REFERENCES

[1] J. C. Dunn, "A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters," *J. Cybern.*, vol. 3, pp. 32-57, 1974.

[2] J. C. Bezdek, "Fuzzy mathematics in pattern classification," Ph.D. dissertation, Appl. Math., Cornell Univ., Ithaca, NY, 1973.

[3] L. A. Zadeh, "Fuzzy sets," *Inform. Contr.*, vol. 8, pp. 338-353, 1965.

[4] J. B. Bezdek, "A convergence theorem for the fuzzy ISODATA clustering algorithms," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-2, pp. 1-8, 1980.

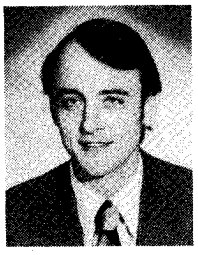
[5] —, "Cluster validity with fuzzy sets," *J. Cybern.*, vol. 3, pp. 58-73, 1973.

[6] —, "Mathematical models for systematics and taxonomy," in *Proc. 8th Int. Conf. on Numerical Taxonomy*, G. Estabrook, Ed. San Francisco, CA: Freeman, 1975, pp. 143-166.

[7] M. P. Windham, "Cluster validity for fuzzy clustering algorithms," *Fuzzy Sets Syst.*, vol. 5, pp. 177-185, 1981.

[8] R. Dubes and A. Jain, "Validity studies in clustering methodologies," *Pattern Recognition*, vol. 11, pp. 235-253, 1979.

[9] E. Anderson, "The iris of the Gaspe peninsula," *Bull. Amer. Iris Soc.*, vol. 59, pp. 2-5, 1935.



Michael P. Windham was born in Houston, TX, on September 23, 1944. He received the M.A. and Ph.D. degrees in mathematics from Rice University, Houston, TX, in 1970.

He was Instructor in Mathematics at University of Miami in 1970. Since 1971 he has been with the Department of Mathematics, Utah State University, Logan, where he is an Associate Professor. He is engaged in teaching and research in stochastic differential equations and cluster analysis.

Medial Axis Transformation of a Planar Shape

D. T. LEE, MEMBER, IEEE

Abstract—**The medial axis transformation is a means first proposed by Blum to describe a shape. In this paper we present a $O(n \log n)$ algorithm for computing the medial axis of a planar shape represented by an n-edge simple polygon. The algorithm is an improvement over most previously known results in terms of both efficiency and exactness and has been implemented in Fortran. Some computer-plotted output of the program are also shown in the paper.**

Index Terms—**Analysis of algorithm, computational complexity, continuous skeleton, divide-and-conquer, medical axis transformation, simple polygon, Voronoi diagram.**

I. INTRODUCTION

THE MEDIAL axis transformation is a technique first proposed by Blum [2] as a means to describe a figure. It is formally defined as follows: given an object represented, say by a simple polygon G, the medial axis $M(G)$ is the set of points $\{q\}$ internal to G such that there are at least two points on the object's boundary that are equidistant from $\{q\}$ and are closest to $\{q\}$. Because of its shape, the medial axis of a figure is also called the skeleton or the symmetric axis of the figure. Associated with the medial axis is a radius function R, which defines for each point on the axis its distance to the boundary of the object. With the axis and the radius function one can reconstruct the figure by taking the union of all circles centered on the points comprising the axis, each with a radius given by the radius function.

Since the introduction of the notion of medial axis, there has been a great deal of work involving the computation of the medial axis reported [1], [3], [4]-[8], [10]-[13]. Most of the previously known results take time proportional to n^2 where n is the number of boundary edges of the figure. Recently, Lee and Drysdale [8] and Kirkpatrick [7] have presented a general algorithm for finding continuous skeletons of a set of disjoint objects. Lee-Drysdale's algorithm runs in $O(n \log^2 n)$ time whereas Kirkpatrick's runs in $O(n \log n)$ time. The $O(n \log n)$ time algorithm by Kirkpatrick [7] is asymptotically optimal but is very tedious to implement. In this paper we shall give an algorithm which is simpler to implement and computes the medial axis of a simple polygon in $O(n \log n)$ time. The output of the algorithm would be precisely the medial axis of the polygon if the computer had arithmetic with infinite precision. The computer-plotted diagrams shown in the paper are exact to within the precision of the computer used. The medial axis of a simple polygon is a tree-like planar graph composed of straight-line segments and portions of parabolic curves. Before we give the description of the algorithm in the next section we shall introduce a few definitions and present some preliminary results.

Definition 1: A *closed line segment* $\overline{a, b}$ consists of two endpoints a and b, and a straight-line portion which is denoted by

Manuscript received December 12, 1980; revised January 20, 1982. This work was supported in part by the National Science Foundation under Grant MCS-7916847.

The author is with the Department of Electrical Engineering and Computer Science, Northwestern University, Evanston, IL 60201.

0162-8828/82/0700-0363$00.75

(a, b) and referred to as an *open segment* or briefly a *segment*. Points or segments are called *elements*. The straight line containing $\overline{a, b}$ is denoted by $\overleftrightarrow{a, b}$.

Definition 2: The projection $p(q, \overline{a, b})$ of a point q onto a closed segment $\overline{a, b}$ is the intersection of the line $\overleftrightarrow{a, b}$ and the line perpendicular to $\overleftrightarrow{a, b}$ and passing through q.

Definition 3: The distance $d(q, \overline{a, b})$ between a point q and a closed segment $\overline{a, b}$ in the Euclidean metric is the distance $d(q, x)$, $x = p(q, \overline{a, b})$, between q and its projection onto $\overline{a, b}$ if x belongs to $\overline{a, b}$ and is $\text{MIN}(d(q, a), d(q, b))$ otherwise, where $d(q, r)$ denotes the Euclidean distance between points q and r. The point of $\overline{a, b}$ which is closest to q is called the *image* $I(q, \overline{a, b})$ of q on $\overline{a, b}$.

