Algorithmica (1998) 20: 319–352

Algorithmica

A New Approach for the Geodesic Voronoi Diagram of Points in a Simple Polygon and Other Restricted Polygonal Domains[1]

E. Papadopoulou[2,3] and D. T. Lee[2]

Abstract. We introduce a new method for computing the geodesic Voronoi diagram of point sites in a simple polygon and other restricted polygonal domains. Our method combines a sweep of the polygonal domain with the merging step of a usual divide-and-conquer algorithm. The time complexity is $O((n+k)\log(n+k))$ where n is the number of vertices and k is the number of points, improving upon previously known bounds. Space is $O(n+k)$. Other polygonal domains where our method is applicable include (among others) a polygonal domain of parallel disjoint line segments and a polygonal domain of rectangles in the L_1 metric.

Key Words. Geodesic Voronoi diagrams, Shortest paths, Polygon triangulation, Topological plane sweep, Computational geometry.

1. Introduction. The geodesic Voronoi diagram of a set S of k point sites in a polygonal domain P is the partitioning of P into k cells, one to each site, called (geodesic) Voronoi cells. A point $x \in P$ belongs to the geodesic Voronoi cell of a site s if and only if the (geodesic) distance of x from s is less than (or equal to) the geodesic distance of x from any other site, where the geodesic distance between two points in P is the length of the shortest path between the points. The Voronoi cell of a site s is further partitioned into finer regions by the *shortest path map* from s, where the shortest path map from s in $P' \subseteq P$ is the subdivision of P' so that all points in a region have the same predecessor along their shortest path from s.

As is discussed in Section 9, the geodesic Voronoi diagram can answer several proximity questions involving point sites in a polygonal domain. Examples include the minimum spanning tree, finding the closest pair of sites, and finding the nearest neighbor for every site. This is similar to the ordinary Voronoi diagram case, however, it is not a straightforward generalization (see Section 9). Furthermore, the geodesic Voronoi diagram encodes shortest path information from the sites to all points in the polygonal domain, and can thus be used to answer efficiently shortest path queries from multiple sources. In particular, given the geodesic Voronoi diagram of a set of point sites S in P

[1] An extended abstract appears in the *Proceedings of ESA '95*. Work by the second author was supported in part by the National Science Foundation under Grant CCR-9309743, and by the Office of Naval Research under Grant Nos. N00014-93-1-0272 and N00014-95-1-1007.
[2] Department of Electrical and Computer Engineering, Northwestern University, Evanston, IL 60208, USA. dtlee@ece.nwu.edu.
[3] Present address: IBM TJ Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598, USA. evanthia@watson.ibm.com.

Received February 15, 1996; revised November 2, 1996. Communicated by B. Chazelle.

(sources) and a query point t, finding the site s nearest to t and the shortest path from s to t basically reduces to locating t in the diagram.

In this paper we introduce a new approach for computing the geodesic Voronoi diagram of a set of point sites in a simple polygon and some other restricted polygonal domains. Our method combines a sweep of the polygonal domain and the merging step of a usual divide-and-conquer strategy. The time complexity is $O((n+k)\log(n+k))$, where n is the number of vertices of the polygonal domain and k is the number of sites. In the case of a simple polygon, it improves upon the $O((n+k)\log(n+k)\log n)$ previously best known bound [1]. Our method can extend to polygonal domains where the shortest path between any two points is monotone with respect to a constant number of directions. This includes a polygonal domain of parallel disjoint line segments and a polygonal domain of rectangular obstacles in the L_1 metric. For the last two cases, the algorithm is asymptotically optimal. Moreover, the algorithm can be generalized to some very restricted polygons with holes (see Section 8).

The ordinary Voronoi diagram of point sites in the plane is a classical mathematical object and can be computed in optimal $O(n\log n)$ time by a divide-and-conquer [18] or sweepline approach [7]. For more information about Voronoi diagrams refer to the survey by Aurenhammer [3]. Aronov used a divide-and-conquer approach [1] for computing the geodesic Voronoi diagram of point sites in a simple polygon. The merge step in Aronov's algorithm is preceded by a procedure that extends a recursively computed diagram of the subset of sites inside half the polygon to the full polygon, which is the reason for the additional logarithmic factor in the time complexity. An earlier result by Asano and Asano [2] solves the problem in time $O(nk + n\log\log n + k\log k)$. For the special case where the set of sites includes *all* the reflex vertices of the polygon, the extension phase of [1] can be skipped and, thus, the geodesic Voronoi diagram can be computed in $O((n+k)\log(n+k))$ time by divide and conquer [1]. For the special case where the set of sites coincides with *the vertices* of a simple n-gon, an alternative $O(n\log n)$ algorithm can be obtained from the *generalized Delaunay triangulation* of a simple polygon [5], [13], [20], which was recently improved to $O(n)$ by Wang and Chin [19].

In a polygonal domain of n rectangles in the L_1 metric (paths are rectilinear), Guha and Suzuki used Aronov's approach, i.e., divide-and-conquer followed by an "extension phase," to derive an $O((n+k)\log(n+k)\log n)$-time algorithm [8]. For the more general polygonal domain of rectilinear obstacles in the L_1 metric, Mitchell has given an $O((n+k)\log^2(n+k))$-time algorithm for computing the geodesic Voronoi diagram [15], based on the *continuous Dijkstra paradigm*. Mitchell's algorithm can be improved to $O((n+k)\log(n+k))$ as he showed in [16] and [17]. The *continuous Dijkstra paradigm* simulates the effect of "wavefronts" propagating out of sites. Our $O((n+k)\log(n+k))$-time algorithm for the case of rectangular obstacles introduces a different approach.

In a general polygonal domain of n vertices, there is basically one subquadratic result by Mitchell which computes the geodesic Voronoi diagram of a set of k sites in $O((n+k)^{3/2+\varepsilon})$ time and $O(n)$ space [16], [17]. The algorithm is based on the *continuous Dijkstra paradigm*. In a recent manuscript, Hershberger and Suri [11] give an $O(n\log n)$-time and -space algorithm to compute the shortest path map of a single source point in a general polygonal domain of n vertices, i.e., $k=1$. (A preliminary version of their work appears in [11] and reports $O(n\log^2 n)$-time and $O(n\log n)$-space bounds.) This is a major improvement over all previous shortest path algorithms in the Euclidean plane.

Their approach also uses the *continuous Dijkstra paradigm* and propagation is done on a cell-by-cell manner.

In this paper we present our $O((n+k)\log(n+k))$-time and $O(n+k)$-space algorithm for computing the geodesic Voronoi diagram of points in a simple polygon. In Section 8 we extend to additional restricted polygonal domains. In Section 9 we show how the geodesic Voronoi diagram can provide answers to proximity questions efficiently.

2. Preliminaries. Let P be a simple polygon of vertex set V, $|V| = n$, and let $S = \{s_1, \ldots, s_k\}$ be a set of k point sites inside P. We denote the shortest path between two points $p, q \in P$ that lies totally in P as $\pi(p, q)$. The length of $\pi(p, q)$ is the *geodesic distance* between p and q, and is denoted by $d_g(p, q)$. The Euclidean distance between p and q is denoted as $d(p, q)$. The last vertex on $\pi(p, q)$ before q is called the *anchor* of q (with respect to p).

