Algorithmica (1987) 2: 153-174

Algorithmica

A Sweepline Algorithm for Voronoi Diagrams

Steven Fortune[1]

Abstract. We introduce a geometric transformation that allows Voronoi diagrams to be computed using a sweepline technique. The transformation is used to obtain simple algorithms for computing the Voronoi diagram of point sites, of line segment sites, and of weighted point sites. All algorithms have $O(n \log n)$ worst-case running time and use $O(n)$ space.

Key Words. Voroni diagram, Delaunay triangulation, Sweepline algorithm.

1. Introduction. The Voronoi diagram of a set of sites in the plane partitions the plane into regions, called Voronoi regions, one to a site. The Voronoi region of a site s is the set of points in the plane for which s is the closest site among all the sites.

The Voronoi diagram has many applications in diverse fields. One application is solving closest-site queries. Suppose we have a fixed set of sites and a query point, and would like to know the closest site to the query point. If the Voronoi diagram of the set of sites is constructed, then this problem has been reduced to determining the region containing the query point. If in fact the number of query points is large relative to the number of sites, then the construction of the Voronoi diagram is worthwhile. The papers by Preparata [16] and by Green and Sibson [8] contain references to other applications.

We present simple sweepline algorithms for the construction of Voronoi diagrams when sites are points and when sites are line segments. The proposed algorithms are based on the sweepline technique [17], [20]. The sweepline technique conceptually sweeps a horizontal line upward across the plane, noting the regions intersected by the line as the line moves. Computing the Voronoi diagram directly with a sweepline technique is difficult, because the Voronoi region of a site may be intersected by the sweepline long before the site itself is intersected by the sweepline. Rather than compute the Voronoi diagram, we compute a geometric transformation of it. The transformed Voronoi diagram has the property that the lowest point of the transformed Voronoi region of a site appears at the site itself. Thus the sweepline algorithm need consider the Voronoi region of a site only when the site has been intersected by the sweepline. It turns out to be easy to reconstruct the real Voronoi diagram from its transformation; in fact in practice the real Voronoi diagram would be constructed, and the transformation computed only as necessary. The sweepline algorithms compute the Voronoi diagram of n sites in time $O(n \log n)$ and space usage $O(n)$.

[1] AT&T Bell Laboratories, Murray Hill, NJ 07974, USA.

Received April 6, 1986; revised 12 October, 1986. Communicated by Bernard Chazelle.

Previous algorithms for Voronoi diagrams fall into two categories. First are incremental algorithms, which construct the Voronoi diagram by adding a site at a time. These algorithms are relatively simple, but have worst-case time complexity $O(n^2)$. However, the algorithms may have good expected time behavior [2], [8], [15].

The second category of algorithms are divide-and-conquer algorithms. The set of sites is split into two parts, the Voronoi diagram of each part computed recursively, and then the two Voronoi diagrams merged together. If the sites are points, then they can be split simply by drawing a line that separates the sites into halves [18]. If the sites are line segments, then more complex partitioning is necessary [21]. With care, the divide-and-conquer algorithms can be implemented in worst-case time $O(n \log n)$. The difficulty of the divide-and-conquer algorithms is generally with the merge step that combines two Voronoi diagrams together. While the time required for this step is only linear in the number of sites, the details of the merge are complex and hard to implement.

The sweepline algorithms presented in this paper are competitive in simplicity with the incremental algorithms. Since they avoid the merge step, they are much simpler to implement than the divide-and-conquer algorithms. But they have the same worst-case time complexity as the divide-and-conquer algorithms.

We also present an algorithm to compute the Voronoi diagram of weighted point sites, in which each site has an additive weight associated with it. The algorithm uses exactly the same sweepline technique, and has time complexity $O(n \log n)$. The best previously known algorithm for this problem has time complexity $O(n \log^2 n)$.

The algorithms in this paper do not assume that the points are in general position. Thus sites can be arbitrarily collinear or cocircular. In order to prove that the algorithms are correct, even with degeneracies, we assume that arithmetic is performed exactly. An interesting problem is to show that the algorithms perform correctly in more reasonable models of computer floating-point arithmetic. The algorithm for the case of point sites has been implemented; there are no known examples that cause it to fail.

2. Point Sites. We first consider the case that all sites are points in the plane. This simple case has been discussed extensively in the literature. We summarize the definitions and elementary properties below; for more details, see, for example, [18] or [11]. For a more general situation than point sites, see [14].

If $p \in R^2$, then p_x and p_y are the x- and y-coordinates of p, respectively. Points $p, q \in R^2$ are *lexicographically ordered*, $p < q$, if $p_y < q_y$ or $p_y = q_y$ and $p_x < q_x$. A line or segment l is *below* $p \in R^2$ if there is a point $q \in l$ with $p_x = q_x$ and $q_y < p_y$. For $p, q \in R^2$, $e(p, q)$ is the Euclidean distance between p and q.

Let S be a set of n points in the plane, called *sites*. For $p \in S$, d_p: $R^2 \to R$ is the (Euclidean) distance from a point in R^2 to p, and d: $R^2 \to R$ is $\min_{p \in S} d_p$. The *Voronoi circle* at $z \in R^2$ is the circle centered at z of radius $d(z)$. The *bisector* B_{pq} of $p, q \in S$ is $\{z \in R^2: d_p(z) = d_q(z)\}$; B_{pq} is of course a line, the usual perpendicular

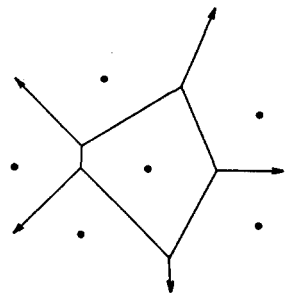

Fig. 2.1. Voronoi diagram V.

R_p, is $\bigcap_{q \neq p} R_{pq}$. R_p is a convex, possibly unbounded polygon containing p. The *Voronoi diagram* $V(S)$, or V for short, is

$$\{z \in R^2: \text{ there is } p \neq q \text{ with } d(z) = d_p(z) = d_q(z)\}.$$

See Figure 2.1.

V consists of the union of segments, half-lines, and lines.[2] A *vertex* of V is a point of V with at least three incident segments or half-lines; equivalently, a vertex is a point equidistant from at least three sites. An *edge* of V is a maximal connected segment, half-line, or line in V; an edge does not contain any vertices in its interior. An edge is either a line, has two vertices as endpoints, or is a half-line with one vertex as endpoint. $B_{pq} \cap V$ is always connected; if it properly contains a point, then it is an edge, labeled e_{pq}. Moreover, each pair $p, q \in S$ label at most one edge of V. Edge e_{pq} forms part of the boundary of R_p and R_q. There are at most $O(n)$ edges and vertices of V, since V forms a planar graph and the degree of every vertex is at least 3.

2.1. The Mapping $*$. The mapping $*: R^2 \to R^2$ defined by $*(z) = (z_x, z_y + d(z))$ is central to the algorithm. This section develops the properties of this mapping; Section 5 contains a geometric interpretation of $*$ that may be helpful. Notice that $*$ maps the point $z \in R^2$ to the topmost point of the Voronoi circle at z. Furthermore, $*$ is continuous, fixes all sites, and maps each vertical line into itself. In general we refer to $*(B)$ as B^*, for B a subset of the plane. To analyze $*$, we first consider the auxiliary function $*_p: R^2 \to R^2$, defined by $*_p(z) = (z_x, z_y + d_p(z))$. Clearly, $* = *_p$ on R_p and $z^* = \min_{p \in S} *_p(z)$.

LEMMA 2.1. *Suppose l is a line and is not vertical. Then $*_p$ is one-to-one on l and $*_p(l)$ is a hyperbola.*

PROOF. Mapping $*$ is clearly one-to-one on a nonvertical line, since it affects only the y-coordinate. Suppose line l has equation $y = mx + b$. Then $*_p(l) = \{(x, z): z = mx + b + ((mx + b - p_y)^2 + (x - p_x)^2)^{1/2}\}$. Thus $(z - mx - b)^2 = (mx + b - p_y)^2 + (x - p_x)^2$ and we see that $*_p(l)$ is a conic section. Consider $H = \{(x, y + e((x, y), r)): (x, y) \in l\}$, where $r \in l$ is the point of l closest to p. H is the union of two half-lines with endpoint r. The two half-lines constituting H must

[2] V contains lines only if all sites are collinear.

