Discrete Comput Geom 21:405–420 (1999)

Discrete & Computational
Geometry

Finding the Medial Axis of a Simple Polygon in Linear Time*

F. Chin,[1] J. Snoeyink,[2] and C. A. Wang[3]

[1]Department of Computer Science,
University of Hong Kong,
Hong Kong
chin@csd.hku.hk

[2]Department of Computer Science, University of British Columbia,
Vancouver, British Columbia, Canada V6T 1Z4
snoeyink@cs.ubc.ca

[3]Department of Computer Science, Memorial University of Newfoundland,
St. John's, Newfoundland, Canada A1C 5S7
wang@garfield.cs.mun.ca

Abstract. We give a linear-time algorithm for computing the medial axis of a simple polygon P. This answers a long-standing open question—previously, the best deterministic algorithm ran in $O(n \log n)$ time. We decompose P into pseudonormal histograms, then influence histograms, then xy monotone histograms. We can compute the medial axes for xy monotone histograms and merge to obtain the medial axis for P.

1. Introduction

The *medial axis* of a simple plane polygon P goes by many names, including *symmetric axis* or *skeleton*. One of the more picturesque is the *grassfire transform*: Imagine igniting all boundary points of P. If the flame burns inward at a uniform rate, then the *quench points* where the flame meets and extinguishes itself define the medial axis. Equivalently, the medial axis is the locus of all centers of circles inside P that touch the boundary of P in two or more points.

The medial axis was proposed and named by Blum [3] in a 1967 article entitled, "A transformation for extracting new descriptors of shape." The pattern recognition literature uses it heavily as a one-dimensional structure that represents two-dimensional shape [4],

* The second author was partially supported by an NSERC grant and a BC ASI Fellowship. The third author's work was partially supported by NSERC Grant OPG0041629.

[8], [18], [22], [24]; it has also been used in solid modeling [27], mesh generation [10], pocket machining [11], etc.

To a computational geometer, the medial axis of an n-gon P is a Voronoi diagram [2], [21] whose sites are the open edges and the vertices of the boundary. In 1982 Lee [17] developed an $O(n \log n)$ algorithm to compute the medial axis. (Yap's [29] or others' [16], [20] Voronoi diagram algorithms for line segments can also be used, although implementing a general algorithm to handle the degenerate cases of segments with the same endpoints is a headache.)

Since that time, it has been an open question to determine the time required to compute the medial axis. Up to 1995 there were two significant milestones: In 1987 Aggarwal et al. [1] first published an algorithm that can compute the medial axis of a *convex* polygon P in linear time. Their algorithm computes the Voronoi diagram of the vertices of a convex polygon in $O(n)$ time; the medial axis are obtained by erecting planes in space through the edges of P at 45° angles to the plane of P and using the dual algorithm to compute the intersection of the half-spaces that contain P. In 1991 Devillers [6] first published a *randomized* algorithm for the medial axis that runs in $O(n \log^* n)$ expected time and uses Seidel's acceleration technique [25].

Decompositions of a simple polygon P into histograms have been profitably applied to the *constrained Voronoi diagram*: the Voronoi diagram of the vertices of P where distance is measured along a shortest path inside P. (This is the problem that Aggarwal et al. [1] solve in linear time for convex polygons.) Klein and Lingas [14] extended histogram partitions from orthogonal polygons to simple polygons. They developed a randomized algorithm for the constrained Voronoi diagram of a histogram and, by merging diagrams for histograms, computed the constrained Voronoi diagram of P in $O(n)$ expected time. Wang and Chin [28] established a deterministic linear-time algorithm for the constrained Voronoi diagram by further decomposing histograms.

We extend Wang and Chin's decomposition to compute the medial axis of a simple polygon P. Our algorithm decomposes P into normal histograms, then into influence histograms, and xy monotone histograms. It computes the Voronoi diagrams of xy monotone histograms, and merges to obtain the medial axis of P. After reviewing definitions and known results about Voronoi diagrams and histogram decompositions in Section 2, we describe the new steps in reverse order: In Section 3 we extend the algorithm of Aggarwal et al. [1] to compute the Voronoi diagram of selected edges and vertices of an xy-monotone histogram and to extend Voronoi diagrams. In the process, we simplify part of the analysis and note that a randomized incremental construction may be simpler to implement. In Section 4 we compute the Voronoi diagram of a histogram by decomposing it into influence histograms and xy monotone histograms. We conclude in Section 5 by mentioning some extensions and applications.

Klein and Lingas have recently extended their work on histogram decompositions to obtain the medial axis of P in *expected* linear time [15]. Their algorithm adds edges to close off all histogram polygons and applies randomization twice: once to compute the medial axis of all edges of a histogram polygon, and again when non-P edges are removed and the medial axes are merged to obtain the medial axis of P. The second could be replaced by a deterministic step along the lines of Section 3, but it is not clear what to do about the first, which is strongly predicated on the fact that all edges affect the medial axis. Addition and deletion of edges also adds to the programming complexity.

2. Preliminaries

Let P be a simple polygon with n vertices, $\{p_1, p_2, \ldots, p_n\}$. The boundary $\partial(P)$ consists of these vertices and the edges (open line segments) between consecutive vertices. We assume that the vertices and edges of P are in general position: no two vertices on the same horizontal or vertical line, no two edges parallel, no three vertices colinear, and no set of four vertices or edges cocircular (i.e., no circle through i vertices is tangent to more than $3 - i$ edges). General position can be simulated by (actual or conceptual) perturbation of the input [9].