The *geodesic Voronoi diagram* of S in P is the partitioning of P into k cells $Vr(s_1), \ldots, Vr(s_k)$ such that a point $x \in P$ belongs to $Vr(s_i)$ if and only if the geodesic distance of x from s_i is less than (or equal to) the geodesic distance of x from any other site $s_j \in S$. $Vr(s_i)$ is referred to as the Voronoi cell of s_i, and s_i is referred to as the *owner* of $Vr(s_i)$. Figure 1 illustrates the geodesic Voronoi diagram of a set of point sites; the whole shaded region is $Vr(s_i)$. The Voronoi cell of a site s_i, $Vr(s_i)$, is partitioned into finer regions by the *shortest path map* from s_i which partitions $Vr(s_i)$ into maximal sets of points that have the same anchor v with respect to s_i. We refer to such a maximal set of points as the *subcell* of $Vr(s_i)$ induced by vertex v. Vertex v is called the *anchor* of the subcell, and is weighted with its *geodesic distance* from s_i ($w(v) = d_g(s_i, v)$) (see Figure 1). It is well known [9], [1] that the edges of the shortest path map is a collection of *extension segments* emanating from reflex vertices of P in $Vr(s_i)$, where the extension segment emanating from a vertex v is the initial part of a half-line collinear with the last edge on $\pi(s_i, v)$, extending in the direction away from s_i, until it hits the boundary of the polygon or the cell $Vr(s_i)$. In Figure 1 the edges of the shortest path map in each Voronoi cell are shown in dashed lines.

The boundary shared by two Voronoi cells $Vr(s_i)$ and $Vr(s_j)$ consists of points that are of the same geodesic distance from s_i and s_j, and is the union of portions of bisectors

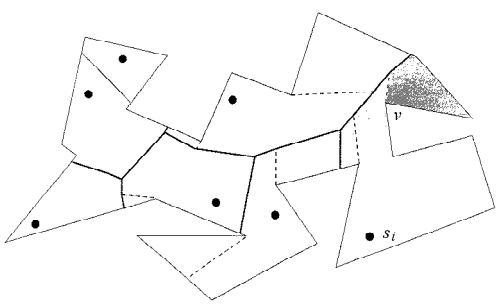

Fig. 1. The geodesic Voronoi diagram of S in P, $Vor_P(S)$. Dashed lines indicate the shortest path map in Voronoi cells; the shaded region is $Vr(s_i)$, $s_i \in S$; the darkest shaded part is the subcell of $Vr(s_i)$ induced by v.

of p and q, where $p, q \in V \cup S$. Given $p, q \in V \cup S$ weighted by $w(p), w(q) \geq 0$, the Euclidean bisector of p and q is $b(p, q) = \{z \mid d(z, p) + w(p) = d(z, q) + w(q)\}$, and belongs to one of the two branches of the hyperbola with foci p, q and eccentricity $d(p, q)/|w(p) - w(q)|$ [14]. If $w(p) < w(q)$, $b(p, q)$ belongs to the branch closer to q, otherwise it belongs to the branch closer to p. For $w(p) = w(q)$, $b(p, q)$ is a straight line. A point inside P which is common to three or more Voronoi cells, or a point on the boundary of P which is common to two or more Voronoi cells, is a *Voronoi vertex*. Bisectors that are common to two Voronoi cells are called *Voronoi edges*. Endpoints of bisector portions which are not Voronoi vertices are called *breakpoints*. A breakpoint is the point of intersection of a Voronoi edge and an edge of the shortest path map of the incident Voronoi cell. An important property of Voronoi cells is that they are *star-shaped* around the cell owner [18], [1]. A set $Q \subset P$ is *star-shaped* around $y \in P$ (with respect to the geodesic distance) if $\pi(y, x) \subset Q$, $\forall x \in Q$. In other words, for every point $x \in Vr(s_i)$, $\pi(s_i, x)$ lies entirely in $Vr(s_i)$.

As was shown in [1], the complexity of the geodesic Voronoi diagram is $O(n + k)$. Augmenting Voronoi cells with their shortest path map yields the *augmented geodesic Voronoi diagram*. Our algorithm computes the *augmented geodesic Voronoi diagram* of S in P. Throughout this paper, when we refer to the geodesic Voronoi diagram of S in P, we imply the augmented structure, and we denote it by $Vor_P(S)$ (see Figure 1). Given $Vor_P(S)$ and a query point t, a point location query [12] yields in $O(\log(n + k))$ time the site $s_i \in S$ nearest to t. We can thus retrieve $\pi(s_i, t)$ in additional time proportional to the number of links in the path.

Throughout this paper, we make the following assumptions:

ASSUMPTION A. No two sites in S are equidistant (in the geodesic sense) from the same polygon vertex.

ASSUMPTION B. No four sites of S lie on the same geodesic circle. In other words, there is no point in P equidistant (in the geodesic sense) from four sites in S.

Assumption A is the *general position* assumption of [1] for the geodesic Voronoi diagram and is made in order to avoid having Voronoi cells that overlap nontrivially. We make Assumption B following the assumption of [18] for the ordinary Voronoi diagram in order to simplify the presentation.

$Vor_P(S)$ can be viewed as a planar graph embedded in P whose vertices are the Voronoi vertices and breakpoints, and whose edges are the Voronoi edges and shortest path map edges. For notational convenience, we identify the Voronoi diagram with the induced planar graph and use the same notation ($Vor_P(S)$) to denote both. Under Assumption B, Voronoi vertices are the common intersection of exactly three Voronoi edges [18], and thus they have degree three. A breakpoint is incident to two bisector arcs (in the same Voronoi edge) and an edge of the shortest path map. Thus, in the augmented structure a breakpoint has degree three. Therefore, all vertices of $Vor_P(S)$ have degree three (under Assumption B). Given a subset of sites $S' \subseteq S$ and some part of the polygon $P' \subseteq P$, the geodesic Voronoi diagram of S' restricted to P' is denoted as $Vor_{P'}(S')$. The planar graph induced by $Vor_{P'}(S')$ consists of the Voronoi vertices and breakpoints

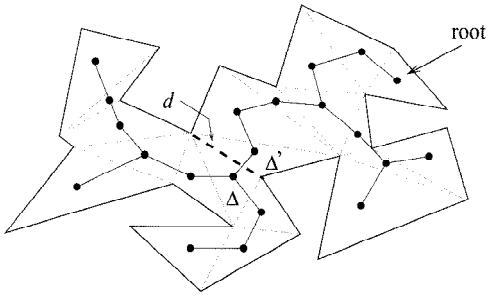

Fig. 2. The dual tree of P.

of $Vor_P(S')$ that lie in P' and the incident Voronoi edges truncated by the sides of P'. $Vor_{P'}(S')$ denotes both the diagram and the induced planar graph.

The geodesic bisector of two disjoint sets of sites, $S_1, S_2 \subset S$ is $b_g(S_1, S_2) = \{x \in P \mid d_g(x, S_1) = d_g(x, S_2)\}$, where $d_g(x, S_i) = \min_{s \in S_i}\{d_g(x, s)\}$. $b_g(S_1, S_2)$ is the set of Voronoi edges of $Vor_P(S_1 \cup S_2)$ that are shared by Voronoi cells $Vr(s_i)$ and $Vr(s_j)$, for $s_i \in S_1$ and $s_j \in S_2$. As shown in [1], $b_g(S_1, S_2)$ can be decomposed into edge-disjoint simple paths and cycles (under Assumption A). If S_1 and S_2 lie on opposite sides of a polygon diagonal, $b_g(S_1, S_2)$ contains no cycles. In that case, $b_g(S_1, S_2)$ is a simple path whose vertices (except the endpoints) have degree two (under Assumptions A and B).

The following lemma can be easily derived by the star-shape property of Voronoi cells.