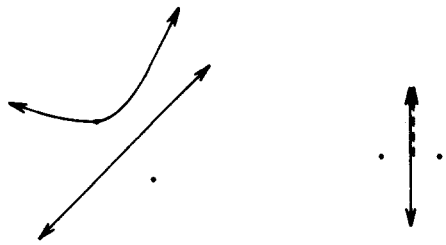

Fig. 2.2. Bisectors and transformed bisectors.

be asymptotes to $*_p(l)$, since for $(x, y) \in l$ sufficiently far from p, $e((x, y), r)$ approximates $e((x, y), p)$. Hence $*_p(l)$ is a hyperbola. □

Figure 2.2 depicts the mapping $*$ when line l is the bisector of two sites (which is the case of interest). We must also consider the case when the line l is vertical. The proof of the following proposition is straightforward.

PROPOSITION 2.2. *Suppose line l is vertical. If $p \notin l$ then $*_p$ is one-to-one on l and $*_p(l)$ is the open half-line above p_y. If $p \in l$ then $*_p(l)$ is the closed half-line above p, points below p are mapped to p, and $*_p$ is one-to-one above p.*

LEMMA 2.3. *If e_{pq} is an edge of V, then e^*_{pq} is a section of a hyperbola or a vertical line.*

PROOF. Since $e_{pq} \subseteq B_{pq}$ is an edge of V, it must be that $d = d_p = d_q$ on e_{pq}. Since B_{pq} is a line, the claim follows from Lemma 2.1 and Proposition 2.2. □

LEMMA 2.4. *Point p is the unique lowest point of R^*_p.*

PROOF. Clearly, $p \in R_p$ and $p^* = p$. By Proposition 2.2, if $z \in R_p$ and z is not on the vertical line through p, then $z^* = *_p(z)$ is above p, and if z is on the vertical line through p then $z^* = *_p(z)$ is mapped to p or above. □

A consequence of Lemma 2.4 is that if edge e_{pq} is below p, then p must lie on e^*_{pq} and in fact must be the unique lowest point on it. Thus the edge that lies below p forms part of the lower boundary of R^*_p. (Notice that there is no edge below p exactly if p has minimal y-coordinate among the set of sites S.) Figure 2.3 depicts V^* for the same set of points as depicted in Figure 2.1.

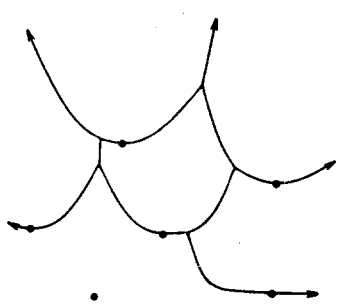

Fig. 2.3. V^* for points in Figure 2.1.

LEMMA 2.5. *Mapping $*$ is one-to-one on V.*

PROOF. Consider $*$ on a vertical line l. We show that $*$ is one-to-one on $V \cap R_p \cap l$, and, furthermore, that if $R_p \cap l$ is a segment, then $*(R_p \cap l)$ does not collapse to a point. Since $*$ is continuous, it preserves the order of points along l, and the lemma follows.

Mapping $*_p$ is one-to-one everywhere except on the vertical half-line below p. Since $* = *_p$ on $R_p \cap l$, $*$ fails to be one-to-one on $R_p \cap l$ only if p lies on $R_p \cap l$ and then only on the section of $R_p \cap l$ below p. If there is no edge of V below p, then trivially $*$ is one-to-one on $V \cap R_p \cap l$. Otherwise some edge lies below p; it cannot be vertical, hence there is only a single point of $V \cap R_p \cap l$ below p. For the second claim, if $R_p \cap l$ is a segment, either p does not lie on l or there is some subsegment of $R_p \cap l$ above p. In either case there is a segment of $R_p \cap l$ on which $*$ is one-to-one, and $*(R_p \cap l)$ is not a point. □

LEMMA 2.6. *Suppose v is a vertex of V. Let r and s be the sites on the Voronoi circle at v counterclockwise and clockwise of v^*, respectively. If v^* is not a site, then edge e^*_{rs} extends upward from v^* and the rest extend downward. If v^* is a site, then edges $e^*_{rv^*}$ and $e^*_{v^*s}$ extend upward from v^* and the rest extend downward.*

PROOF. Every edge e_{pq} incident to v is a segment of the bisector B_{pq} of two sites p and q adjacent around the Voronoi circle at v. In fact, B_{pq} is split into two half-lines by v; e_{pq} is contained in the half-line intersecting A_{pq}, where A_{pq} is the arc of the circle connecting p and q not containing any other sites. We establish that e^*_{pq} extends upward from v^* iff $v^* \in A_{pq}$; the Lemma follows.

First suppose that neither p nor q is v^*. Now v splits B_{pq} into two half-lines. Let h be the half-line that intersects the arc of the circle connecting p and q that contains v^*; let h' be the other. If $p_y \neq q_y$, say $p_y > q_y$, then $*_p(B_{pq})$ is a hyperbola through v^* with minimum point p, $*_p(h)$ extends upward from v^*, and $*_p(h')$ extends downward from v^* to p. If $p_y = q_y$, then $*_p(B_{pq})$ is a section of the vertical half-line above v, $*_p(h)$ is the half-line above v^*, and $*_p(h')$ is the segment from the midpoint of pq to v^*. In either case, if A_{pq} is the arc containing v^*, then $e_{pq} \subseteq h$, $*_p = *$ on e_{pq}, and e^*_{pq} extends upward from v^*; if A_{pq} does not contain v^*, then $e_{pq} \subseteq h'$, $*_p = *$ on e_{pq}, and e^*_{pq} extends downward from v^*. Finally, if one of p or q is v^*, say p, then e^*_{pq} is a section of the hyperbola with minimum point p, A_{pq} always contains v^*, and e^*_{pq} always extends upward from v^*. □

Using Lemma 2.6 we can analyze every point x in the range of $*$. The following cases are mutually exclusive and exhaustive: x is in the interior of some region R^*_p; x lies on some edge e^*_{pq} but is neither a site nor a vertex; x is a site, is not a vertex, and is the minimum point of both R^*_x and e^*_{xp}, some p; x is a vertex, not a site, with at least two edges incident downward and exactly one incident upward; and x is a vertex and a site, with exactly two edges incident upward and at least one incident downward, and is the minimum point of R^*_x.

2.2. The Algorithm for Calculating V^* and V. A sweepline algorithm suffices to compute V^*. The algorithm conceptually moves a horizontal line upward across

the plane, maintaining the regions of V^* intersected by the horizontal line. A region is encountered for the first time at a site, and a region disappears at the intersection of two edges. The coordinates of these two events are easily computed, since all sites are known initially, and since intersections of edges can be computed as these edges become newly adjacent along the sweepline.

The algorithm is given formally as Algorithm 1 (Figure 2.4). Algorithm 1 produces the Voronoi diagram V^* as a list of bisectors. Each bisector is marked with the vertices that are the endpoints of the corresponding Voronoi edge. If a bisector is marked with only a single vertex, then the corresponding edge is a half-line.

Algorithm 1: Computation of $V^*(S)$.
Input: S is a set of $n \geq 1$ points with unique bottommost point.
Output: The bisectors and vertices of V^*.
Data structures: Q: a priority queue of points in the plane, ordered lexicographically. Each point is labeled as a site, or labeled as the intersection of a pair of boundaries of a single region. Q may contain duplicate instances of the same point with distinct labels; the ordering of duplicates is irrelevant.
L: a sequence $(r_1, c_1, r_2, \ldots, r_k)$ of regions (labeled by site) and boundaries (labeled by a pair of sites). Note that a region can appear many times on L.

1. initialize Q with all sites
2. $p \leftarrow$ extract_min(Q)
3. $L \leftarrow$ the list containing R_p.
4. **while** Q is not empty **begin**
5. $p \leftarrow$ extract_min(Q)
6. **case**
7. p is a site:
8. Find an occurrence of a region R_q^* on L containing p.
9. Create bisector B_{pq}^*.
10. Update list L so that it contains $\ldots, R_q^*, C_{pq}^-, R_p^*, C_{pq}^+, R_q^*, \ldots$ in place of R_q^*.
11. Delete from Q the intersection between the left and right boundary of R_q^*, if any.
12. Insert into Q the intersection between C_{pq}^- and its neighbor to the left on L, if any, and the intersection between C_{pq}^+ and its neighbor to the right, if any.
13. p is an intersection:
14. Let p be the intersection of boundaries C_{qr} and C_{rs}.
15. Create the bisector B_{qs}^*.
16. Update list L so it contains $C_{qs} = C_{qs}^-$ or C_{qs}^+, as appropriate, instead of C_{qr}, R_r^*, C_{rs}.
17. Delete from Q any intersection between C_{qr} and its neighbor to the left and between C_{rs} and its neighbor to the right.
18. Insert any intersections between C_{qs} and its neighbors to the left or right into Q.
19. Mark p as a vertex and as an endpoint of B_{qr}^*, B_{rs}^*, and B_{qs}^*.
20. **end**

Fig. 2.4. Algorithm 1: computation of $V^*(s)$.