Definition 4: The bisector $B(e_i, e_j)$ of two elements e_i and e_j is the locus of points equidistant from e_i and e_j. The bisector $B(X, Y)$ of two sets of elements X and Y is defined to be the locus of points equidistant from X and Y, where the distance $d(q, X)$ of a point q and a set of elements X is defined to be $\text{MIN}_{e \in X}\, d(q, e)$, i.e., $B(X, Y) = \{q \mid \text{MIN}_{e \in X}\, d(q, e) = \text{MIN}_{e \in Y}\, d(q, e)\}$. The bisector $B(e_i, e_j)$ is said to be *oriented* if a direction is imposed upon it so that element e_i and e_j lie to the *left* and to the *right* of it, respectively. An *oriented* bisector $B(X, Y)$ is defined similarly.

For example, in Fig. 1 the bisector $B(q, \overline{a, b})$ of a point q and a closed segment $\overline{a, b}$ has three components, i.e., $B(q, a)$, $B(q, b)$, and a portion of the parabola whose focus and directrix are the point q and the line $\overleftrightarrow{a, b}$, respectively. Fig. 2 shows the bisector $B(\overline{a, b}, \overline{c, d})$ of two closed segments $\overline{a, b}$ and $\overline{c, d}$, where one of the components is a portion of the angular bisector of the angle formed by the lines $\overleftrightarrow{a, b}$ and $\overleftrightarrow{c, d}$. Note that the bisector $B(a, \overline{a, b})$, instead of being a parabola, is a line perpendicular to $\overline{a, b}$ and passing through a.

Definition 5: Given two elements e_i and e_j, the half-plane associated with e_i, *denoted* $h(e_i, e_j)$, is the locus of points closer to e_i than to e_j, i.e., $h(e_i, e_j) = \{x \mid d(x, e_i) \leq d(x, e_j)\}$.

Definition 6: Given n elements, $e_1, e_2, \cdots, e_n$, the Voronoi polygon associated with e_i, denoted $V(e_i)$, is the locus of points closer to e_i than to any other element, i.e., $V(e_i) = \cap_{j \neq i}\, h(e_i, e_j)$. The boundary edges of $V(e_i)$ are not necessarily straight-line segment and are called Voronoi edges and the vertices of $V(e_i)$ are called Voronoi points. The collection of the Voronoi polygons associated with each of the n elements is called the *Voronoi diagram* $\text{VOD}(S)$ of $S = \{e_1, e_2, \cdots, e_n\}$.

Definition 7: A simple polygon G of n vertices, denoted $G = (q_0, q_1, \cdots, q_{n-1})$, is a sequence of n points such that $\overline{q_i q_{i+1}}$ is a closed segment for $i = 0, 1, \cdots, n-1$, called an *edge*, and no two nonconsecutive edges intersect.[1] In other words, a simple polygon G is a closed plane figure composed of straight-line edges and divides the plane into two regions, interior and exterior.

Definition 8: Given a simple polygon $G = (q_0, q_1, \cdots, q_{n-1})$, a vertex q_i is called *convex* if the internal angle at q_i is less than π and is called *reflex* otherwise. We assume without loss of generality that no internal angle has a measure π, i.e., no three consecutive vertices are collinear.

Definition 9: Given a simple polygon $G = (q_0, q_1, \cdots, q_{n-1})$, let e_i denote the segment (q_i, q_{i+1}), for $i = 0, 1, \cdots, n-1$. A *chain* of G is a sequence of elements $e_j, q_{j+1}, e_{j+2}, \cdots, q_{k-1}, e_{k-1}$ such that the vertices q_j and q_k are convex and $q_{j+1}, \cdots, q_{k-1}$ are reflex.

[1] Index additions and subtractions are taken modulo n.

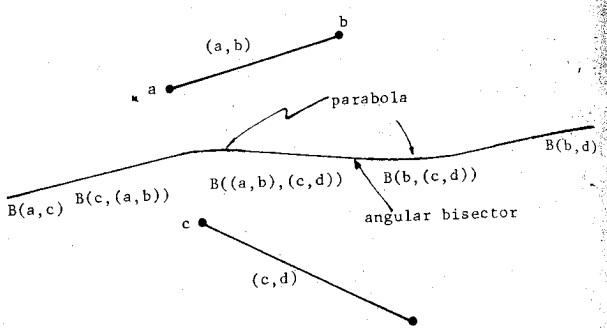


Fig. 1. Bisector of a point and a closed segment.

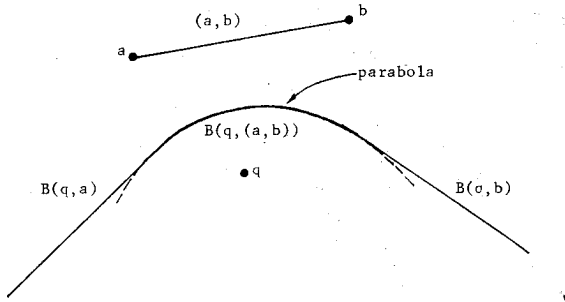


Fig. 2. Bisector of two closed segments.