In this section we define the medial axis of P and its connection to the Voronoi diagram of the vertices and edges of P. We also state a lemma on merging. Almost all deterministic algorithms for medial axis and Voronoi diagram computation are merge based. We also define pseudonormal histograms [14] and review how to decompose polygon P into pseudonormal histograms (PNHs) and how to merge medial axes of PNHs to obtain the medial axis of P.°

2.1. *The Medial Axis and Voronoi Diagram of P*

The *Voronoi diagram* [2], [21] of a set of *sites* is the partition of the plane into connected regions having the same set of closest sites. This partition consists of *Voronoi cells*, which are the regions with one closest site, *Voronoi edges*, which have two closest sites, and *Voronoi vertices*, which have three or more closest sites—these are illustrated in Fig. 1.

If we select the vertices and edges (open line segments) of polygon P to be the sites, then our general position assumption ensures that Voronoi vertices are defined by three sites. Voronoi edges become segments of straight lines or parabolas that are bisectors of two sites—straight lines when the two closest sites are two points, two open segments, or an open segment and one of its incident vertices, and parabolas when the two closest sites are an open segment and a nonincident vertex.

The *medial axis* of P is the locus of all centers of circles contained in P that touch $\partial(P)$ in two or more points. Thus, the medial axis consists of Voronoi vertices and Voronoi edges. The only Voronoi edges that are not part of the medial axis are the bisectors of an edge and an incident vertex (which is the perpendicular to the segment through the vertex.) We, therefore, concentrate on computing the Voronoi diagram $V(P)$ in this paper, and obtain the medial axis by removing these Voronoi edges.

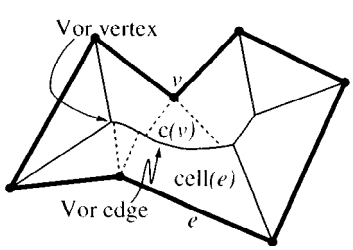

Fig. 1. Medial axis (solid) and Voronoi diagram (solid and dotted).

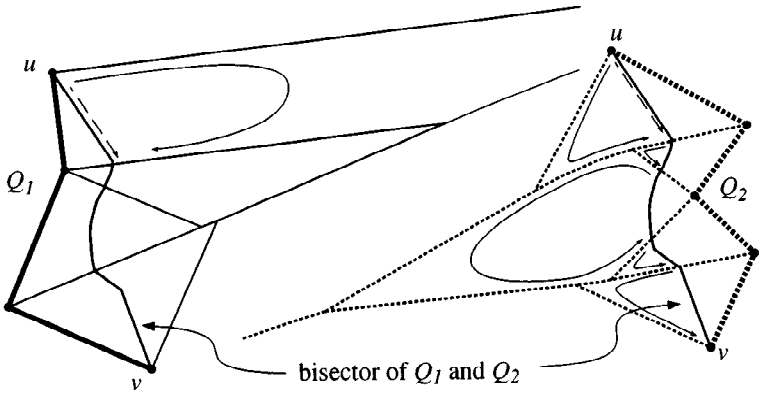

Fig. 2. Merging Voronoi diagrams.

The *constrained Voronoi diagram* of a set of sites on the boundary of P is the Voronoi diagram in which distance is measured along a shortest path inside P. All of our Voronoi diagrams should be considered constrained Voronoi diagrams, even though we typically omit the word "constrained." In our algorithms, the reflex vertices are always sites, so the shortest paths are line segments.

Algorithms to merge Voronoi diagrams have been important since Shamos and Hoey's divide and conquer algorithm [26], [23]. See also [12], [17], or [13].

Lemma 2.1. *Let Q be a polygon that is divided into Q_1 and Q_2 by a diagonal $\overline{uv}$. Let subsets of vertices and edges S_1, S_2, and $S = S_1 \cup S_2$ be the sites in Q_1, Q_2, and Q, respectively. Given the Voronoi diagrams of S_1 in Q_1 and S_2 in Q_2, the Voronoi diagram of S in Q can be obtained in time proportional to the number of Voronoi edges that intersect $\overline{uv}$ and the number of new edges added.*

Proof. We sketch a proof; refer to references [12], [13], [17], [23], and [26] for more details. Figure 2 illustrates polygonal chains Q_1 and Q_2 formed by dividing the polygon in Fig. 1 along a diagonal $\overline{uv}$. Assume that the Voronoi diagram of Q_i is extended across the segment $\overline{uv}$ to a new half-space (which may be considered to lie on a separate sheet from Q_i if Q_i also extends across the line $\overline{uv}$). We can use the routine of Theorem 3.6 to compute the extension, if necessary, in time proportional to the number of Voronoi edges that intersect $\overline{uv}$.

The bisector between sites in Q_1 and sites in Q_2 is a curve from u to v. We can trace this curve starting with the line segment that bisects the edges of Q incident to u, if u is a convex corner of Q as in Fig. 2, or by tracing the perpendicular to an edge at u, if u is a reflex corner.