LEMMA 1. *Let S_1, S_2 be two disjoint subsets of S and let x_1, x_2 be points in P. If $s_i \in S_1$ and $s_j \in S_2$ are the nearest sites to x_1 and x_2, respectively, among $S_1 \cup S_2$, then $\pi(s_i, x_1)$ and $\pi(s_j, x_2)$ do not intersect.*

In what follows, we assume that polygon P is arbitrarily triangulated.[4] The *dual graph* of the triangulated polygon is a tree, such that each node of the graph corresponds to a triangle of the triangulation and each edge of the graph connects two nodes if and only if the corresponding two triangles share a diagonal of the polygon. The dual graph of a triangulated polygon contains no cycles, and therefore it is a tree, referred to as the *dual tree* (see Figure 2).

We assign an arbitrary node of degree one as the *root* of the tree. (A node of degree one corresponds to a triangle with two edges belonging to the polygon boundary.) Then there is a unique directed path from the root to every triangle which defines a parent–child relation between the triangles. For a diagonal d, let $\Delta(d)$ and $\Delta'(d)$ denote the triangles adjacent to d, such that $\Delta'(d)$ is on the path from the root to $\Delta(d)$. We say that $\Delta(d)$ is the triangle "below" d and that $\Delta'(d)$ is the triangle "above" d. We also say that $\Delta'(d)$ is the *parent* of $\Delta(d)$ and that d is the *root diagonal* of $\Delta(d)$. Furthermore,

[4] A triangulated polygon is partitioned into triangular regions by nonintersecting diagonals, i.e, straight-line segments with endpoints at vertices. A polygon can be triangulated in linear time [4]. There are also several simpler $O(n \log n)$-time algorithms (e.g., [18]).

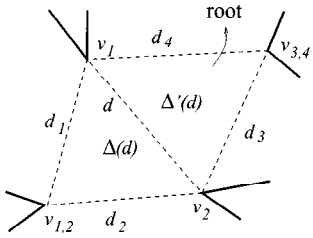

Fig. 3. d, $\Delta(d)$, $\Delta'(d)$.

d partitions the polygon into two parts: The part of the polygon "below" d, denoted by $P(d)$, which corresponds to the subtree of the dual tree rooted at $\Delta(d)$, and the part of the polygon "above" d, denoted by $P'(d)$, the complement of $P(d)$. We assume that $d \in P(d)$. Similarly, let $S(d) = S \cap P(d)$ be the set of sites "below" d and let $S'(d) = S \cap P'(d)$ be the set of sites "above" d. The set of sites in $\Delta(d)$ is denoted as $S(\Delta(d)) = S(d) - S(d_1) - S(d_2)$.

Throughout this paper, we adopt the following convention: Given a diagonal $d = \overline{v_1 v_2}$, let d_1, d_2 denote the remaining diagonals of $\Delta(d)$, and let d_3 and d_4 denote the remaining diagonals of $\Delta'(d)$. We assume that d_4 is the diagonal shared by $\Delta'(d)$ and its parent. We further assume that d_1, d, and d_4 are incident to vertex v_1, and that d_2, d, and d_3 are incident to vertex v_2. We denote the vertex shared by d_1 and d_2 as $v_{1,2}$ and the vertex shared by d_3 and d_4 by $v_{3,4}$ (see Figure 3). To avoid ambiguities, a site on diagonal d is assumed to belong in $S(d)$ (note that $d \in P(d)$). Furthermore, if a site coincides with a vertex of the polygon, we assume that it belongs to only one of the two polygon edges incident to it.

3. An Overview of the Algorithm. In the algorithm we use the rooted dual tree of the triangulation as a guide to sweep the polygon. We compute the geodesic Voronoi diagram of S in P in three phases: In the first phase we sweep the triangulated polygon in the order specified by the postorder traversal of the rooted dual tree. For each triangle of root diagonal of d, we compute (in the triangle) the geodesic Voronoi diagram of all sites "below" d (i.e., $Vor_{\Delta(d)}(S(d))$). We create an $O(n+k)$ subdivision of P, $\mathcal{SD}(P)$, which contains the above information for every triangle. In the second phase we sweep the triangulated polygon in the order specified by the preorder traversal of the rooted dual tree. For each triangle of root diagonal of d, we compute (in the triangle) the geodesic Voronoi diagram of all sites "above" d (i.e., $Vor_{\Delta(d)}(S'(d))$) and store the information in another $O(n+k)$ subdivision, $\mathcal{SD}'(P)$. Finally, we merge the two subdivisions and obtain the geodesic Voronoi diagram of S in P. It is easy to see that merging in $\Delta(d)$ the Voronoi diagram of sites "below" d with the Voronoi diagram of sites "above" d yields the part of the geodesic Voronoi diagram of all the sites that falls in $\Delta(d)$.

4. Definition of the Subdivisions. Subdivision $\mathcal{SD}(\mathcal{P})$ is defined as follows. For every triangle of root diagonal d, $\mathcal{SD}(P)$ contains $Vor_{\Delta(d)}(S(d))$, i.e., the Voronoi diagram of $S(d)$ that lies in $\Delta(d)$ (see Figure 4). Consider the points of intersection of d

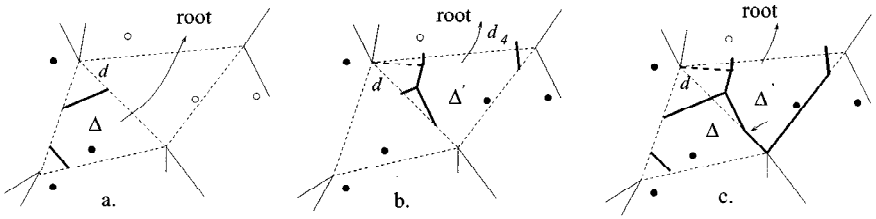

Fig. 4. (a) $Vor_{\Delta(d)}(S(d))$, (b) $Vor_{\Delta'(d)}(S(d_4))$, (c) $\mathcal{SD}(P) \cap \{\Delta, \Delta'\}$; border vertices are indicated by arrows.

with the Voronoi diagrams in $\Delta(d)$ and $\Delta'(d)$ (i.e., $Vor_{\Delta(d)}(S(d))$ and $Vor_{\Delta'(d)}(S(d_4))$). Intersection points that are common to both diagrams and belong to the same Voronoi or shortest-path-map edge in both $\Delta(d)$ and $\Delta'(d)$ are not considered. Intersection points that belong to only one of the diagrams or intersection points that are common to both diagrams but belong to different Voronoi or shortest-path-map edges induce vertices which are referred to as *border* vertices. (In Figure 4(c), the border vertex is indicated by an arrow.) Any two consecutive border vertices on d (including the endpoints of d) are joined by an edge, if the incident Voronoi cells in $\Delta(d)$ and $\Delta'(d)$ belong to different sites. These edges are called *border* edges. A border edge on d implies that the owner of the adjacent Voronoi cell in $Vor_{\Delta'(d)}(S(d_4))$ has not been considered in $P(d)$, and that its Voronoi cell should expand in $P(d)$ (see Figures 4(c) and 5).

$\mathcal{SD}'(P)$ is defined similarly. For every triangle of root diagonal d, $\mathcal{SD}'(P)$ contains the Voronoi diagram of $S'(d)$ that lies in $\Delta(d)$, $Vor_{\Delta(d)}(S'(d))$ (see Figure 6). The Voronoi diagrams in $\Delta(d)$ and $\Delta'(d)$ (i.e., $Vor_{\Delta(d)}(S'(d))$ and $Vor_{\Delta'(d)}(S'(d_4))$) induce *border* vertices and *border* edges on d which are defined in exactly the same manner as for $\mathcal{SD}(P)$ (see Figure 6(c)). In $\mathcal{SD}'(P)$ a border edge on a diagonal d implies that the site of the adjacent Voronoi cell in $\Delta(d)$ has not been considered in $P'(d)$ and that its Voronoi cell should expand in $P'(d)$ (see Figures 6(c) and 7).