Note that if G is a convex polygon, i.e., every vertex of G is convex, then each segment e_i by itself forms a chain. Suppose now that G has m reflex vertices, $q_{i_1}, q_{i_2}, \cdots, q_{i_m}$. In the following we shall represent G by $(e_{i_1-1}, q_{i_1}, e_{i_1}, \cdots, e_{i_2-1}, q_{i_2}, e_{i_2}, \cdots, e_{i_m-1}, q_{i_m}, e_{i_m}, \cdots, e_{i_1-2})$ which is a sequence of n segments interspersed with m reflex vertices. For convenience, we further assume that q_0 is convex and q_1 is reflex if any exists, so that G can be written as $G = (e_0, q_1, e_1, \cdots, e_{n-1})$. Since in the case of convex polygons the medial axes can be obtained in $O(n \log n)$ time using an algorithm given by Preparata [13], we shall consider the case when the given polygon is not convex. The algorithm presented here is certainly applicable to convex polygons. Consider now the set of m reflex vertices and n edges $S = \{e_{i_1-1}, q_{i_1}, e_{i_1}, \cdots, e_{i_2-1}, q_{i_2}, e_{i_2}, \cdots, e_{i_m-1}, q_{i_m}, e_{i_m}, \cdots, e_{n-1}\}$. The Voronoi diagram $\text{VOD}(S)$ will partition the interior (and exterior) of G into $m + n$ Voronoi polygons, each associated with an element in S. Fig. 3 shows the Voronoi diagram of a simple polygon (restricted only to the interior of the polygon hereafter). Interestingly enough the medial axis of a polygon is totally contained in the set of Voronoi edges of the polygon. Since the definition of the medial axis requires that the circles centered at points on the axis with radii specified by the radius function be tangent to at least two boundary points of G, we can obtain the medial axis by removing the two Voronoi edges incident with each reflex vertex. Fig. 4 shows the medial axis of the polygon shown in Fig. 3.

We give below some properties of the Voronoi diagram of a polygon G. Detailed proofs can be found in [8].

Lemma 1: Let e_i be a segment or a reflex vertex of G. For

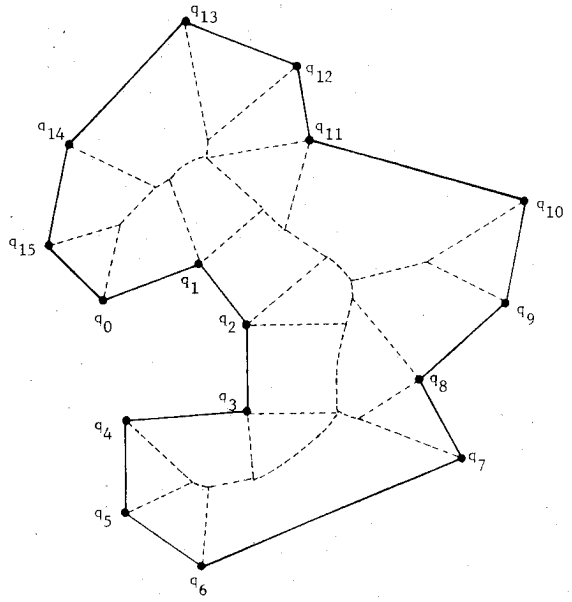


Fig. 3. Voronoi diagram of a simple polygon.

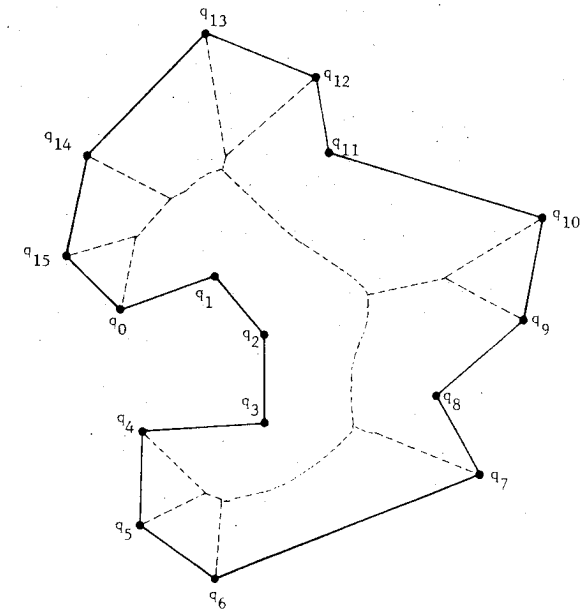


Fig. 4. Medial axis of a simple polygon.

any point z in $V(e_i)$, the line segment $\overline{z, t}$, where t is the image of z on e_i, is entirely contained in $V(e_i)$ [see Fig. 5(a)].

Corollary 1: Each Voronoi polygon $V(e_i)$ is path-connected, i.e., for any two points u and v in $V(e_i)$, there exists a path connecting u and v that is totally contained in $V(e_i)$.

Proof: Let u' and v' be the images of u and v on e_i, respectively. Since $\overline{uu'}$ and $\overline{vv'}$ lie completely in $V(e_i)$, the path $u, u', v'; v$, connecting u and v lies in $V(e_i)$. Note that if e_i is a reflex vertex, $u' = v' = e_i$ and that if e_i is an edge, then line segment $\overline{u', v'}$ is contained in e_i. □

The lemma implies that if we move a point z along the boundary Voronoi edges of $V(e_i)$ in a counterclockwise (clockwise) direction, then the image $I(z, e_i)$ also moves in the same direction along e_i. If e_i is a vertex, then e_i itself is the image $I(z, e_i)$ for all $z \in V(e_i)$.

Lemma 2: The Voronoi polygons $V(e_i)$ and $V(e_j)$ share an edge if and only if there exists a point z such that the circle centered at z with radius $d(z, e_i) = d(z, e_j)$ does not include any boundary point of G in its interior [see Fig. 5(b)].