The bisector follows line segments and parabolas, changing whenever it leaves a cell of the Voronoi diagram of Q_1 or of Q_2. We can determine when it leaves by walking the portions of the cell boundaries that will be discarded, as indicated in Fig. 2. These discarded portions are forests (trees, if we include a vertex at infinity) whose complexity

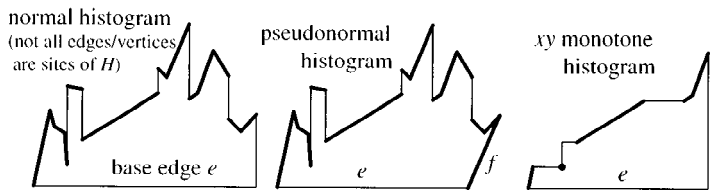

Fig. 3. Histograms.

is proportional to the number of their leaves, which is the number of new Voronoi edges added in the bisector. □

2.2. *Histograms*

A *normal histogram* (NH) is a simple polygon H whose boundary consists of a *base edge* e and a chain that is monotone with respect to e—that is, any line perpendicular to e that intersects $\partial(H)$ intersects e and at most one other point or segment. Typically, we rotate NHs so that the base is along the x axis and the rest of the polygon is in the positive quadrant, as in Fig. 3. The *base line* is the line through e. When computing Voronoi diagrams of histograms, we also compute the constrained Voronoi diagram in the half-space below the base line when e is not a site.

A *pseudonormal histogram* (PNH), defined by Klein and Lingas [14], can be viewed as a NH with a missing corner. That is, a PNH can be turned into an NH by replacing an edge f incident to the base with a segment perpendicular to the base and an extension of the base. Because of the merge lemma, an NH is as good as a PNH with respect to computing a Voronoi diagram.

Corollary 2.2. *The constrained Voronoi diagram of selected sites of an n-vertex PNH can be obtained from the diagram of the corresponding NH in $O(n)$ time.*

Proof. The removed edge f can be reintroduced into the PNH as a diagonal by walking through Voronoi cells in $O(n)$ time. Then, if f is a site in the PNH, the polygon containing only f can be merged to the rest of the PNH according to Lemma 2.1. □

An NH is an *xy-monotone histogram* if, after putting the base along the x axis, the y coordinates of nonbase vertices are monotone increasing or monotone decreasing.

2.3. *Decomposing P into Pseudonormal Histograms*

Klein and Lingas' algorithm for the constrained Voronoi diagram [14] is based on decomposing a polygon P into PNHs, computing their Voronoi diagrams (via the corresponding NHs), and then merging. For completeness, we briefly sketch their decomposition and note that only two calls to a linear-time trapezoidation algorithm are needed.

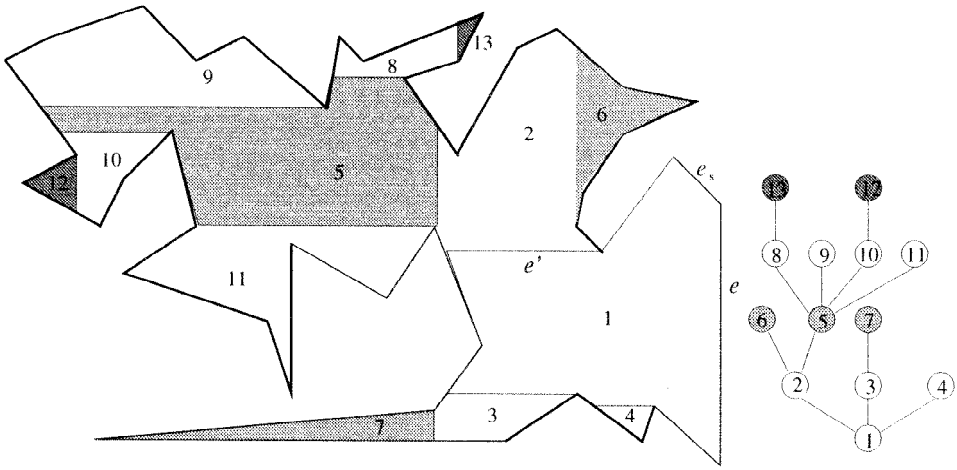

Fig. 4. A decomposition of P into a tree of PNHs.

Figure 4 shows a polygon P decomposed into 13 PNHs. PNH_e is associated with the vertical base e missing its upper corner; $\text{PNH}_{e'}$ with the horizontal base e' is missing its left corner, etc. The decomposition can be represented as a rooted tree whose nodes are PNHs and whose edges represent adjacency of two PNHs. The base edge of each child PNH is a trapezoid edge of the parent PNH.

Lemma 2.3. *An n-vertex simple polygon P can be decomposed, in linear time, into a set of PNHs having a linear number of vertices.*

Proof. The decomposition starts with an arbitrary edge e of P as the base of the first PNH, PNH_e. Make this edge horizontal, then use Chazelle's algorithm [5] to compute horizontal and vertical trapezoidations of P, denoted $\mathcal{H}$ and $\mathcal{V}$, in $O(n)$ time. Store trapezoids with links to adjacent neighbors.

Now, for the horizontal base edge e, form PNH_e from all vertical trapezoids in $\mathcal{V}$ that are incident on e or e_s, where e_s is one of the two edges (if any) incident to e at an interior angle between 90° and 180°. Remove PNH_e from P; boundary edges of PNH_e that are not edges of P will be base edges of PNHs in the next level of the decomposition, after we repair the horizontal trapezoidation $\mathcal{H}$ both inside and outside of PNH_e.

