3. *If one contour's* ($C1$) *projection is inside the other* ($C2$), *then they have the same orientation, $C1$ has at least one negative vertex, $C2$ has at least one positive vertex, and there is no contour insulating $C1'$ and $C2$. In other words, there exists a nonoverlapping vertex $V_2 \in C2$ such that $\mathcal{NEC}(V_2') = \mathcal{NEC}(C1)$, and $C2 = \mathcal{NEC}(C1')$.*

THEOREM 7. *If any of the three conditions of Theorem 6 holds, then there exists a path on the reconstructed surface linking these two contours. This does not imply that a tiling triangle always exists in between.*

Theorems 6 and 7 state the necessary and sufficient conditions for the correspondence of two contours in adjacent slices.

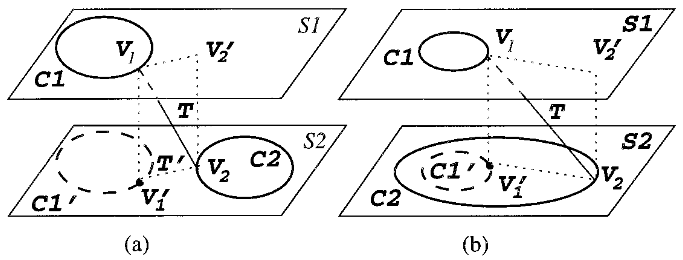

FIG. 10. (a) C' and $C2$ are disjoint; (b) $C1'$ is inside $C2$.

4. IMPLEMENTATION

Implementation of the correspondence and tiling rules of the preceding section are discussed in Section 4.1. This implementation guarantees that the reconstructed surfaces satisfies the three criteria stated in Section 3. It does not, however, guarantee numerical stability. Two minor modifications, which are discussed in Section 4.2, do make the implementation stable.

4.1. *General Implementation*

Section 3 provides precise rules to determine correspondences between contours on adjacent slices. The tiling rules on the other hand only prohibit bad tilings—they do not suggest good tilings. It was thus necessary to develop a multipass tiling algorithm to achieve reasonably good tiling. It first constructs tilings for any regions not violating any of the tiling rules. Regions that violate these rules correspond to holes, branching regions and dissimilar portions of contours. They are processed by tiling to their medial axes.

In addition to the notation in Section 3, we define OTV to be an optimal tiling vertex. $V_1 = OTV(V_2)$ if $\overline{V_1V_2}$ is the shortest among all slice chords incident with V_2 that satisfy Theorems 2 and 4. An OTV pair (U, V) implies $OTV(V) = U$ and $OTV(U) = V$.

Our implementation has the following major steps:

Step 1: form closed contours from image slices.

Step 2: create any required augmented contours (Definition 3).

Step 3: find correspondences between contours.

Step 4: form the tiling region (Theorem 2) of each vertex.

Step 5: form the OTV table.

Step 6: construct the tiling.

Step 7: collect the boundaries of untiled regions.

Step 8: form triangles to cover untiled regions based on their edge Voronoi diagram (EVD).

As mentioned in Section 2, the overall procedure of our algorithm is very similar to the approach of Barequet *et al.* [1]. The effect of our Step 2 is equivalent to enforcing short matches at the intersection of the contour projections in their algorithm. Our Step 5 is similar to the norm distance calculation between the consistently oriented contour portions of their approach. We employ a multipass tiling algorithm to collect pieces of tiling triangles while their algorithm uses voting to find a long match between contours. Our tiling algorithm gives much smaller untiled regions than their approach because of the dependence of their approach on the tolerance used in the matching. The use of a small tolerance in their approach results in large unmatched regions, while a large tolerance could result in multiply matched regions. We use Theorem 2 to avoid incorrect tiling while they rely on a tolerance selection. The procedure we use in Step 7, which collects the boundaries of untiled regions, is similar to Barequet's algorithm. The major difference between our approach and theirs is that we have theoretical foundation to justify our approach. Our postprocessing of untiled regions is similar to Geiger's approach [12]. The primry difference is that the *EVD* added in our postprocessing is a subset of the added external Voronoi skeleton (*EVS*) in the preprocessing of Geiger's approach.

Our program can have as input either image slices or contour data. In the case of contour data, Step 1 is skipped. Step 2 is required because our theorems are based on augmented contours. Step 3 finds the correspondences required by the calculation of the *OTV* table and tiling. Step 4 produces the data structure for the verification of Theorem 2. Step 5 precalculates the *OTV*s of every vertex. The *OTV* table is required for determining the optimality of a proposed slice chord during Step 6, which generates tiling triangles. Holes, branching regions, and dissimilar portions of contours cannot be tiled, so they are detected in Step 7 and postprocessed in Step 8. The detailed implementation of each step is described in the following paragraphs.

The 2D marching cubes algorithm [18] is used to generate contour segment from an image slice. We assume that the image objects do not intersect the slice boundary. Therefore, all generated contour segments can be linked to form simple polygons. Each contour segment from the 2D marching cubes algorithm is miniscule, so the contours are approximated by fewer contour segments under an error tolerance [9]. Choosing a half-pixel as the approximation tolerance effectively eliminates 75–90% of contour segments. If the approximation causes intersections between contours of the same slice, the approximation process for the intersecting contours is repeated with a smaller tolerance.

Based on Lemma 3, augmented contours are formed. For the example in Fig. 11a, $c_1' \cap c_2 = V_1'$, so V_1' is added to c_2 as an overlapping vertex. Similarly, V_1 is added to c_1. Although Lemma 3 does not require the insertion of any new vertices if the intersection of one contour segment projection with another contour segment is one line segment (e.g., Fig. 11b), we still insert new vertices (V_1' and V_2') so that the overlapping part is a contour segment in both slices. This allows the tiling algorithm to guarantee that the overlapping part in both slices will be used. An untiled region will thus not cross itself.

Theorem 6 judges the correspondence between contours on different slices. One table for each slice stores the contour relationships (disjointedness or enclosure) of that slice. These tables are used to derive the $\mathscr{NEC}$ (Definition

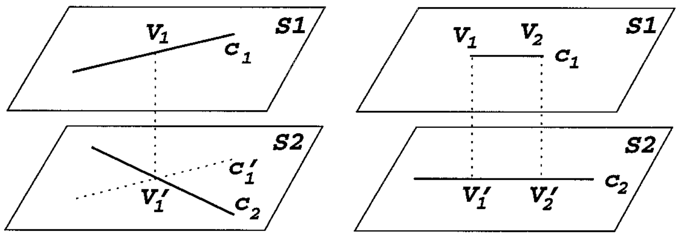

FIG. 11. Projection of contour segment c_1 onto slice $S2$ containing contour segment c_2: (a) intersection; (b) overlap.

4) of a vertex V or a contour C. Another table stores the contour relationships (intersection, disjointedness, or enclosure) between adjacent slices. The $\mathscr{NEC}$ of every vertex projection is searched on the projection slice. Once the $\mathscr{NEC}$ of a vertex projection is available, whether the vertex is *positive* or *negative* can be determined by Definition 5. Finally, Theorem 6 determines the correspondence between contours of adjacent slices. This takes $O(n^2)$ time because checking a point inside a contour is an $O(n)$ time algorithm and there are $O(n)$ vertices.

The tiling region of each vertex is defined in Theorem 2. Two line equations and an AND–OR flag define the tiling region described by Theorem 2.1. In the example of Fig. 7b, the tiling region of the vertex q is on the left side of line pq or line qr. If V is an overlapping vertex, then (V, V') is an *OTV* pair. So V' can be indexed from the *OTV* table formed in Step 5, and no extra data structure is needed to verify Theorem 2.2.

The *OTV* table stores the *OTV*s of all vertices. The optimality is in terms of the shortest slice chord, as used by Christiansen *et al.* [4]. The *OTV* on the adjacent slice is searched for each vertex V. The current implementation sorts the candidate vertices based on their distance to V. Then the candidate vertices, starting from the closest vertex, are checked until one satisfies Theorems 2 and 4. Usually, the closest candidate is qualified. So the average time complexity is $O(n^2 \log n)$. But in the worst case of no qualified vertex, all vertices on the corresponding contour are tested. This implementation therefore has an $O(n^3)$ worst case time complexity. (There are $O(n)$ vertices. Checking Theorems 2 and 4 requires $O(1)$ and $O(n)$ time, respectively, and there are $O(n)$ checks for each V in the worst case.) This step is usually the most time consuming part of the whole process. Our method can be sped up by presorting all vertices into an RPO tree [5] so the closest-point query can be done in $O(\log n)$ time. Furthermore, the $O(\log n)$ algorithm described by Preparata [22] can be used to check Theorem 4. The time complexity of this approach is $O(n \log n)$ average case and $O(n^2 \log n)$ worst case.