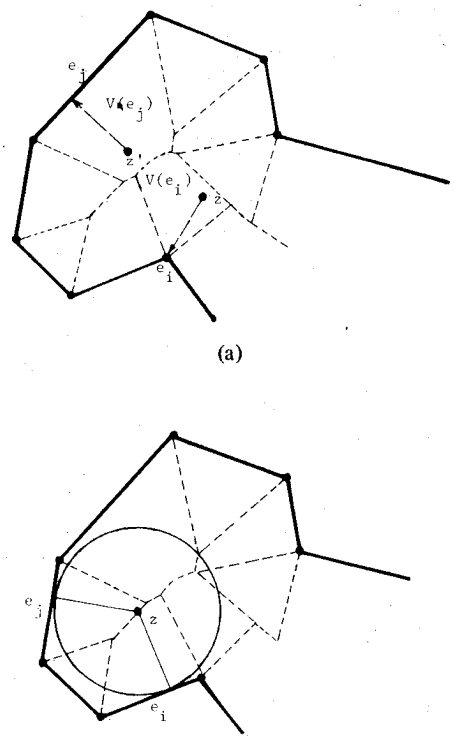


Fig. 5. Illustration of Lemmas 1 and 2.

Corollary 2: Let z be any point on a Voronoi edge $\bar{B}(e_i, e_j)$ of $V(e_i)$. The circle centered at z with radius $d(z, I(z, e_i))$ is *tangent* to elements e_i and e_j at points $I(z, e_i)$ and $I(z, e_j)$.

Since for $z \in \bar{B}(e_i, e_j)$, $I(z, e_i)$ and $I(z, e_j)$ are identical only when e_i is an endpoint of e_j or vice versa, we can obtain the medial axis of G by removing those Voronoi edges $\bar{B}(e_i, e_j)$ where e_i is an endpoint of e_j or vice versa. Thus, we have the following.

Corollary 3: The medial axis of G is the set of Voronoi edges less the edges incident with reflex vertices.

Lemma 3: The Voronoi diagram of a simple polygon with n edges and m reflex vertices is planar and has at most $2(n + m) - 3$ Voronoi edges and $n + m - 2$ Voronoi points.

Proof: The planarity follows from the fact that each Voronoi polygon is path-connected and the numbers of Voronoi edges and Voronoi points can be obtained by Euler's formula.

II. The Algorithm

We shall now describe the algorithm for constructing the Voronoi diagram of a simple polygon. As before we assume that G is represented by a list of $N = n + m$ elements $e_1, e_2, \cdots, e_N$ where n is the number of edges and m the number of reflex vertices. The algorithm to be described is based on the divide-and-conquer technique. That is, we shall divide G into two lists $G_1 = (e_1, e_2, \cdots, e_{\lfloor N/2 \rfloor})$ and $G_2 = (e_{\lfloor N/2 \rfloor + 1}, \cdots, e_N)$ and recursively construct the Voronoi diagrams VOD(G_1) and VOD(G_2). Then we merge VOD(G_1) and VOD(G_2) to form

the diagram VOD(G). As we shall see later, since the merge process takes $0(N)$ time, the overall running time is $0(N \log N)$.

Because we are interested in only the portion of the diagram that is internal to G, we shall restrict our discussion to the interior of G. To distinguish the interior of G from the exterior we shall assume that G is traversed in a counterclockwise direction so that the interior of G always lies to the *left*. Furthermore, when we say the Voronoi diagram of a list of elements, we mean the portion that is to the left of the list of elements. Fig. 6(a) and (b) illustrate the Voronoi diagrams of $G_1 = (e_1, e_2, \cdots, e_{10})$ and of $G_2 = (e_{11}, e_{12}, \cdots, e_{21})$ where $(e_1, e_2, \cdots, e_{21}) = G$ is a polygon shown in Fig. 3. For implementation purposes we instead partition the list of elements of G into chains $C_1, C_2, \cdots, C_h$ (cf. Definition 9) and apply the divide-and-conquer technique to $S = \{C_1, C_2, \cdots, C_h\}$. The reason for it is that the Voronoi diagram of a chain can be computed straightforwardly in time proportional to the number of elements in the chain, and yet the running time of the modified algorithm remains the same, i.e., $0(N \log N)$. Referring to Fig. 6(a), we can partition G_1 into 4 chains, i.e., $C_1 = e_1$, $C_2 = e_2$, $C_3 = e_3$ and $C_4 = e_4, e_5, \cdots, e_{10}$. The Voronoi diagram of C_4 consists of six (6) Voronoi edges $\bar{B}(e_4, e_5)$, $\bar{B}(e_5, e_6)$, $\bar{B}(e_6, e_7)$, $\bar{B}(e_7, e_8)$, $\bar{B}(e_8, e_9)$, and $\bar{B}(e_9, e_{10})$. Note that these Voronoi edges involve two elements one of which is an endpoint of the other. To be consistent with the implementation we shall describe the algorithm using the latter approach, i.e., we first partition G into a set $S = \{C, C_2, \cdots, C_h\}$ of chains, recursively construct VOD(S_1) and VOD(S_2), where $S_1 = \{C_1, C_2, \cdots, C_{\lfloor h/2 \rfloor}\}$ and $S_2 = \{C_{\lfloor h/2 \rfloor + 1}, \cdots, C_h\}$ and then merge VOD(S_1) and VOD(S_2) together to form VOD(S).