We can trace the boundary $\partial(\text{PNH}_e)$ through $\mathcal{H}$ in time proportional to the number of trapezoids intersected—each trapezoid edge is cut at most twice—and split trapezoids that are cut by vertical boundary segments. We can then remove horizontal trapezoidation edges that no longer contain a vertex and merge adjacent trapezoids. All intersected trapezoids will become part of a PNH at the next level.

The motivation for PNHs instead of NHs is that every PNH defined in P uses a vertex along its base. Since a vertex is in at most two PNHs, the construction terminates. Also, the work per trapezoid is constant, so the total amount of work performed is $O(n)$. □

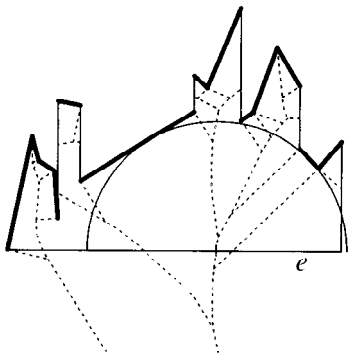

Fig. 5. The constrained Voronoi of selected sites.

Klein and Lingas proved that the Voronoi diagrams of vertices of two PNHs do not interfere unless these two PNHs (i) are parent and child, or (ii) are children of the same parent that face each other [14]. Their proof extends easily to edge sites.

Lemma 2.4 [14, Theorem 4.6]. *Given the Voronoi diagrams for the PNHs in a decomposition of an n-vertex simple polygon P, the Voronoi diagram for P can be computed in linear time.*

Proof (*Sketch*). The children of a PNH can be separated into left and right groups. The Voronoi diagram of the PNH is first merged with the Voronoi diagrams of all its left children and then with those of its right. This merging of Voronoi diagrams can be done in time linearly proportional to the total size of the PNH and all its children. □

The task that remains is depicted in Fig. 5: to compute the constrained Voronoi diagram of selected vertex and edge sites inside a normal histogram and in the half-space across its base edge. The latter we do in the next section; the former requires us to compute the Voronoi diagram of an xy monotone histogram.

3. Computing the Voronoi of an *xy* Monotone Histogram and the Voronoi Extension

In this section we provide efficient subroutines for two Voronoi diagram problems. See Fig. 6.

Voronoi of *xy* monotone histogram. Compute the constrained Voronoi diagram of selected sites in an xy monotone histogram H.

Voronoi extension. Given the intersection of the Voronoi diagram of the sites in a histogram H with the base edge e, compute the Voronoi diagram in the half-space below the base line.

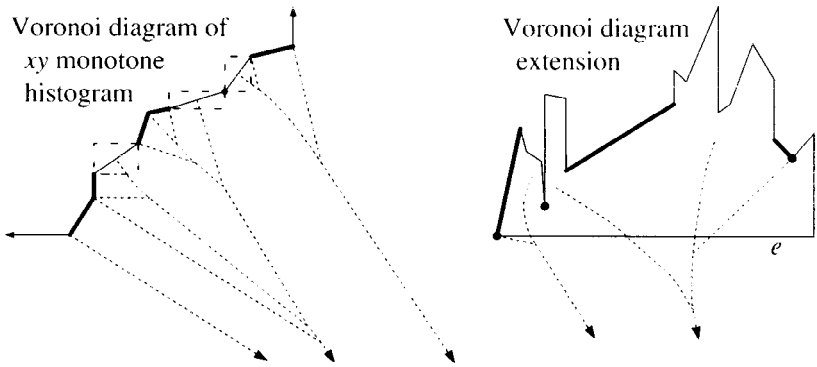

Fig. 6. In both problems, the Voronoi diagrams are trees with known leaves.

In both problems the sites have an order given by a curve and the intersection of the Voronoi diagram with the curve is known. In this paper we focus on the xy monotone histogram H, where the curve is the monotone portion of the boundary of H. We direct initial and final rays to $x = -\infty$ and $y = \infty$ as shown in Fig. 6.

Monotonicity in both x and y implies that the boundary of H does not leave the bounding box of the diagonals that join consecutive sites in x coordinate order. Inside such a box, the Voronoi diagram is determined by the endpoints of the diagonal. We can replace portions of the boundary that are not sites with these diagonals. To avoid a degenerate case, if an edge and its lower endpoint are both sites, we consider them as a unit.

To make sure the problems are clear, Section 3.1 sketches a randomized incremental construction that can solve our two Voronoi diagram problems in expected linear time. Section 3.2 gives deterministic algorithms by adapting the work of Aggarwal et al. [1].

3.1. *Randomized Incremental Construction*

The randomized algorithm for these problems is simpler to implement than the deterministic. The "tracing" in this lemma is standard [6], [13], [16] and is essentially the computation of the merge Lemma 2.1; finding where to start tracing in constant time gives us an algorithm with expected linear running time. Klein and Lingas [15] have independently given another.

Lemma 3.1. *The Voronoi diagram of an xy monotone histogram and the Voronoi diagram below the base can both be computed by a randomized incremental construction in expected time that is linear in the number of sites.*

Proof. Each problem involves sites with a given order along a simple curve; we order subsets of sites consistent with the curve order.

Randomly delete the sites one at a time, storing the predecessor and successor of a site when it is deleted. Then insert the sites according to the reverse permutation and compute the portion of the Voronoi cell of each new site as it is added.

