International Journal of Computational Geometry & Applications

Triangulation of Branching Contours using Area Minimization

Thomas W. Sederberg, Krzysztof S. Klimaszewski, and Mu Hong
Computer Science Department, Brigham Young University
Provo, Utah 84602 United States of America

and

Kazufumi Kaneda
Hiroshima University, 4-1 Kagamiyama 1 chome
Higashi-hiroshima 724, Japan

Received 15 August 1996
Revised 7 March 1997
Communicated by Editor's name

ABSTRACT

This paper presents a new method for reconstructing piecewise linear surfaces from planar polygonal contours that branch. For non-branching contours, experience has shown that the piecewise linear surface of minimum surface area which connects a pair of contours often provides a good solution. The current algorithm extends this idea by searching for the surface of minimal area which connects two contours comprised of more than one polygon. Several examples that justify this heuristic are provided.

Keywords: Surface reconstruction, triangulation, contours, area minimization

1. Introduction

The problem of interpolating a surface from a set of contour curves on parallel planes is addressed by a large and growing literature[13]. Such contour data may be generated from tomography scans, ultrasonic and nuclear magnetic resonance devices, or topographic elevation maps. Its reconstruction into surfaces has many applications, perhaps foremost being medical imaging.

In this paper, the contours are taken to be polygons and the extrapolated surface is piecewise triangular. We will refer to this surface extrapolation procedure as contour triangulation. Figure 1 gives an example of contour triangulation in which a single contour on one plane is triangulated with a single contour on an adjacent parallel plane.

In this paper, we adopt a few rules that any acceptable triangulation must follow. Given two polygons, $\mathbf{P}$ and $\mathbf{Q}$, that lie on parallel planes, we define a *legal* triangulation of $\mathbf{P}$ and $\mathbf{Q}$ to be one in which

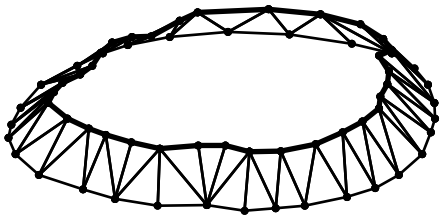

Figure 1: Simple triangulation

1. all triangle vertices are constrained to lie on vertices of $\mathbf{P}$ and $\mathbf{Q}$;
2. if $\mathbf{Q}_j$–$\mathbf{P}_i$ forms the edge of one triangle in the triangulation, then either $\triangle \mathbf{Q}_j \mathbf{P}_i \mathbf{Q}_{j+1}$ or $\triangle \mathbf{Q}_j \mathbf{P}_i \mathbf{P}_{i+1}$ also belong to the triangulation (see Figure 2);
3. each triangle has at least one vertex on $\mathbf{P}$ and at least one on $\mathbf{Q}$.

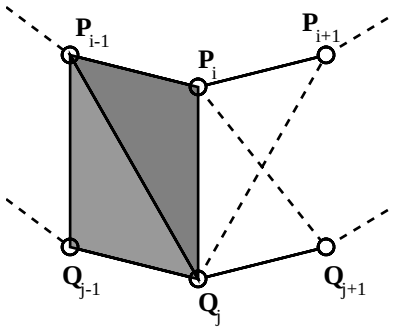

Figure 2: Adjacency constraint

The seminal works by Keppel[9] and Fuchs et al.[5] observe that legal triangulations have a simple graph structure, as illustrated in Figure 3. The columns of the graph represent the vertices of $\mathbf{P}$ and the rows represent the vertices of $\mathbf{Q}$. Vertex (i, j) of the graph (the intersection of column i and row j) represents the line segment $\mathbf{Q}_i$–$\mathbf{P}_j$. In graph jargon, the line segment connecting two horizontally or vertically adjacent vertices is called an *arc*; each arc represents a triangle, two of whose edges are represented by the arc's endpoints. A *path* is any continuous set of arcs that begins at the upper-left corner and ends in the lower-right corner of the graph. Then, any legal triangulation is represented by a path whose arcs proceed only to the right or down. Hence, a legal triangulation consists of $n_P + n_Q$ triangles where n_P and n_Q are the number of vertices in $\mathbf{P}$ and $\mathbf{Q}$ respectively.

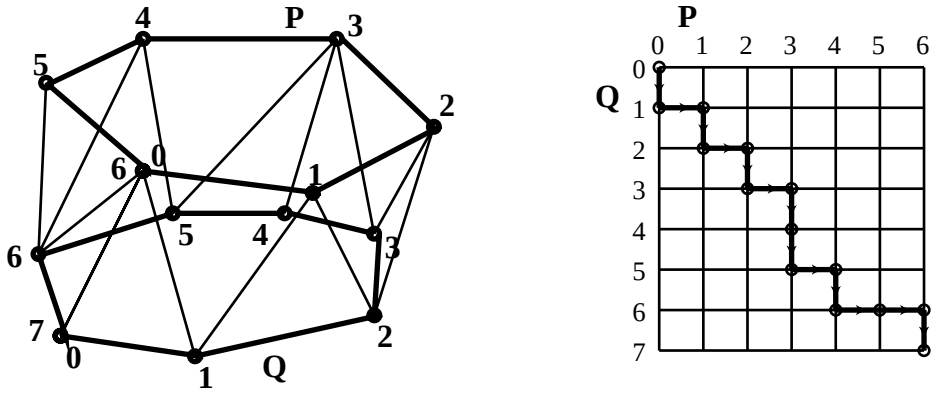

Figure 3: Graph representation

The contour triangulation problem does not generally have a unique "correct" solution. Figure 4 shows two different legal triangulations of the same pair of contours. Both are reasonable triangulations, and circumstances can be imagined in which either would be deemed best. If a complete description was available of the underlying surface from which the contours were extracted a "best" triangulation could be identified in an approximation-theoretic sense, though the choice of "best" would depend on the particular norm used. In practice, the choice of "best" is a rather subjective matter.

