On the choice of a cost function for the reconstruction of surfaces by triangulation between contours

W.R.J. Funnell

BioMedical Engineering Unit & Department of Otolaryngology
McGill University
3655 Drummond Street
Montréal, Québec
Canada H3G 1Y6

Computers & Structures 18(1): 23-26 (1984)
Submitted 1982 June 22
Revised 1982 September 18

ABSTRACT

Methods for the reconstruction of three-dimensional objects by triangulating between pairs of two-dimensional contours are discussed. Alternative cost functions for optimizing the triangulation are considered, and new cost functions are proposed that measure the narrowness of the triangles. These narrowness functions are particularly well suited to the generation of finite-element meshes.

1. INTRODUCTION

The problem of reconstructing the shapes of three-dimensional objects from series of two-dimensional contours arises in a number of areas, including many biomedical applications involving the forms and interrelationships of complex three-dimensional anatomical structures studied by serial histological sections.

Given a set of N planar polygonal closed contours, each described by the x and y coördinates of its vertices and by a single z coördinate, one approach to defining the surface of a corresponding three-dimensional solid is to define non-overlapping triangular 'tiles' or facets joining pairs of contours. For each triangle, two of its vertices will be neighbouring points on one polygonal contour, and the third vertex will be some point on a neighbouring contour.

In Section 2 I shall review the methods that have been described in the literature for choosing such a triangulation: it will be seen that the algorithm of Fuchs *et al.* [4936] offers a robust and reasonably efficient method. Their presentation of the method took surface area as the cost function to be minimized, and this has been repeated in subsequent uses of the method [1744,4264,4940,4931], but it is not essential. In Section 3 I shall discuss the use of different cost functions, and present new ones that may be preferable, particularly when the triangulation is to be used for generating a finite-element mesh for subsequent structural or other analysis.

2. REVIEW OF PUBLISHED METHODS

The basic computational problem involves choosing a particular order in which to join up nodes on neighbouring contours so as to form sequences of non-overlapping triangles. Suppose that we are given the contours S^A and S^B, consisting of the points $\{A_i, i = 1, 2, ... m\}$ and $\{B_j, j = 1, 2, ... n\}$. One must choose vertices A_{i0} and B_{j0} at which to start, and then choose a particular sequence of triangulation. In the following review of previously published methods, I shall consider the triangulation itself first, and afterwards discuss the problem of choosing a pair of starting points.

Triangulation algorithms

Keppel [4367] published the first description of a triangulation algorithm of which I am aware, although he mentioned (without references) earlier uses of triangulation by U. Rosenow, by J. Weinkam, and by W.A. Hunt. Keppel introduced the use of concepts from graph theory to describe the problem. He adopted a simple optimization criterion, namely, the maximization of the volume enclosed by the surface defined by the triangulation, and showed that the problem of finding the optimal triangulation is equivalent to finding a maximum-cost path through a graph which turns out to have a particularly simple structure. Each triangle is associated with an arc in the graph, and has a cost function, in this case volume, associated with it. Dijkstra [1001] had already described an algorithm for solving this problem in a more general class of graph, and Keppel presented an even simpler algorithm for the special case involved here. Unfortunately, these algorithms rely for their efficiency on the choice of a cost function which is never negative. The volume cost function is guaranteed to be non-negative only when the contour is convex, and as a result Keppel was forced to subdivide his contours, treating only convex subsets, and iterating until all contour points had been treated. The algorithm as a result is difficult to programme, and in fact Keppel did not give the details about the most difficult aspects of combining the treatments of the convex and concave parts of the contours.

Fuchs, Kedem & Uselton [4936] followed Keppel in formulating the problem in terms of graph theory. One of the two main contributions in their paper was in choosing to minimize surface area rather than maximizing volume. The surface-area cost function is always non-negative, and thus they avoided most of the complexities of Keppel's algorithm. The basic graph-traversal algorithm they used was similar in spirit to that described by Keppel although organized somewhat differently: whereas Keppel advanced through the rectangular grid of the graph with a diagonal 'front', Fuchs *et al.* advanced row by row [1002]. As discussed below, they incorporated into this row-by-row algorithm a second major contribution, making the choice of starting points much more efficient.