The new site will be a nearest neighbor to some point on the segment joining its predecessor and successor. One can, therefore, compute the new cell by tracing the portion of the Voronoi diagram to be deleted, starting from the Voronoi edge that crosses this segment. The Voronoi edges deleted form a forest, whose complexity is proportional to the Voronoi edges that bound the new Voronoi cell. Because the Voronoi cell for a randomly chosen site has constant complexity, the expected time to trace all cells is proportional to the number of sites. □

3.2. *Deterministic Algorithms*

Djidjev and Lingas [7] pointed out that the algorithm of Aggarwal et al. [1] could find the Voronoi diagram of the vertices of an xy monotone histogram. We show that the algorithm can be adapted for segment sites, and simplify part of the analysis. The basic idea of Aggarwal et al. is to identify a constant fraction of the Voronoi cells that are not adjacent to each other, remove their corresponding sites, recursively compute the Voronoi diagram of the remaining sites, and then independently merge in the nonadjacent cells. Identifying nonadjacent cells is complicated by the fact that one does not have the cell descriptions until the algorithm has completed its task.

Let $s_1, s_2, \ldots, s_k$ be the list of sites in order along the curve. (Include sites at infinity as s_0 and s_{k+1}.) We mark sites red and blue to satisfy three rules:

1. No two adjacent sites are marked *red*.
2. No three adjacent sites are marked *blue*.
3. If the portion of the boundary below five sites ($s_{i-2}, s_{i-1}, s_i, s_{i+1}, s_{i+2}$) has a circle below that touches s_{i-1} and s_{i+1} and does not contain any point of s_{i-2}, s_i, or s_{i+2}, then site s_i must be *red*.

Figure 7 shows two histograms with possible markings. Recall that if an edge and its lower endpoint are both sites, we consider them as a unit—otherwise symbolic perturbation would be needed so that two open edge sites with the same (nonsite) endpoint could not be cocircular with two other sites. At the left of Fig. 7, marks on the concave chain are constrained only by rules 1 and 2; at the right, the reds are forced by empty circles.

Lemma 3.2. *A marking of sites can be computed in linear time.*

Proof. The first and third rules do not conflict because the third cannot apply to two adjacent sites. Such a configuration would require two circles that were centered inside H, one touching s_{i-1} and s_{i+1} and the other touching s_i and s_{i+2}, but this would imply that all four sites were cocircular or that two circles intersected in four points.

Thus, we can mark sites in linear time by initializing all sites to blue, then marking the sites that rule 3 says must be red. For each sequence of $i > 2$ blues that remain, we mark every other site red, starting with the second and ending one or two before the last. □

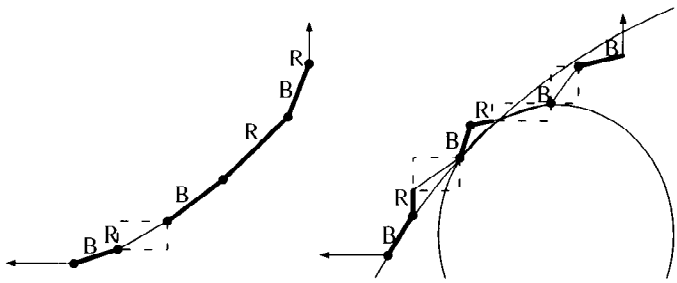

Fig. 7. Possible markings.

This coloring has the following independence property:

Lemma 3.3. *Consecutive red sites cannot have Voronoi cells that are adjacent.*

Proof. Consider two sites s and s' that have adjacent Voronoi cells. There is a circle in H that touches s and s' and excludes all other sites.

If s and s' have a single site t between them, then by rule 3, t is marked red and by rule 1, both s and s' are blue. On the other hand, if two sites t and t' lie between s and s', then move the circle center along the bisector of s and s' toward t and t', as in Fig. 8. The new circle exits the old only between s and s'. Therefore, by monotonicity, the circle encounters t or t' before encountering any other site. Suppose that it encounters t', then t is red by rule 3 and s must be blue. Therefore, either s and s' are not both red or they are not consecutive. □

These two lemmas are sufficient to use the algorithm of Aggarwal et al. [1], as we briefly describe. They prove the following combinatorial lemma.

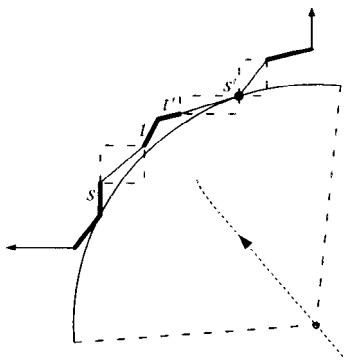

Fig. 8. s or s' is blue.

Lemma 3.4 [1]. *Let T be a binary tree embedded in the plane. Each leaf of T has an associated "neighborhood," which is a connected subtree rooted at that leaf, and leaves adjacent in the topological order around the tree have disjoint neighborhoods. Then there are a fixed fraction of the leaves with disjoint, constant-size neighborhoods, and such leaves can be found in linear time* (*assuming that neighborhoods can be traced out in breadth-first order*).

The rest of the proof follows Aggarwal et al. [1].