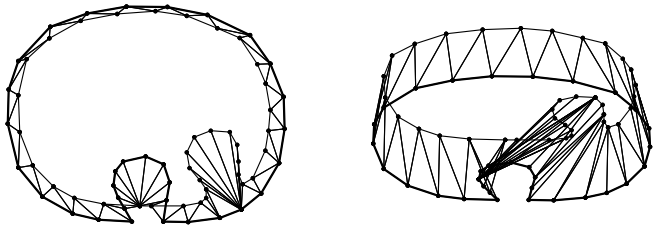

Figure 4: Ambiguous triangulation

Since the only data available in the triangulation problem are the contour polygons, one or more heuristics must be invoked that will identify an "acceptable" legal triangulation in most cases. A heuristic that in general yields good results when triangulating a single pair of polygons is to search for the triangulation of least surface area[11]. This approach has the appeal that it is a simple, single rule, plus it has a physical analog in minimal surfaces. The area of a triangulation is the sum of the areas of its constituent triangles. Denote by $A_{min}(\mathbf{P}, \mathbf{Q})$ the smallest area of all legal triangulations of $\mathbf{P}$ and $\mathbf{Q}$. A minimal-area triangulation (there may be more than one) is any legal triangulation whose area is $A_{min}(\mathbf{P}, \mathbf{Q})$. The

number of legal triangulations is huge — $\frac{(n_P+n_Q)!}{n_P!n_Q!}$[5]. Hence, an exhaustive search of all legal triangulations grows prohibitive ($\approx 10^{59}$ possibilities for $n_P = n_Q = 100$). Fortunately, dynamic programming reduces the search for the minimal-area triangulation to an $O(n_P n_Q)$ algorithm, assuming that $\mathbf{P}_0$–$\mathbf{Q}_0$ is an edge on the optimal triangulation[5,9], or to roughly $O(n_P n_Q \log(\min(n_P, n_Q)))$ if $\mathbf{P}_0$–$\mathbf{Q}_0$ is not an edge on the optimal triangulation[5]. We assume that the reader is somewhat familiar with this algorithm.

The minimal-area heuristic is not flawless. For example, consider the case where the two contours are identical circles (approximated by regular polygons) on parallel planes, with centers offset by the diameter of the circle. In this case, the surface of minimal area turns out to be two cones[11]. Even if the offset is not that great, the minimal-area triangulation is not a smooth cylinder, but a cylinder containing two grooves. This problem can be mitigated by first scaling and translating the contours so that their bounding boxes align[4].

1.1. Other Hueristics

In addition to minimal-area, several alternative criteria have been suggested in the literature for determining a good triangulation between two polygons. Keppel[9] proposed that if the two polygons are convex, the triangulation of maximum volume works well. If the polygons are not convex, the algorithm begins by triangulating their convex hulls (using the maximum-volume criterion) and then triangulates the concave regions in a subsequent pass.

Area-minimization and volume-maximization are examples of global optimization methods, which are typically solved using dynamic programming. Other algorithms have been devised based on a local advancing rule. For example, Christiansen and Sederberg[4] describe an $O(n_P + n_Q)$ algorithm that chooses each new triangle based on minimum edge length. Ganapathy and Dennehy[6] compute an arclength parametrization of $\mathbf{P}$ and $\mathbf{Q}$, and then assign triangles based on proportional parameter values.

Some algorithms include a rule that the vertical projection of the two contours must intersect at polygon vertices (adding new vertices if needed) and those pairs of vertically-aligned vertices form edges of triangles[1,2]. These algorithms are noteworthy because they handle any number of contour polygons.

It is evident from Figure 4 that no single set of rules can always yield the "best" solution. All of the algorithms cited would tend to triangulate the contours in Figure 4 more like case (a) than like case (b).

1.2. Overview

This paper investigates the application of the minimal-area heuristic to triangulations involving branching contours. Section 2 motivates the problem, and section 3 discusses how to adapt the minimum-area idea to the case of branching contours. A known weakness of the minimal-surface heuristic is that in the presence of severe convolutions, it can lead to a self-intersecting surface[1]. Section 4 discusses this

weakness and proposes a fix. Other approaches to triangulating branching contours such as those in Figure 7 — approaches that introduce additional vertices not in the original data set — are surveyed in section 5.

Readers who desire a review of literature dealing with the general problem of surface reconstruction from contours can study [1,2,12,13].

2. Branching

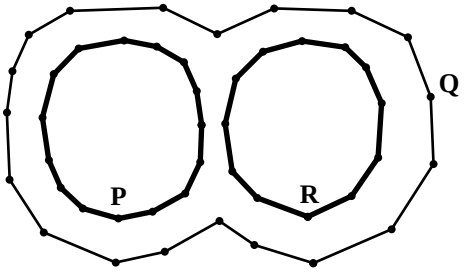

Figure 5: One-to-two branching

A simple case of contour branching is shown in Figure 5, where the contour comprises two polygons, **P** and **R**. We will refer to this situation with one polygon on one plane branching into two polygons on an adjacent plane as a case of single branching. The earliest proposed solution to the single-branching problem[4] is to connect the two polygons together with a line segment "bridge" (see Figure 6). The

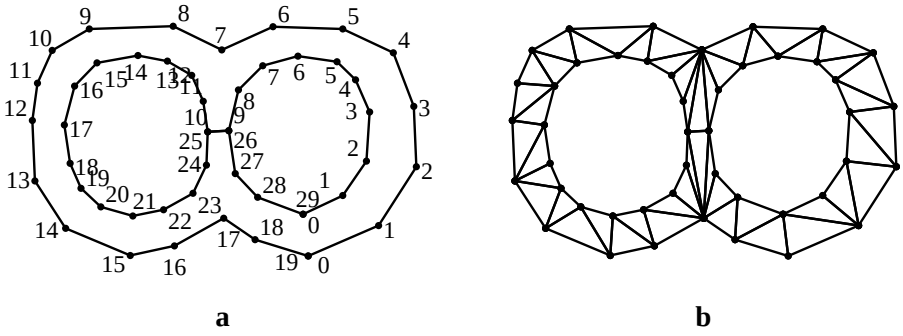

Figure 6: One-to-two branching

two polygons are then considered to be a single polygon which has been pinched together at the bridge, and triangulation proceeds as in the non-branching case.

This simple idea only works in simple cases; Figure 7 shows a more convoluted case in which a line-segment bridge yields an unpleasing result (the triangulation is

not shown because it is too noisy).