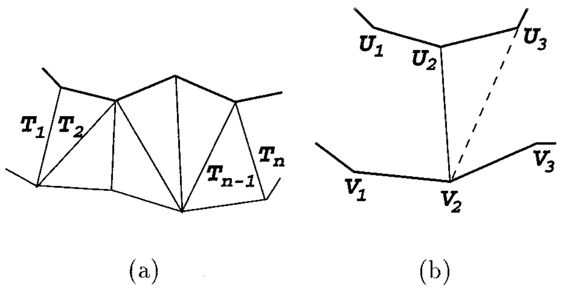

FIG. 12. (a) Only the boundary slice chords T_1 and T_n are needed to verify Theorem 5; (b) the optimality of forming $\triangle U_2V_2U_3$ can be classified into six cases.

During the tiling, the *legality* of each slice chord is checked frequently for compliance with Theorems 2–5. Verifying Theorems 2 and 3 for a proposed slice chord takes $O(1)$ time. Theorem 4 takes $O(n)$ or $O(\log n)$ time depending on the implementation. Verifying Theorem 5 takes $O(n)$ time because the number of existing slice chords is always increasing and they cannot be preprocessed. Fortunately, it is not necessary to check for intersections of proposed slice chords with all existing slice chords. Only the boundary slice chords of tiled regions need to be verified. For the example in Fig. 12a, only the two boundary slice chords T_1 and T_n need to be checked. The existing slice chords T_2 to T_{n-1} are bounded by T_1, T_n and contours, so it is impossible for the projection of the proposed slice chord to cross any of T_2 to T_{n-1} without crossing T_1, T_n, or any contour.

There are four passes in tiling. The tiling sequence is based on the optimality of the tiling pair, with optimality defined in terms of the shortest slice chord. Suppose $\overline{U_2U_3}$ and $\overline{V_2V_3}$ are two contour segments of two corresponding contours, as shown in Fig. 12b. We are going to form a tiling triangle $\triangle U_2V_2V_3$ or $\triangle U_2V_2U_3$. Suppose slice chords $\overline{U_2V_2}$ and $\overline{U_2U_3}$ and legal, and $\overline{U_2U_3}$ has not been used. Considering only one triangle $\triangle U_2V_2U_3$, we can classify it into six cases in decreasing order of degree of optimality.

Case 1: (U_2, V_2) is an OTV pair, and so is (U_3, V_3).

Case 2: (U_2, V_2) is an overlapping vertex pair.

Case 3: (U_2, V_2) is an OTV pair, and $V_2 = OTV(U_3)$.

Case 4: (U_2, V_2) is an OTV pair, $V_2 \neq OTV(U_3)$, and V_2 and $OTV(U_3)$ are on the same contour.

Case 5: (U_2, V_2) in not an OTV pair, and V_2 $OTV(U_2)$ and $OTV(U_3)$ are on the same contour.

Case 6: All other cases.

Cases 1–3 are all considered to be optimal. Cases 4, 5, and 6 are not optimal and their degree of optimality decreases with increasing case index.

There are four passes in tiling: The first pass handles Cases 1–3, the second pass handles Case 4, and so on.

In the first pass, the OTV table is scanned to look up an optimal tiling pair. Lemma 4 states that there are only two spanning directions for tiling. If Case 1 is encountered, then the quadrilateral $U_2V_2V_3U_3$ is divided into two triangles using the shorter slice chord. In Cases 2 or 3, only one triangle ($\triangle U_2V_2U_3$ or $\triangle U_2V_2V_3$) is chosen in one span direction. The selection is based on the shorter of $\overline{U_2V_3}$ and $\overline{V_2U_3}$ if both triangles satisfy Cases 2 or 3. Tiling triangles are tested and may be formed on both sides of $\overline{U_2V_2}$. The boundary slice chords and their directions (spanning right or left) are put into a boundary slice chord array. If the to-be-stored boundary slice chord is already in the array, it is already shared by two tiling triangles, and thus it is deleted from the boundary slice chord array.

Cases 4–6 are handled in the three subsequent passes. One starting tiling pair with its direction is popped out from the boundary slice chord array, and the tiling spans one direction until no satisfying case is available. The starting tiling pairs of the second pass also come from the OTV table. As in the first pass, the boundary slice chords are stored in an alternative boundary slice chord array. The tiling takes $O(n^2)$ time because checking Theorem 5 takes $O(n)$ time, and there are $O(n)$ proposed slice chords. Compared to making only one pass to do all tiling, our multipass approach does not significantly increase the number of proposed tiling pairs, which is proportional to the processing time.

Figure 13 shows a rather complicated many-to-many branching tiling. Fig. 13a shows the top view of contours of two slices. Two bottom contours (C_2^1 and C_2^2) branch into five top contours ($C_1^1 - C_1^5$). In addition to this complicated branching, a very dissimilar contour C_1^4 and a hole C_1^2 are involved. The tiling result after the first pass is shown in (b); only good tiling triangles are formed. As each pass proceeds, the algorithm loosens the optimality requirements in order to form more tiling triangles. The last pass uses any legal triangle. The result after the last pass is shown in (c).

After all four passes, there may be untiled regions in dissimilar contours, holes or branching regions. The processing of untiled regions is illustrated in Figs. 13c–13f. The untiled regions can be traced from the unused contour segments (the dotted lines in (c)) and the boundary slice chord array. The boundary slice chord is assigned a direction from the bottom slice to the top slice if it spans left, or from top to bottom if it spans right. The directions of the top unused contour segments are reversed. These directions are required to trace a closed untiled polygon when more than two untiled contour segments or slice chords share the same vertex. Figure 13d shows the top view of the traced untiled polygons.

If the projection of an untiled region is convex, it is

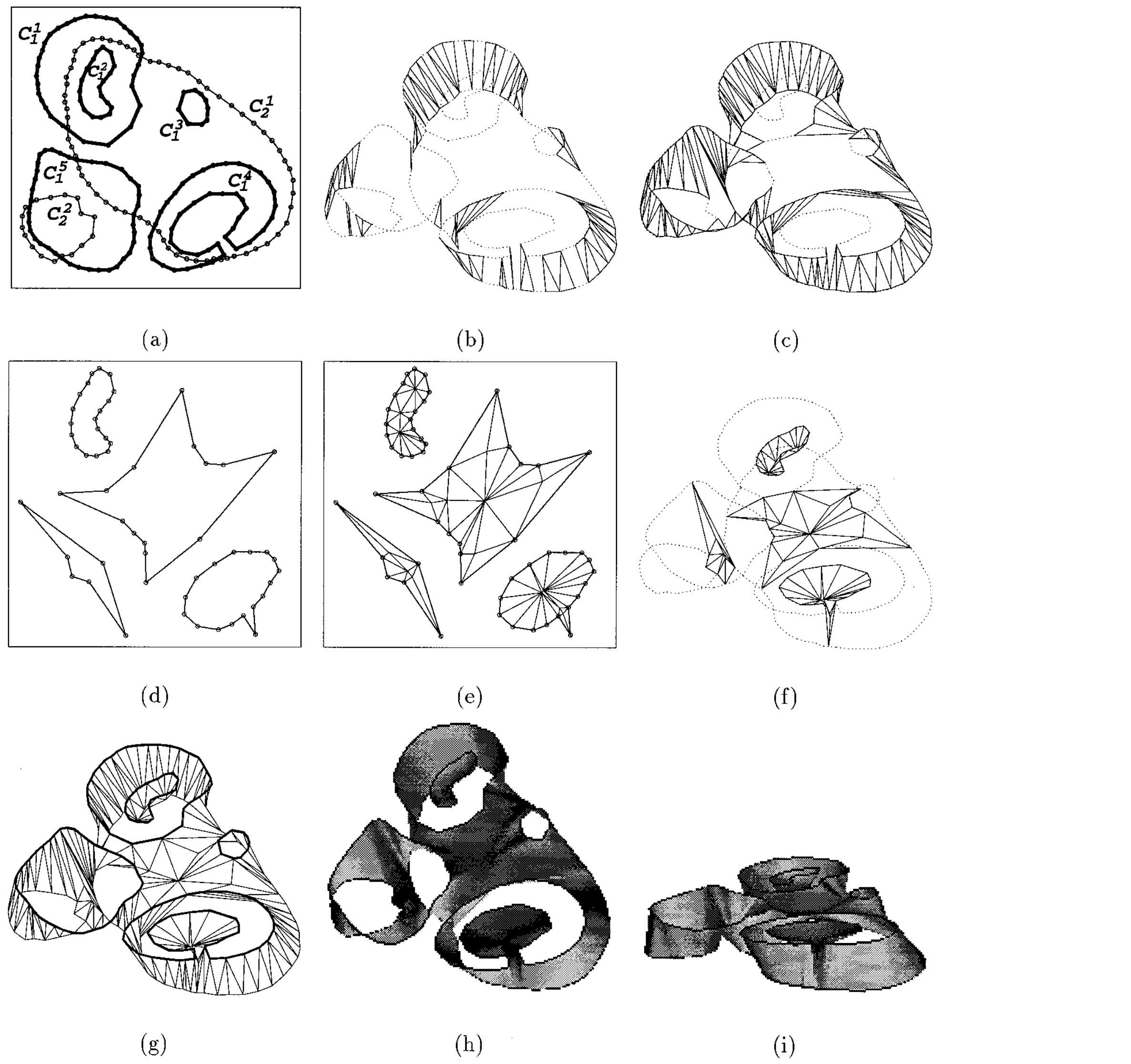

FIG. 13. (a) Two slices of contours. Thicker contours are from the top slice. The small circles denote vertices. (b) Result of the first tiling pass. Only good tiling triangles are formed. (c) The result of all tiling passes. (d) Top view of untiled regions. (e) Top view of untiled region triangulation by edge Voronoi diagram. (f) The perspective view of (e) with hidden lines removed. (g) The final result. (h) and (i) Two different shaded views of the tiled surface shown in (g).