Algorithm 1 does not explicitly test for degeneracies, where a degeneracy is a site lying on a bisector, four or more cocircular sites, or three or more collinear sites. As Theorem 2.7 shows, Algorithm 1 is adequate to compute the Voronoi diagram even in the presence of degeneracies. However, one consequence of the lack of explicit tests for degeneracies is that Algorithm 1 can produce zero length edges, specifically bisectors with the same point as the two endpoints. These edges can be removed if necessary. Alternatively, Algorithm 1 could be written to test explicitly for degeneracies, though it then becomes more complex. Also, Algorithm 1 assumes that the set S of sites contains a unique bottommost site. This assumption is not essential; lines 2 and 3 need to be modified to initialize the list L correctly if S contains several sites with identical minimal y-coordinate.

If $p_y > q_y$, then $*_p(B_{pq})$ is a hyperbola open upward, and a horizontal line can intersect it at two points. To simplify the presentation of Algorithm 1, we split $*_p(B_{pq})$ into two pieces, C^-_{pq} to the left of and containing p, and C^+_{pq} to the right of and containing p. Then C^-_{pq} is monotonically decreasing, C^+_{pq} is monotonically increasing, and a horizontal line can intersect either of them at most once. If $p_y = q_y$ then we set $C^-_{pq} = \emptyset$ and $C^+_{pq} = *_p(B_{pq})$. We call C^-_{pq} and C^+_{pq} *boundaries*. We use C_{pq} to denote one of C^-_{pq} and C^+_{pq} when the choice is unimportant or can be determined from context. For example, in line 15 of Algorithm 1, C_{qs} is C^+_{qs} either if p is to the right of the higher of q and s or if q and s are cohorizontal; otherwise C_{qs} is C^-_{qs}.

Line 8 of Algorithm 1 is a search to find the region containing a newly encountered site. The search can be implemented as a binary search on list L, since L contains the regions and boundaries in order on the horizontal line. If the site actually falls on a boundary, the search can return the region on either side of the boundary. Note that the actual x-coordinate where a boundary intersects the horizontal line is determined by the y-coordinate of the line.

THEOREM 2.7. *Let S be a set of point sites, with unique bottommost site. Then Algorithm 1 computes $V^*(S)$.*

PROOF. Say a region or boundary T of V^* is *active* if T intersects the horizontal line through p to the left of or at p and extends above the line, or if T intersects the horizontal line to the right of p, and also extends below the line. (Thus a boundary with minimum point on the line to the right of p is not active, and a boundary with maximum point on the line to the left of p is not active.) We claim that statement I3 following is an invariant of the **while** loop, and that statements I1 and I2 following are intermittent invariants of the loop. Specifically, I1 and I2 are true after the last iteration of the loop that extracts a particular point p from Q.

(I1) List L contains all regions and boundaries of V^* active at p, in the order intersected by the horizontal line through p.

(I2) If e_{rs} is an edge of V (r and s are arbitrary sites) and e^*_{rs} contains a point lexicographically less than or equal to p, then bisector B^*_{rs} has been created. If v is a vertex of V and v^* is lexicographically less than or equal to p, then

v^* has been marked as a vertex and as an endpoint of all bisectors containing an edge of V^* incident to v^*.

(I3) If two boundaries are adjacent on L and intersect above the horizontal line through p, then the intersection is in Q.

Invariant I1 is true initially, since the horizontal line through the bottommost site b intersects R_b^* at b itself. Invariant I2 is true initially, since no edge or vertex of V^* contains a point lexicographically less than the minimal site. Invariant I3 is true initially since no boundary intersects the horizontal line through b. Invariant I3 is clearly maintained by lines 11, 12, 17, and 18.

We argue invariant I1 explicitly; invariant I2 follows because we examine the diagram V^* in lexicographic order and always correctly create bisectors and label vertices. To argue that I1 is invariant, first note that the set of active regions and boundaries changes only at a site or a vertex of V^*. Now a vertex of V^* must have degree at least three; by Lemma 2.6, if it is not also a site, it must have at least two incident edges emanating downward, and the two edges must be boundaries active when intersected by the sweepline. Hence it suffices to argue that L is updated correctly at a site and at the intersection of boundaries.

First suppose that p is a site not lying on a boundary. By Lemma 2.4, p is the minimal point of R_p^*. Furthermore, if p lies in the interior of R_q^* when it was first encountered, then p is above edge e_{pq}, and p lies on edge e_{pq}^*. Hence lines 8–10 correctly update L.

Now suppose that p is not a site but is the intersection of boundaries. Let boundaries $C_{q_1q_2}, C_{q_2q_3}, \ldots, C_{q_{m-1}q_m}$, $m \geq 3$, all intersect at p, in this order from left to right just below p, and suppose that p is about to be extracted from Q for the first time. By the induction hypothesis I1, L contains the sequence $\ldots, R_{q_1}^*, C_{q_1q_2}, \ldots, C_{q_{m-1}q_m}, R_{q_m}^*, \ldots$; call this subsequence L_0. By Lemma 2.6, L_0 should be replaced with $R_{q_1}^*, C_{q_1q_m}, R_{q_m}^*$ after the last time p is extracted from Q. We claim the following assertion is an invariant of the **while** loop until the last time p is extracted from Q:

(I4) $R_{q_1}^*$ and $R_{q_m}^*$ are never deleted from L_0, and if $R_{q_i}, C_{q_iq_j}, R_{q_j}, C_{q_jq_k}, R_{q_k}$ are adjacent on L_0, then $C_{q_iq_j}$ and $C_{q_jq_k}$ intersect at point p.

(Notice that by I3, Q contains one copy of p for each consecutive pair of boundaries in L_0.) Invariant I4 is clearly true before p is extracted for the first time. Each iteration of the **while** loop replaces a consecutive pair of boundaries intersecting at p with a single boundary. However, since all sites $q_1, \ldots, q_m$ are equidistant from $*^{-1}(p)$, the new boundary intersects its left and right neighboring boundaries at p, and invariant I4 is maintained. After p is extracted from Q for the last time, only $R_{q_1}, C_{q_1q_m}, R_{q_m}$ remain, and invariant I1 is established.

Now suppose that p is a site lying on a boundary, or p is a site and two or more boundaries intersect at p. Let $C_{q_1q_2}, \ldots, C_{q_{m-1}q_m}$ be the boundaries incident at p, $m \geq 2$. By Lemma 2.6, this sequence of boundaries and enclosed regions should be replaced on L by C_{q_1p}, R_p, C_{pq_m}. We show that I4 is again an invariant of the **while** loop for the iterations in which p is extracted. The only new case is if p, labeled as a site, is extracted from Q. Then p is determined to lie in some

region R_{q_i}, and boundaries $C^-_{q_i p}$ and $C^+_{q_i p}$ are created. However, p and $q_1, \ldots, q_m$ are all equidistant from $*^{-1}(p)$. Hence if $i \neq 1$, then $C^-_{q_i p}$ intersects its left neighboring boundary at p, and if $i \neq m$ then $C^+_{q_i p}$ intersects its right neighboring boundary at p. □

We see from the proof of Theorem 2.7 that it does not matter in what order multiple events at a single site are processed. This is why Algorithm 1 can handle degeneracies in the placements of the sites without being explicitly coded to do so. Of course, there is a tradeoff between the complexity of the proof of Theorem 2.7 and the complexity of Algorithm 1. The proof would be simpler if Algorithm 1 explicitly handled multiple events.

THEOREM 2.8. *Algorithm* 1 *can be implemented to run in time* $O(n \log n)$ *and space* $O(n)$.

PROOF. We first claim that the number of iterations of the **while** loop is at most $O(n)$; it then follows that the number of bisectors ever created is $O(n)$. By an analysis similar to Theorem 2.7, the number of iterations of the loop for a point p not a site is one less than the number of boundaries intersecting at p from below, and for p a site the number of iterations is one more than the number of boundaries p intersecting at p from below. Since the number of intersecting boundaries at p is bounded above by the degree of p as a vertex of V, the number of iterations summed over all vertices and sites is $O(n)$.