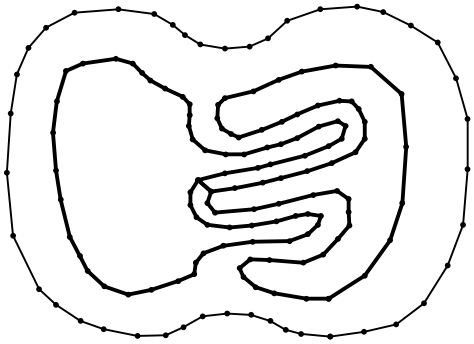

Figure 7: Case where line-segment bridge fails

If we continue to restrict our triangles to have vertices lying on existing contour vertices, but allow for triangles having all three vertices on the same contour level (we will call them *bridge triangles*), a more acceptable solution to the example in Figure 7 is obtained by forming a polygonal bridge between the two branching polygons, as shown in Figure 8. Triangles with vertices on both contour planes will be called *face triangles*.

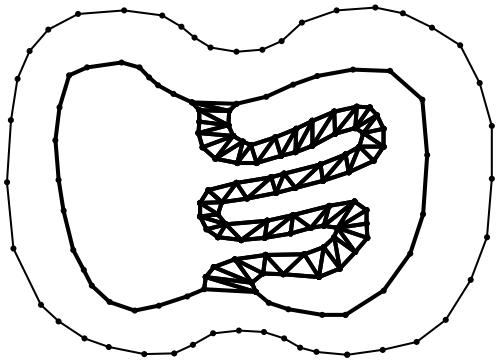

Figure 8: Polygonal bridge

Polygonal bridges are promoted in [10], where the observation is made that an automatic algorithm for generating polygonal bridges must involve some criterion for determining the bridge limits. The contribution of this paper is to demonstrate that area minimization (the area of the bridge triangles plus the area of the face triangles) is an excellent criterion. Figure 9 shows minimal-area triangulations of several branching contours. (Other authors have called bridges "canyons" — an equally valid metaphor.)

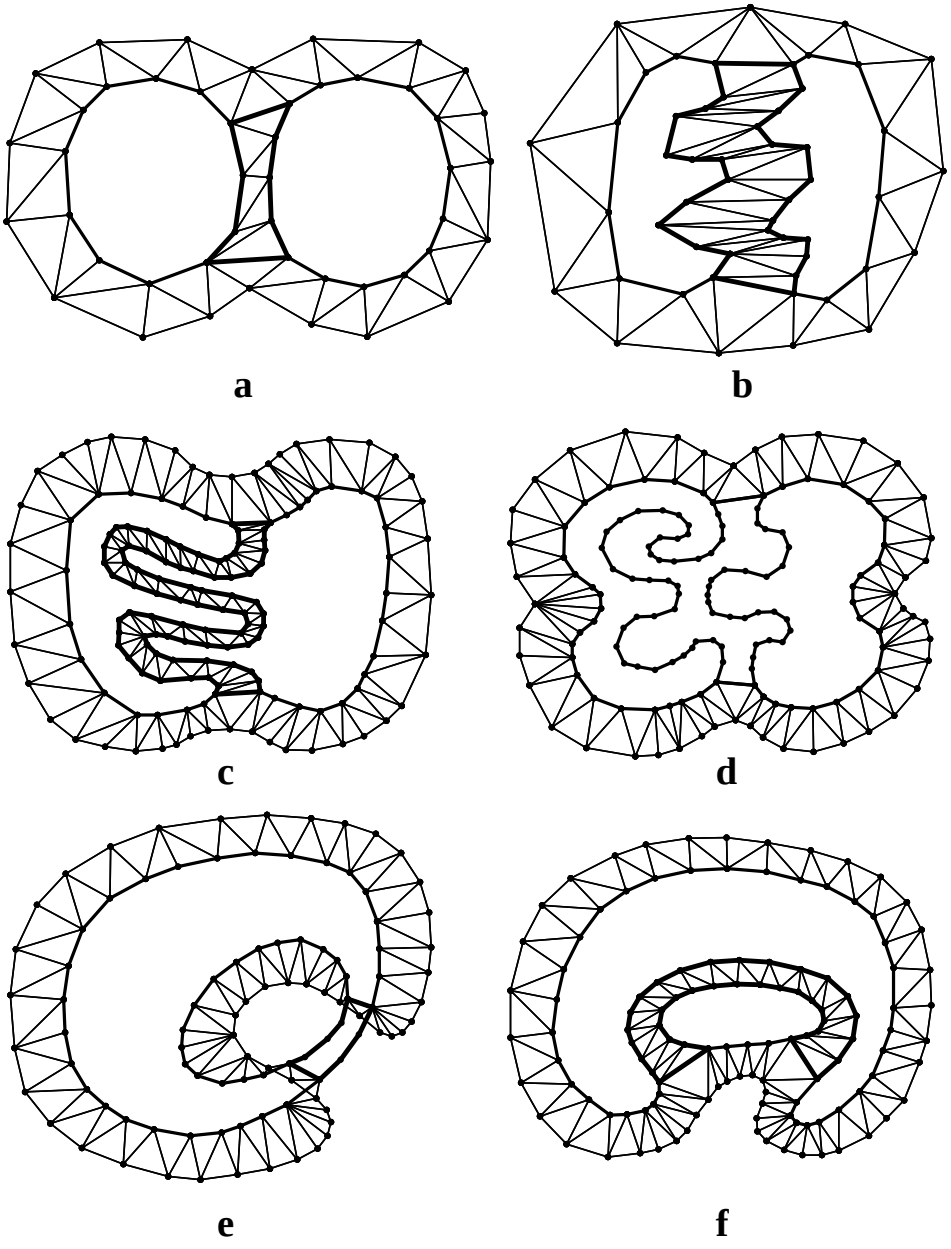

Figure 9: Examples of minimal-area branching contours

3. Algorithm Development

In this section, we discuss algorithms for computing the minimal-area triangulation for the case of one contour branching into two contours. Section 3.2 presents an exhaustive search algorithm that is guaranteed to find the optimal solution. Section 3.3 presents a much faster algorithm that gives no guarantee of finding the optimal solution, yet has yielded excellent results in practice.

3.1. Bridge Delimiters

Denote by $\mathbf{P}$ and $\mathbf{R}$ two polygons on the same plane, by $\mathbf{Q}$ a polygon on a parallel plane. We assume that $\mathbf{P}$ and $\mathbf{R}$ do not intersect (or self-intersect). We also assume that the vertices of $\mathbf{P}$, $\mathbf{R}$ and $\mathbf{Q}$ are labeled counter-clockwise. We allow for the possibility that $\mathbf{R}$ lies completely inside of $\mathbf{P}$ (see Figure 9(e)), in which case the vertices of $\mathbf{R}$ are numbered clockwise.