triangulated with its center of gravity. Otherwise, we triangulate using its medial axis. We use Lee's algorithm [16] to find the *EVD* (edge Voronoi diagram or medial axis). The *EVD* is approximated by a smaller number of line segments so fewer triangles are required to cover the untiled region. The *Z* values of the *EVD* are set to the middle of two slices so the reconstructed model corresponds to our expectations of the physical object. The result is shown in Fig. 13e. The perspective view of the triangulated untiled regions is shown in Fig. 13f in which the contours are plotted as dotted line segments. Figure 13g shows all traingles from tiling and the *EVD*. Two different perspective views of the shaded result are presented in (h) and (i).

In the case of one untiled region's projection enclosing the projection of another untiled region, one can use the *EVD* algorithm of Srinivasan *et al.* [27] to calculate the medial axis of nested polygons. Because our algorithm tries to do as much tiling as possible during the last tiling pass, the number of untiled regions left is near minimum. The case of nested untiled regions rarely happens; in fact, we have never encountered it in real image data.

4.2. *Numerically Stable Implementation*

Two minor modifications of the algorithm sketched in Section 4 help avoid numerical instability. The first moification abolishes some augmented contours, and the second develops an algorithm to find the rough medial axis.

The augmented contours are required because of Lemma 3. Augmenting a contour may add vertices which are very close to existing vertices. This results in numerical instability in programming because it is difficult to consistently treat two very close vertices as one vertex or two separated vertices. Besides, this results in very sharp triangles. These problems can be solved by not requiring Criterion 2 at *crossed* contour segments. This minor modification affects the reconstructed surface only at crossed contour segments; the overall shape is not changed.

Lemma 3 shows that two crossed contour segments cannot be tiled because of Criterion 2, so augmented contours are formed so there are no crossed contour segments. If we waive the Criterion 2 requirement only at crossed contour segments and directly tile crossed contour segments, there is no need to form augmented contours. The non-planar quadrilateral formed by two crossed contour segments c_1 and c_2 in Fig. 11a is divided into two triangles. In the case of overlapping projection contour segments (e.g., Fig. 11b), the quadrilateral is planar and is triangulated. This is done before the first tiling pass, and then the same four-pass tiling algorithm applies. If a contour segment has n intersections (counting multiplicities) with the projections of other contours of the adjacent slice, then $n - 1$ new vertices are inserted to break this contour segment into n contour segments. A new vertex is placed at the middle of every two adjacent intersections. This step is necessary to ensure that one contour segment can cross at most one contour segment projection.

Because the added vertices of this implementation are at the middle of two intersections of one contour segment, they are rarely close to any existing vertex. This also adds many fewer new vertices compared to the general implementation because most intersected contour segments have only one intersection and no vertex is added. The lack of Criterion 2 in the numerically stable implementation is limited to crossed segments, and it does not affect the formation of other regions. It has few effects on the overall shape, and the problems of very close vertices are solved. Figure 14 shows the results of the general implementation and the numerically stable implementation.

It is difficult to implement a numerically stable *EVD* algorithm for different situations. So we developed a program to find the rough medial axis. A polygon is decomposed into two polygons by adding a cutting edge (the dahsed line in Fig. 15). The decomposition is repeated until all generated polygons are convex. The rough medial axis (shown as the darker line segments in Fig. 15) is formed by linking the centers of the convex polygons and the middle points of the cutting edges.

5. RESULTS

The algorithm has been implemented in *C*, and runs on Sun Sparc and Silicon Graphics Indigo2 workstations. It has also been ported to an Intel Paragon. Both the synthetic and real cases used to test these implementations are presented in this section.

Figure 13 illustrates the capabilities of our algorithm when many-to-many branching, dissimilar contours and holes are present. As can be seen from Fig. 13, C_1^2 is reconstructed as a shallow hole since it has no corresponding contour on the bottom slice. The dissimilar portion of C_1^4 does not tile to the bottom contour C_2^1. It tiles to its medial axis and forms a shallow hole with a link to C_2^1. This is a highly likely topology. The handling of branching is shown in Fig. 13f and in Fig. 16. The added curve could be a point to model the saddle point, as in Fig. 16b, if the branching region is not complicated, or it could be line segments to model canyons between contours, as shown in Fig. 16d.

Figure 17 shows a dissimilar contour case which causes many tiling algorithms to fail. Figure 17a illustrates the top view. The wire frame result of the minimum surface area optimizing algorithm is shown in (b). The result is redisplayed in (c) with hidden lines removed. The arrow points to the abnormality. Figure 17d is the result of the shortest slice chord heuristic algorithm. It is much worse than (c). Figure 17e shows at least one polyhedron solution. The result of our algorithm is shown in (f). Heuristic methods fail badly because (1) they do not check the limited region associated with a slice chord (Theorem 2), and (2) they cannot change any previous improper slice chord selection. The first drawback is also associated with the optimal algorithm, but the optimal algorithm can recover from this drawback because it selects the optimal one from a large set of possible solutions. Our algorithm guarantees that the results are polyhedra because of Criterion 1. It forms a natural shape in the dissimilar region because of Criterion 2.

For a test with real data, we used the tiling algorithm to reconstruct different parts of the human body in order to build a database for a human body walkthrough. The algorithm worked very well in dense data, as will be seen with results from three sets of medical data.

The rendering tool (see Acknowledgments) is based on the hardware platform independent graphic library *XS* developed at the *Shastra* laboratory at the Computer Sciences Department, Purdue University. Figure 18 shows the Gouraud shading and the wire frame of the reconstructed surface of a brain hemisphere. It was generated from a set of contour data that was manually traced from 52 MRI image slices. Figure 19a shows a reconstructed skull. The noise around the teeth was present in the original image slices. The surface was automatically generated from 112

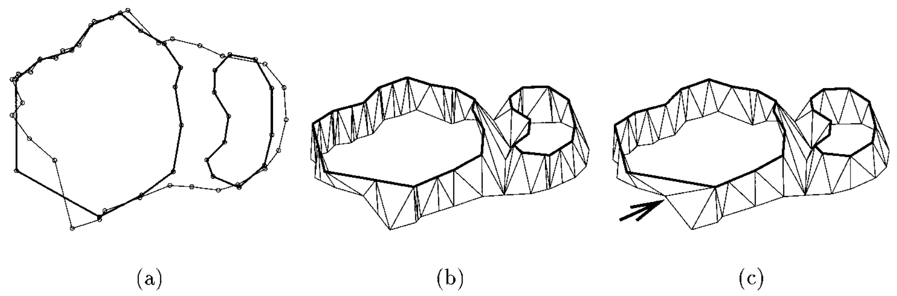

FIG. 14. Results of the general implementation and the numerically stable implementation: (a) Top view of contours. Top contours are drawn in thick line segments. (b) The general implementation: it results in a very sharp triangle if the added vertex is very close to an existing vertex. (c) The numerically stable implementation: it does not generate very sharp triangles, but surfaces may be distorted (indicated by the arrow) because of not satisfying Criterion 2.

256 × 256 CT slices. The image slices were first enhanced by processing with a 3 × 3 median filter before automatic contour segmentation.

Figures 19b–d show two adjacent slices around the nasal area and the tiling between them. As can be seen from (b) and (c), there are numerous holes, dissimilar areas, and branching between two adjacent slices. Figure 20 shows three different views of the skeleton reconstruction of the human cadaver Freddy. The tiling of some cross sections is shown in Fig. 21. The image volume consisted of 920 256 × 256 slices. The resolution was 2.16 mm in all dimensions. Each image slice was first enhanced by processing with a Gaussian weighted low pass filter before automatic contour segmentation. The global thresholding scheme employed in the 2D marching cubes approach did not work well in regions which were barely visible, such as at the shoulder bones. However, our tiling algorithm takes whatever contours are generated and produces surfaces based on those contours. Therefore, these three examples show the capability of our algorithm to handle complicated topologies.