Theorem 3.5. *The Voronoi diagram of an xy monotone histogram can be computed in linear time.*

Proof. By rule 2, a constant fraction of the sites are marked blue. We compute their Voronoi diagram recursively and let T be the tree of Voronoi edges that start between blues sites that are separated by reds (i.e., eliminate edges that are between adjacent sites that are both marked blue). Now, the "neighborhood" of a leaf is the portion of the Voronoi edge that is farther from the blue sites that define it than from the red site that is being inserted. Lemma 3.3 says that adjacent neighborhoods are disjoint, so Lemma 3.4 says that a constant fraction of the red sites with disjoint, constant-size neighborhoods can be found. These red sites can be merged into the blue diagram in constant time apiece.

Finally, a constant fraction of the sites remain red; compute their Voronoi diagram recursively and merge it into the blue Voronoi diagram—we can do this in linear total time if we merge connected portions starting and ending on the histogram boundary. □

We can deterministically compute the Voronoi diagram extension by a similar algorithm.

Theorem 3.6. *Given the intersection Voronoi diagram of a histogram with the base edge, the extension below the base can be computed in linear time.*

Proof. Let $s_1, s_2, \ldots, s_k$ be the sites whose Voronoi cells intersect the base edge in the order of intersection. Mark the sites to satisfy rules 1, 2, and a modified rule 3: if a circle centered below the base line touches s_{i-1} and s_{i+1} and does not contain any point of s_{i-2}, s_i, or s_{i+2}, then site s_i must be *red*.

We can establish analogues of Lemmas 3.2 and 3.3: Labeling can be accomplished in linear time using a constant time implementation of rule 3. In proving that consecutive red sites are independent, the difficult case is when an empty circle C, centered below the base line, touches sites s and s' that have two sites t and t' between them. As before, we can move the center of C along the the bisector of s and s', maintaining tangency with sites s and s' until we encounter t or t'. We must encounter some site, because the Voronoi cells of s and s' are not adjacent along the base line, and t and t' are the only candidates because the circle pokes out of C only between s and s'. This circle will certify that t or t' is red and thus s or s' is blue.

The rest of the computation continues as before. □

4. Finding the Voronoi Diagram of a Normal Histogram

The key property of the histogram decomposition of Klein and Lingas is that the influence of a site is limited—the Voronoi cell of site does not extend beyond parents, children, or siblings of histograms containing the site. Following Wang and Chin [28], we show that limiting the influence is also key to computing the Voronoi diagram of an NH.

Let H be an NH with horizontal base edge e. We can identify the sites of H whose influence (i.e., Voronoi cell) extends across base edge e by considering circles centered on e that are empty of sites.

Lemma 4.1. *In an NH H, the sites whose Voronoi cells extend below the base line are those that can be touched by a circle centered at the base edge and empty of other sites.*

Proof. Voronoi cells are simply connected, so a cell extends below the base edge iff it intersects the base edge. The empty circle certifies that such an intersection exists. □

We assume that H has been decomposed into a linear number of horizontal trapezoids, which is the result of our decomposition into PNHs or of running Chazelle's algorithm [5]. We call the horizontal segments introduced by trapezoidation *chords*. The dual graph of the trapezoidation of H is a tree rooted below the base edge whose edges correspond to chords, as illustrated in Fig. 9.

The *influence region* of H is the union of all circles centered at the base edge whose interiors do not intersect a site. The *influence histogram*, IH, consists of all horizontal trapezoids that intersect the influence region.[1] We find the IH by exploring the dual tree of the trapezoidation and maintaining stacks of sites whose Voronoi cells may intersect the base.

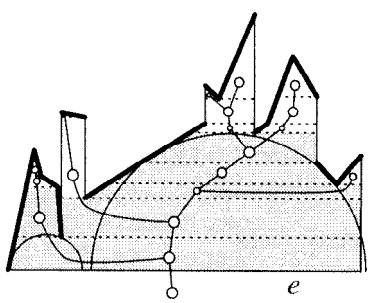

Fig. 9. Trapezoids in an IH.

[1] This influence region and IH are slightly different from Wang and Chin's corresponding ones for the constrained Voronoi diagram of vertices of P [28].

Lemma 4.2. *The IH of H can be computed in time linearly proportional to the number of its trapezoids.*

Proof. All circles in this algorithm will be centered on the base edge e. Any two such circles intersect in at most one point above the base line. Let C_s denote the largest circle that crosses a chord s and whose interior does not contain a site on or below s. If C_s exists, it either touches sites to the left and right of its center, or is centered at a (nonsite) endpoint of e, or is infinite. The first case corresponds to the intersection of e with the bisector of the two sites touched, as can be seen in Fig. 5.

We compute all such largest circles. Initialize empty stacks L and R, for left and right. If the left endpoint of the base edge is a site, insert it into L; similarly, insert the right endpoint into R. Let chord $s = e$.

Now, we maintain two invariants:

1. With respect to the histogram consisting of the trapezoids output so far, the stacks contain the sites whose Voronoi cells intersect e in order.
2. The largest circle C_s is determined by the sites at the tops of L and R.

Let Δ_s be the trapezoid above s and let t and u be the at most two other chords of Δ_s as illustrated in Figure 4. Include Δ_s in the output. If the segment of circle C_s above s is contained in Δ_s, then this branch of the IH is complete. If C_s crosses chord t but does not contain a site in Δ_s, then let $s = t$ and continue. Otherwise, we need to reestablish the invariants, as at the right of Fig. 10.