Given points $\mathbf{P}_i$ and $\mathbf{P}_j$ on $\mathbf{P}$, $\mathbf{P}_{ij}$ denotes the polyline consisting of vertices $\mathbf{P}_i, \mathbf{P}_{i+1}, \ldots, \mathbf{P}_{j-1}, \mathbf{P}_j$ with polygon indices taken modulo n_P. Note that $\mathbf{P}$ is the union of $\mathbf{P}_{ij}$ and $\mathbf{P}_{ji}$

A bridge $B(ijkl)$ is a planar polygon comprised of polylines $\mathbf{P}_{ij}$ and $\mathbf{R}_{kl}$ joined by two line segments $\mathbf{P}_j\mathbf{R}_k$ and $\mathbf{R}_l\mathbf{P}_i$ as shown in Figure 10. If $B(ijkl)$ satisfies the following three requirements, we will say that $B(ijkl)$ is a legal bridge:

1. $B(ijkl)$ is not self-intersecting;
2. $\mathbf{P}_j\mathbf{R}_k$ and $\mathbf{R}_l\mathbf{P}_i$ do not intersect $\mathbf{P}$ or $\mathbf{R}$ except at their endpoints;
3. No point on polylines $\mathbf{P}_{ji}$ or $\mathbf{R}_{lk}$ lie on the interior of $B(ijkl)$.

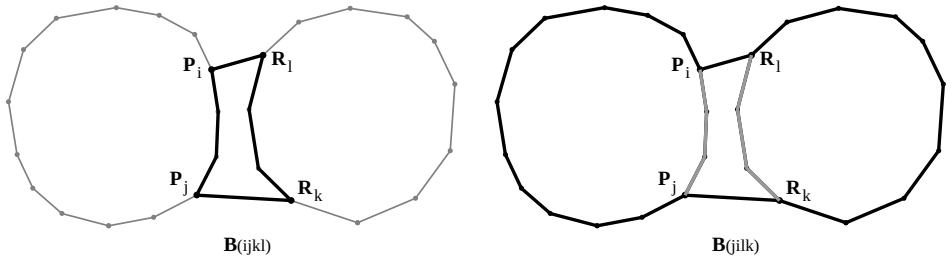

Figure 10: Bridge delimiters

Requirement 3 is needed because if $B(ijkl)$ is a polygon that satisfies requirements 1 and 2, then $B(jilk)$ is also a polygon which satisfies those requirements. However, as illustrated in Figure 10, only one of those two will satisfy requirement 3.

We can now define a legal triangulation of single-branching contours to consist of a tessellation of a legal bridge $B(ijkl)$ and a legal triangulation of the polygons $B(jilk)$ and $\mathbf{Q}$. The tessellation of $B(ijkl)$ is performed using any polygon tessellator. Our objective is to determine a set of bridge delimiters for which the area of $B(ijkl)$ plus the area of the minimal-area triangulation of $B(jilk)$ and $\mathbf{Q}$ is smallest.

Of course, we don't need to tessellate each candidate bridge in order to compute its area.

3.2. Exhaustive Search Algorithm

We noted that for a polygon **Q** (with n_Q vertices) being triangulated with a single polygon **P** (with n_P vertices), graph theory leads to an elegant $O(n_Q n_P)$ algorithm for finding the minimal-area triangulation. Unfortunately, if **Q** branches into two polygons, **P** and **R**, an elegant dynamic programming solution has thus far eluded the authors. An exhaustive search for the minimal-area triangulation would consider all possible pairs of bridge delimiters, of which there are $O(n_P^2 n_R^2)$. For each pair of bridge delimiters, the algorithm must compute the area of the bridge (which can be done in $O(n_P + n_R)$ time) and the minimal-area triangulation for $B(jilk)$ and **Q** ($O(n_Q*(n_P+n_R))$ time, using the dynamic programming algorithm). So, if $n \approx n_P \approx n_Q \approx n_R$, this exhaustive search algorithm is $O(n^6)$.

3.3. Approximate Algorithm

Shortcuts can be introduced to reduce the time complexity of the algorithm. For example, the DP algorithm for computing a minimal-area triangulation of the face triangles can be reduced from $O(n_Q n_P)$ to $O(n_P)$ by limiting our search of the cost matrix to a constant width band along the diagonal[14]. This is almost always justified, since the least cost path normally does not deviate very far from the major diagonal of the graph. This shortcut does relinquish the iron-clad guarantee that the algorithm will return the absolute minimal-area triangulation.

A major time savings can be obtained by observing that if an optimal triangulation T_{ijkl} has been computed that involves bridge $B(ijkl)$, the optimal triangulation $T_{i+1,j,k,l}$ involving bridge $B(i+1,j,k,l)$ will generally differ from T_{ijkl} in only the few triangles with vertices on $\mathbf{P}_i$ and $\mathbf{P}_{i+1}$. Consider the portion of a triangulation shown in Figure 11a which shows one bridge delimiter $\mathbf{P}_1\mathbf{R}_8$. If the bridge delimiter is moved to $\mathbf{P}_2\mathbf{R}_7$ as shown in Figure 11b, only three triangles will be changed in the least cost triangulation: triangles $\triangle\mathbf{Q}_2\mathbf{P}_2\mathbf{P}_1$, $\triangle\mathbf{Q}_2\mathbf{P}_1\mathbf{R}_8$, and $\triangle\mathbf{Q}_2\mathbf{R}_8\mathbf{R}_7$ are replaced with triangle $\triangle\mathbf{Q}_2\mathbf{P}_2\mathbf{R}_2$ and quadrilateral $\mathbf{P}_2\mathbf{P}_1\mathbf{R}_8\mathbf{R}_7$ (which lies on the bridge). This observation, which experience suggests is pretty typical, leads to the following heuristic algorithm for finding the optimal bridge. The algorithm makes use of a *visibility table* VT which is a two-dimensional matrix whose elements VT_{ij} are set to one if line segment $\mathbf{P}_i - \mathbf{R}_j$ does not intersect any contour, and set to zero otherwise.