Some authors have attempted to simplify the triangulation problem by making a series of local decisions about which triangles to use, rather than performing a global optimization. For example, one can decide which of two alternative triangles to use at each step on the way around the contours by choosing the triangle with the shorter diagonal [4935, 1003]. In terms of the graph-theoretic formulation, this is equivalent to using a cost function equal to triangle perimeter (which is non-negative), but making a final decision about direction at each step along the path rather than looking at all the possibilities and making a globally optimal decision. When it works, this local strategy is of course much faster, but it can sometimes give unacceptable results even for relatively simple contours with mild concavities. Before triangulating, Christiansen & Sederberg [4935] scaled the x and y coördinates separately, and translated them, so as to map the contours onto unit squares. This improves the behaviour at least for some contours, but does not guarantee acceptable results for complex contours.

Ganapathy & Dennehy [5021] made local decisions based on a cost function involving the cumulative sum of the lengths of the contour segments (normalized by dividing by the overall contour perimeter), which works well for pairs of contours that are quite similar. The authors pointed out that, unlike other methods that make local decisions, this method has memory in the sense that its cost function depends on previous decisions; it is not clear how much of an advantage this is. They also pointed out that the decision criterion does not depend on the distance between contours. This means that the same triangulation is obtained even if the planes of consecutive contours are not parallel; other cost functions would presumably work for nonparallel planes but the particular triangulation obtained would depend on their relative positions and orientations.

Batnitzky *et al.* [3923] described another local method. They first mapped each contour (in some unspecified way) onto a contour that was star convex relative to the centroid of the original contour, that is, a contour on which every point could be joined to the centroid by a straight line which did not cross over the contour. They decided which pairs of points to join based on the directions from the contour centroids: of the two candidate pairs of points at each stage, they joined the pair for which the difference in direction of the points from their respective centroids was smaller. This method of choosing obviously depends on the property of star convexity, and such a mapping is only possible for contours which are fairly smooth and well behaved, without abrupt concavities.

Choosing starting points

Keppel [4367] did not address the problem of choosing a pair of starting points. One possibility [4935, 1003] is to choose the two vertices which are closest together, that is, to choose i_0 and j_0 such that A_{i0} and B_{j0} are closer together than any other pair A_i and B_j. (One can either look at all *mn* pairs of points before deciding which two points are closest, or, with some risk, arbitrarily choose to use the point on S^A which was used as starting point when triangulating between it and the previous contour, and only look at the *n* distances between it and the points on S^B.) This use of proximity runs into trouble if the two contours are skewed with respect to each other, and can be improved by aligning the centroids of the two contours before choosing the starting points [1004].

Another approach is to choose points A_{i0} and B_{j0} that lie in almost the same direction from the centroids of their respective contours [3923]. This method is suitable only for a contour which is star convex, or which can at least be mapped onto a contour which is star convex.

A third approach is to choose points with, for example, minimal *x* values [5021].

None of these approaches is suitable for contours with complex concavities. An approach which is more reliable, albeit considerably more expensive, is to choose one point arbitrarily on S^A to begin with and consider joining it to all *n* points on S^B, performing a new triangulation for all *n* such pairs of starting points, finally choosing the triangulation which is best in some sense. For the triangulation algorithm of Fuchs *et al.*, which performs a global path optimization for each starting point, it can be seen that by choosing A_{i0} arbitrarily and trying all B_j one is guaranteed to find the best overall path. However, in the case of an algorithm which makes a series of local decisions in determining a path, a given set of *n* starting pairs can easily give rise to *n* paths of which none is as good as might be obtained with some other starting pair.

Fuchs *et al.* showed how one can greatly improve the efficiency of trying *n* starting pairs, by making use of previous results in order to reduce the number of possible paths to be considered for each starting pair. Their method reduces the time increase for trying *n* starting pairs from a factor of *n* to a factor of $\log_2 n$.