The error tolerance during the approximation of the skull and Freddy contours was 0.5 pixel. The marching cubes approach generated 554,500 triangles from the skull data and 1.4 million triangles from the Freddy data, respectively. Our approach generated significantly fewer (54,071 and 285,349, respectively) triangles than the marching cubes approach. Table 1 summarizes our results. The CPU time in the table is based on SGI Indigo2 HIIMPACT workstations and does not include the image segmentation time.

6. CONTRIBUTIONS AND LIMITATIONS

The theoretical approach and the algorithms developed in this paper differ significantly from those currently available. It is thus important to clearly describe the benefits they provide to both the theory and practice of surface reconstruction. It is equally important to describe the limi-

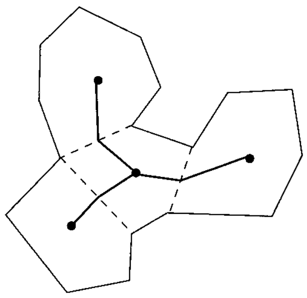

FIG. 15. Finding the rough medial axis by repeatedly decomposing a polygon into two polygons.

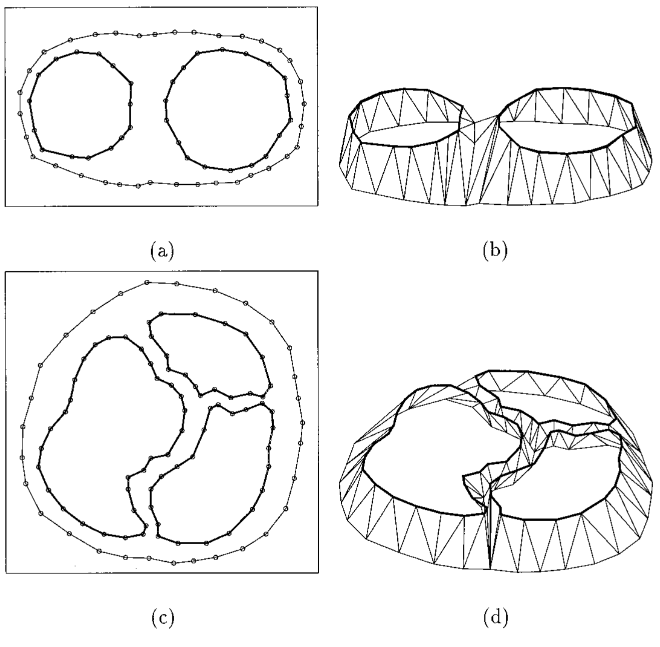

FIG. 16. Tiling in the presence of branching: (a) and (c) Top view of two different cases of branching contours. The thicker line segments are top contours and small circles represent vertices. (b) and (d) Our results with hidden lines removed.

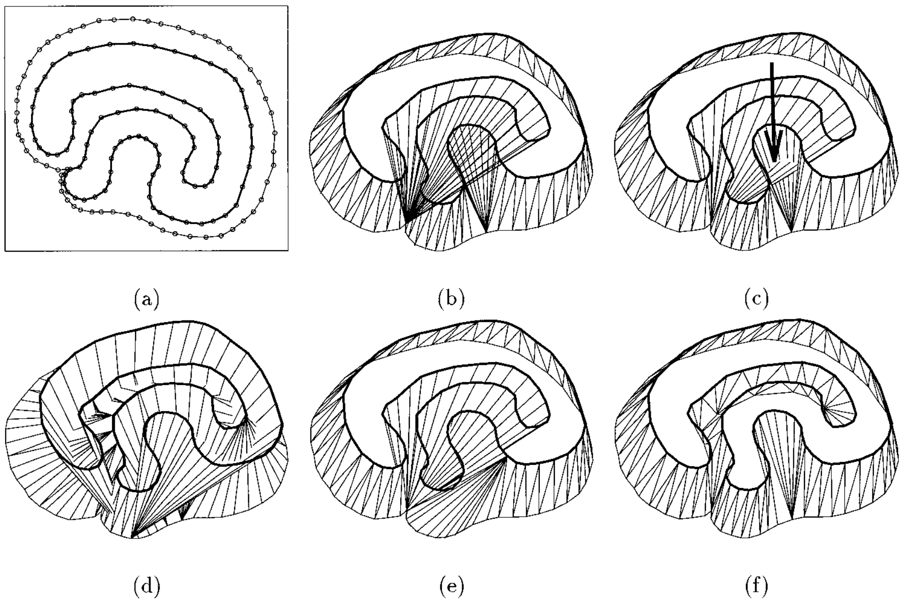

FIG. 17. Tiling of two dissimilar contours: (a) Top view of two contours. The thicker contour is the top contour and small circles represent vertices. (b) Result of minimum surface area tiling. (c) Same as (b) with hidden lines removed. The arrow points to the abnormality where triangles intersect non-adjacent triangles. (d) Result of shortest slice chord heuristic algorithm. (e) Existence of a nonselfintersecting surface. (f) Our result.

tations of these results, both to ensure that the algorithms are used appropriately, and to clarify issues that should motivate future work.

6.1. *Contributions*

The primary contribution of this paper is the theoretical foundation supporting our surface reconstruction algorithm. It consists of the three constraints imposed on the reconstructed surface and the derivation of precise correspondence and tiling rules that satisfy them. This constraint-based approach produces a reconstruction algorithm which can generate an expected physical surface from any topology that is encountered. This is not the case, for example, with algorithms based on heuristic methods for tiling and reconstruction. Such algorithms often fail on topologies that were not directly anticipated (see Section 2).

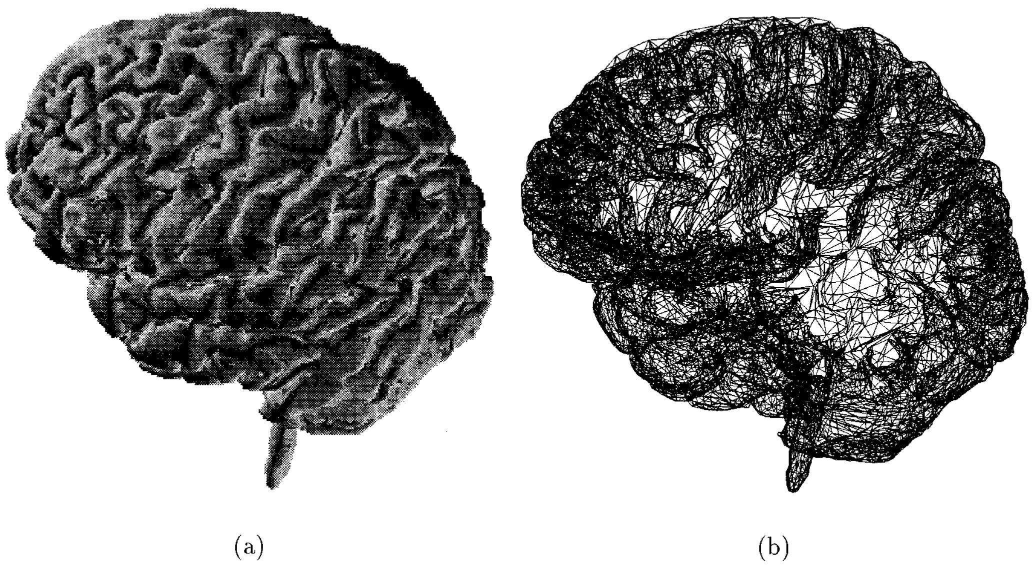

FIG. 18. Visualization of a reconstructed brain hemisphere: (a) Gouraud shading; (b) wire frame.