Now Q contains at most two entries per boundary and one per site, hence at most $O(n)$ entries. Priority queue Q needs operations inset, delete, and extract-min; Q can be implemented as a heap at time cost $O(\log n)$ per operation and storage cost $O(n)$. Similarly, L can contain at most $O(n)$ entries, since a horizontal line can intersect each bisector at most twice. List L needs operations insert, delete, and search (for line 10); a balanced tree scheme can implement these at time cost $O(\log n)$ per operation and storage cost $O(n)$. Thus each iteration of the **while** loop takes time $O(\log n)$ for a total time of $O(n \log n)$. □

THEOREM 2.9. *Algorithm* 1 *can be modified to compute* V *in time* $O(n \log n)$ *and space* $O(n)$.

PROOF. We claim that Algorithm 1 can be modified to use only untransformed bisectors and vertices. Algorithm 1 creates a Voronoi edge by first creating the bisector containing the edge and then later marking the bisector with the endpoints of the edge; these operations can be performed directly. Event queue Q contains sites and the intersections of boundaries. The intersection of two boundaries can be obtained by computing the intersection of two untransformed bisectors and then adding to the y-coordinate of the intersection the distance to any of the sites determining the bisectors. Similarly, list L can contain untransformed regions and bisectors; the mapping can be computed explicitly during the search of line 10. □

The *Delaunay triangulation* is the geometric dual of the Voronoi diagram. A Delaunay edge is a segment joining two sites whose Voronoi regions share a common bounding edge. There is a one-to-one correspondence between the vertices of the Voronoi diagram and the regions resulting from the subdivision of the plane induced by the Delaunay edges. If a Voronoi vertex has degree three, then the corresponding Delaunay region is a triangle; if the vertex has degree exceeding three, then the corresponding region has more than three bounding Delaunay edges, and diagonals need to be added to obtain a triangulation. The Delaunay triangulation has many interesting properties; for example, it maximizes the minimum angle of a triangle over all triangulations of the sites. The paper by Lee and Schacter [12] is a general reference on Delaunay triangulations.

Algorithm 1 can be easily modified to compute the Delaunay triangulation. In the case of the intersection of boundaries, starting at line 12, the Delaunay triangle corresponding to vertices q, r, and s should be output. This will actually produce a full triangulation, with diagonals added arbitrarily to Delaunay regions with more than three sides. If Delaunay regions without additional diagonals are desired, Algorithm 1 can be modified to detect simultaneous intersection of more than two boundaries.

3. Line Segment Sites. Algorithm 1 can be extended to compute the Voronoi diagram of a set of line segments. The general idea of the algorithm is the same; we transform the Voronoi diagram so that the lowest point of a Voronoi region appears at the Voronoi site, and then use a sweepline technique. However, the details are more complex, because the Voronoi diagram itself is more complex, and because the transformation $*$ is not as well behaved.

The Voronoi diagram is constructed using the bisectors between pairs of sites, where the "bisector" is the locus of points equidistant from the two sites. The bisector of two points is a line. While the bisector of two disjoint segments is still a simple curve, it can have up to seven sections, each a section of a line or a parabola. Worse, the bisector of two segments sharing a common endpoint need not even be one-dimensional, since there is a two-dimensional region for which the common endpoint is the closest point of both sites.

We can simplify the situation somewhat by giving a slightly modified definition of the Voronoi diagram. We require that every segment be split into three sites, two for the endpoints and one for the segment itself. Then we distinguish between the endpoints of the segment or the segment itself being closest. As will be seen, this has the consequence that the bisector between two sites is always a section of a line or a parabola. Also, the two-dimensional region that used to be the bisector of two coincident segments becomes the Voronoi region of the common endpoint of the segments. A similar idea has been used in previous papers on the Voronoi diagram of line segments [9], [21].

The transformation $*$ in the line segment case is not always one-to-one, as it was in the point site case. This does not alter the asymptotic time or space complexity of the algorithm, but more care is needed in handling special cases.

3.1. The Voronoi Diagram of Line Segments. We summarize the definition and elementary properties of the Voronoi diagram of a set S of point and closed line segment sites in the plane. For more details, see, for example, [11] or [9]. We assume S contains n sites (at least one of which is a line segment), that line segments intersect only at endpoints, and that every endpoint of a line segment is a point site. Furthermore, we assume that not all sites in S are collinear,

For $p \in S$ and p a point, d_p: $R^2 \to R$ is the Euclidean distance to p, as before. If p is a line segment, then we define d_p to be the distance to the closest point of p; there is a closest point because p is closed. It is possible that the closest point of line segment p to some point of $z \in R^2$ is an endpoint of p. In this case we wish to assign z to the region of the endpoint of p (always a site by the conditions on S) rather than to p; hence we define the *tangent contact region* of p, t_p. For p a point t_p is just R^2. For p a segment t_p is the closed band containing p bounded by the two lines perpendicular to p through the endpoints of p. (Thus a circle centered at $z \in t_p$ of radius $d_p(z)$ is tangent to p.) Now we define d: $R^2 \to R$ by $d(z) = \min_{p:\, z \in t_p} d_p(z)$. The *bisector* B_{pq} is $\{z \in t_p \cap t_q : d_p(z) = d_q(z)\}$. R_{pq} is $\{z \in t_p : d_p(z) \leq d_q(z)\}$ and the *Voronoi region* R_p is $\bigcap_{q \neq p} R_{pq}$. The *Voronoi diagram* $V = V(S)$ is $\{z \in B_{pq} : p, q \in S \text{ and } d(z) = d_p(z)\}$.

The bisector B_{pq} falls into one of four categories, depending on whether p and q are two points, a segment and its endpoint, a segment and a disjoint point, or two segments. If p and q are distinct point sites, then B_{pq} is the perpendicular bisector of p and q, as before. If p is an endpoint of segment q, then B_{pq} is the line perpendicular to q through p. If p is a point not collinear with segment q, then B_{pq} is a section of a parabola. (Note that if p is a point collinear with segment q, not an endpoint, then B_{pq} is empty.) Finally, if p and q are segment sites, then B_{pq} is a point, a segment, possible empty if $t_p \cap t_q$ contains no points equidistant from p and q, or if p and q are collinear sharing a common endpoint, then B_{pq} is the line perpendicular to p and q through the common endpoint. See Figure 3.1.

The Voronoi region R_p need not be convex; however, it is *starshaped* from p: for each $x \in R_p$, there is $y \in p$ so that the line segment xy is contained in R_p (in fact, y can be chosen to be the point of p closest to x) [11].

It is possible that the Voronoi region of a point site p is degenerate. If p is the common endpoint of two collinear segments, then R_p is contained in the line through p perpendicular to the two segments. If p is the common endpoint of three or more segments and every angle formed by two consecutive segments is less than π, then R_p is p itself.

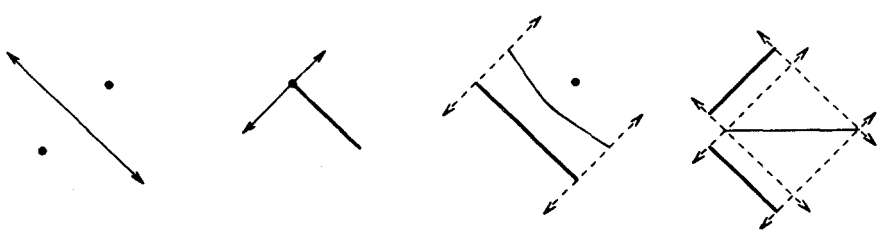

Fig. 3.1. Bisectors. Dotted lines outline tangent contact region.

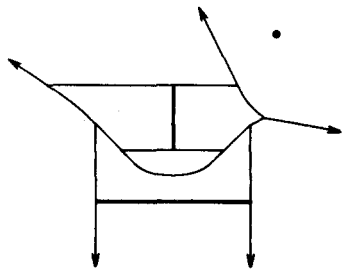

Fig. 3.2. Voronoi diagram of line segments.

A *vertex* of V is a point of V incident to three distinct curves (line segments or parabolas) contained in V. A vertex is always a point equidistant from three sites and in all three tangent contact regions. (A point equidistant from three sites and in all three contact regions need not be a vertex. Consider as sites two collinear coincident segments and their common endpoint. In general, points on the line through the common endpoint perpendicular to the segments are not vertices.) An *edge* of V is a maximal connected subset of a bisector $B_{pq} \cap R_p \cap R_q$ contained in V. Notice that $B_{pq} \cap V$ may not be connected, though it can have at most n components, each an edge. (Consider a "sandwich" consisting of two long segments and a row of sparsely spaced point sites between the two segments.) There are at most $O(n)$ edges and vertices of V, since V forms a planar graph with vertices of degree at least three. Figure 3.2 is a simple example of the Voronoi diagram.