We refer to faces of the two subdivisions as *regions*. Every region of $\mathcal{SD}(P)$ and

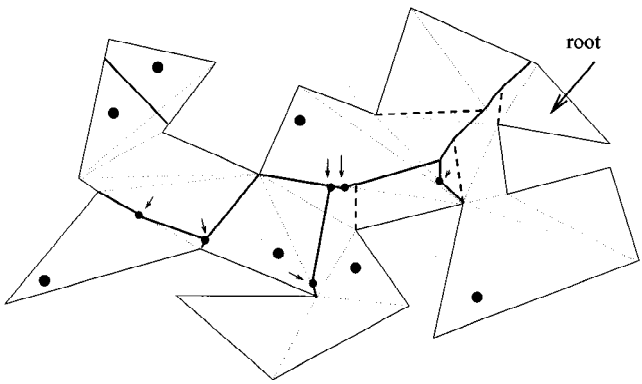

Fig. 5. $SD(P)$: border vertices are indicated by arrows.

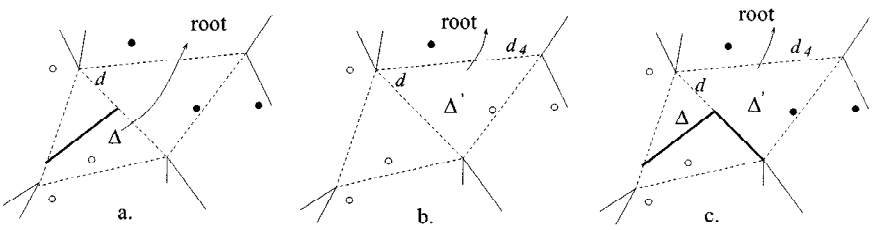

Fig. 6. (a) $Vor_{\Delta(d)}(S'(d))$, (b) $Vor_{\Delta'(d)}(S'(d_4))$, (c) $\mathcal{SD}'(P) \cap \{\Delta, \Delta'\}$.

$\mathcal{SD}'(P)$ has an *owner* (a site in S) and a distinct *anchor* (a polygon vertex or site in S). Recall that the anchor of a region is the last vertex on the shortest path from the owner to every point in the region, and it is weighted by its geodesic distance from the owner of the region. A region of anchor y, where $y \in V \cup S$, is denoted as $r(y)$, and the union of regions that have owner $s \in S$ is denoted as $R(s)$. In Figure 7 the shaded part is $R(s_i)$ and the region shaded darkest indicates $r(y) \in R(s_i)$.

Given a diagonal d, $\mathcal{SD}(d)$ denotes the part of $\mathcal{SD}(P)$ "below" d excluding border vertices and edges on d, i.e., $\mathcal{SD}(d) = \mathcal{SD}(P) \cap \{P(d) - d\}$. $\mathcal{SD}'(d)$ denotes the part of $\mathcal{SD}'(P)$ "above" d excluding border vertices and edges on d, i.e., $\mathcal{SD}'(d) = \mathcal{SD}'(P) \cap P'(d)$. There are two types of border vertices on d: border vertices incident to exactly one border edge which are referred to as *basic*, and border vertices incident to two border edges. By definition of border vertices, a nonbasic border vertex of $\mathcal{SD}(P)$ on d must be induced by a Voronoi edge in $Vor_{\Delta(d)}(S(d))$ which does not appear in $Vor_{\Delta'(d)}(S(d_4))$. Clearly, this Voronoi edge must be shared by $s_i, s_j \in S(d)$. On the other hand, a *basic* border vertex of $\mathcal{SD}(P)$ on d must be induced by a Voronoi edge in $Vor_{\Delta'(d)}(S(d_4))$ shared by $s_i \in S(d)$ and $s_j \in S'(d) \cap S(d_4)$. For example, in Figure 4(c) the border vertex is basic. Similarly in $\mathcal{SD}'(d)$, a nonbasic border vertex on d must be induced by a Voronoi edge in $Vor_{\Delta'(d)}(S'(d_4))$ which does not continue in $Vor_{\Delta(d)}(S'(d))$. A *basic* border vertex of $\mathcal{SD}'(P)$ on d must be induced by a Voronoi edge in $Vor_{\Delta(d)}(S'(d))$

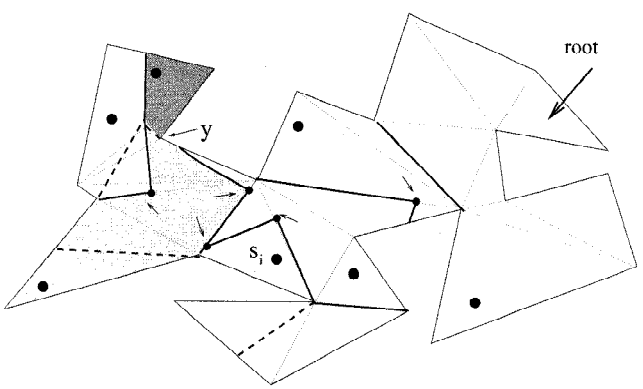

Fig. 7. $\mathcal{SD}'(P)$. The shaded portion is $R(s_i)$. The face shaded darkest is $r(y) \in R(s_i)$.

shared by the Voronoi regions of $s_i \in S'(d_4)$ and $s_j \in S(d_4) \cap S'(d)$. A basic border vertex on d must have degree two unless it coincides with a Voronoi vertex in the Voronoi diagrams of $\Delta(d)$ or $\Delta'(d)$. Clearly, nonbasic border vertices have degree at least three. As is shown later basic border vertices are useful for merging the two subdivisions.

The following lemmas show that the two subdivisions have linear complexity.

LEMMA 2. *The number of basic border vertices in $\mathcal{SD}(P)$ and $\mathcal{SD}'(P)$ is $O(n+k)$.*

PROOF. We only prove for $\mathcal{SD}(P)$; $\mathcal{SD}'(P)$ can be shown similarly.

Consider a triangle Δ of root diagonal d, and let d_1, d_2 be the remaining diagonals of Δ. We assume that border vertices along d_2 are ordered in the direction toward d. Let b be a basic border vertex on d_2 and let e be the incident border edge. The border vertex b is the intersection point of d_2 and the bisector $b(y_i, x_j)$, where y_i and x_j are the anchors of the regions in $\mathcal{SD}$ that are adjacent to e in Δ and $\Delta(d_2)$, respectively. Note that y_i is a vertex or site in $P(d_1) \cup \Delta$ and x_j is a vertex or site in $P(d_2)$. We say that b is induced by the pair (y_i, x_j). For simplicity we would also say b is induced by y_i or by x_j. Since $b(y_i, x_j)$ is a hyperbolic arc, it may intersect d_2 at most twice, i.e, pair (y_i, x_j) may induce at most two border vertices on d_2.