First, the new sites in Δ_s may be closer to e than sites in L or R—new Voronoi cells may crowd out old ones. If the largest circle determined by the next-to-top site in L and sites in Δ_s does not contain the top site in L, then pop the top of L. The first site to remain on the stack certifies that all sites beneath also remain, since stacked sites appear in the same order as their Voronoi cells intersect e. Handle R similarly.

Next, look at the new empty circles crossing chords t and u—if there are none, then the branch of the IH ends here. If only one chord is crossed, then we push the new sites whose Voronoi cells intersect e on the L and/or R stacks and continue. Otherwise, both chords are crossed, as in Fig. 10; some site, call it q, touches C_t on the right and C_u on the left. We continue building the IH across t using circle C_t and stack L and a new stack R' that contains site q alone. Building IH across u uses C_u, R, and a new stack L' containing only q. Site q will not be popped off these stacks because among the

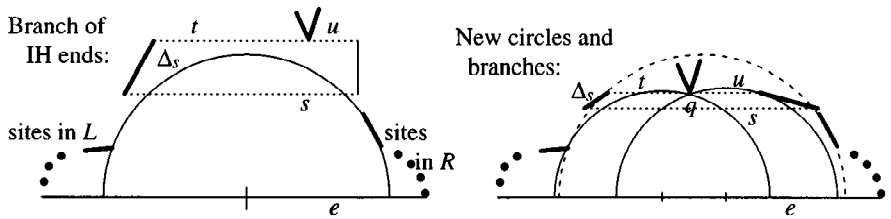

Fig. 10. The largest circle C_s and the trapezoid Δ_s.

empty circles between C_t and C_u there must be one that touches q directly above the center—this circle, tangent to t and u, certifies that the Voronoi cell of q in histogram H intersects the base edge e. Thus, the branches can be computed independently and the invariants can be maintained. □

The IH contains all sites whose Voronoi cells intersect the base edge. It may also contain other sites, but they can be grouped into xy monotone histograms.

Lemma 4.3. *The Voronoi diagram of an IH H can be computed in time linearly proportional to the number of vertices in H.*

Proof. Consider a trapezoid at which the IH branches, as at the right of Fig. 10. Let p be the site that forced the branch. We know that the perpedicular from p to the base edge e is contained in the Voronoi cell of site p—we can cut the IH along this segment.

After all cuts, we obtain a set of histograms; in each histogram an empty circle intersects all horizontal trapezoids. Thus, the histograms are *bitonic* and can be cut into two xy monotone histograms.

Finally, compute Voronoi diagrams of bitonic histograms by merging diagrams for xy monotone histograms in linear time. The Voronoi diagram of the IH is obtained from those of the bitonic histograms by simply rejoining along the vertical cuts. □

At last, we return to the problem of computing the Voronoi diagram of an NH H with a horizontal trapezoidation.

Theorem 4.4. *The Voronoi diagram of selected sites in an NH H can be computed in time linearly proportional to the number of vertices in H.*

Proof. Again, we assume that H is an NH with horizontal base edge e and that H has been decomposed into a linear number of horizontal trapezoids. We decompose H into a tree of IHs as follows. First, compute in IH I in H. Connected sets of trapezoids in H that are not in I form NHs; make I the parent of recursively computed subtrees of IHs for these NHs. Lemma 4.2 says that each IH takes time proportional to the number of its trapezoids, so the total time is linear in the size of H.

By Lemma 4.3 the Voronoi diagrams of all IHs in the tree can be computed in linear time. By the definition of IHs (see Lemma 4.1), the Voronoi cell of a site intersects only its own IHs (at most two) and that of its parent. Thus, the cost to merge Voronoi diagrams of IHs to form the Voronoi diagram of H is also linear. □

From this theorem, with the lemmas from Section 2.3, we obtain the final result.

Corollary 4.5. *The Voronoi diagram of the vertices and edges of a polygon P can be computed in time linearly proportional to the number of vertices in P.*

5. Conclusion

We have given an optimal linear-time algorithm for computing the Voronoi diagram or medial axis of a simple polygon. There are two complex steps: using Chazelle's algorithm [5] to compute two trapezoidations, and using the technique of Aggarwal et al. [1] to compute Voronoi diagrams of xy monotone histograms. Both steps can be replaced by simpler randomized algorithms, although the asymptotic running time of trapezoidation increases by a $\log^* n$ factor.

Several problems for simple polygons can be solved in linear time based on this result: computing the largest inscribed circle, building a query structure for the closest boundary point, and determining the buffer zone of all points within ε of a simple polygonal curve. This algorithm also applies to other L_p metrics and constant-complexity convex distance functions. If the unit ball is a convex polygon of m vertices that contains the origin, the necessary primitives can be implemented in $O(\log m)$ time apiece, giving an $O(n \log m)$-time medial axis algorithm.

One open question is whether the Voronoi diagram of k polygons with n vertices total can be computed in $O(n + k \log k)$ time. It can be when the polygons are convex [19].

Acknowledgments

We thank Otfried Schwartzkopf and David Kirkpatrick for discussions on medial axis algorithms, the anonymous referees, Bethany Chan, Siu-Wing Cheng, and Michael McAllister for their careful reading and comments on drafts of this paper.