1. Create the visibility table VT for **P** and **R**.
2. Define a vector ZR for which $ZR_i = 0$ if every element of row i in VT is zero. Otherwise, $ZR_i = 1$.
3. Define a vector ZC for which $ZC_i = 0$ if every element of column i in VT is zero. Otherwise, $ZC_i = 1$.

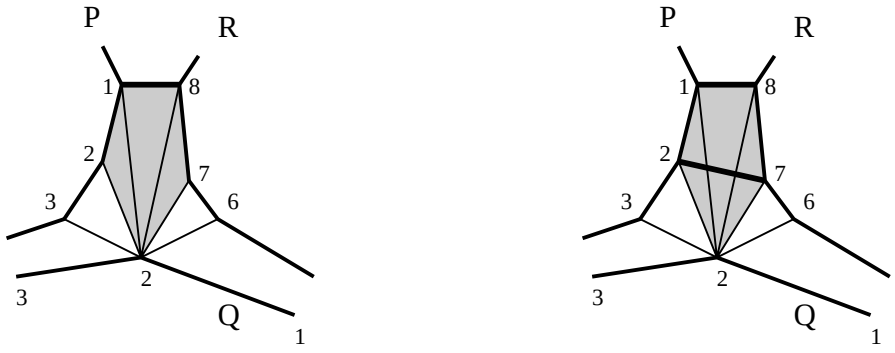

Figure 11: Adjacent bridge delimiters

4. Identify the largest rectangle of elements in VT for which each row and column are not all zero. Find an element $VT_{i,l}$ near the center of that rectangle whose value is 1. This represents the initial bridge delimiter.

5. Triangulate using line segment $\mathbf{P}_i\mathbf{R}_l$ as the bridge. Define this to be the "current triangulation". Set $C = C_{min}$ = area of current triangulation; Set $i_0 = j_0 = i$; $k_0 = l_0 = l$.

6. Set $j = i$, $k = l$.
Repeat steps 7 and 8 until done:

7. Find $\mathbf{P}_J\mathbf{R}_K$, the next "adjacent" bridge delimiter to $\mathbf{P}_j\mathbf{R}_k$. This is done by searching VT for an element $\mathrm{VT}_{JK} = 1$ for which the value $(J - j \bmod n_P) + (K - k \bmod n_R)$ is smallest.

8. Determine which triangles in the current triangulation will become invalid if the bridge delimiter is moved from $\mathbf{P}_j\mathbf{R}_k$ to $\mathbf{P}_J\mathbf{R}_K$. Determine the vertices on $\mathbf{Q}_a \ldots \mathbf{Q}_b$ which are also vertices of those triangles. Determine if C_{min} will decrease if those triangles are replaced by a larger bridge with a delimiter $\mathbf{P}_J\mathbf{R}_K$ and the triangles needed to fill the region $\mathbf{Q}_a \ldots \mathbf{Q}_b\mathbf{P}_J\mathbf{R}_K$. If so, store the fact that we have just found a better bridge and update the value of C_{min}.

9. Repeat steps 7 and 8 once again, except expand the bridge in the other direction.

This procedure is illustrated in Figure 12. Table 1 presents the change in total area as the bridge delimiters are moved. The initial bridge delimiter is chosen to be $\mathbf{P}_9\mathbf{R}_4$. The next adjacent bridge delimiter is $\mathbf{P}_{10}\mathbf{R}_3$. Shifting the bridge delimiter requires triangles $\mathbf{P}_{10}\mathbf{P}_9\mathbf{Q}_2$, $\mathbf{P}_9\mathbf{R}_4\mathbf{Q}_2$, and $\triangle\mathbf{R}_4\mathbf{R}_3\mathbf{Q}_2$ to be replaced with quadrilateral $\mathbf{P}_{10}\mathbf{P}_9\mathbf{R}_4\mathbf{R}_3$ and triangle $\triangle\mathbf{P}_{10}\mathbf{R}_3\mathbf{Q}_2$. The cost of the new triangulation is found quickly by just computing the areas of those four triangles and one quadrilateral. The process continues until the bridge delimiter is $\mathbf{P}_{11}\mathbf{R}_2$. The