3.2. The mapping $*$. As before, we define $*: R^2 \to R^2$ by $*(z) = (z_x, z_y + d(z))$. Again mapping $*$ is central to the algorithm for the Voronoi diagram of line segments. It has many of the same properties as mapping $*$ in Section 2: it is continuous, fixes all sites, and maps each vertical line into itself. Furthermore, the lexicographically least point of a region R_p^* is the lexicographically least point of p. This last property is of course crucial to the sweepline algorithm. Unfortunately, $*$ fails to be one-to-one on V. This makes the geometric details of the sweepline algorithm more complicated. This section contains a detailed analysis of the properties of $*$. Figure 3.3 depicts the transformed Voronoi diagram of Figure 3.2.

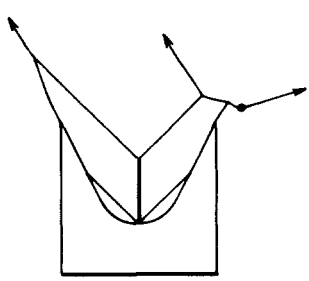

Fig. 3.3. Transformed Voronoi diagram.

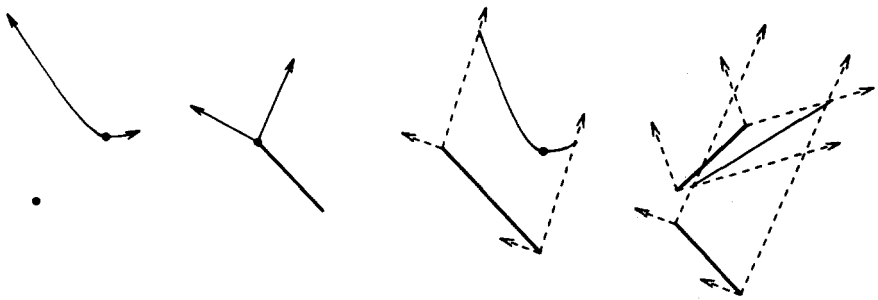

Fig. 3.4. Transformed bisectors.

We begin by studying the mapping $*_p$, defined by $*_p: R^2 \to R^2$ by $*_p(z) = (z_x, z_y + d_p(z))$. Again $* = *_p$ on R_p. Notice that $*_p$ is one-to-one everywhere except on a vertical segment below the lowest point of p, or below all of p if p is a horizontal segment. The following lemma describes the effect of $*_p$ on bisectors. See Figure 3.4.

LEMMA 3.1. *Let $p, q \in S$.*

1. *If p and q are points, then $*_p(B_{pq})$ is a hyperbola or an open vertical half-line.*
2. *Suppose point p is an endpoint of segment q. If q is not horizontal, then $*_p(B_{pq})$ is two half-lines with endpoint p, one directed up and to the left, the other directed up and to the right. If q is horizontal, then the vertical half-line below p contained in B_{pq} collapses to p, and $*_p(B_{pq})$ is the vertical half-line with bottom endpoint p.*
3. *Suppose p is a point not collinear with segment q. Then $*_p(B_{pq})$ is a section of a parabola, except if q is horizontal and $q_y > p_y$, in which case $*_p(B_{pq})$ is q.*
4. *If p and q are both segment sites, and B_{pq} is a segment, then in general $*_p(B_{pq})$ is also a segment. However, if p and q are both horizontal segments sharing a common endpoint, then again the vertical half-line below the common endpoint contained in B_{pq} collapses to the common endpoint, and $*_p(B_{pq})$ is the vertical half-line above the common endpoint.*

PROOF. In all cases $*_p(B_{pq}) = *_q(B_{pq})$.

(1) This follows from Lemma 2.1 and Proposition 2.2.

(2) If q is not horizontal, then for each side of q, $*_q$ is a linear function on t_q, hence on B_{pq}. Furthermore, for $z \in B_{pq}$, $*_p(z)$ must be strictly above p, so both sections of $*_p(B_{pq})$ extend upward from p. If q is horizontal, then $*_q$ is linear on t_q above q and collapses the half-line below p to p itself.

(3) Notice B_{pq} is a section of a parabola. If q is not horizontal or B_{pq} is above q, then $*_q$ is a linear transformation on t_q. Since linear transformations preserve asymptotes (or the lack thereof) the image of a parabola under a linear transformation is a parabola, and B^*_{pq} is a section of a parabola. If q is horizontal and B_{pq} is below q, then $*_q$ collapses the portion of t_q below q to q, hence it also collapses B_{pq} to q.

(4) Similar. □

COROLLARY 3.2. *If e is an edge of V, then e^* is a section of a line, parabola, or hyperbola.*

We give a characterization of every point w in the range of $*$. This characterization is used to construct the sweepline algorithm for computing V^*. The first five possibilities are the same as in Section 2: w is an interior point of some region R_p^*; w lies on an edge e^* but is neither a site nor a vertex; w is an isolated point site, not a vertex; w is a vertex, not a site; and w is an isolated point site and a vertex. Lemma 2.6 applies to these cases. The remaining possibilities are that w is the interior point of some segment site or that w is a point site with incident segment sites. The next two lemmas describe these cases.

LEMMA 3.3. *Suppose w is a point site with at least one incident segment site. Let E_s and E_b be the segment sites incident to w with w as the lexicographically greater and lesser endpoints, respectively. (E_s is the "smaller" edges; E_b is the "bigger" edges.) The edges of V^* extending lexicographically upward from w are the bisectors among the clockwise-most segment of E_s (if $E_s \neq \emptyset$), point site w, E_b, and the counterclockwise-most segment of E_s (if $E_s \neq \emptyset$), and possibly more edges described below. Furthermore:*

1. *If there is a segment on E_s that is not horizontal, then $*^{-1}(w)$ is w itself and no other edges of V^* extend upward from w. The edges of V^* incident to w from below are the bisectors of the segments in E_s, assuming E_s contains at least two segments. See Figure* 3.5(a).
2. *If E_s is empty or is a single horizontal segment, and there is no site in S with y-coordinate smaller than w_y, then $*^{-1}(w)$ is the closed vertical half-line below w. No other edges of V^* extend upward from w and no edges of V^* are incident from below. See Figure* 3.5(b).
3. *If E_s is empty or is a single horizontal segment, and there is a site with y-coordinate smaller than w_y, then $*^{-1}(w)$ is a closed vertical segment wz, z below w. Point z lies on an edge e_{wx} of V, where site x is not a segment incident to w. Let t and u be the sites intersecting the Voronoi circle at z counterclockwise and clockwise of w, respectively. (Notice either $t = u = x$ or $t \neq u$.)*
   (a) *If there is no horizontal segment in $E_s \cup E_b$ then wz is not an edge of V. The edges extending upward from w are the ones described above, the bisector between t and w, and the bisector between u and w. If $t \neq u$ then there is an edge of V^* incident to w from below, and z is a vertex of V. Furthermore, z is the endpoint of all the edges incident to w from below in V^* as well as the bisectors between t and w and between u and w. See Figure* 3.5(c) *if $t = u = x$ and Figure* 3.5(d) *if $t \neq u$.*
   (b) *If there is a horizontal segment incident to w, then wz is an edge of V and z is a vertex of V. The edges of V^* extending lexicographically upward from w are the ones described above, the bisector between t and w if there is no horizontal segment in E_s with right endpoint w, and the bisector between u and w if there is no horizontal segment in E_b with left endpoint w, or the bisector between w and such a segment if it exists. Again z is the endpoint*