Let us now assume that VOD(S_1) and VOD(S_2) are available and describe how these two diagrams can be merged in $0(N)$ time. Suppose that S_1 contains the elements $e_1, e_2, \cdots, e_j$ and S_2 contains $e_{j+1}, e_{j+2}, \cdots, e_N$, where e_1 and e_j are, respectively, the first and last elements of C_1 and $C_{\lfloor h/2 \rfloor}$ and e_{j+1} and e_N are, respectively, the first and last elements of $C_{\lfloor h/2 \rfloor + 1}$ and C_h. To merge VOD(S_1) and VOD(S_2) we need to construct the merge curve M which is the bisector $B(S_1, S_2)$ of two sets *of* S_1 and S_2 of elements. Since e_j and e_{j+1} share a vertex q which is convex, the bisector $B(e_j, e_{j+1})$ which is an angle bisector is a component bisector of M and will be our *starting bisector*. Similarly, the vertex shared by e_1 and e_N is convex and the angle bisector $B(e_1, e_N)$ which is also a component bisector of M will be our *terminating bisector*, i.e., when $B(e_1, e_N)$ is constructed during the merge process, we know that the merge is completed. The terminating bisector is easy to identify at the very last phase of merging. During the recursion, the terminating bisector will be one, say $B(e_s, e_t)$ such that both e_s and e_t are vertices of G and the line l determined by e_s and e_t has the property that all the elements involved in the current merge process lie on one side of l. The merge process will terminate once the stopping condition is met. With the starting (oriented) bisector $B(e_j, e_{j+1})$ we shall scan the edges of the Voronoi polygon $V(e_j)$ in *counterclockwise* (CCW) direction to find the edge $\bar{B}(e_j, e_s)$ which intersects $B(e_j, e_{j+1})$ and similarly scan the edges of the Voronoi polygon $V(e_{j+1})$ in a *clockwise* (CW) direction to find the edge

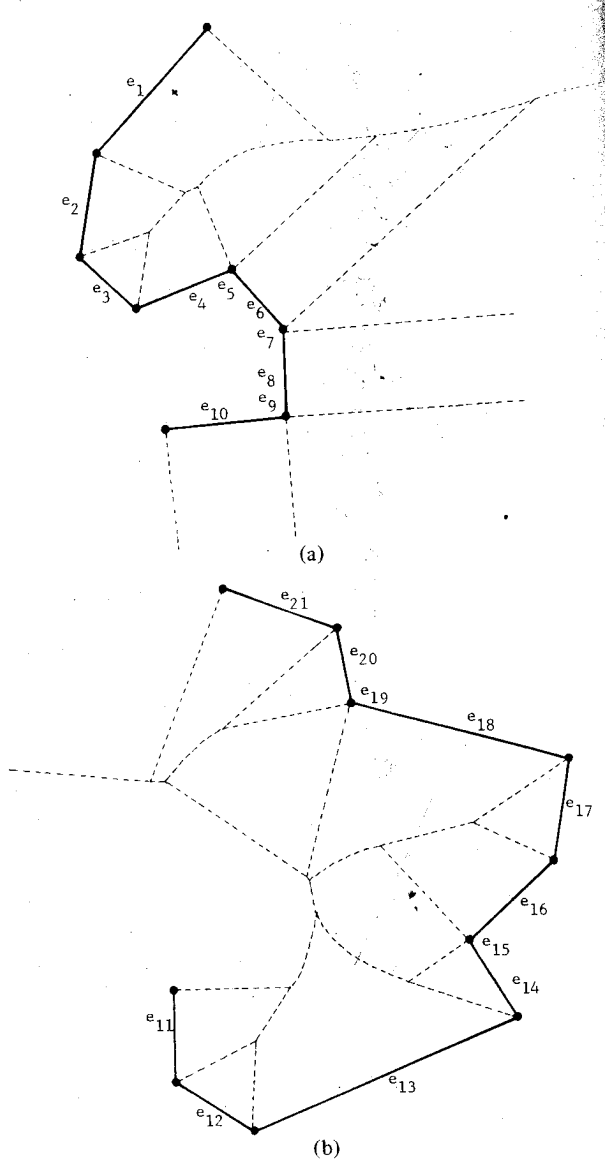


Fig. 6. Voronoi diagrams of two lists of elements.

$\bar{B}(e_{j+1}, e_t)$ which intersects $B(e_j, e_{j+1})$. With an additional comparison we can determine between the two edges $\bar{B}(e_j, e_s)$ and $\bar{B}(e_{j+1}, e_t)$ which intersects $B(e_j, e_{j+1})$ first. If $\bar{B}(e_j, e_s)$ intersects $B(e_j, e_{j+1})$ before $\bar{B}(e_{j+1}, e_t)$, then the next bisector to consider is $B(e_s, e_{j+1})$, otherwise it will be $B(e_j, e_t)$. Ties are broken arbitrarily. In general, in each step there is a triplet $(B(e_u, e_v), V(e_u), V(e_v))$, i.e., a current bisector and two Voronoi polygons whose edges are to be tested for intersection with the current bisector. The scanning scheme is that for current oriented bisector $B(e_u, e_v)$ we scan the edges of the polygon associated with the element e_u to the *left* of $B(e_u, e_v)$ in a CCW direction starting with the last examined edge and scan the edges of the polygon associated with the element e_v to the *right* of $B(e_u, e_v)$ in a CW direction starting with the last examined edge. If $B(e_u, e_v)$ intersects the edge $\bar{B}(e_u, e_s)$ of $V(e_u)$ before the edge $\bar{B}(e_v, e_t)$ of $V(e_v)$, then the next triplet to consider is $(B(e_s, e_v), V(e_s), V(e_v))$ otherwise the triplet is $(B(e_u, e_t), V(e_u), V(e_t))$. If during the scan none of the edges of $V(e_u)$ and of $V(e_v)$ intersects $B(e_u, e_v)$, then $B(e_u, e_v)$ will be the last component bisector of M. It can be shown that at that

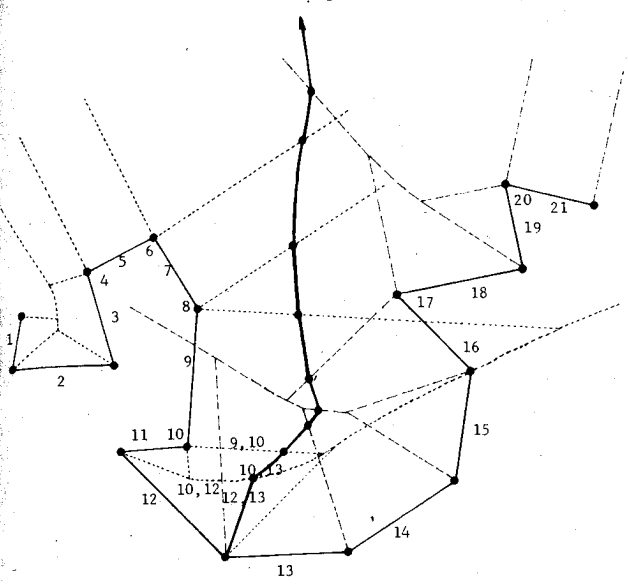


Fig. 7. Merge of two Voronoi diagrams.