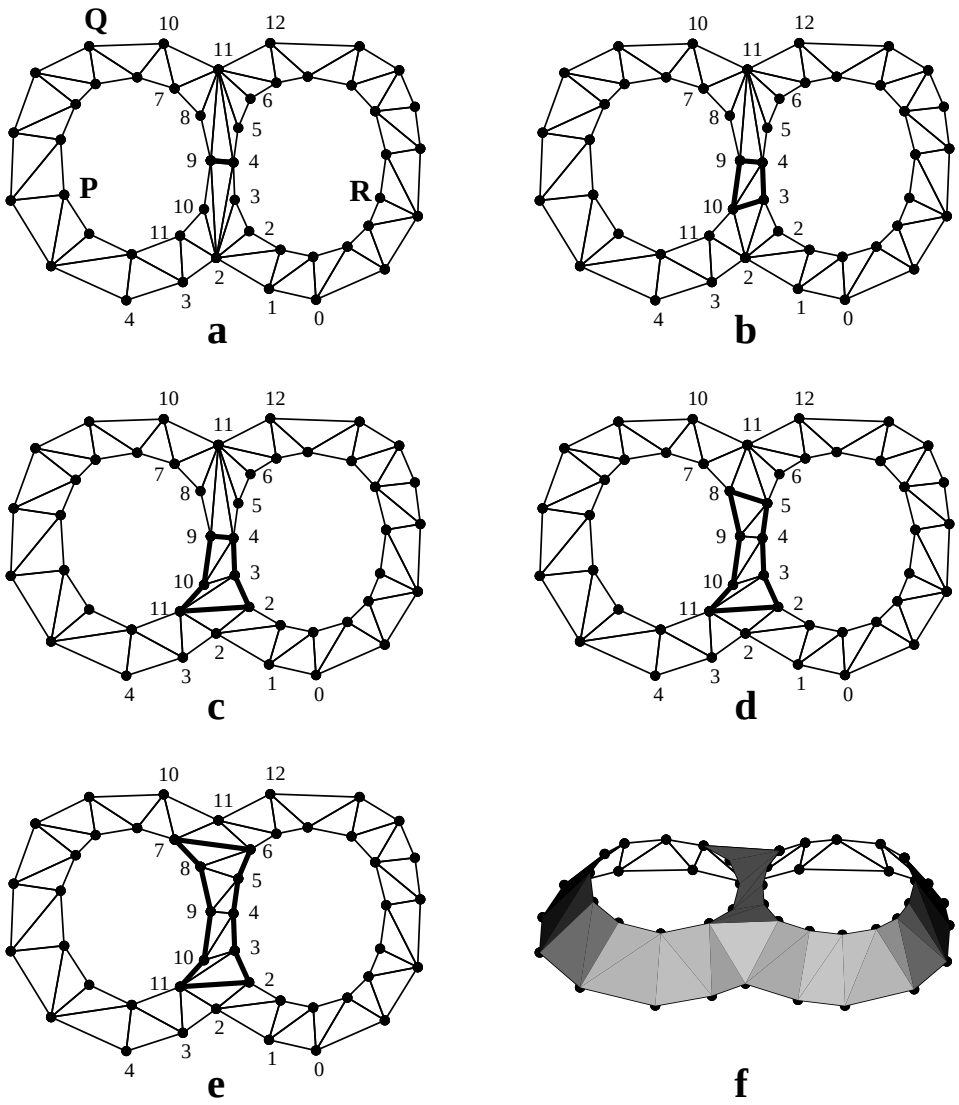

Figure 12: Fast heuristic for finding optimal bridge delimiters

next adjacent legal bridge delimiter is $\mathbf{P}_{12}\mathbf{R}_1$, for which the total area is larger than it was using $\mathbf{P}_{11}\mathbf{R}_2$. After that, there are only a few more legal bridge delimiters to check, and the total area cannot be decreased. The algorithm then attempts to expand the bridge in the other direction. It first determines that an improvement arises by moving delimiter $\mathbf{P}_9\mathbf{R}_4$ to $\mathbf{P}_8\mathbf{R}_5$, and further improvement comes by moving to $\mathbf{P}_7\mathbf{R}_6$.

Table 1: Numerical values for fast heuristic example

Bridge delimiter index				See	Face	Bridge	Total	Area
Contour **P**		Contour **Q**		Figure	area	area	area	change
$\mathbf{P}_i$	$\mathbf{P}_j$	$\mathbf{R}_k$	$\mathbf{R}_l$					
9	9	4	4	a	237895	0	237895	-
9	10	3	4	b	229786	3316	233102	Decrease
9	11	2	4	c	225042	7439	232481	Decrease
9	12	1	4	none	222868	13234	236102	Increase
8	11	2	5	d	217960	10873	228832	Decrease
7	11	2	6	e	212880	1503	228183	Decrease
6	11	2	7	none	210613	19406	230018	Increase

This algorithm, while not guaranteeing the minimal area, runs in $O(n_P n_R)$ time. In all of the examples in Figure 13, this fast heuristic produced the same triangulations as did the exhaustive search algorithm.

It is of some interest to examine the geometric relationships that underly the minimal-area heuristic. Figure 13 plots the algebraic curve **C** defined

$$\mathbf{C} = \{\mathbf{Q}_b | \mathbf{A}(\mathbf{P}_1\mathbf{P}_2\mathbf{Q}_b) + \mathbf{A}(\mathbf{R}_8\mathbf{P}_1\mathbf{Q}_b) + \mathbf{A}(\mathbf{R}_7\mathbf{R}_8\mathbf{Q}_b) = \mathbf{A}(\mathbf{R}_7\mathbf{P}_2\mathbf{Q}_b) + \mathbf{A}(\mathbf{R}_7\mathbf{R}_8\mathbf{P}_1\mathbf{P}_2)\} \tag{1}$$

where $\mathbf{A}(\mathbf{P}_1\mathbf{P}_2\mathbf{Q}_b)$ denotes the area of triangle $\mathbf{P}_1\mathbf{P}_2\mathbf{Q}_b$ etc. and where $\mathbf{Q}_b$ is constrained to lie on the plane of **Q**. If $\mathbf{Q}_2$ lies to the right of **C**, a net decrease in surface area will occur when bridge delimiter $\mathbf{P}_1\mathbf{R}_8$ is shifted to $\mathbf{P}_2\mathbf{R}_7$. Clearly, **C** is a function of $\mathbf{P}_1$, $\mathbf{P}_2$, $\mathbf{R}_7$, $\mathbf{R}_8$ and of the elevation difference between the two contour planes.

4. Non-branching contours with severe convolutions

Figure 14.a shows an interesting example noted in [1] of how the minimal-area triangulation can lead to self intersections. Of course, this raises the point that the triangulation in Figure 14 is minimal-area only if we restrict ourselves to triangles with at least one vertex on each contour. However, if we enlarge our search for minimal-area triangulations to include triangles with all three vertices on a single contour, we can produce a more satisfying triangulation. Figure 14.b shows the same contour pair after undergoing a true area minimizing triangulation.

This triangulation was computed using a modification of the optimal bridge delimiter search, in which the visibility table is used to determine which vertices on **P** are visible from other vertices on **P**. Observe that in the simple branching