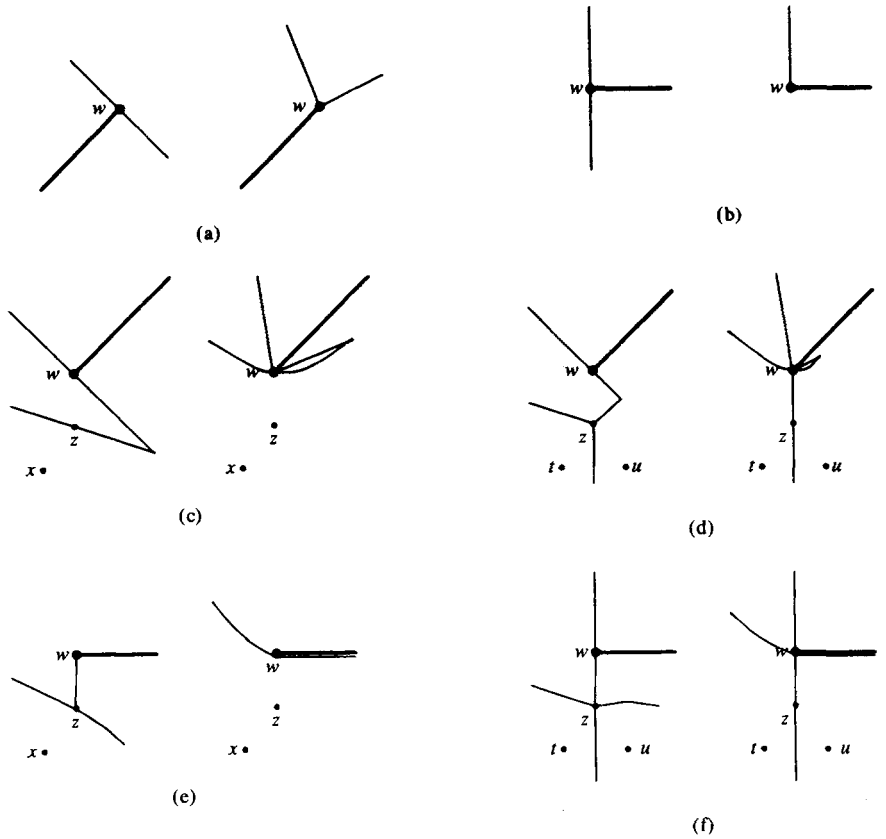

Fig. 3.5

of edges involving t or u and all edges incident to w from below in V. See Figure* 3.5(e) *if* $t = u = x$ *and Figure* 3.5(f) *if* $t \neq u$.

PROOF. We show part 3(a); the others are similar. Since there is a site with y-coordinate smaller than w_y and since there are no segment sites with upper endpoint w, there is an edge e_{wx} of V below w and a vertical segment $wz \subseteq R_w$ with z lying on edge e_{wx}. Mapping $*_w$ collapses the vertical half-line below w to w itself and $* = *_w$ on R_w, so $*(wz) = w$. Mapping $*$ is one-to-one above w and below z, so $*^{-1}(w) = wz$.

Since there are no horizontal segments incident to w, segment wz must be contained in the interior of R_w except at w and z. Hence wz is not an edge of V.

Since z lies on edge e_{wx}, site x must intersect the Voronoi circle at z. If x is the only site below w intersecting the Voronoi circle at z, then z lies in the interior of edge e_{wx}, and e^*_{wx} extends upward both to the left and right of $z^* = w$. If more than one site below w intersects the Voronoi circle at z, then z is a vertex of V. As in the proof of Lemma 2.6, all the bisectors among the sites from t to u counterclockwise around the Voronoi circle at z must be incident to $z^* = w$ from below. □

LEMMA 3.4. *Suppose w is an interior point of segment site s.*

1. *If s is not horizontal then $*^{-1}(w)$ is w itself and w lies in the interior of both R_s and R_s^*.*
2. *Suppose s is horizontal. Then $*^{-1}(w)$ is a vertical segment or half-line with topmost point w. If $*^{-1}(w)$ is a segment its bottommost endpoint z is a vertex exactly if there are at least two edges in V^* incident to w (one of the edges is a horizontal edge to the left of w). If z is a vertex then there is a single horizontal edge to the right of z and z is the endpoint in V of all edges whose images are incident to w in V^*.*

PROOF. Similar to the preceding lemma. □

3.3. The Algorithm for Calculating V

THEOREM 3.5. *Let S be a set of n point and line segment sites. There is an algorithm that computes the Voronoi diagram V of S in time $O(n \log n)$ and space $O(n)$.*

PROOF. The algorithm is similar to Algorithm 1; we give a sketch of it here.

We split edges of V^* into monotonic pieces called *boundaries* and define *active* regions and boundaries as in the proof of Theorem 2.7. We claim that the set of active regions and boundaries only changes at a point site or at a vertex that is the intersection of boundaries incident from below. This follows from an examination of all the cases of points in the range of $*$, given by Lemmas 2.6, 3.3, and 3.4. Hence the points at which the set of active regions changes can be determined using a sweepline algorithm.

The algorithm uses a list L that contains active regions and boundaries and a priority queue Q that contains all point sites and intersections of boundaries active on L. Assertions I1, I2, and I3 from the proof of Theorem 2.7 should be maintained as invariants. To do this the algorithm chooses the lexicographically least point w in the priority queue and performs all the updates to L necessary to reflect the regions and boundaries newly active and inactive at w. We claim that L can be correctly updated using only knowledge of the sites incident to w and the boundaries incident to w from below. This follows by examining all cases in Lemmas 2.6, 3.3, and 3.4. After updating L, the algorithm deletes from Q intersections among boundaries no longer active and inserts into Q all intersections among boundaries newly active. This process iterates until Q is empty.

As described the algorithm sweeps through V^*. We claim that as L is updated at each point w of V^*, the inverse image of w in V can be constructed. Once again, this follows by examining the cases of Lemmas 2.6, 3.3, and 3.4. (Note that there is a subtlety to invariant I2 not apparent in Theorem 2.6: since more edges may be incident to a vertex in V^* than in V (case 3 of Lemma 3.3), the algorithm must mark endpoints of edges according to the incidence structure in V, not V^*.) Hence the algorithm can actually output V, rather than V^*.

The analysis of the time and space bounds of the algorithm is similar to Theorem 2.8. The number of accesses needed to update L at a point w is proportional to the number of boundaries and segment sites incident to w; the total number of updates summed over all vertices and sites of V is $O(n)$. If L

is implemented as a balanced tree then accesses cost time $O(\log n)$ each, for a total time of $O(n \log n)$. Similarly, Q can be implemented as a heap, so the $O(n)$ total accesses each cost time $O(\log n)$, for a total of $O(n \log n)$. Both Q and L require $O(n)$ space. □

It was possible to implement Algorithm 1 as if intersections of boundaries and sites could not occur simultaneously, and still have list L updated correctly. In the case of segments, there seems to be no corresponding way to avoid considering simultaneous intersections and sites. However, the following steps are a reasonably methodical way of updating L at a point w. First, if two or more boundaries are incident to w from below, replace them with a single boundary. Such boundaries may arise in two ways: either edges of V can intersect at a point in $*^{-1}(w)$ strictly below w (this may happen in the cases covered by Lemmas 2.6, 3.3(3), and 3.4(2)) or bisectors of segment sites with upper endpoint w can intersect at w both in V and V^* (Lemma 3.3(1)). The second step in updating L is to create the bisectors resulting from the point site at w and segment sites with w as lower endpoint, if any. There are two cases here, depending on whether w lies in the interior of some region or on a boundary (possibly the boundary is the result of the first step). By Lemma 3.3 all such bisectors extend upward from w in V^*, except the bisector between w and a horizontal segment with left endpoint w, if any. The last step is to consider the edge resulting from a bisector between w and a horizontal edge with left endpoint w. This edge extends vertically downward and is collapsed into w by $*$. However, as in Lemma 3.3(3b), its lower endpoint may be a vertex of V, and there may be another boundary extending upward from w in V^* (the boundary is actually horizontal with left endpoint w).

4. Weighted Point Sites. As a final, easy, application of the sweepline technique we consider the Voronoi diagram of weighted point sites. Now every site is a point and has a nonnegative weight associated with it. We wish to partition the plane into regions so that each region consists of points closest to a particular site, where the metric is the sum of the weight of the site and the distance to the site. The Voronoi diagram that results has a similar appearance to the unweighted point site case, except that bisectors are sections of hyperbolas. Also, the region of a site may be empty if its weight is bigger than the weighted distance from some other site.

The weighted Voronoi diagram turns out to be identical to the Voronoi diagram of circles. The metric in the circle case is the Euclidean distance to the center of the circle less the radius of the circle. Thus outside the circle the metric is the Euclidean distance to the closest point of the circle and inside the circle the metric is the negative of the Euclidean distance to the closest point of the circle. The Voronoi diagram of a set of weighted point sites is just the Voronoi diagram of a set of (possibly intersecting) circles, where the site at p of weight w_p is represented by the circle with center p and radius $W - w_p$, $W = \max\{w_q\}$. Previous algorithms for the Voronoi diagrams of circles have running time $O(n \log^2 n)$;

Lee and Drysdale [11] consider the case of nonintersecting circles and Sharir [19] considers the case of possibly intersecting circles.

The sweepline algorithm for the weighted point site case has running time $O(n \log n)$. It is similar to Algorithm 1, with two minor differences. First, the appropriate mapping $*$ no longer fixes sites, but maps them upward by a distance equal to their weight. This does not affect the algorithm, however. Second, a test is necessary to see if the region of a site is empty; this turns out to be a simple byproduct of the mapping $*$.