References

1. A. Aggarwal, L. J. Guibas, J. Saxe, and P. W. Shor. A linear-time algorithm for computing the Voronoi diagram of a convex polygon. *Discrete Comput. Geom.*, 4:591–604, 1989.
2. F. Aurenhammer. Voronoi diagrams—a survey of a fundamental geometric data structure. *ACM Comput. Surveys*, 23(3):345–405, 1991.
3. H. Blum. A transformation for extracting new descriptors of shape. In W. Whaten-Dunn, editor, *Proc. Symp. Models for Perception of Speech and Visual Form*, pages 362–380. MIT Press, Cambridge, MA, 1967.
4. L. Calabi and W. E. Hartnett. Shape recognition, prairie fires, convex deficiences and skeletons. *Amer. Math. Monthly*, 75:335–342, 1968.
5. B. Chazelle. Triangulating a simple polygon in linear time. *Discrete Comput. Geom.*, 6:485–524, 1991.
6. O. Devillers. Randomization yields simple $O(n \log^* n)$ algorithms for difficult $\Omega(n)$ problems. *Internat. J. Comput. Geom. Appl.*, 2(1):97–111, 1992.
7. H. Djidjev and A. Lingas. On computing the Voronoi diagram for restricted planar figures. In *WADS '91: Second Workshop on Data Structures and Algorithms*, number 519 in LNCS, pages 54–64. Springer-Verlag, Berlin, 1991.
8. R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. Wiley, New York, 1973.
9. H. Edelsbrunner and E. P. Mücke. Simulation of simplicity: a technique to cope with degenerate cases in geometric algorithms. *ACM Trans. Graph.*, 9(1):66–104, 1990.
10. H. N. Gürsoy and N. M. Patrikalakis. An automatic coarse and fine surface mesh generation scheme based on medial axis transform: Part I, algorithm. *Engrg. Comput.*, 8:121–137, 1992.
11. M. Held. *On the Computational Geometry of Pocket Machining*. Number 500 in LNCS. Springer-Verlag, Berlin, 1991.

12. D. G. Kirkpatrick. Efficient computation of continuous skeletons. In *Proc*. 18*th FOCS*, pages 162–170, 1977.
13. R. Klein. *Concrete and Abstract Voronoi Diagrams*. Number 400 in LNCS. Springer-Verlag, Berlin, 1989.
14. R. Klein and A. Lingas. A linear-time randomized algorithm for the bounded Voronoi diagram of a simple polygon. In *Proc*. 9*th Ann*. *ACM Symp*. *Comp*. *Geom*., pages 124–132, 1993.
15. R. Klein and A. Lingas. Fast skeleton construction. In P. Spirakis, editor, *Algorithms–ESA*'95, number 979 in LNCS, pages 582–595. Springer-Verlag, Belin, 1995.
16. R. Klein, K. Mehlhorn, and S. Meiser. Randomized incremental construction of abstract Voronoi diagrams. *Comput*. *Geom*. *Theory Appl*., 3(3):157–184, 1993.
17. D. T. Lee. Medial axis transformation of a planar shape. *IEEE Trans*. *Pattern Anal*. *Mach*. *Intell*., 4(4):363–369, 1982.
18. S. Lu, H. Xu, and C. Wang. Detecting and eliminating false strokes in skeletons by geometric analysis. In *Vision Geometry*, *SPIE Proc*., Volume 1832, pages 312–323, 1993.
19. M. McAllister, D. Kirkpatrick, and J. Snoeyink. A compact piecewise-linear Voronoi diagram for convex sites in the plane. *Discrete Comput*. *Geom*., 15:73–105, 1996.
20. K. Mehlhorn, S. Meiser, and C. Ó'Dúnlaing. On the construction of abstract Voronoi diagrams. *Discrete Comput*. *Geom*., 6:211–224, 1991.
21. A. Okabe, B. Boots, and K. Sugihara. *Spatial Tessellations*: *Concepts and Applications of Voronoi Diagrams*. Wiley, New York, 1992.
22. T. Pavlidis. A review of algorithms for shape analysis. *Comput*. *Graph*. *Image Process*, 7:243–258, 1978.
23. F. P. Preparata and M. I. Shamos. *Computational Geometry—An Introduction*. Springer-Verlag, New York, 1985.
24. A. Rosenfeld. Axial representation of shape. *Comput*. *Vis*. *Graph*. *Image Process*, 33:156–173, 1986.
25. R. Seidel. A simple and fast incremental randomized algorithm for computing trapezoidal decompositions and for triangulating polygons. *Comput*. *Geom*. *Theory Appl*., 1:51–64, 1991.
26. M. I. Shamos and D. Hoey. Closest point problems. In *Proc*. 16*th FOCS*, pages 151–162, 1975.
27. P. Vermeer. Two-dimensional MAT to boundary conversion. In *Proc*. 2*nd Symp*. *Solid Model*. *Appl*., pages 493–494, 1993.
28. C. A. Wang and F. Chin. Finding the constrained Delaunay triangulation and constrained Voronoi diagram of a simple polygon in linear time. In P. Spirakis, editor, *Algorithms—ESA* '95, number 979 in LNCS, pages 280–294. Springer-Verlag, Berlin, 1995.
29. C. K. Yap. An $O(n \log n)$ algorithm for the Voronoi diagram of a set of simple curve segments. *Discrete Comput*. *Geom*., 2:365–393, 1987.

Received May 16, 1997, *and in revised form October* 30, 1997.