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 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 too difficult 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 $a, b$ consists of two endpoints $a$ and $b$, and a straight-line portion which is denoted by...
(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 \( \overline{a, b} \).

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

**Definition 3:** The distance \( d(q, a, b) \) between a point \( q \) and a closed segment \( \overline{a, b} \) in the Euclidean metric is the distance \( d(q, x) \), \( x = p(q, a, b) \), between \( q \) and its projection onto \( \overline{a, b} \) if \( x \) belongs to \( \overline{a, b} \) and is \( \min(d(q, a), d(q, b)) \) 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 \( I(q, a, b) \) of \( q \) on \( a, b \).

**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 \( d(X, Y) \) of a point \( q \) and a set of elements \( X \) is defined to be \( \min_{e \in X} d(q, e) \), i.e., \( B(X, Y) = \{ q | \min_{e \in X} d(q, e) = \min_{e \in Y} d(q, e) \} \). 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 \( B(q, a, b) \) 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 \( \overline{a, b} \), respectively. Fig. 2 shows the bisector \( B(a, b, c, d) \) 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 \( \overline{a, b} \) and \( \overline{c, d} \). Note that the bisector \( B(a, a, b) \), 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 \( e_j \) is the locus of points closer to \( e_j \) than to \( e_i \), i.e., \( V(e_j) = \{ x | d(x, e_j) < d(x, e_i) \} \).

**Definition 6:** Given \( n \) elements, \( e_1, e_2, \ldots, e_n \), 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., \( V(e_i) = \cap_{j \neq i} V(e_j) \). The boundary edges of \( V(e_i) \) are not necessarily straight-line segments 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 \( S = \{ e_1, e_2, \ldots, e_n \} \).

**Definition 7:** A simple polygon \( G \) of \( n \) vertices, denoted \( G = (q_0, q_1, \ldots, q_{n-1}) \), is a sequence of \( n \) points such that \( \overline{q_i q_{i+1}} \) is a closed segment for \( i = 0, 1, \ldots, n - 1 \), 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 \( G = (q_0, q_1, \ldots, q_{n-1}) \), a vertex \( q_i \) is called convex if the internal angle at \( q_i \) is less than \( \pi \) and is called reflex otherwise. We assume without loss of generality that no internal angle has a measure \( \pi \), i.e., no three consecutive vertices are collinear.

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

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, \( q_1, q_1, \ldots, q_m \). In the following we shall represent \( G \) by \( (e_1, e_2, \ldots, e_{i_1}, e_{i_1}, e_{i_2}, \ldots, e_{i_2}, \ldots, e_{i_m}, e_{i_m}) \) which is a sequence of 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 \( G = (e_0, q_1, e_2, \ldots, e_{n-1}) \). 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 \( S = \{ e_1, q_1, e_2, \ldots, e_{i_1}, q_{i_1}, e_{i_2}, \ldots, e_{i_2}, \ldots, e_{i_m}, q_{i_m}, e_{i_m}, \ldots, e_{i_m} \} \). 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 \n
\(^1\)Index additions and subtractions are taken modulo \( n \).
Fig. 3. Voronoi diagram of a simple polygon.

Fig. 4. medial axis of a simple polygon.

Fig. 5. Illustration of Lemmas 1 and 2.

my point \( z \) in \( V(e_i) \), the line segment \( z, t \), 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 \( uu' \) and \( vv' \) 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 \( u', v' \) 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)].

Corollary 2: Let \( z \) be any point on a Voronoi edge \( B(e_i, e_j) \) 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 \( z \in B(e_i, e_j) \), \( I(z, e_i) \) 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 \( B(e_i, e_j) \) 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 \( e_1, e_2, \ldots, e_N \) 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 \( G_1 = (e_1, e_2,\ldots, e_{\lfloor N/2 \rfloor}) \) and \( G_2 = (e_{\lfloor N/2 \rfloor + 1}, \ldots, e_N) \) and recursively construct the Voronoi diagrams \( \text{vog}(G_1) \) and \( \text{vog}(G_2) \). Then we merge \( \text{vog}(G_1) \) and \( \text{vog}(G_2) \) to form
the diagram \( \text{VOD}(G) \). As we shall see later, since the merge process takes \( O(N) \) time, the overall running time is \( O(N \log N) \).