An alternative definition of weighted Voronoi diagram has been studied by Aurenhammer and Edelsbrunner [1]. In this model each site has a multiplicative weight: the metric is the Euclidean distance to a site multiplied by its weight. The diagram that arises from this metric is quite different from the usual Voronoi diagram; for example, one Voronoi region may completely encircle another Voronoi region.

4.1. The Voronoi Diagram. We sketch the definition and elementary properties of the Voronoi diagram of weighted point sites. The paper by Sharir [19] contains an extensive presentation of this information, cast in the framework of the Voronoi diagram of circles. S, the set of *sites*, is a set of n points in the plane. Associated with every site p is a nonnegative weight w_p. If p and q are distinct sites, and $d_p(q) + w_p \leq w_q$, then we say p *dominates* q. If neither p nor q dominates the other, then the *bisector* B_{pq} is $\{z \in R^2 : d_p(z) + w_p = d_q(z) + w_q\}$, and R_{pq} is $\{z \in R^2 : d_p(z) + w_p \leq d_q(z) + w_q\}$. If p dominates q, then B_{pq} is empty, R_{pq} is R^2, and R_{qp} is empty. (Note that if $d_p(q) + w_p = w_q$, then $\{z \in R^2 : d_p(z) + w_p = d_q(z) + w_q\}$ is a half-line with endpoint q. We define away this degenerate case to simplify the presentation.) The *Voronoi region* R_p is $\bigcap_{q \neq p} R_{pq}$ and the *Voronoi diagram* $V(S) = V$ is $\{z \in B_{pq} : p, q \in S \text{ and } d_p(z) + w_p = \min_{r \in S} d_r(z) + w_r\}$.

LEMMA 4.1. *Suppose $w_p > w_q$. Then B_{pq} is a branch of a hyperbola open toward p. If $|q_y - p_y| > w_p - w_q$, then one asymptote of B_{pq} extends to the left, the other to the right. If $|q_y - p_y| = w_p - w_q$, then B_{pq} has one vertical and one nonvertical asymptote. If $|q_y - p_y| < w_p - w_q$ then both asymptotes extend left or both extend right.*

PROOF. Consider the defining equation of B_{pq}, $w_p + ((x - p_x)^2 + (y - p_y)^2)^{1/2} = w_q + ((x - q_x)^2 + (y - q_y)^2)^{1/2}$. By moving w_q to the left-hand side and squaring, we obtain an equation with a single square root. Furthermore, the quadratic terms in x and y cancel, leaving only linear terms. By squaring again we can eliminate the remaining square root and obtain terms at most quadratic in x and y. Hence B_{pq} is a conic section.

Now choose points r_1 and r_2 so that pqr_1 and pqr_2 are right triangles with hypotenuse segment pq and with $e(q, r_1) = e(q, r_2) = w_p - w_q$ (so $e(p, r_1) = e(p, r_2) = (e(p, q)^2 - (w_p - w_q)^2)^{1/2}$). Choose ray Y_1 so that it perpendicularly bisects segment pr_1 and is directed away from triangle pqr_1; similarly choose ray Y_2. Now rays Y_1 and Y_2 must be asymptotes to B_{pq}, since for a point z on Y_1

(or Y_2) sufficiently far from p and q, $d_p(z)+w_p$ is arbitrarily close to $d_q(z)+w_q$. Hence B_{pq} is a branch of a hyperbola open toward p.

Suppose $q_y > p_y$, so $|q_y - p_y| = q_y - p_y$. Now r_1 and r_2 both lie on the circle of radius $w_p - w_q$ about q and segments $r_1 p$ and $r_2 p$ are tangent to the circle. If $q_y - p_y > w_p - w_q$, then both r_1 and r_2 must lie above p, so asymptotes Y_1 and Y_2 extend downward, one to the left and one to the right. If $q_y - p_y = w_p - w_q$, then one of r_1 and r_2 has the same y-coordinate as p, and one of Y_1 and Y_2 extends vertically downward and the other is not vertical. If $q_y - p_y < w_p - w_q$, then one of r_1 and r_2 lies below p and the other above p, and Y_1 and Y_2 extend either both to the left or both to the right.

If $q_y \le p_y$, the analysis is similar, with asymptotes extending upward rather than downward. □

A *vertex* of V is a point v of V satisfying $d_p(v)+w_p = d_q(v)+w_q = d_r(v)+w_r$, for three distinct sites p, q, r. Equivalently, a vertex is a point lying on three distinct curves of V. An *edge* of V is a maximal one-dimensional curve contained in V properly containing a point and not containing any vertices in its interior. An edge has two vertices as endpoints or has a single vertex as endpoint and extends to infinity. The intersection of B_{pq} with V need not be connected, as in the case of line segments. The Voronoi diagram V forms a planar graph with vertices of degree at most three and has size $O(n)$. It may not be connected, but it has at most $O(n)$ connected components.

4.2. The transformation $*$. We define $*_p: R^2 \to R^2$ by $*_p(x, y) = (x, y + d_p(x, y) + w_p)$, and $*: R^2 \to R^2$ by $*(x, y) = (x, y + \min_{p \in S} \{d_p(x, y) + w_p\})$.

We can use the transformation $*$ to get a condition when a site p is dominated by some other site. Notice that $*_p(p)_y = p_y + w_p$. If for some other site q, $*_q(p)_y \le p_y + w_p$, then $d_q(p) + w_q \le w_p$ and q dominates p. Hence some site dominates p if $*(p)_y < p_y + w_p$, or if $*(p)_y = p_y + w_p$ with the minimum attained both at p and at some site distinct from p.

LEMMA 4.2. *The unique lowest point of* R_p^* is $(p_x, p_y + w_p)$.

PROOF. Similar to Lemma 2.4. □

LEMMA 4.3.

1. If $w_q + q_y > w_p + p_y$, *then* $*_q(B_{pq})$ *has unique lowest point and unique horizontal tangent at* $(q_x, q_y + w_q)$.
2. *If* $w_p + p_y = w_q + q_y$ *then* $*_q(B_{pq})$ *has no horizontal tangents, lies strictly above* $q_y + w_q$, *and has open endpoint at* $((p_x + q_x)/2, q_y + w_q)$.

PROOF. (1) We show that there is a point of B_{pq} below q. First, suppose $q_y > p_y$. If $w_p > w_q$ then using the fact that $q_y - p_y > w_p - w_q$ and Lemma 4.1, B_{pq} is a hyperbola with asymptotes extending both left and right, hence some point of it

lies below q. If $w_p \leq w_q$, then B_{pq} is either a nonvertical line or a hyperbola open toward q, and some point of it lies below q. Now suppose $q_y \leq p_y$; then $w_q > w_p$. We have $w_q - w_p > p_y - q_y$, hence by Lemma 4.1 with p and q reversed, the asymptotes to B_{pq} extend either both left or both right. Since B_{pq} is open toward q some point of it must lie below q.

Now it is clear that $*_q$ maps points of B_{pq} not on the vertical line through q above $q_y + w_p$ and points on the vertical ray below q to $(q_x, q_y + w_p)$. Let $\hat{q}$ be the point of B_{pq} below q. Then $*_q(\hat{q}) = (q_x, q_y + w_q)$ is the unique minimal point of $*_q(B_{pq})$. It is not hard to see that the tangent to $*_q(B_{pq})$ at $*_q(\hat{q})$ must be horizontal.

Suppose $z \in B_{pq}$ and $z \neq \hat{q}$. We show the tangent at $*_q(z)$ is not horizontal. Notice z is not below p, since there can be a point of B_{pq} below at most one of p and q. Consider the ray Y_p directed from p through z and the ray Y_q directed from q through z. Y_p and Y_q cannot overlap. We assume that Y_p and Y_q are not oppositely directed; a slight modification of the argument is necessary if they are. Let Y_p^+ and Y_q^+ be points of Y_p and Y_q after z and Y_p^- and Y_q^- be points of Y_p and Y_q before z. Since Y_p is not vertical downward, $*_p(Y_p)$ must be a ray extending upward from $*_p(p)$. Similarly, $*_q(Y_q)$ must be a ray extending upward from $*_q(q)$.