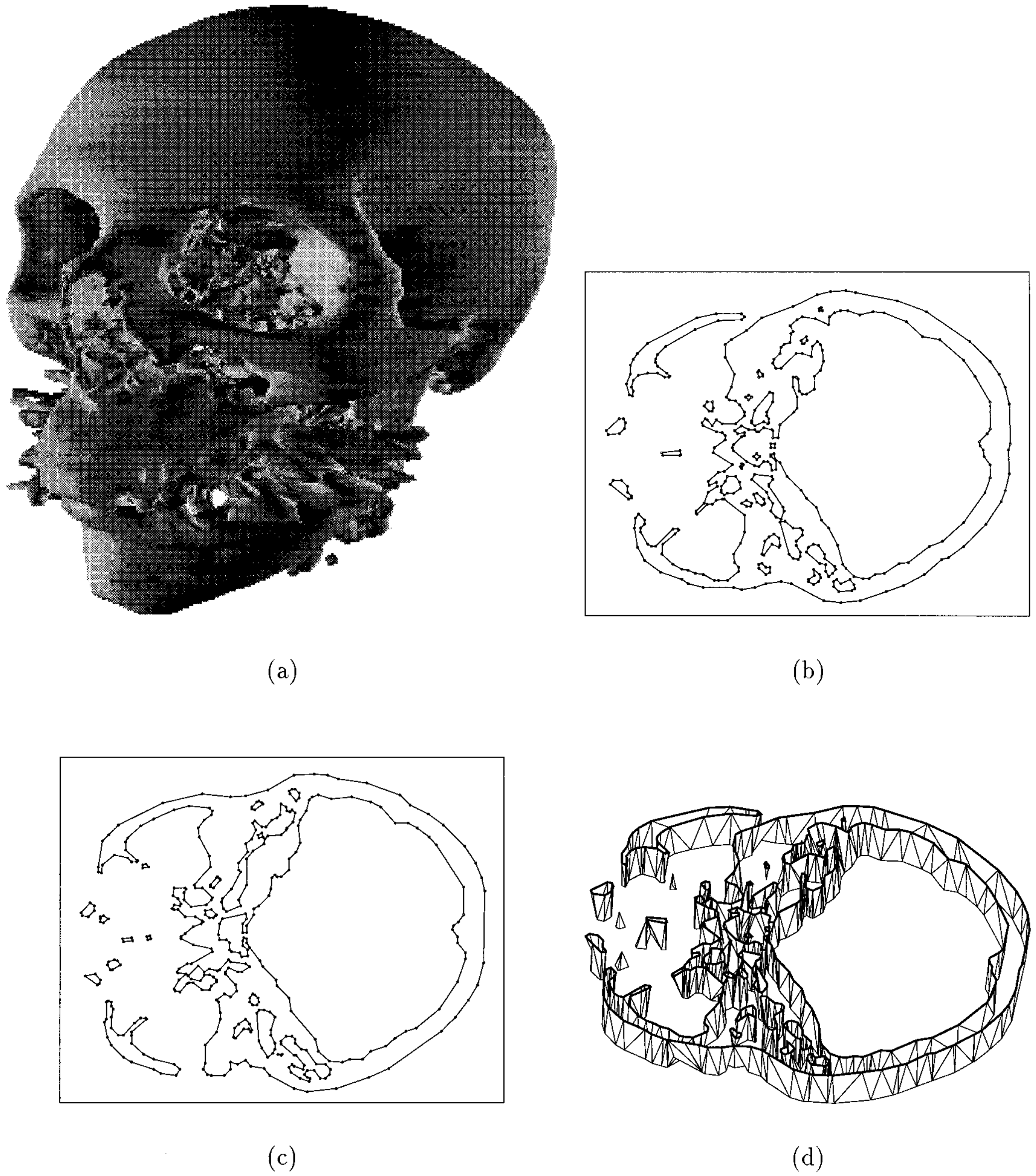

FIG. 19. Visualization of a reconstructed skull: (a) Gouraud shading; (b) and (c) two sample slices; (d) the tiling of (b) and (c).

The importance of these constraints is also made clear by the abnormal surface, shown in Fig. 17c, generated by a minimum-surface-area algorithm.

The second major contribution is the development of a robust algorithm whose reconstructed surfaces correspond well with the surfaces of the actual object. The aspects of our approach which make this possible are: (a) the data of our reconstructed model cannot flip its sign more than once along a perpendicular line between adjacent slices; (b) the resampling of our reconstructed surfaces is guaranteed to produce the original contours; (c) our method of handling branching method works well on very complex branchings; and (d) a region which cannot be tiled using our explicit rules is tiled with its medial axis.

The third major contribution of our research is our multipass tiling algorithm. Traditional tiling algorithms have problems in branching regions because it is difficult to know exactly when to switch from tiling rules based on the assumption of a one-to-one correspondence of contours to rules based on the assumption that a region is branched. Many papers try to avoid this problem by identifying branching regions and then inserting composite contours or intermediate contours in order to reduce the tiling of the branched topology to that of tiling uniquely paried contours. As discussed in Section 2, such approaches have drawbacks. We have avoided them by designing a tiling algorithm that makes several passes. Successive passes tile according to progressively weaker optimality rules—but

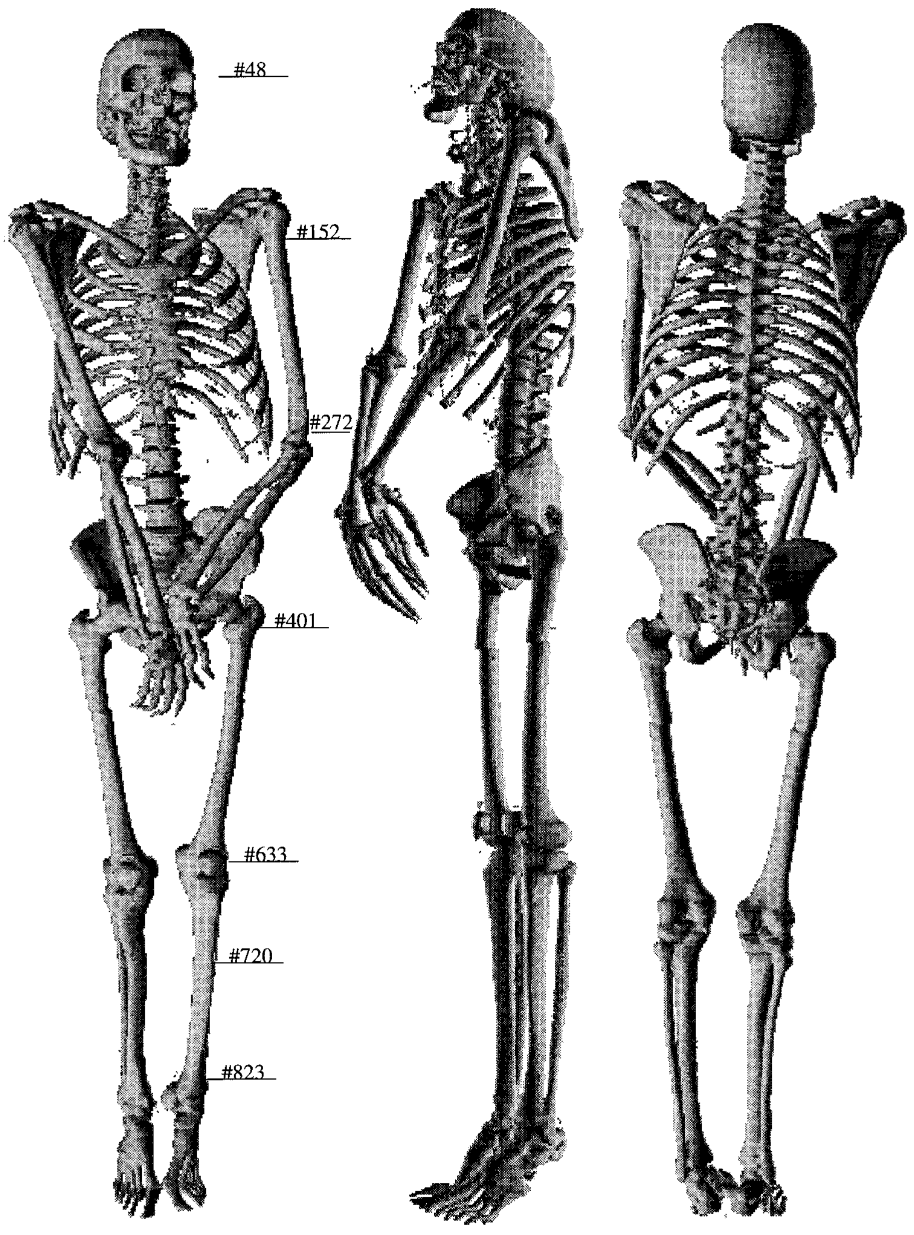

FIG. 20. Tiling based visualization of an entire human skeleton (Freddy). The tilings of the numbered portions are shown in the next figure.

in all cases satisfying the surface reconstruction constraints. At the end of these passes, branching regions, holes and dissimilar areas of contours are left untiled. We then tile these well-defined remaining regions. As discussed in Section 5, this new tiling algorithm produces very reasonable tilings for very complicated branching structures. The most probable reason for the success of this new approach is that it tiles "difficult" regions only after it has tiled as much of the rest of the surface as possible. It thus has as much surface information as possible when it has finally reached the point when it must tile the "difficult" regions.

6.2. *Limitations*

As discussed above, one strength of our algorithm is that we know exactly what type of surfaces it will construct because of the constraints imposed on them.