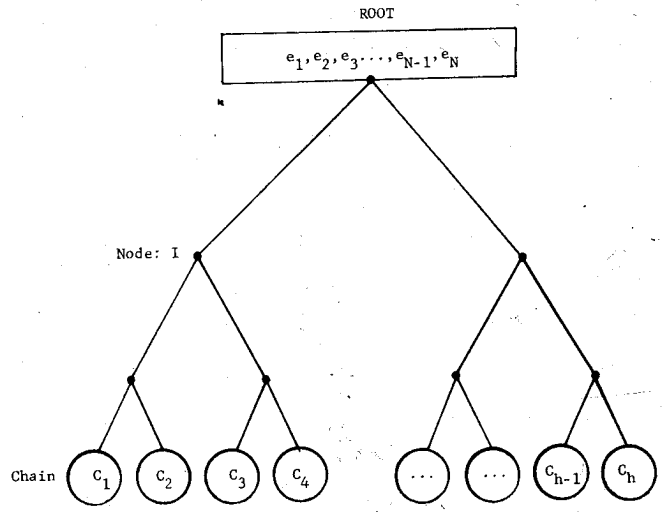


Fig. 8. Binary merge tree for computing the Voronoi diagram of a simple polygon.

time $B(e_u, e_v)$ satisfies the stopping condition. Let us illustrate this idea by an example. Consider the example shown in Fig. 7, where $S_1 = \{C_1 = (1), C_2 = (2), C_3 = (3, 4, \cdots, 11), C_4 = (12)\}$ and $S_2 = \{C_5 = (13), C_6 = (14), C_7 = (15), C_8 = (16, 17, 18), C_9 = (19, 20, 21)\}$. VOD$(S_1)$ and VOD(S_2), are shown in dotted and dashed lines, respectively. $B(12, 13)$ is our starting bisector. We scan the edges of $V(12)$ in CCW direction and find that $B(12, 13)$ intersects $B(10, 12)$. Similarly we scan the edges of $V(13)$ in a CW direction and find that $B(12, 13)$ intersects $B(13, 17)$. Since $B(10, 12)$ intersects $B(12, 13)$ before $B(13, 17)$ does, the next bisector is $B(10, 13)$. Now we scan the edges of $V(10)$ in a CCW direction starting with the last examined edge $B(10, 12)$ and scan the edges of $V(13)$ in a CW direction starting with the last examined edge $B(13, 17)$ to find the edges which intersect $B(10, 13)$. Since $B(9, 10)$ intersects $B(10, 13)$ before $B(13, 14)$ does, the next bisector is $B(9, 13)$. The same process is repeated until the bisector $B(6, 20)$ is reached. Note that when the current bisector $B(6, 20)$ is being constructed, we will scan as before the edges of $V(6)$ and $V(20)$ to find which edges intersect $B(6, 20)$. Since no other edges except $B(17, 20)$ of $V(20)$ intersect $B(6, 20)$, we terminate the merge procedure. All those edges of VOD(S_1) that lie to the right of the oriented bisector $B(S_1, S_2)$ are discarded. Similarly the edges of VOD(S_2) that lie to the left of $B(S_1, S_2)$ are discarded. In fact, these edges can be deleted while applying the scanning scheme to construct the merge curve.

The analysis of the algorithm shows that merging two Voronoi diagrams VOD(S_1) and VOD(S_2) takes time proportional to the total number of elements in S_1 and S_2. Therefore, the entire algorithm runs in time $0(N \log N)$ where $N = n + m$. The proof of the correctness of the algorithm and its running time can be found in [8].

III. Implementation

The algorithm described in Section II has been implemented in Fortran in a nonrecursive manner. The program takes as input a sequence of vertices $q_1, q_2, \cdots, q_n$, each represented by its x- and y-coordinates of a simple polygon and partitions the set of vertices into h chains $C_1, C_2, \cdots, C_h$. Then the Voronoi diagram for each chain, VOD(C_i), $i = 1, 2, \cdots, h$, is computed. Next these h Voronoi diagrams are merged two at a time to form VOD$(C_1 \cup C_2)$, VOD$(C_3 \cup C_4)$, etc., as illustrated by the binary merge tree shown in Fig. 8. In Fig. 8, the leaves are the Voronoi diagrams for chains and the internal nodes represent the Voronoi diagrams for the elements in the corresponding subtrees. Thus, the root of the binary merge tree will represent the final Voronoi diagram of the simple polygon. The height of the merge tree is $\lceil \log_2 h \rceil$ and each internal node I represents work for merging of the two Voronoi diagrams associated with its two sons and is $0(t_l + t_r)$, where t_l and t_r denote the numbers of elements in the left and right subtrees of node I. Thus, the time required for the program is at most $\lceil \log_2 h \rceil * 0(N)$ which is $0(N \log N)$.

To facilitate the computation of the merge curve $B(S_1, S_2)$ for some sets S_1 and S_2 of elements we make use of the following.