3. CHOICE OF COST FUNCTION

As indicated above, the triangulation method of Fuchs *et al.* offers a reasonably efficient way of reliably triangulating even complex contours. However, their use of minimal surface area as an optimization criterion is not necessarily the best choice for all purposes. It is intuitively appealing because it is conceptually similar to stretching a rubber membrane over a wire frame: the elastic forces in the membrane will cause it to adopt something like a minimal-surface-area shape. However, it may result in shapes unlike what one would naturally choose to interpolate between certain contours. For example, Figure 1 shows the result of triangulating between two particular contours using a minimal-area criterion. It can be seen that the urge to minimize surface area has caused a deep concavity between the protrusions on the two contours. (The sample contours used here were all obtained from serial sections of a cat middle ear [4943]. The triangulations were done using a global path optimization algorithm, programmed in FORTRAN on a PDP-11/70.)

Figure 1 suggests that there may be better cost functions than surface area. The maximal-volume criterion would obviously avoid a concavity of this sort, but as discussed above it leads to problems because it can be either positive or negative. The cost function selected must be (1) a quantity associated with an individual triangle independent of its neighbours; and (2) one-signed, that is, either non-negative or non-positive. It makes no real difference whether it is to be minimized or maximized, and in the following I shall arbitrarily assume that all cost functions are to be minimized, for simplicity.

One type of cost function that suggests itself is some measure of the narrowness of the triangles.

Obviously in Figure 1 the avoidance of narrow triangles would have avoided the deep concavity. Furthermore, the avoidance of narrow triangles is to be especially commended if the triangulation of the surface is to be used for creating a finite-element model of the object: it is well known that long narrow elements tend to cause numerical problems in the matrix computations involved in the finite-element method.

The most obvious measure of narrowness is $1/\theta_{\min}$, that is, the inverse of the smallest angle of the triangle. Figure 2 shows the results of using this cost function in triangulating between the same contours as shown in Figure 1. It can be seen that the concavity has been avoided, and the resulting triangulation is excellent.

This cost function is somewhat expensive since it involves computing at least two trigonometric functions for each triangle. (Only two are needed if use is made of the fact that $\theta_1 + \theta_2 + \theta_3 = 180°$.) A simpler measure of narrowness is obtained by using

$$\frac{1}{1-\max\left(\cos\theta_k\right)}, k=1,2,3$$

Here, $\cos\theta_k$ is the cosine of the angle at the k-th vertex of the triangle, and is easily obtained by computing the dot product of the 3-dimensional vectors from that vertex to each of the other two vertices of the triangle. Choosing the maximal value of the cosine corresponds to minimizing θ. Subtracting from 1 and inverting yields a non-negative cost function which approaches infinity as the size of the narrowest angle of the triangle approaches zero, so it is a strong measure of narrowness. This cost function resulted in the same triangulation as shown in Figure 2.

An even less expensive cost function would be the maximal $\cos\theta_k$ itself, but this function is almost constant for angles near zero, suggesting that it might not penalize very narrow triangles enough. In fact it gives the same triangulation as the previous function for the example of Figure 2, and in most other cases that I have tested. In Figure 3, however, is shown a case where the triangulations given by $\cos\theta_k$ and $1/(1 - \cos\theta_k)$ are slightly different. It should be noted that with a triangulation algorithm that makes only local decisions it does not matter what the exact shape of the cost function is: any monotonically decreasing function of $\theta_{\min}$ should work equally well, except for possible numerical problems if the function is nearly flat. However, this is not the case when using a global path optimization algorithm like that of Fuchs *et al.* In any case the possibility of numerical problems is real, and $1/(1–\cos_{\max})$ is probably a safer choice than $\cos_{\max}$.