Let $Y(d_2)$ and $X(d_2)$ denote the set of vertices or sites in $P(d_1) \cup \Delta$ and $P(d_2)$, respectively, that induce basic border vertices on d_2. Because of the star-shape property of Voronoi cells, between any two points along $r(y_i) \cap d_2$ for $y_i \in Y(d_2)$, there can be no point in $r(y_k)$ for $y_k \in Y(d_2) - S(\Delta)$, $y_k \neq y_i$. (Recall that $r(y_k)$ denotes the region of anchor y_k.) Moreover, between any two points along $r(y_i) \cap d_2$ there can be no point in $r(s)$ for any site $s \in S(\Delta)$, since the bisector $b(y_i, s)$ may intersect d_2 at most once. (Note that $b(y_i, s)$ is either a straight line or the branch of a hyperbola closer to y_i.) Thus, the basic border vertices induced by an element of $Y(d_2)$ on d_2 must be consecutive. Similar arguments hold for the border vertices induced by $x_j \in X(d_2)$ and, thus, the number of basic border vertices along d_2 is $O(|Y(d_2)| + |X(d_2)|)$.

Now by the star-shape property of Voronoi cells, among the elements of $\{Y(d_2) - S(\Delta)\}$ at most one may have a region that intersects d. (Note that for $y_i, y_k \in \{Y(d_2) - S(\Delta)\}$ such that the border vertices induced by y_i on d_2 preceed those of y_k, $\pi(y_i, p)$ must pass through $r(y_k)$ for any point $p \in d$.) Hence, among the elements of $\{Y(d_2) - S(\Delta)\}$ at most one may induce border vertices on some diagonal of $P'(d) \cup d$. In other words, the number of elements in $Y(d_2)$ that may induce border vertices in $P'(d) \cup d$ is $O(|S(\Delta)| + 1)$. On the other hand, if an element $x_j \in X(d_2)$ induces a basic border vertex on d_2 with some $y_i \in \{Y(d_2) - S(\Delta)\}$ such that the region of y_i does not intersect d, then the region of x_j cannot intersect d. In other words, among the elements of $X(d_2)$ that induce basic border vertices with elements of $\{Y(d_2) - S(\Delta)\}$, at most one may have a region in $P'(d) \cup d$. Moreover, among the elements of $X(d_2)$ that induce basic border vertices with a site s of $S(\Delta)$, at most two may have regions intersecting d (those that induce the first and last border vertex of s on d_2). Therefore, the number of elements in $X(d_2)$ that induce basic border vertices in $P'(d) \cup d$ is bounded by $O(|S(\Delta)| + 1)$.

It is now clear that the elements of the set $Y(d_2) \cup X(d_2)$ that may induce border vertices in $P'(d) \cup d$ is bounded by $O(|S(\Delta)| + 1)$. Since the number of basic border vertices on a diagonal d_2 is $O(|Y(d_2)| + |X(d_2)|)$, summing over all diagonals of P we derive that the total number of basic border vertices in $\mathcal{SD}(P)$ is $O(n+k)$. □

LEMMA 3. *$\mathcal{SD}(P)$ and $\mathcal{SD}'(P)$ are planar subdivisions of size $O(n+k)$.*

PROOF. $\mathcal{SD}(P)$ and $\mathcal{SD}'(P)$ are planar by definition. The degree of any vertex in $\mathcal{SD}(P)$ and $\mathcal{SD}'(P)$ other than basic border vertices is at least three. (Under the general position assumption the degree is exactly three). However, by Lemma 2, the number of basic border vertices is $O(n+k)$. Ignoring border vertices of degree two (i.e., considering the two adjacent edges as one), we get a planar graph whose vertices are of degree at least three. By Euler's formula, the number of edges must be linear in the number of faces. Since each face has a distinct vertex or site as anchor, the number of faces, and therefore the number of edges, is $O(n+k)$. Considering the vertices that have been ignored, we only add one edge per vertex. Thus, the total number of edges is $O(n+k)$. □

5. Constructing $\mathcal{SD}(P)$. Consider a diagonal d, and the incident triangles $\Delta(d)$ and $\Delta'(d)$. For brevity, let Δ and Δ' denote $\Delta(d)$ and $\Delta'(d)$, respectively.

The Voronoi diagram of $S(\Delta)$ in Δ, $Vor_\Delta(S(\Delta))$, can be easily computed in the ordinary divide-and-conquer way by ignoring the polygon boundary. We only keep the part that is contained in Δ. Let $S(\Delta|d)$, $S(\Delta|d_1)$, and $S(\Delta|d_2)$ denote the list of sites that have a Voronoi cell intersecting d, d_1, and d_2, respectively, in $Vor_\Delta(S(\Delta))$. We assume that $S(\Delta|d)$, $S(\Delta|d_2)$, and $S(\Delta|d_1)$ are ordered in clockwise order around the boundary of Δ starting at v_1. Consider also the Voronoi diagrams of $S(d_1)$ and $S(d_2)$ in Δ, $Vor_\Delta(S(d_i))$, $i=1,2$. $Vor_\Delta(S(d_i))$, $i=1,2$, can be obtained from $\mathcal{SD}(d_i)$ using the extension procedure of [1].

In order to compute $\mathcal{SD}(d)$, we need $Vor_\Delta(S(d))$, the Voronoi diagram of $S(d)$ in Δ. $Vor_\Delta(S(d))$ can be computed by merging $Vor_\Delta(S(d_1))$, $Vor_\Delta(S(\Delta))$, and $Vor_\Delta(S(d_2))$. We define a subdivision, $\mathcal{SD}_0(d)$, to be used as an intermediate step in the construction of $\mathcal{SD}(d)$. $\mathcal{SD}_0(d)$ is defined as $\mathcal{SD}(d)$ would be defined if $S(d_2)$ were empty, i.e, $\mathcal{SD}_0(d) = \mathcal{SD}(d_1) \cup Vor_\Delta(S(d_1) \cup S(\Delta)) \cup \{\textit{border edges on } d_1\}$. Border edges on d_1 are induced by regions of sites in $S(\Delta)$ that are adjacent to d_1 in $Vor_\Delta(S(d_1) \cup S(\Delta))$. Let $\sigma(\Delta)$ denote the set of edges of $\mathcal{SD}(d)$ that are shared by regions $R(s_i)$ and $R(s_j)$, for $s_i \in S(d_1) \cup S(\Delta)$ and $s_j \in S(d_2)$. $\sigma(\Delta)$ consists of $\{b_g(S(d_1) \cup S(\Delta), S(d_2))\} \cap \Delta$ and the incident border edges on d_1 or d_2 (see Figure 8(a)). If $\{b_g(S(d_1) \cup S(\Delta), S(d_2))\} \cap \Delta = \emptyset$, then $\sigma(\Delta)$ is d_1 or d_2. Let $\tau(\Delta)$ denote the set of edges of $\mathcal{SD}_0(d)$ that are shared by regions $R(s_i)$ and $R(s_j)$, for $s_i \in S(d_1)$ and $s_j \in S(\Delta)$. Note that $\tau(\Delta)$ consists of $\{b_g(S(d_1), S(\Delta))\} \cap \Delta$ and the incident border edges on d_1 (if any) (see Figure 8(b)). Under Assumptions A and B, $\sigma(\Delta)$ and $\tau(\Delta)$ have the following properties (see Figures 8 and 10).