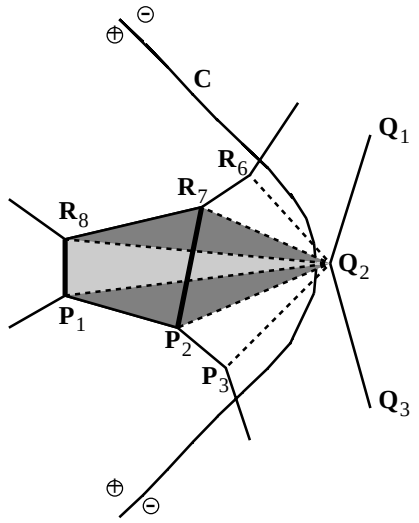

Figure 13: Break-even curve

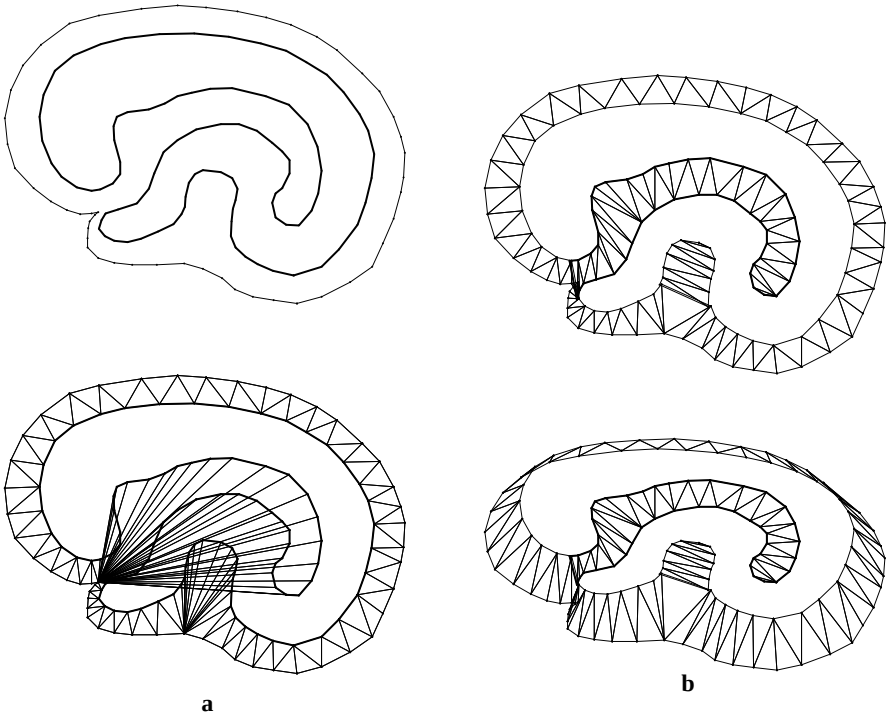

Figure 14: Self-intersecting "minimal-area" triangulation and true minimal-area triangulation

case, once polygons $\mathbf{P}$ and $\mathbf{R}$ are joined using segment $\mathbf{P}_i\mathbf{R}_l$ as the bridge (step 5 in the algorithm) the result is virtually identical to a non-branching case involving a non-convex polygon. Three minor changes are made to the algorithm. First, the initial bridge delimiter is chosen as a line segment near the center of the concavity (as measured by cumulative chord length). Second, step 9 is not needed. Finally, this procedure can be applied to each concavity.

It should be noted that cases exist for which *every* triangulation that requires its triangles to have at least one vertex on each contour will self intersect[7]. However, it is clear that if we allow the use of triangles with all three vertices on the same contour, that a non-self-intersecting triangulation can be found for *any* pair of contours: merely triangulate their convex hulls then fill in the concave regions with triangles.

5. Discussion

We have tested our algorithm on numerous hand-made models and on actual biological data. In all cases, the results earned our subjective approval.

Figures 15 and 16 show various examples of branching from a set of mouse embryo cross sections. In Figure 16(b), the branching consists of a hole. All of there figures were created using the fast heuristic algorithm, and are identical to those obtained using the slow-but-exact algorithm. In all cases, the algorithm ran in less than one second on a 75 MHz workstation on these cases that involve polygons of up to 200 vertices.

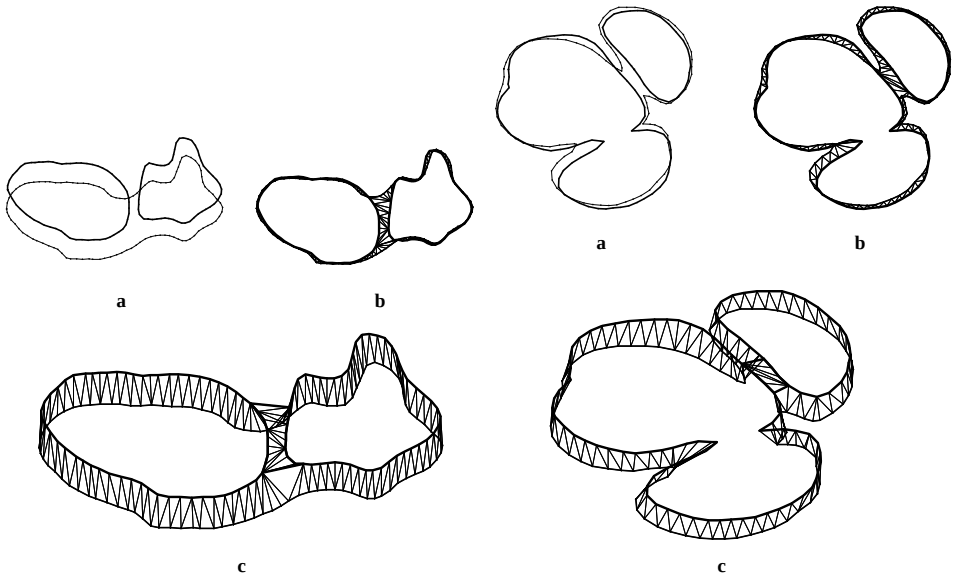

Figure 15: Mouse embryo examples 1 and 2

This paper describes an algorithm for triangulating contours that involve double-

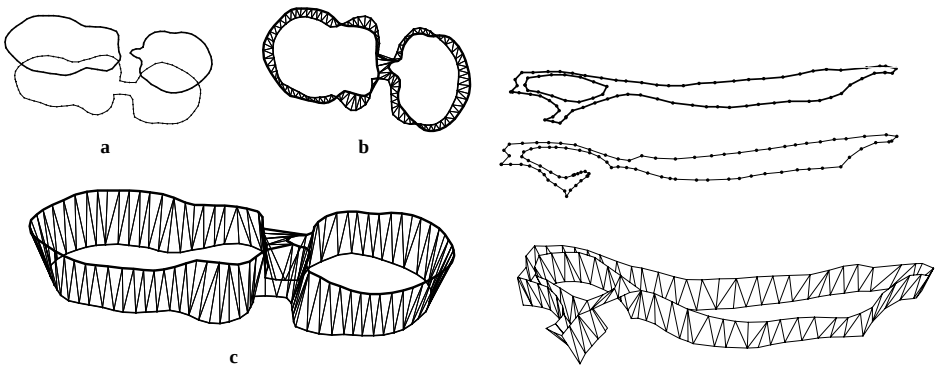

Figure 16: Mouse embryo examples 3 and 4

branching. The strategy uses triangles whose vertices lie at existing polygon vertices, forming a bridge between the two branching contours. Meyers, et al.[11] break the problem of surface reconstruction from planar contours into four sub-problems: correspondence, tiling, branching, and surface-fitting problems. They discuss tiling the bridge between two branches in conjunction with a piecewise parametric surface-fitting algorithm. Their observation that a criterion is needed for determining the extent of the bridge motivated this paper.