It is not hard to see that B_{pq} can be split into two pieces near z: B_{pq}^+ contained in the cone bounded by Y_p^+ and Y_q^+ and B_{pq}^- contained in the one bounded by Y_p^- and Y_q^-. We claim B_{pq}^+ is either vertical upward or is above one of Y_p^+ or Y_q^+. If Y_p^+ and Y_q^+ extend both to the left or both to the right, then B_{pq}^+ must be above the lower of the two. If one of Y_p^+ and Y_q^+ extends to the left and the other to the right, or one is vertical, then either B_{pq}^+ is vertical or B_{pq}^+ is above the ray on the same side of the vertical line through z as B_{pq}^+. Similarly, B_{pq}^- is either vertical downward or is below one of Y_p^- and Y_q^-.

Now if B_{pq} is vertical, then $*_q(B_{pq})$ must be vertical. If B_{pq}^+ is above Y_p^+, say, then since $*_p$ is continuous, $*_p(B_{pq}^+) = *_q(B_{pq}^+)$ must be above $*_p(Y_p^+)$. Similarly, $*_p(B_{pq}^-)$ must be below $*_p(Y_p^-)$ or $*_q(Y_q)$. Since $*_p(Y_p)$ extends upward from p and $*_q(Y_q)$ extends upward from q, the tangent to B_{pq} at z cannot be horizontal.

(2) The argument is similar. If $w_p = w_q$ and $p_y = q_y$ then B_{pq} is the vertical line through $(p_x + q_x)/2$. If $w_p \neq w_q$, then using Lemma 4.1 it can be seen that one of the asymptotes to B_{pq} is a ray vertically downward through $(p_x + q_x)/2$. In either case $*_q$ maps all points of B_{pq} above $q_y + w_q$. □

THEOREM 4.4. *Let S be a set of weighted point sites, and $V = V(S)$. There is a sweepline algorithm that computes V in time $O(n \log n)$ and space $O(n)$.*

PROOF. Similar to Theorems 2.7, 2.8, and 2.9. □

5. A Geometric Interpretation of $*$. The mapping $*$ may appear to be somewhat mysterious. A three-dimensional version of the Voronoi diagram may elucidate the role of $*$. This interpretation was first suggested by Edelsbrunner [5].

Fix a set of point sites S. We view the sites as lying in the $z=0$ plane of R^3. For p a site, let the *cone* of p, c_p, be $\{(x, y, z) \in R^3: d_p(x, y) = z\}$. Let C be the lower envelope of these cones, i.e., C is $\{(x, y, z): (x, y, z) \in c_p$, some p, and for all q, if $(x, y, z') \in c_q$ then $z \leq z'\}$. Finally, let D be the subset of points of C contained in two or more cones. Clearly the Voronoi diagram V is the projection of D onto the plane $z=0$ in the direction parallel to the z-axis. Consider the "oblique" projection of R^3 onto the plane $z=0$ in the direction parallel to the line $\{x=0, y+z=0\}$. Then V^* is just the oblique projection of D, and R_p^* just the oblique projection of $c_p \cap C$.

This interpretation of V in three dimensions gives a different geometric explanation of the sweepline algorithm. Suppose the sweepline is the line $y=c$ (in the $z=0$ plane). Let P_c be the plane $y+z=c$. Notice that the portion of the Voronoi diagram V^* intersected by the sweepline is just the oblique projection of $P_c \cap D$. Hence, rather than translating a line in the $z=0$ plane, we can imagine translating a plane parallel to P_c. The intersection of the plane with D gives exactly the sequence of edges of V^* intersected by the sweepline. It is clear that the first time that the translating plane intersects a cone C_p, the plane is tangent to the cone, and the oblique projection of the intersection is the site p.

The extension to additively weighted point sites is immediate from this interpretation. The cone for each site is simply pushed in the positive z direction by a distance equal to the weight of the site.

6. Open Problems. One application of Voronoi diagrams is nearest-neighbor searching in the plane. This use requires a method to search the planar polygonal regions defined by the Voronoi diagram. Two data structures have recently been proposed to assist in polygonal region searching. The first, suggested by Edelsbrunner *et al.* [6], requires linear storage, linear time to construct, and results in logarithmic query time. The second, proposed by Driscoll *et al.* [4], also requires linear space and results in logarithmic query time, but takes time $O(n \log n)$ to construct. However, the second is built using a sweepline approach, like the Voronoi diagram algorithms presented here. A direct application of either data structure would require two passes: one to construct the Voronoi diagram, and the second to build the search data structure. An interesting question is whether the search data structure could be built directly, and the Voronoi diagram eliminated.

The combination of transformation and sweepline is a powerful technique, and the question arises of what other problems can be solved with it. In a companion paper we will consider the case of Voronoi diagrams of line segments under polygonal convex distance functions [3]. Convex polygonal distance functions generalize the metrics L_1 and L_∞ and have applications to versions of the piano-mover's problem. (A preliminary version of the companion paper appeared in [7], and Leven and Sharir [13] obtain a similar result, though with a more complex algorithm.) Perhaps the sweepline algorithm is adequate for even more general metrics.

References

[1] F. Aurenhammer and H. Edelsbrunner, An optimal algorithm for constructing the weighted Voronoi diagram in the plane, *Pattern Recognition*, **17** (1984), 251–257.

[2] J. L. Bentley, B. W. Weide, and A. C. Yao, Optimal expected-time algorithms for closest-point problems, *ACM Trans. Math. Software*, **6** (1980), 563–580.

[3] L. P. Chew and R. L. Drysdale, Voronoi diagrams based on convex distance functions, *Proceedings of the Symposium on Computational Geometry*, 1985, pp. 235–244.

[4] J. R. Driscoll, N. Sarnak, D. D. Sleator, and R. E. Tarjan, Making data structures persistent, *Proceedings of the 18th Annual ACM Symposium on Theory of Computing*, 1986, pp. 109–121.

[5] H. Edelsbrunner, private communication, 1985.

[6] H. Edelsbrunner, L. J. Guibas, and J. Stolfi, Optimal point location in a monotone subdivision, Technical Report, DEC Systems Research Center, Palo Alto, CA, 1984.

[7] S. J. Fortune, Fast algorithms for polygon containment, *Automata, Languages, and Programming, 12th Colloquium*, Lecture Notes in Computer Science, Vol. 194, Springer-Verlag, New York, pp. 189–198.

[8] P. J. Green and R. Sibson, Computing Dirichlet tesselations in the plane, *Comput. J.*, **21** (1977) 168–173.

[9] D. Kirkpatrick, Efficient computation of continuous skeletons, *Proceedings of the 20th Annual Symposium on Foundations of Computer Science*, 1979, pp. 18–27.

[10] D. T. Lee, Medial axis transformation of a planar shape, *IEEE Trans. Pattern Analysis Machine Intel.*, **4** (1982), 363–369.

[11] D. T. Lee and R. L. Drysdale, Generalizations of Voronoi diagrams in the plane, *Siam J. Comput.*, **10** (1981), 73–87.

[12] D. T. Lee and B. J. Schacter, Two algorithms for constructing a Delauney triangulation, *Internat. J. Comput. Inform. Sci.*, **9** (1980), 219–227.

[13] D. Leven and M. Sharir, Planning a purely translational motion for a convex object in two-dimensional space using generalized Voronoi diagrams, Technical Report 34/85, Tel Aviv University, 1985.

[14] D. Leven and M. Sharir, Intersection problems and applications of Voronoi diagrams, in *Advances in Robotics*, Vol. I (J. Schwartz and C. K. Yap, eds), Lawrence Erlbaum, 1986.

[15] T. Ohya, M. Iri, and K. Murota, Improvements of the incremental method for the Voronoi diagram with computational comparison of various algorithms, *J. Oper. Res. Soc. Japan*, **27** (1984), 306–336.

[16] F. P. Preparata, The medial axis of a simple polygon, *Proceedings of the Sixth Symposium on Mathematical Foundations of Computer Science*, Lecture Notes in Computer Science, Vol. 53, Springer-Verlag, New York, 1977, pp. 443–450.

[17] F. P. Preparata and M. I. Shamos, *Computational Geometry: An Introduction*, Springer-Verlag, New York, 1985.

[18] M. I. Shamos and D. Hoey, Closest-point problems, *Proceedings of the 16th Annual Symposium on Foundations of Computer Science*, 1975, pp. 151–162.

[19] M. Sharir, Intersection and closest-pair problems for a set of planar discs, *SIAM J. Comput.*, **14** (1985), 448–468.

[20] R. Sedgewick, *Algorithms*, Addison Wesley, Reading, MA, 1983.

[21] C. K. Yap, An $O(n \log n)$ algorithm for the Voronoi diagram of a set of simple curve segments, NYU-Courant Robotics Report No. 43 (submitted to *SIAM J. Comput.*).