LEMMA 4. *If $S(\Delta) = \emptyset$, $\sigma(\Delta)$ is a simple path with one endpoint at vertex $v_{1,2}$ (the vertex incident to d_1 and d_2) and another on diagonal d; $\sigma(\Delta)$ has exactly one common point with d.*

PROOF. Since $S(d_1)$ and $S(d_2)$ are separable by d_1 and d_2, the (geodesic) bisector $b_g(S(d_1), S(d_2))$ must be a simple path with endpoints on the polygon boundary. Let x be the point of intersection of $b_g(S(d_1), S(d_2))$ and d (if any). Let $s_i \in S(d_1)$ and $s_j \in S(d_2)$ be the nearest neighbors of x. Since $s_i, s_j \notin \Delta$, $\pi(s_k, y)$ for any $s_k \in S(d_1)$

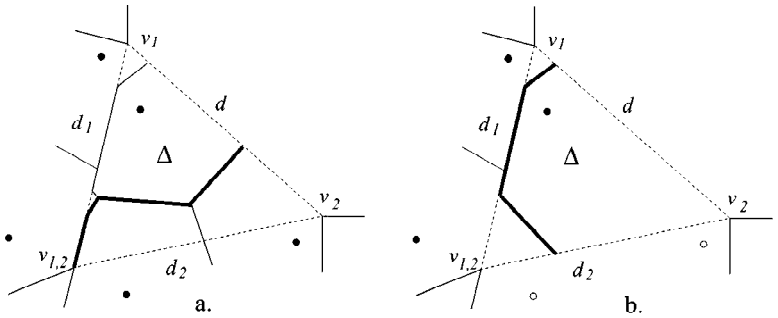

Fig. 8. (a) $\mathcal{SD}(d) \cap \Delta$; the thick curve is $\sigma(\Delta)$. (b) $\mathcal{SD}_0(d) \cap \Delta$; the thick curve is $\tau(\Delta)$.

and any point $y \in d$ between x and v_2 (the vertex incident to d and d_2) must intersect $\pi(s_j, x)$ (see Figure 9). Thus, by Lemma 1 no site in $S(d_1)$ can have a region in $\mathcal{SD}(P)$ intersecting d between x and v_2. Thus, $b_g(S(d_1), S(d_2))$ may intersect d at most once. However, it may intersect d_1 and d_2 a number of times. If $v_{1,2}$ belongs in the region of a site $s_k \in S(d_1)$ (resp. $s_l \in S(d_2)$), then s_k (resp. s_l) induces a border edge on d_2 (resp. d_1) incident to $v_{1,2}$. By definition, this border edge belongs to $\sigma(\Delta)$, i.e, one endpoint of $\sigma(\Delta)$ is $v_{1,2}$. Even if $v_{1,2} \in S$, it is assumed that $v_{1,2}$ belongs to only one of $S(d_1)$ or $S(d_2)$ and thus it induces a border edge on d_2 or d_1, respectively.

Let p be a point of $\sigma(\Delta)$ on d_2 other than the endpoints of d_2. If p is incident to $b_g(S(d_1), S(d_2))$, let $s_i \in S(d_1)$ and $s_j \in S(d_2)$ be the nearest neighbors of p. However, then s_i induces a border edge on d_2 incident to p, i.e., p cannot be an endpoint of $\sigma(\Delta)$. If $p \notin b_g(S(d_1), S(d_2))$, then p must belong in a border edge of d_2, i.e., p cannot be an endpoint of $\sigma(\Delta)$. Thus, $\sigma(\Delta)$ cannot have an endpoint on d_2 other than $v_{1,2}$ and v_2. Similarly, $\sigma(\Delta)$ cannot have an endpoint on d_1 other than $v_{1,2}$ and v_1. Since moreover $b_g(S(d_1), S(d_2))$ may intersect d at most once and $\sigma(\Delta)$ must have an endpoint on $v_{1,2}$, $\sigma(\Delta)$ must be a simple path with endpoints at $v_{1,2}$ and d. (Note that v_1 and v_2 are valid endpoints of $\sigma(\Delta)$.) □

LEMMA 5. *If $S(\Delta) \neq \emptyset$, $\sigma(\Delta)$ is a subset of edges of $\mathcal{SD}(d)$ with the following properties*:

- *$\sigma(\Delta)$ is a collection of disjoint simple paths, none of which is a cycle.*
- *There is one component of $\sigma(\Delta)$, referred to as the* initial *component, with one endpoint*

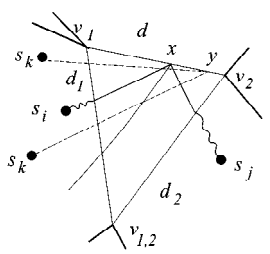

Fig. 9. $\pi(s_k, y)$ must intersect $\pi(s_j, x)$ for every $y \in d$ between x and v_2.

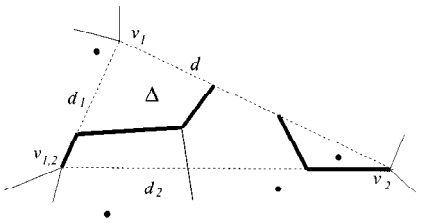

Fig. 10. $\sigma(\Delta)$.

at vertex $v_{1,2}$ and another on d. The endpoint of the initial component on d is the first endpoint of $\sigma(\Delta)$ on d in the direction from v_1 toward v_2.

- *The components of $\sigma(\Delta)$ other than the initial one have both their endpoints on d and are induced by sites in $S(\Delta)$ and $S(d_2)$. The endpoints of these components belong to edges shared by regions $R(s_i)$ and $R(s_j)$, for $s_i \in S(\Delta|d)$ and $s_j \in S(d_2)$.*
- *All components of $\sigma(\Delta)$ intersect d transversely.*

PROOF. $S(d_1) \cup S(\Delta)$ and $S(d_2)$ are separable by d_2, thus, the bisector $b_g(S(d_1) \cup S(\Delta), S(d_2))$ is a simple path with endpoints on the polygon boundary without any cycles. Following the same reasoning as in Lemma 4, $v_{1,2}$ must be an endpoint of $\sigma(\Delta)$, and $\sigma(\Delta)$ cannot have endpoints on d_1, d_2 other than $v_1, v_2, v_{1,2}$. Because of sites in $S(\Delta)$, unlike in Lemma 4, $b_g(S(d_1) \cup S(\Delta), S(d_2))$ may intersect d more than once, i.e., $b_g(S(d_1) \cup S(\Delta), S(d_2))$ may have more than one endpoint on d. In other words, $\sigma(\Delta)$ consists of one or more components, where one component has an endpoint at $v_{1,2}$, and all other component have their endpoints on d.

Let x_1 be the endpoint on d of the initial component of $\sigma(\Delta)$. If $x_1 \neq v_1$, let $s_i \in S(d_1) \cup S(\Delta)$ and $s_j \in S(d_2)$ be the nearest neighbors of x_1. Similarly as in Lemma 4, $\pi(s_k, y)$ for any $s_k \in S(d_1)$ must intersect $\pi(s_j, x_1)$ for any $y \in d$ between v_2 and x. Thus, in $\mathcal{SD}(P)$, no site of $S(d_2)$ has a region on d between v_1 and x_1, and no site of $S(d_1)$ has a region on d between x and v_2. Therefore x_1 is the first endpoint of $\sigma(\Delta)$ on d (ordered from v_1 to v_2). Moreover, the initial component of $\sigma(\Delta)$ is the only component that may border regions of $s_i \in S(d_1)$. Thus, the edges of any component of $\sigma(\Delta)$ other than the initial one must be shared by $R(s_i)$, $R(s_j)$ for $s_i \in S(\Delta)$ and $s_j \in S(d_2)$. Since the endpoints of a component, other than the initial one, lie on d, the edge incident to the endpoint must border the region of a site in $S(\Delta|d)$.