1) The merge curve $B(S_1, S_2)$ consists of component bisectors of the form $B(e_i, e_j)$ where $e_i \in S_1$ and $e_j \in S_2$. To compute the curve imagine moving a point z along $B(e_i, e_j)$. In order for z to be in $B(e_i, e_j)$ the projections of z onto e_i and e_j must belong to e_i and e_j, respectively. That is, if e_i is a segment, then z must be within the region defined by two parallel lines through endpoints of e_i and perpendicular to e_i and if e_i is a reflex vertex, then z must be within the wedge defined by the lines perpendicular to e_{i-1} and to e_{i+1} and by the apex e_i.

2) In computing $B(S_1, S_2)$, where $S_1 = \{e_i, e_{i+1}, \cdots, e_j\}$ and $S_2 = \{e_{j+1}, e_{j+2}, \cdots, e_k\}$, we terminate the computation when the stopping condition is met, i.e., when $B(e_u, e_v)$ is reached where $e_u \in S_1$ and $e_v \in S_2$ and all the elements in S_1 and S_2 lie on one side of $\overleftrightarrow{e_u, e_v}$. It can be shown that this can only happen when both e_u and e_v are reflex vertices. Note that e_u can be q_i which is an endpoint of e_i and e_v can be q_{k+1} which is an endpoint of e_k.

Figs. 9-11 are the output of the program run under FTN with PLOT-10.

CP TIME: .569 SEC
>

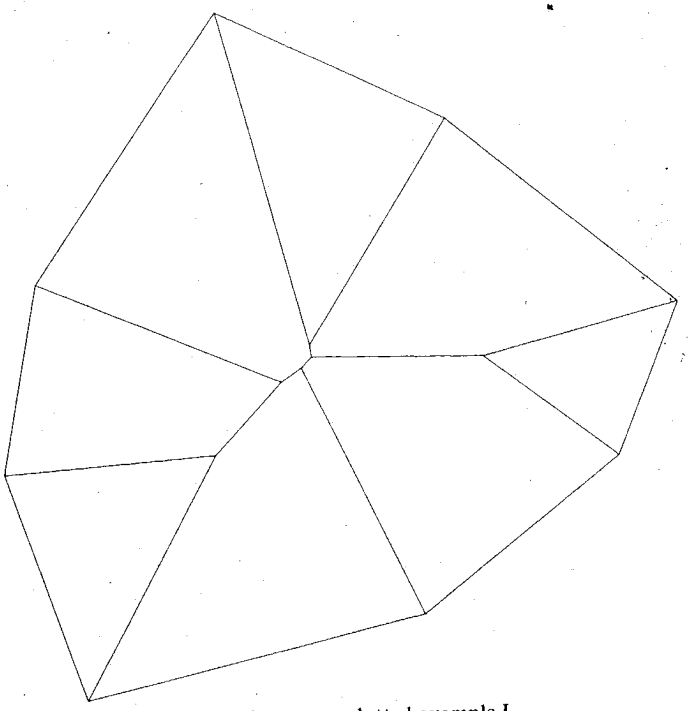


Fig. 9. Computer-plotted example I.

> OK
>

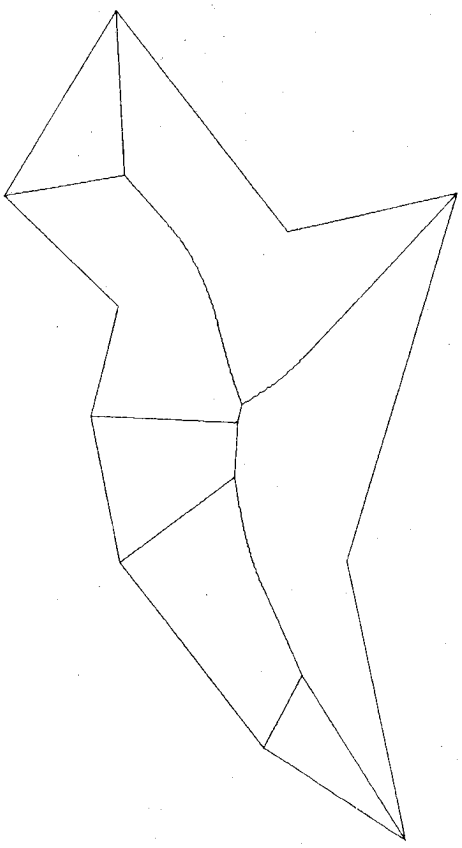


Fig. 10. Computer-plotted example II.

> OK
> OK
>

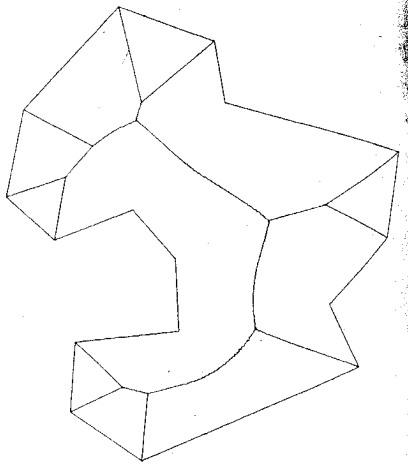


Fig. 11. Computer-plotted example III.

IV. Conclusion

We have presented a divide-and-conquer algorithm for constructing the medial axis of a simple polygon with n edges in $O(n \log n)$ time. The algorithm has been implemented in Fortran and satisfactory results have been obtained. Whether or not its time complexity is optimal remains to be an open problem.

As a passing remark, the medial axis not only describes the figure represented by a simple polygon, but also contains information about where the *closest* distance (or clearance) between nonconsecutive vertices and/or edges occurs. It therefore has an application in the design rule checking (clearance test) of artworks in very large scale integrated (VLSI) circuit design [9].