The paper whose objective most closely parallels that of this paper (i.e., a method for determining bridge delimiters) is Herbert et al.[8]. Their method is based on finding the shortest distances between vertices of **Q** and vertices of **P** and **R**. The paper observes that the method is known to not be robust.

Other approaches to triangulating branching contours have been proposed that involve the introduction of new vertices. Instead of forming a bridge between **P** and **R**, Choi and Park[3] split **Q** into two polygons, one of which gets triangulated with **P** and one with **R**. The medial axis of **P** and **R** is taken to be the polyline that splits **Q**.

Two recent papers[2,1] take fresh approaches to the triangulation problem. These papers are noteworthy for directly addressing the general triangulation problem between any two sets of any number of polygons. Those algorithms, as well as[3], discuss the problem in which any number of contour polygons on one plane are to be triangulated with any number of polygons on the other plane. These algorithms include a rule that the vertical projection of the two contours must intersect at polygon vertices (adding new vertices if needed) and those pairs of vertically-aligned vertices form edges of triangles.

5.1. Future Work

We feel that the area-minimization criterion has proven itself effective in the double branching problem, at least when the two cross-sections are similar enough. In

most contour data, branching between adjacent contour planes is less common than non-branching, and multiple branching is far less common than double branching. Even in contours comprising several polygons, any particular polygon on one level most commonly connects with just one polygon on adjacent levels. Nonetheless, further work needs to be done to consider how well the area-minimization concept handles multiple branchings. We ran an example to test a simple idea of treating a multiple-branching problem as a series of double-branching problems. In Figure 17, first the bridge between **Q** and **S** was formed by treating **P**, **Q**, and **S** as a double branching case; then **Q** and **S** were treated as a single polygon $\mathbf{Q} - \mathbf{S}$ and $\mathbf{P}$, $\mathbf{Q} - \mathbf{S}$ and **R** were treated as another single branching case. Clearly, much more work is needed to develop robust algorithms for the multiple branching case, and to verify that area-minimization is an appropriate heuristic in such cases.

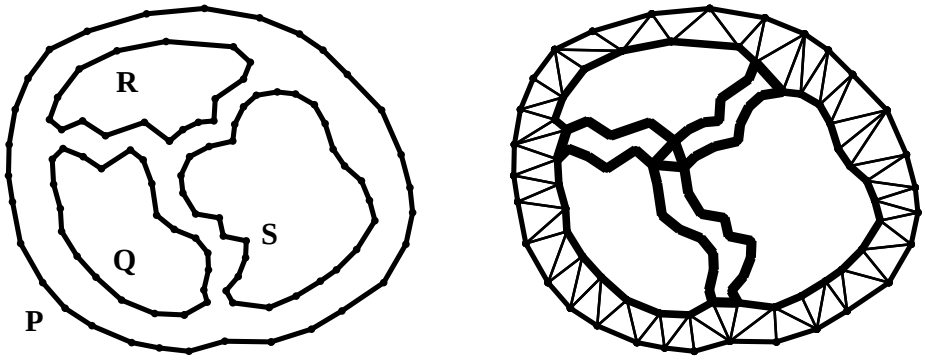

Figure 17: Triple branching

Another problem that deserves study is motivated by Figure 14. In this example, the self-intersection was eliminated by finding a minimal-area surface in which triangles are allowed with all three vertices lying on the same polygon. Can cases be found in which a true minimal-area surface self-intersects? If so, can an efficient algorithm be devised which solves for the triangulation of minimal area that does not self-intersect?

Acknowledgements

Effort sponsored by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF, under grant number F49620-95-1-0052, US-Japan Center of Utah. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon.

The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Office of Scientific Research or the U.S. Government.

Partial support was also provided by the Office of Naval Research.

Two exceptionally helpful referees contributed several comments that have improved the paper significantly.

References

1. C. L. Bajaj, E. J. Coyle, and K.-N. Lin. Arbitrary topology shape reconstruction from planar cross-sections. *Graphical Models and Image Processing*, 58, 6:524–543, 1996.
2. G. Barequet and M. Sharir. Piecewise-linear interpolation between polygonal slices. *Computer Vision and Image Understanding*, 63:251–272, 1996.
3. Y.-K. Choi and K.H. Park. A heuristic triangulation algorithm for multiple planar contours using an extended double branching procedure. *Visual Computer*, 10:372–387, 1994.
4. H.N. Christiansen and T.W. Sederberg. Conversion of complex contour line definitions into polygonal element mosaics. *Computer Graphics*, 12:187–192, 1978.
5. H. Fuchs, Z.M. Kedem, and S.P. Uselton. Optimal surface reconstruction from planar contours. *Commun. ACM*, 20:693–702, 1977.
6. S. Ganapathy and T.G. Dennehy. A new general triangulation method for planar contours. *Computer Graphics*, 16:69–75, 1982.
7. C. Gitlin, J. O'Rourke, and V.Subramanian. On reconstruction of polyhedra from parallel slices. *International Journal of Computational Geometry & Applications*, 6:103–122, 1996.
8. M.J. Herbert, C.B. Jones, and D.S. Tudhope. Three-dimensional reconstruction of geoscientific objects from serial sections. *The Visual Computer*, 11:343–359, 1995.
9. E. Keppel. Approximating complex surfaces by triangulation for contour lines. *IBM J. Res. Develop*, 19:2–11, 1975.
10. D. Meyers, S. Skinner, and K. R. Sloan. Surfaces from contours: The correspondence and branching problems. In Wizard V. Oz and Mihalis Yannakakis, editors, *Proceedings, Graphics Interface '91*, pages 246–254, 1991.
11. D. Meyers, S. Skinner, and K. R. Sloan. Surfaces from contours. *ACM Transactions on Graphics*, 11:228–258, 1992.
12. L. L. Schumaker. Reconstructing 3D objects from cross-sections. In W. Dahmen, M. Gasca, and C. A. Micchelli, editors, *Computation of Curves and Surfaces*, NATO ASI Series, pages 275–309. Kluwer Academic Publishers, Boston, 1990.
13. L. L. Schumaker. Triangulations in CAGD. *IEEE Computer Graphics & Applications*, 13:47–52, 1993.
14. T.W. Sederberg and E. Greenwood. A physically based approach to 2–d shape blending. *Computer Graphics*, 26:25–34, 1992.