Since all sites lie on the same side of d, $b_g((S(d_1) \cup S(\Delta), S(d_2))$ cannot share a segment with d. Thus, $\sigma(\Delta)$ cannot share a segment with d since there are no border edges on d that are shared by regions of sites in $S(d)$. □

LEMMA 6. *$\tau(\Delta)$ is a subset of edges of $\mathcal{SD}_0(d)$ with the following properties*:

- *$\tau(\Delta)$ is a collection of disjoint simple paths with endpoints on $\{d, d_2\}$ without any cycles.*
- *Each component of $\tau(\Delta)$ contains an edge which is shared by regions $R(s_i)$ and $R(s_j)$, for $s_j \in S(\Delta|d_2) \cup S(\Delta|d)$ and $s_i \in S(d_1)$.*

PROOF. $S(d_1)$ and $S(\Delta)$ are on opposite sides of d_1 thus, under the general position assumptions, the bisector $b_g(S(d_1), S(\Delta))$ is a simple path with endpoints on the polygon boundary. Since there are no border edges on d or d_2 the only common points of $\tau(\Delta)$ with d, d_2 are those on $b_g(S(d_1), S(\Delta))$. Since $b_g(S(d_1), S(\Delta))$ must intersect d, d_2 transversely [1] (if it intersects at all), the only points of $\tau(\Delta)$ on d, d_2 are the endpoints of $\tau(\Delta)$. Following the same reasoning as in Lemma 4, $\tau(\Delta)$ may not have endpoints on d_1 other than $v_{1,2}$, v_1. Since all endpoints of $\tau(\Delta)$ lie on $\{d, d_2\}$, the edge incident to an endpoint on d must be shared by $R(s_i)$ and $R(s_j)$, where $s_i \in S(d_1)$ and $s_j \in S(\Delta|d)$, and the edge incident to an endpoint on d_2 must be shared by $R(s_i)$ and $R(s_j)$, where $s_i \in S(d_1)$ and $s_j \in S(\Delta|d_2)$. □

LEMMA 7. *The number of connected components of $\sigma(\Delta)$ and $\tau(\Delta)$ is $O(1+|S(\Delta)|)$.*

PROOF. By the star-shape property of Voronoi cells, a site $s_i \in S(\Delta)$ can induce an edge in at most one component of $\sigma(\Delta)$ or $\tau(\Delta)$. Thus, by Lemmas 4–6 the claim follows. □

$\mathcal{SD}(P)$ is kept as a doubly connected edge list (DCEL) [18]. For each diagonal d, we compute $\mathcal{SD}(d)$. Let $T(d)$ be the ordered list of regions of $\mathcal{SD}(d)$ intersecting d, arranged in a search tree that supports insertions, deletions, splitting, and merging in time logarithmic of its size (for example, the red–black tree of [10]). As in [1], every triangle is associated with a bucket to hold bisector intersection points. The algorithm is as follows:

- For every triangle $\Delta = \Delta(d)$ do
  - — Compute the ordinary Voronoi diagram of $S(\Delta)$, $Vor(S(\Delta))$, ignoring the polygon boundary.
  - — Locate in the triangulation the Voronoi vertices of $Vor(S(\Delta))$ that do not lie in Δ, and store them in the associated buckets. Voronoi vertices lying in $P'(d)$ are considered for constructing $\mathcal{SD}(P)$. Voronoi vertices lying in $P(d_1)$ and $P(d_2)$ are considered for constructing $\mathcal{SD}'(P)$.
  - — Keep the part of $Vor(S(\Delta))$ that lies in Δ ($Vor_\Delta(S(\Delta))$). Identify $S(\Delta|d_1)$, $S(\Delta|d_2)$, $S(\Delta|d)$, the ordered list of regions in $Vor_\Delta(S(\Delta))$ intersecting d_1, d_2, and d, respectively.
- Call procedure *Construct-$\mathcal{SD}(d)$* where d is one of the two boundary edges of the root triangle. Procedure *Construct-$\mathcal{SD}(d)$* does a postorder traversal of the dual tree.

Algorithm *Construct-$\mathcal{SD}(d)$*
Input: A triangulated polygon P and a diagonal d.
Output: $\mathcal{SD}(d)$ and $T(d)$.

begin

1. If d is a boundary edge such that there is no triangle "below" d, then, if there is no site on d, let $T(d) = \emptyset$. If d is a boundary edge and $S \cap d \neq \emptyset$, then initialize $T(d)$ to the ordered list of regions induced by the sites on d. (If an endpoint v_i, $i = 1, 2$, of d is included in S and v_i has been

marked do not consider v_i as a site in the computation of $T(d)$. If $v_i \in S$ but v_i is not marked, then mark v_i and include it in the computation of $T(d)$.) Let $\mathcal{SD}(d) = d$. Return $T(d)$ and $\mathcal{SD}(d)$.

2. Let $\Delta = \Delta(d)$.
   (a) Recursively call *Construct-*$\mathcal{SD}(d_1)$. Let $T(d_1)$ be the ordered list of regions of $\mathcal{SD}(d_1)$ intersecting d_1 returned by the recursive call.
   (b) If $T(d_1) \neq \emptyset$, extend $T(d_1)$ into Δ using the algorithm of [1]. Split the extended tree at v_2 (the vertex shared by d and d_2). Let $T(d_1|d)$ be the part of $T(d_1)$ intersecting d, and let $T(d_1|d_2)$ be the part of $T(d_1)$ intersecting d_2. Store $T(d_1|d_2)$ in order to be used in the construction of $\mathcal{SD}'(P)$. While extending $T(d_1)$, compute $\mathcal{SD}(d_1) \cup Vor_\Delta(S(d_1))$.
   (c) Recursively call *Construct-*$\mathcal{SD}(d_2)$. Let $T(d_2)$ be the ordered list of regions of $\mathcal{SD}(d_2)$ intersecting d_2 returned by the recursive call.
   (d) If $T(d_2) \neq \emptyset$, extend $T(d_2)$ into Δ using the algorithm of [1]. Split the extended tree at v_1 (the vertex shared by d and d_1). Let $T(d_2|d)$ be the part of $T(d_2)$ intersecting d, and let $T(d_2|d_1)$ be the part of $T(d_2)$ intersecting d_1. Store $T(d_2|d_1)$ in order to be used in the construction of $\mathcal{SD}'(P)$. While extending $T(d_2)$, compute $\mathcal{SD}(d_2) \cup Vor_\Delta(S(d_2))$.
   (e) If $S(\Delta) = \emptyset$, merge $\{\mathcal{SD}(d_1) \cup Vor_\Delta(S(d_1))\}$ and $\{\mathcal{SD}(d_2) \cup Vor_\Delta(S(d_2))\}$ to get $\mathcal{SD}(d)$ (basically compute $\sigma(\Delta)$). While merging, update $T(d_1|d)$ and $T(d_2|d)$ every time a region is trimmed by $\sigma(\Delta)$. At the end, merge $T(d_1|d)$ and $T(d_2|d)$ to derive $T(d)$. Compute the intersection of the bisector of $\sigma(\Delta)$ intersecting d with its neighboring bisectors in $T(d)$. Locate the intersections in the triangulation (without checking feasibility) and assign them to the corresponding buckets.
   (f) If $S(\Delta) \neq \emptyset$, do the following:
      i. Merge $\{\mathcal{SD}(d_1) \cup Vor_\Delta(S(d_1))\}$ and $Vor_\Delta(S(\Delta))$ to derive $\mathcal{SD}_0(d)$ (basically compute $\tau(\Delta)$). While merging, update $T(d_1|d)$ and compute $T_0(d)$. ($T_0(d)$ is the list of regions of $\mathcal{SD}_0(d)$ intersecting d.)
      ii. Compute the points of intersection of the bisectors of $\tau(\Delta)$ intersecting d with the neighboring bisectors in $T_0(d)$, locate them in the triangulation, and assign them to the corresponding buckets.
      iii. Merge $\mathcal{SD}_0(d)$ and $\{\mathcal{SD}(d_2) \cup Vor_\Delta(S(d_2))\}$ to derive $\mathcal{SD}(d)$ (basically compute $\sigma(\Delta)$). While merging, update $T_0(d)$ and $T(d_2|d)$ and merge to compute $T(d)$.
      iv. Compute the points of intersection of the bisectors of $\sigma(\Delta)$ intersecting d with the neighboring bisectors in $T(d)$, locate them in the triangulation, and assign them to the corresponding buckets.
   (g) Return $T(d)$ and $\mathcal{SD}(d)$.

end.