References

[1] N. I. Badler and C. Dane, "The medial axis of a coarse binary image using boundary smoothing," in *Proc. Pattern Recognition and Image Processing*, Aug. 1979, pp. 286–291.

H. Blum, "A transformation for extracting new descriptors of shape," in *Proc. Symp. Models for Perception of Speech and Visual Form*, W. Whaten-Dunn, Ed. Cambridge, MA: M.I.T. Press, 1967, pp. 362-380.
H. Blum and R. N. Nagel, "Shape description using weighted symmetric axis features," *Pattern Recognition*, vol. 10, pp. 167-180, 1978.
F. L. Bookstein, "The line-skeleton," *Comput. Graphics Image Processing*, vol. 11, pp. 123-137, 1979.
L. Calabi and W. E. Hartnett, "Shape recognition, prairie fires, convex deficiences and skeletons," *Amer. Math. Month.*, vol. 75, pp. 335-342, Apr. 1968.
P. V. de Souze and P. Houghton, "Computer location of medial axis," *Comput. Biomed. Res.*, vol. 10, pp. 333-343, 1977.
D. G. Kirkpatrick, "Efficient computation of continuous skeletons," in *Proc. 20th Annu. Symp. Found. Computer Sci.*, Oct. 1979, pp. 18-27.
D. T. Lee and R. L. Drysdale, "Generalization of Voronoi diagrams in the plane," *SIAM J. Comput.*, vol. 10, pp. 73-87, Feb. 1981.
C. Mead and L. Conway, *Introduction to VLSI Systems.* Reading, MA: Addison-Wesley, 1980.
U. Montanari, "A method for obtaining skeletons using a quasi-Euclidean distance," *J. Ass. Comput. Mach.*, vol. 15, pp. 600-624, Oct. 1968.
U. Montanari, "Continuous skeletons from digitized images," *J. Ass. Comput. Mach.*, vol. 16, pp. 534-549, Oct. 1969.
J. L. Pfaltz and A. Rosenfeld, "Computer representation of planar regions by their skeletons," *Commun. Ass. Comput. Mach.*, vol. 10, pp. 119-125, Feb. 1967.
F. P. Preparata, "The medial axis of a simple polygon," in *Proc. 6th Symp. Math. Foundations of Comput. Sci.*, Sept. 1977, pp. 443-450.
M. J. Shamos, "Problems in computational geometry," Dep. Comput. Sci., Yale Univ., New Haven, CT, May 1975.
M. I. Shamos and D. Hoey, "Closest-point problems," in *Proc. 16th Annu. Symp. Foundations of Comput. Sci.*, Oct. 1975, pp. 151-162.

D. T. Lee (S'76-M'78) received the B.S. degree in electrical engineering from the National Taiwan University in 1971 and the M.S. and Ph.D. degrees in computer science from University of Illinois at Urbana-Champaign in 1976 and 1978, respectively.

He joined the Department of Electrical Engineering and Computer Science at Northwestern University, Evanston, IL, in September 1978, where he is currently an Associate Professor. He spent two summers (1977 and 1979) as a Researcher at IBM T. J. Watson Research Center, Yorktown Heights, NY, and was a Consultant for General Electric Company, Space Division, Daytona Beach, FL, in 1977 and 1979. His research interests include design and analysis of algorithms in computational geometry and VLSI systems, data structures, discrete mathematics for computation, and computer graphics.

Dr. Lee is a member of the Association for Computing Machinery.

Rule-Based Learning for More Accurate ECG Analysis

KENNETH P. BIRMAN

Abstract—Long-term electrocardiograms exhibit a small number of *QRS morphologies* (waveform shapes) whose analysis can reveal cardiac abnormalities. We considered the problem of accurately identifying instances of each in 24-h ECG recordings.

A new learning algorithm was developed. Each *QRS* morphology is represented as a tree of *rule activations*, which associate attribute measurements with a rule. Each rule has a *syntactic pattern* together with a *semantic procedure* which manages and applies the knowledge stored in the activation. A single rule may be activated several times to learn different waveform segments. *Delineation refinement* improves each hypothesized signal interpretation. A simple *conflict resolution* mechanism resolves conflicting interpretations into a single unambiguous one. Comparison of the system with an existing program confirmed the promise of the new approach.

Index Terms—Electrocardiogram (ECG) analysis, interactive signal processing, knowledge representation, learning, pattern recognition languages (SEEK), rule-based learning, syntax directed inference.

Manuscript received June 19, 1981; revised February 3, 1982. This work was supported in part by an ROLM Corporation fellowship.

The author was with the Computer Science Division, Department of Electrical Engineering and Computer Science, University of California, Berkeley, CA 94720. He is now at 100 Wellington Avenue, New Rochelle, NY 10804.

I. Introduction

We BEGIN by summarizing the structure of the paper. Sections I and II review the problem and then present our rule-based knowledge representation technique. Interactive secondary analysis strategies are covered in Section III. Section IV presents other contemporary algorithms, and Section V compares our work with these alternative approaches. Section VI presents a simple *R*-wave recognition program coded in the SEEK pattern recognition language, which we designed for concise specification of rule-based algorithms. Section VII concludes the paper and outlines problems for future study.

A. The Recorded Electrocardiogram

Each time the heart beats, a *QRS waveform* is induced on the electrocardiogram. Usually, there are three major waves present: the *P*-wave, corresponding to the contraction of the atria (upper chambers of the heart), the *R*-wave, corresponding to contraction of the ventricles (lower chambers of the heart), and the *T*-wave, corresponding to the repolarization of the ventricles. The *R*-wave is said to begin with the *Q*-point and to

0162-8828/82/0700-0369$00.75 © 1982 IEEE