One of these constraints, though, is actually a requirement on the sampling rate used when imaging an object. Specifically, Criterion 2—which states that the reconstructed surface between adjacent slices can have at most one intersection with any line perpendicular to the slices—requires that the inner-slice spacing of the data be fine enough that certain topologies are unlikely. Examples of

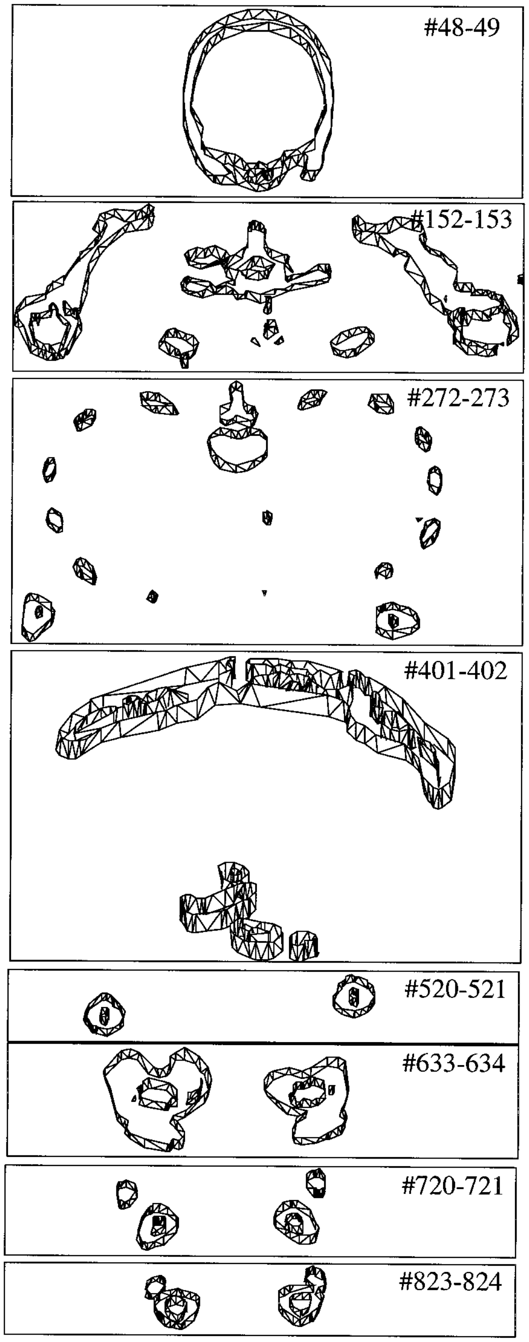

FIG. 21. Multiple contours with tiling for different horizontal cross sections through the human skeleton model.

topologies which violate Criterion 2 because of undersampling in a given direction are provided in Fig. 22. In each case, the true surface of the object cannot be reconstructed by our algorithm because the surface it produces must satisfy Criterion 2. Our algorithm will still finish, but the surfaces produced in these cases will be incorrect.

In cases (a) through (c) in Fig. 22, a preprocessing stage would enable our algorithm to reconstruct the actual surface. For example, the surface in Fig. 22a can be generated if we apply our algorithm only to those contours with the same number of enclosing contours (see Definition 4). This function has been implemented in our program. The surfaces in Figs. 22b and 22c could be correctly generated by translating and scaling the contours until Theorem 6 generates the actual correspondence, and then translating and scaling back the tiling results. O'Rourke [21] points out a potential flaw in scaling and suggests using a uniform scaling (with the same amount in both x and y). In this latter case, though, a significant amount of side informtion must be present in order to determine that the actual object must be as shown in the figure.

No amount of preprocessing would enable our algorithm to correctly reconstruct the surface of Fig. 22d. In such badly undersampled data, the aliasing is severe enough that it is completely unclear what to do. The *best* way to handle this case would be to acquire additional data in order to decrease the interslice distance.

TABLE 1
Results

	Segment. method	# of slices	# of $\triangle$'s	CPU time (s)	Step 3 time	Step 5 time	Step 6 time	Marching cubes $\triangle$'s
Brain	Manually	52	37,992	90	27%	60%	10%	N/A
Skull	Auto.	112	54,071	104	31%	51%	15%	554,500
Freddy	Auto.	920	285,349	248	43%	32%	20%	1,416,584

7. CONCLUSION

This paper presented a robust algorithm for reconstructing surfaces from a set of planar contours or image slices. The theoretical derivation of the correspondence and tiling rules allowed our algorithm, given any input data, to generate a unique topology satisfying the desired surface criteria. This new, unified approach led to reconstructed surfaces which correspond well with the surface of the physical objects that were imaged.

Our algorithm is not guaranteed to produce an optimal solution, but it does achieve the following improvements over previous surface-based techniques:

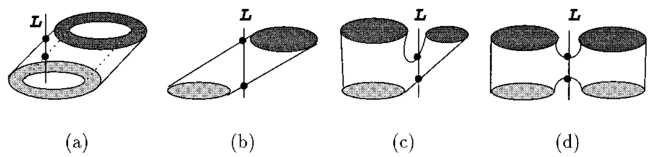

FIG. 22. Some examples of topologies that cannot be processed because they violate Criterion 2 by having two points of intersection with a line perpendicular to the slices.

• It avoids such major drawbacks of general heuristic tiling procedures as the generation of twisted surfaces because of lack of global information.

• Unlike other tiling approaches which may generate self-intersecting surfaces, it guarantees that the tiling result is a polyhedron surface.

• It produces appropriate tilings in branching regions.

• It generates significantly fewer triangles than the marching cubes approach.

These improvements were all demonstrated with both real and synthetic medical data.

The strength of our approach is its ability to handle dissimilar contours and complicated branching. Its primary drawback is one faced by any surface-based algorithm—when the interslice distance is too large, it becomes very difficult to solve the correspondence problem without additional algorithms or human intervention. In our case, this problem manifests itself as a violation of Criterion 2. The probability that these violations occur increases as the interslice distance increases.

APPENDIX

In addition to the definitions in Section 3, we further define $\mathcal{DISK}(P)$ as the disk region on a slice centered at a point P. The radius is arbitrarily small. We use $\mathcal{SURF}$ to represent any reconstructed surface satisfying criteria 1–3. We consider the scalar data to be positive, neutral or negative if it is inside, on the boundary of, or outside the polyhedra stated in Criterion 1, respectively.

Proof of Lemma 1. Suppose L does not intersect any solid region but intersects $\mathcal{SURF}$. Because of Criterion 2, this intersection must be exactly one line segment or one point. The scalar data along L is negative at both ends and is neutral at one line segment or one point between two slices. This means that $\mathcal{SURF}$ would need to curve back at L. Thus there exists a vertical line $L1$ very close to L such that $L1$ intersects $\mathcal{SURF}$ twice. This violates Criterion 2. So L must intersect at least one solid region.

Suppose L intersects two solid regions. We can apply the same argument to the complement (dual) of the solid to contradict this assumption. So L must intersect exactly one solid region. ■

Proof of Lemma 2. $N = 1 \Rightarrow M = 1$ (Lemma 1). If $M = 1$, then the scalar data at the intersection end of L is positive, and that of the other end is negative. This implies that there is a zero crossing (intersection) in between. So $M = 1 \Rightarrow N = 1$, thus $M = 1 \Leftrightarrow N = 1$. Lemma 2.2 is proved by inverting both sides of Lemma 2.1. ■

Proof of Lemma 3. As shown in Fig. 23, c_1 and c_2 are two contour segments on different slices. Suppose c_1' crosses c_2 at p', then $\mathcal{DISK}(p')$ is divided into R_1, R_2, R_3, and R_4. The solid regions are shown as the shadow regions. Based on Lemma 2, the projection of the $\mathcal{SURF}$ is in $R_1 \cup R_3$, and not in $R_2 \cup R_4$. However the projection of the reconstructed triangle containing c_1 should either be in $R_1 \cup R_4$ or in $R_2 \cup R_3$. Hence c_1 cannot be tiled. A similar argument applies to c_2. ■

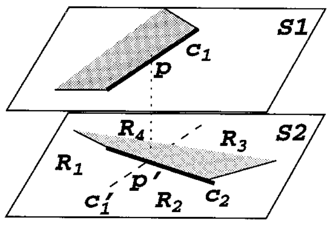

FIG. 23. Crossed contour segments cannot be tiled.

Proof of Theorem 1. Let V be an overlapping vertex. A vertical line passing through V has two point intersections with contours. In order to satisfy Criterion 2, the intersection with $\mathcal{SURF}$ must be one line segment containing V and V'. V' is also a vertex due to augmented contours. Hence V tiles to V'. ■

Proof of Theorem 2. 1. Suppose $\mathcal{NEC}(V')$ is CW as shown in Fig. 24a. Then $\mathcal{DISK}(V')$ does not intersect any solid region. There exists a point $V_1 \in (\mathcal{DISK}(V) \cap \mathcal{RS}(V))$ such that V_1 and V_1' are not on any contour. Let L be the vertical line passing through V_1. L does not intersect any solid region because the solid region incident with V is on $\mathcal{LS}(V)$ (see Definition 6). So, L does not intersect $\mathcal{SURF}$ according to Lemma 2. This implies $T' \not\subset \mathcal{RS}(V)$. Hence $T' \subset (\mathcal{S} - \mathcal{RS}(V))$. The case of CCW $\mathcal{NEC}(V')$ is proven in a similar way.