To test whether the results could be improved upon by simple modifications of the cost function, triangulations were done with 8 different contour pairs using the cost function $1/(1–\cos_{\max})$ raised to various powers between 0.1 and 5.0. In all cases tried, the triangulation results were the same over the whole range of powers. It appears that a drastic change in the shape of a cost function is needed in order to change its behaviour. Changing the function from being convex upwards ($\cos\theta$) to being convex downwards ($1/(1 – \cos\theta)$ or $1/\theta$) is sufficient to cause at least small changes in the triangulation.

Another cost function that is suggested by Figure 1 is triangle perimeter. The resulting triangulation is in fact the same as that of Figure 2. This cost function is related to the shorter-diagonal criterion used by Christiansen & Sederberg [4935] and Chawla [1003]: the two are equivalent for an algorithm making local decisions, but they may be slightly different for a global optimization. Comparing lengths of diagonals obviously requires less computation than comparing perimeters, since one needs to consider only the leading edge of each triangle rather than all three edges. One can simplify the diagonal-length calculation further by ignoring the z coördinates [1004]; again, the results will be identical when using local decisions but may differ slightly with global optimization.

4. CONCLUSIONS

The global path optimization algorithm of Fuchs *et al.* yields a robust triangulation procedure that works fairly well even for complex contours. However, the area-minimization criterion that has generally been used with it sometimes yields undesirable results. The volume-maximization criterion of Keppel [4367] may give better shapes but is much more complicated to use. The alternative cost functions suggested here give better results than the area function, without introducing extra difficulties. The narrowness function is especially well suited to applications where the triangulation is to be used as a finite-element mesh, since finite-element computations are sensitive to the existence of long narrow elements.

These alternative cost functions may also be desirable when the triangulation is being used specifically for volume and surface estimations, as by Marino *et al.* [4942], since they avoid the built-in biases of the maximal-volume and minimal-area cost functions.

In some applications one can get away with using an algorithm that uses local decisions rather than performing a global path optimization. This is true if the contours are relatively smooth and do not change shape too drastically from one contour to the next. These conditions may require using more closely spaced contours and more points per contour. If the computational burden imposed by these extra points is excessive (as it may well be if subsequent finite-element or hidden-surface computations are required) then it may be better to stay with the more expensive but more reliable global optimization. In either case the narrowness and perimeter cost functions may be used. The differences among different cost functions will be more pronounced with less well behaved contours.

ACKNOWLEDGEMENTS

This work was supported by the Medical Research Council of Canada. I should like to thank the reviewer for pointing out the existence of reference [5021].

CAPTIONS

Figure 1. Example of a triangulation using surface area as the cost function. On the left the two contours are shown projected onto the x-*y* plane, on the right they are projected onto the *y*-*z* plane.

Figure 2. Example of a triangulation using a narrowness cost function, for the same contours as in Figure 1. The same results were obtained for the cost functions $1/\theta$, $1/(1 - \cos\theta)$ and $\cos\theta$, and also for the triangle perimeter.

Figure 3. An example of triangulation in which different results were obtained by the different narrowness cost functions. Part *a* shows the triangulation obtained using $1/(1 - \cos\theta)$; the same results were obtained using $1/\theta$. Part *b* shows the upper part of the triangulation obtained using either $\cos\theta$ or triangle perimeter; the lower part of the triangulation is the same as in Part *a*. Part *c* shows the upper part of the triangulation obtained using area; again the lower part is the same as in Part *a*.

REFERENCES

3923. S. Batnitzky, H.I. Price, P.N. Cook, L.T. Cook and S.J. Dwyer III. Three-dimensional computer reconstruction from surface contours for head CT examinations. *J. Comp. Assist. Tomography* 5(1), 60 - 67 (1981).

1003. S.D. Chawla. *An Interactive Computer Graphics System for 3-D Stereoscopic Reconstruction from Serial Sections: An Application in the Study of Pulmonary Metastatic Growth*. M.Eng. thesis, McGill University, Montréal, iv + 62 pp (1979).