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 \( G_1 = (e_1, e_2, \ldots, e_{10}) \) and of \( G_2 = (e_{11}, e_{12}, \ldots, e_{21}) \) where \( (e_1, e_2, \ldots, e_{21}) = G \) is a polygon shown in Fig. 3. For implementation purposes we instead partition the list of elements of \( G \) into chains \( C_1, C_2, \ldots, C_n \) (cf. Definition 9) and apply the divide-and-conquer technique to \( S = \{C_1, C_2, \ldots, C_n\} \). 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., \( O(N \log N) \). 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 \( C_4 = e_4, e_5, \ldots, e_{10} \). The Voronoi diagram of \( C_4 \) consists of six (6) Voronoi edges \( B(e_4, e_5), B(e_5, e_6), B(e_6, e_7), B(e_7, e_8), B(e_8, e_9), \) and \( B(e_9, e_{10}) \). 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 \( S = \{C_1, C_2, \ldots, C_n\} \) of chains, recursively construct \( \text{VOD}(S_1) \) and \( \text{VOD}(S_2) \), where \( S_1 = \{C_1, C_2, \ldots, C_{[n/2]}\} \) and \( S_2 = \{C_{[n/2] + 1}, \ldots, C_n\} \), and then merge \( \text{VOD}(S_1) \) and \( \text{VOD}(S_2) \) together to form \( \text{VOD}(S) \).

Let us now assume that \( \text{VOD}(S_1) \) and \( \text{VOD}(S_2) \) are available and describe how these two diagrams can be merged in \( O(N) \) time. Suppose that \( S_1 \) contains the elements \( e_1, e_2, \ldots, e_i \) and \( S_2 \) contains \( e_{i+1}, e_{i+2}, \ldots, e_N \), where \( e_i \) and \( e_{i+1} \) are, respectively, the first and last elements of \( C_1 \) and \( C_{[n/2]} \) and \( e_{i+2} \) and \( e_N \) are, respectively, the first and last elements of \( C_{[n/2] + 1} \) and \( C_n \). To merge \( \text{VOD}(S_1) \) and \( \text{VOD}(S_2) \) 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 \( e_i \) and \( e_{i+1} \) share a vertex \( q \) which is convex, the bisector \( B(e_i, e_{i+1}) \) which is an angle bisector is a component bisector of \( M \) and will be our starting bisector. Similarly, the vertex shared by \( e_i \) and \( e_N \) is convex and the angle bisector \( B(e_i, e_N) \) which is also a component bisector of \( M \) will be our terminating bisector, i.e., when \( B(e_i, e_N) \) 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 \( B(e_i, e_{i+1}) \) we shall scan the edges of the Voronoi polygon \( V(e_i) \) in counterclockwise (CCW) direction to find the edge \( B(e_i, e_j) \) which intersects \( B(e_i, e_{i+1}) \) and similarly scan the edges of the Voronoi polygon \( V(e_{i+1}) \) in a clockwise (CW) direction to find the edge \( B(e_{i+1}, e_j) \) which intersects \( B(e_i, e_{i+1}) \). With an additional comparison we can determine between the two edges \( B(e_i, e_j) \) and \( B(e_{i+1}, e_j) \) which intersects \( B(e_i, e_{i+1}) \) first. If \( B(e_i, e_j) \) intersects \( B(e_i, e_{i+1}) \) before \( B(e_{i+1}, e_j) \), then the next bisector to consider is \( B(e_i, e_{i+1}) \), otherwise it will be \( B(e_{i+1}, e_j) \). 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 \( e_v \) 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 \( e_u \) 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 \( B(e_u, e_v) \) of \( V(e_u) \) before the edge \( B(e_{i+1}, e_j) \) of \( V(e_v) \), then the next triplet to consider is \( (B(e_u, e_v), V(e_u), V(e_v)) \) otherwise the triplet is \( (B(e_{i+1}, e_j), V(e_u), V(e_v)) \). 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 the
Fig. 7. Merge of two Voronoi diagrams.

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

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 \( q_1, q_2, \ldots, q_n \), each represented by its \( x \)- and \( y \)-coordinates of a simple polygon and partitions the set of vertices into \( h \) chains \( C_1, C_2, \ldots, C_h \). Then the Voronoi diagram for each chain, \( \text{vod}(C_i) \), \( i = 1, 2, \ldots, h \), is computed. Next these \( h \) Voronoi diagrams are merged two at a time to form \( \text{vod}(C_1 \cup C_2), \text{vod}(C_3 \cup C_4), \ldots \), 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 \( \log_2 h \) and each internal node \( I \) represents work for merging of the two Voronoi diagrams associated with its two sons and is \( O(t_I + t_r) \), where \( t_I \) 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 \( [\log_2 h] \ast \Omega(N) \) which is \( O(N \log N) \).

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 \( S_1 = \{ e_i, e_{i+1}, \ldots, e_j \} \) and \( S_2 = \{ e_{j+1}, e_{j+2}, \ldots, e_k \} \), 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 \( e_u, e_v \). 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 \( q_k \) which is an endpoint of \( e_k \).

Figs. 9–11 are the output of the program run under Fortran with PLOT-10.
Fig. 9. Computer-plotted example I.

> OK

Fig. 10. Computer-plotted example II.

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

Rule-Based Learning for More Accurate ECG Analysis

KENNETH P. BIRMAN

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...