2. If $T_1' \subset (\mathcal{LS}(V) \cap \mathcal{LS}(V'))$ as shown in Fig. 24b, there eixsts a vertical line L_1 passing through the non-vertex points on $(T_1' \cap \mathcal{DISK}(V))$ such that L_1 does not intersect with any contour segment. Hence L_1 has two intersections with solid regions on both slices. Besides, L_1 has one intersection with $\mathcal{SURF}$ because it passes through T_1. This also contradicts Lemma 2. In the case of $T_2' \subset$

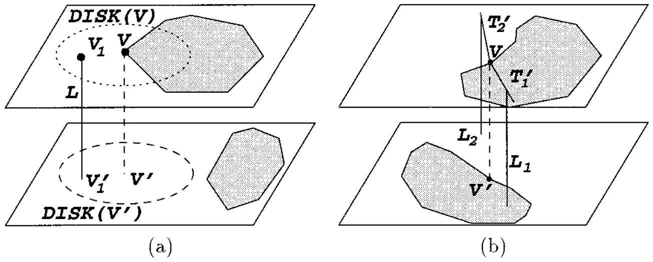

FIG. 24. Shadow regions represent solid regions. Both figures are used in the proof of Theorem 2.

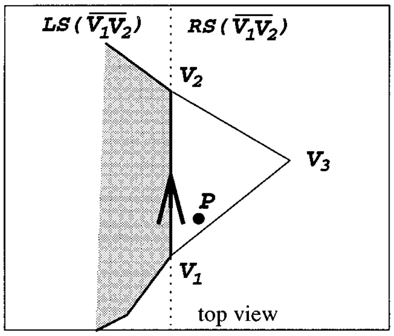

FIG. 25. $\triangle V_1V_2V_3$ is a tiling triangle. $\overline{V_1V_2}$ is a contour segment, and V_3 is on the adjacent slice.

$(\mathcal{RS}(V) \cap \mathcal{RS}(V'))$ as shown in Fig. 24b, L_2 has no intersection with solid regions on both slices but one intersection with $\mathcal{SURF}$. This contradicts Lemma 2. ■

Proof of Theorem 3. Referring to Fig. 25, suppose V_1 is not an overlapping vertex and V_1' has a CW $\mathcal{NEC}$. From Definition 6, the solid region is on $\mathcal{LS}(\overline{V_1V_2})$. Because $\mathcal{NEC}(V_1')$ is CW, there is no solid region within $\mathcal{DISK}(V_1')$. Suppose $V_3' \in \mathcal{RS}(\overline{V_1V_2})$ and let P be any point in $\mathcal{I}(\triangle V_1'V_2'V_3) \cap \mathcal{DISK}(V_1')$. The vertical line passing through P doesn't intersect any solid region of the slice containing V_1 and V_2 because $P' \in \mathcal{RS}(\overline{V_1V_2})$. Furthermore, it does not intersect any solid region of the other slice either because $P \in \mathcal{DISK}(V_1')$, but does have one intersection with $\triangle V_1V_2V_3$. This violates Lemma 2. So Case 1 is proved. Case 2 is proven in a similar way. ■

Proof of Theorem 4. This is proved by contradiction. Suppose $T' \cap \mathcal{I}(C) \neq \phi$ and $T' \cap \mathcal{O}(C) \neq \phi$. T' has three ordered sections t_1, t_2 and t_3 as shown in Fig. 26 such that $t_1 \subset \mathcal{I}(C)$, $t_2 \subset C$, $t_3 \subset \mathcal{O}(C)$, and $t_1 \cup t_2 \cup t_3$ is a line segment. Note, t_2 could be a line segment or a point, and both t_1 and t_3 are line segments. Suppose C is on slice $S1$, and the adjacent slice is denoted $S2$ and let $V1$ and V_2 be the two endpoints of t_2, and V_1 is closer to t_1. (In the case that t_2 is a point, V_1 and V_2 are the same point.) Let $L1$ be a vertical line passing through $P_1 \in (t_1 \cap \mathcal{DISK}(V_1))$. $L1$ does not intersect with any contour, and $L1$ intersects with $T \subset \mathcal{SURF}$. So $L1$ must intersect with

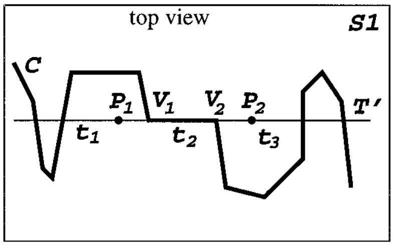

FIG. 26. Used in the proof of Theorem 4.

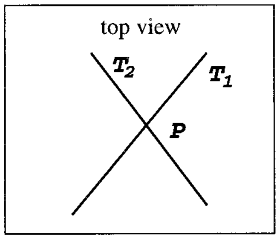

FIG. 27. The projections of two slice chords T_1 and T_2 cross each other at point P.

one solid region (Lemma 1). Hence P_1 and P_1' have opposite orientations. The same argument applies to $P_2 \in (t_3 \cap \mathcal{DISK}(V_2))$. Furthermore P_1 and P_2 have opposite orientations because $P_1 \in \mathcal{I}(C)$ and $P_2 \in \mathcal{O}(C)$. Hence P_1' and P_2' have opposite orientations. The radius of $\mathcal{DISK}$ is arbitrary small, so P_1' and P_2' have the same orientations as V_1' and V_2', respectively. Hence V_1' and V_1' have opposite orientations. This implies that there is a contour intersecting $\overline{V_1'V_2'}$. Thus there is at least one overlapping vertex in $\overline{V_1'V_2'}$. However, there cannot be any overlapping vertex on T_2, otherwise T intersects the vertical slice chord incident with this overlapping vertex (Theorem 1), and thus violates the polyhedron surface. ■

Proof of Theorem 5. 1. Let T_1' and T_2' cross each other at P (see Fig. 27). P and P' should not be on any contour, otherwise at least one of T_1 and T_2 will violate Theorem 4, T_1' and T_2' divide $\mathcal{DISK}(P)$ into four regions. Similar to the proof of Lemma 3, we can have one region R that any vertical line passing through R has two intersections with $\mathcal{SURF}$. However, this violates Criterion 2.

2. If T_1 crosses T_2 (in 3D), their point intersection is not a contour vertex. Then the tiling triangle containing T_1 and T_2 intersect at a nonvertex. Thus it violates the polyhedron surface property. ■

Proof of Lemma 4. The solid is on only one side of the surface represented by polygon $V_iV_j\underline{U_lU_k}$ of Fig. 9. So the solid is on the same side of $\overline{V_iV_j}$ and $\overline{U_lU_k}$. This implies $i - j = k - l = 1$ or -1. ■

Proof of Lemma 5. From Definition 5, there are two cases to form a positive vertex V.

Case 1: V is CCW, and $\mathcal{NEC}(V')$ is CW.

Case 2: V is CW, and $\mathcal{NEC}(V')$ is CCW.

In Case 1, $T' \subset (\mathcal{S} - \mathcal{RS}(V))$ according to Theorem 2. $T' \subset (\mathcal{I}(C) \cup C)$ because the $\mathcal{RS}$ of a CCW vertex is outside the contour.

In Case 2, $T' \subset (\mathcal{S} - \mathcal{LS}(P))$ according to Theorem 2. $T' \subset (\mathcal{I}(C) \cup C)$ because the $\mathcal{LS}$ of a CW vertex is outside the contour.

The second part of this lemma is proven similarly. ■

Proof of Theorem 6. Condition 1: It is obvious based on Theorem 1.

Condition 2: Suppose $C1$ and $C2$ are disjoint, and T is a legal slice chord incident with V_1 of $C1$ and V_2 of $C2$ as shown in Fig. 10a. $T' \subset (\mathcal{O}(C1) \cup C1)$, and $T' \subset (\mathcal{O}(C2) \cup C2)$. Hence V_1 and V_2 are negative vertices (Lemma 5). Furthermore, there is no contour intersecting the open line segments (V_1, V_2') or (V_1', V_2) (Theorem 4). If a contour contains V_1', then V_1 is an overlapping vertex and there exists a slice chord $\overline{V_1V_1'}$ (Theorem 1). Also, there must be three surfaces sharing the contour segment incident with V_1. One contains $\overline{V_1V_1'}$, one contains $\overline{V_1V_2}$ and the other is on a solid region, but this violates the polyhedron surface. Hence no contour contains V_1'. The same argument applies V_2'. Hence no contour except $C1$ and $C2$ intersects $\overline{V_1V_2'}$ or $\overline{V_1'V_2}$. Thus $\mathcal{NEC}(V_1') = \mathcal{NEC}(C2)$ and $\mathcal{NEC}(V_2') = \mathcal{NEC}(C1)$. A vertical line segment L intersecting the open line segment (V_1, V_2) has one intersection with the solid region based on Lemma 2. Hence the scalar data at both ends of L must have different signs. $C1$ has a different orientation from the point at $(S1 \cap L)$, and $C2$ has different orientation from the point at $(S2 \cap L)$. Hence $C1$ and $C2$ have different orientations.