4935. H.N. Christiansen and T.W. Sederberg. Conversion of complex contour line definitions into polygonal element mosaics. *Comp. Graph. (SIGGRAPH-ACM)* 12(3), 187 - 192 (1978).

4940. 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.* BME-27(9), 493 - 500 (1980).

1001. E.W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik* 1, 269 - 271 (1959). Cited by A.V. Aho, J.E. Hopcroft and J.D. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, Reading (Mass.), pp. 207, 223 (1974).

4931. R.D. Fargason, S. Jacques, R.W. Rand, C.H. Shelden, G.D. McCann and S. Linn. Visualization and three-dimensional reconstruction of pituitary microadenomas from CT data: a technical report. *Surg. Neurol.* 15, 450 - 454 (1981).

4936. H. Fuchs, Z.M. Kedem and S.P. Uselton. Optimal surface reconstruction from planar contours. *Comm. ACM* 20(10), 693 - 702 (1977).

1002. Z.M. Kedem. Personal communication (1982).

4367. E. Keppel. Approximating complex surfaces by triangulation of contour lines. *IBM J. Res. & Develop.* 19(1), 2 - 11 (1975).

4942. T.A. Marino. The use of a computerized algorithm to determine single cardiac cell volumes. *J. Microscopy (Oxford)* 122(1), 65 - 73 (1981).

1744. M.J. Shantz and G.D. McCann. Computational morphology: three-dimensional computer graphics for electron microscopy. *IEEE Trans.* BME-25(1), 99 - 103 (1978).

4264. J.C.M. Antunez, F.R. Galey, F.H. Linthicum, and G.D. McCann. Computer-aided and graphic reconstruction of the human endolymphatic duct and sac: A method for comparing Meniere's and non-Meniere's disease cases. *Ann. Otol. Rhino. Laryngol.* 89(6) Part 3 Suppl. 76, 23 - 32 (1980).

1004. I. Sobel. Paper in progress (1982).

4943. W.R.J Funnell and K.E. Phelan. Finite-element modelling of the middle-ear ossicles. *J. Acoust. Soc. Am.* 69 Suppl. 1, S14 (1981).

5021. S. Ganapathy and T.G. Dennehy. A new general triangulation method for planar contours. *Computer Graphics* 16(3), 69 - 75 (1982).