Condition 3: Suppose $C1'$ is inside $C2$, and let T be a legal slice chord incident with V_1 of $C1$ and V_2 of $C2$ as shown in Fig. 10b. V_1 is a negative vertex, and V_2 is a positive vertex based on Lemma 5. With the same reasoning as in the proof of Theorem 6.2, no contour except $C1$ and $C2$ intersects $\overline{V_1V_2'}$ or $\overline{V_1'V_2}$. Hence $\mathcal{NEC}(V_2') = \mathcal{NEC}(C1)$ and $C2 = \mathcal{NEC}(C1')$. Based on Lemma 2, a vertical line segment L intersecting the open line segment (V_1, V_2) has one intersection with the solid region. Hence the points at both ends of L have different orientations. $C1$ has a different orientation from the point at $(S1 \cap L)$, and $C2$ has the same orientation as the point at $(S2 \cap L)$. Hence $C1$ and $C2$ have the same orientation. ■

Proof of Theorem 7. If $C1'$ intersects $C2$, it is obvious that there is a path linking $C1$ and $C2$ (Theorem 1).

Suppose Condition 2 of Theorem 6 holds, and let V_1 be a negative vertex on $C1$. We draw a line segment T, which initially is in $\mathcal{O}(C1)$ from $V1$ to $C2$. Some contours might go across T, because $C1$ and $C2$ are not insulated by their $\mathcal{NEC}$s. So any contour that crosses T' must either cross T' an even number of times or intersect $C2'$ as shown in Fig. 28. Hence $C1$ and $C2$ have different orientations. A vertical line, which passes through the common outside region of $C1$ and $C2$ and is not in the inside of the sibling contours of $C1$ and $C2$, intersects the solid region at one point, and it has an intersection with $\mathcal{SURF}$. Thus $\mathcal{SURF}$ always has a projection on T except at between the odd and even crossings (shown as a dashed line in Fig. 28) We start from $V1$ and walk along the projection of T onto $\mathcal{SURF}$. If we encounter a contour, then we walk along that contour. It either leads to $C2$, or comes back to L. If it leads to $C2$, then we are done. If it comes back to L, we again walk along the projection of L on $\mathcal{SURF}$. Because $C2$ or a contour having an intersection with C_2' is within the $(2n)$th and $(2n + 1)$th crossing of L, we can always reach $C2$. So there exists a path between $C1$ to $C2$.

The case of condition 3 is proven in the similar way. ■

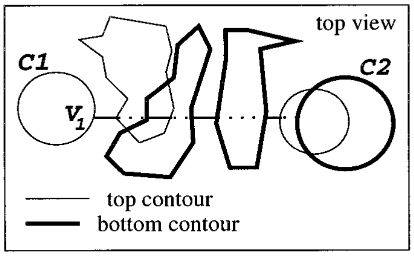

FIG. 28. There exists a path on $\mathcal{SURF}$ linking $C1$ and $C2$,

ACKNOWLEDGMENTS

We are grateful to the University of North Carolina for anonymous ftp access to the skull and brain data sets and Johns Hopkins University for the entire human cadaver data sets. Our thanks also to the reviewers for several constructive suggestions, to Dan Schikore for his development of rendering tools, to Dr. Bernhard Geiger for his insightful suggestions for improving this paper, and to Dr. Gershon Elber for his IRIT 5.0 tool to illustrate many tiling results. This research was supported in part by NSF Grants CCR 92-22467, AFOSR Grants F49620-93-10138 and F49620-94-10080, and ONR Grant N00014-91-1-0370.

REFERENCES

1. G. Barequet and M. Sharir, Piecewise-linear interpolation between polygonal slices, in *10th Conputational Geometry, 1994,* pp. 93–102.
2. J. D. Boissonnat, Shape reconstruction from planar cross sections, *Comput. Vision Graphics Image Process.* **44,** 1988, 1–29.
3. Y. Bresler, J. A. Fessler, and A. Macovski, A Bayesian approach to reconstruction from incomplete projections of a multiple object 3D domain, *IEEE Trans. Pattern Anal. Mach. Intell.* **11**(8), Aug. 1989, 840–858.
4. H. N. Christiansen and T. W. Sederberg, Conversion of complex contour line definitions into polygonal element mosaics, *Comput. Graphics* **12,** Aug. 1978, 187–192.
5. K. L. Clarkson, A randomized algorithm for closest-point queries, *SIAM J. Comput.* **17**(4), 1988, 830–847.
6. 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. Biomed. Eng.* **BME-27**(9), Sept. 1980, 493–499.
7. A. B. Ekoule, F. C. Peyrin, and C. L. Odet, A triangulation algorithm from arbitrary shaped multiple planar contours, *ACM Trans. Graphics* **10**(2), Apr. 1991, 182–199.
8. T. T. Elvins, A survey of algorithms for volume visualization, *Comput. Graphics* **26**(3), Aug. 1992, 194–201.
9. D. Eu and G. T. Toussaint, On approximating polygonal curves in

two and three dimensions, *CVGIP: Graphical Models Image Process.* **56**(3), May 1994, 231–246.

10. H. Fuchs, Z. M. Kedem, and S. P. Uselton, Optimal surface reconstruction from planar contours, *Commun. ACM* **20**(10), Oct. 1977, 693–702.
11. S. Ganapathy and T. G. Dennehy, A new general triangulation method for planar contours, *Comput. Graphics* **16,** 1982, 69–75.
12. B. Geiger, *Three-Dimensional Modeling of Human Organs and Its Application to Diagnosis and Surgical Planning,* Technical Report 2105, INRIA, France, 1993.
13. C. Gitlin, J. O'Rourke and V. Subramanian, On reconstructing polyhedra from parallel slices, *Int. J. Comput. Geom. Appl.* **6**(1), 1996, 103–122.
14. N. Kehtarnavaz, L. R. Simar, and R. J. P. De Figueiredo, A syntactic/semantic technique for surface reconstruction from cross-sectional contours, *Comput. Vision Graphics Image Process.* **42,** 1988, 399–409.
15. E. Keppel, Approximating complex surfaces by triangulation of contour lines, *IBM J. Res. Dev.* **19,** Jan. 1975, 2–11.
16. D. T. Lee, Medial axis transformation of a planar shape, *IEEE Trans. Pattern Anal. Mach. Intell.* **PAMI-4**(4), July 1982, 363–369.
17. W. C. Lin and S. Y. Chen, A new surface interpolation technique for reconstructing 3D objects from serial cross-sections, *Comput. Vision Graphics Image Process.* **48,** 1989, 124–143.
18. W. E. Lorensen and H. E. Cline, Marching cubes: A high resolution 3D surface construction algorithm, *Comput. Graphics* **21**(4), 1987, 163–169.
19. D. Meyers, *Reconstruction of Surfaces From Planar Contours,* Ph.D. thesis, Univ. of Washington, 1994.
20. D. Meyers, S. Skinner, and K. Sloan, Surfaces from contours, *ACM Trans. Graphics* **11**(3), Jul. 1992, 228–258.
21. J. O'Rourke, On the scaling heuristic for reconstruction from slices, *CVGIP: Graphical Models Image Process.* **56**(3), May 1994, 420–423.
22. F. P. Preparata and M. I. Shamos, *Computational Geometry: An Introduction,* Springer-Verlag, Berlin/New York, 1985.
23. M. Shantz, Surface definition for branching contour-defined objects, *Comput. Graphics* **15**(2), July 1981, 242–270.
24. Y. Shinagawa and T. L. Kunii, The homotopy model: A generalized model for smooth surface generation from cross sectional data, *Visual Comput.* **7,** 1991, 72–86.
25. K. R. Sloan and J. Painter, Pessimal guesses may be optimal: A counterintuitive search result, *IEEE Trans. Pattern Anal. Mach. Intell.* **10**(6), Nov. 1988, 949–955.
26. B. I. Soroka, Generalized cones from serial sections, *Comput. Vision Graphics Image Process.* **15,** 154–166. 1981.
27. V. Srinivasan and L. R. Nackman, Voronoi diagram for multiply-connected polygonal domains, I: Algorithm, *IBM J. Res. Dev.* **31**(3), May 1987, 361–372.
28. Y. F. Wang and J. K. Aggarwal, Surface reconstruction and representation of 3-D scenes, *Pattern Recongition* **19**(3), 1986, 197–207.