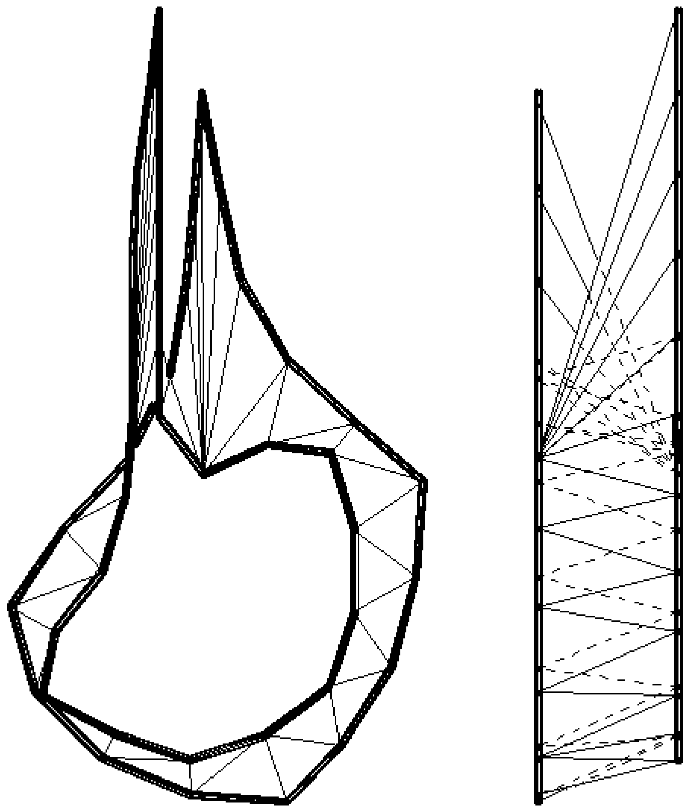

Figure 1

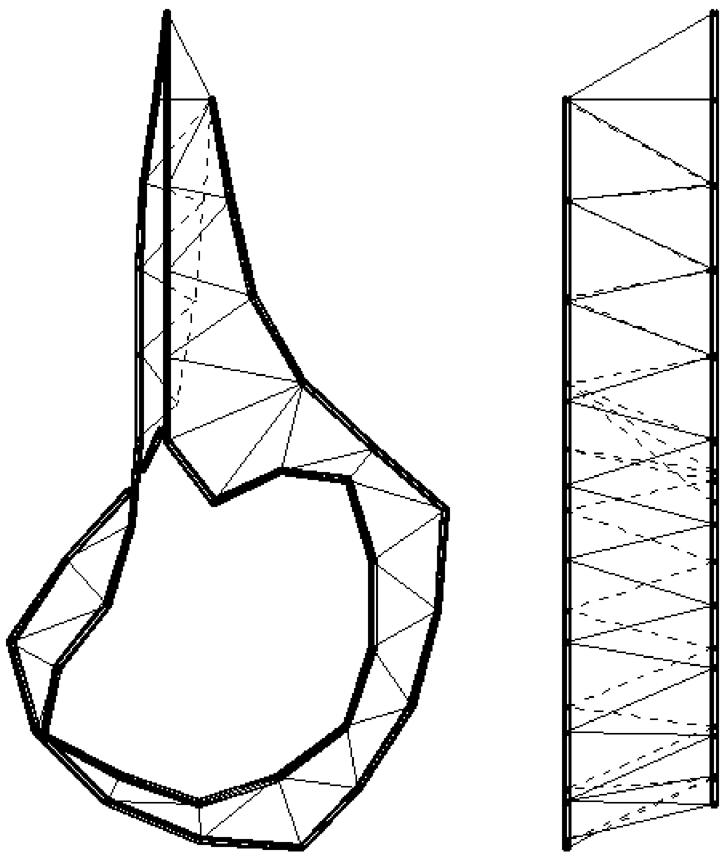

Figure 2

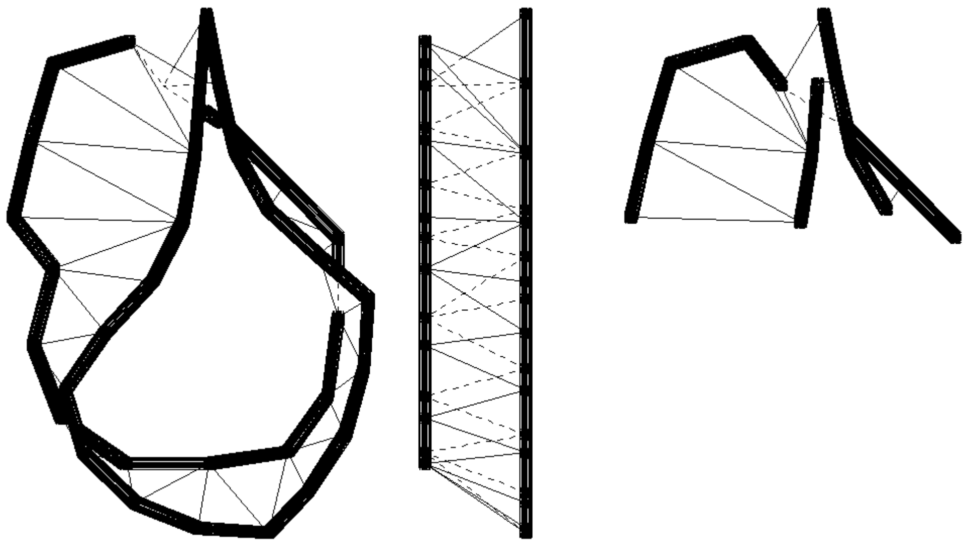

Figure 3 (corrupted)