To extend $\mathcal{SD}(d_1)$ and $\mathcal{SD}(d_2)$ into Δ we use the *extension* procedure of [1], which given a triangulated polygon separated by a triangulation diagonal e into subpolygons P_1 and P_2, and the planar map induced by the Voronoi diagram of a set of sites in P_1, finds the planar map induced on P by the Voronoi diagram of the same set of sites. We briefly review here the main ideas of this procedure.

Each triangle in the triangulation of P_2 is associated with a bucket to hold bisector intersection points. From the Voronoi diagram in P_1, extract the ordered list of regions adjacent to e, and arrange them in a search tree T. For each pair of bisectors bounding a single region in T compute their point of intersection (without checking feasibility), locate them in the triangulation of P_2, and store them in the associated buckets. (Here we are given $T(d_1)$ and $T(d_2)$, thus we do not extract the regions; intersections of neighboring bisectors have also been computed.) Move from triangle to neighboring triangle starting at the triangle adjacent to e. Upon entering a triangle through edge e, consider the extension segments of the endpoints of e from their owner in the diagram. (Here we only move into Δ.) Compute their point of intersection with the neighboring bisectors and locate them in the triangulation. Process the intersection points in the bucket of the current triangle in order of increasing distance from e. Process only the feasible intersection points. Each feasible intersection point corresponds to a region deletion and the generation of a new bisector. When construction inside the current triangle is complete, split T at the apex of the triangle giving two subtrees, one for each of the remaining edges of the current triangle. For more details see [1]. Slightly modifying the analysis of [1] we have the following.

LEMMA 8. *The time to extend $T(d_i)$, $i = 1, 2$, into Δ (and therefore compute $\mathcal{SD}(d_i) \cup \mathrm{Vor}_\Delta(S(d_i))$) is*

$$O((|I_{\mathrm{f}}(\Delta)| + 1)\log|T(d_i)| + |I_{\mathrm{new}}(\Delta)|\log n + |I_{\mathrm{inf}}(\Delta)| + (|I(\Delta)|\log|I(\Delta)|)),$$

where $I_{\mathrm{f}}(\Delta)$ denotes the feasible intersection points in the bucket associated with Δ, $I_{\mathrm{new}}(\Delta)$ denotes the intersection points generated while extending in Δ, $I_{\mathrm{inf}}(\Delta)$ denotes the infeasible intersection points located in the bucket associated with Δ, and $I(\Delta) = I_{\mathrm{f}}(\Delta) \cup I_{\mathrm{inf}}(\Delta)$.

To merge $\mathcal{SD}(d_1) \cup \mathit{Vor}_\Delta(S(d_1))$ with $\mathit{Vor}_\Delta(S(\Delta))$, we construct $\tau(\Delta)$, split the diagrams along the components of $\tau(\Delta)$, and clean up dominated regions. Similarly, to merge $\mathcal{SD}(d_2) \cup \mathit{Vor}_\Delta(S(d_2))$ with $\mathcal{SD}_0(d)$ (resp. $\mathcal{SD}(d_1) \cup \mathit{Vor}_\Delta(S(d_1))$ if $S(\Delta) = \emptyset$) we construct $\sigma(\Delta)$, split along the components of $\sigma(\Delta)$, and clean up. Lemmas 4–6 provide the means to identify a point on every component of the merge curves. In particular, the initial component of $\sigma(\Delta)$ has an endpoint at vertex $v_{1,2}$, while the other components of $\sigma(\Delta)$ (if any) have their endpoints on d. On the other hand, all components of $\tau(\Delta)$ have their endpoints on $d \cup d_2$. The components of $\sigma(\Delta)$ (resp. $\tau(\Delta)$) can be ordered according to the order that their endpoints appear on d (resp. $\{d, d_2\}$) starting at v_1 and moving clockwise. We refer to the endpoint of a component of $\sigma(\Delta)$ (resp. $\tau(\Delta)$) closer to v_1 as the *first* endpoint of the component, and to the endpoint closer to v_2 (resp. $v_{1,2}$) as the *second* endpoint.

We can identify the first endpoint of a component of $\sigma(\Delta)$ (other than the initial one) as follows: Let x_k denote the first endpoint of the component of $\sigma(\Delta)$ under consideration.

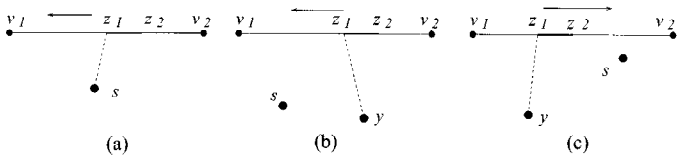

Fig. 11. Binary search in $T(d_2|d_s)$.

x_k is the intersection of bisector $b(s,u)$ with d, where $s \in S(\Delta|d)$ and u is a vertex or site in $P(d_2)$. We say that x_k is induced by s and u. To identify s and u, let s be the first site in $S(\Delta|d)$ following the site that has been involved in the construction of the previous component. (If no site in $S(\Delta|d)$ has been involved in the construction of a prior component, let s be the first site in $S(\Delta|d)$.) Let $\overline{d_s}$ denote the segment on d that lies in the Voronoi cell of s in $Vor_\Delta(S(\Delta))$, i.e., $\overline{d_s} = Vr(s) \cap d$. If the region of s in $\mathcal{SD}(d)$ intersects d, then clearly $x_k \in \overline{d_s}$. Let $T(d_2|d_s)$ denote the part of $T(d_2|d)$ intersecting $\overline{d_s}$. $T(d_2|d_s)$ can be extracted from $T(d_2|d)$ in $O(\log|T(d_2|d)|)$ time by splitting $T(d_2|d)$ at the endpoints of $\overline{d_s}$. Using binary search in $T(d_2|d_s)$, we can determine in $O(\log|T(d_2|d_s)|)$ time whether x_k is induced by s, and, if so, determine the vertex or site $u \in P(d_2)$ that also induces x_k. Given a region of anchor y in $T(d_2|d_s)$, let z_1 and z_2 be the endpoints of the region on $\overline{d_s}$, ordered from v_1 to v_2. We can determine whether the region of u is before or after the region of y in $T(d_2|d_s)$ or whether $y = u$ as follows:

- If z_1 is closer to s than to y (i.e., $d(s,z_1) < w(y) + d(y,z_1)$), then x_k must be induced by s. Thus, x_k must precede z_1 on $\overline{d_s}$, i.e., the region of u, $r(u)$, must precede the region of y, $r(y)$, in $T(d_2|d_s)$ (see Figure 11(a)).
- If s lies on the opposite side of segment $\overline{yz_1}$ to z_2, then x_k must be before z_1, i.e., $r(u)$ must be before $r(y)$ in $T(d_2|d_s)$ (see Figure 11(b)).
- If z_1 is closer to y than s (i.e., $w(y) + d(y,z_1) < d(s,z_1)$) and s lies on the same side of segment $\overline{yz_1}$ as z_2, then x_k must be after z_1 (see Figure 11(c)). If moreover z_2 is closer to s than y (i.e., $d(s,z_2) < w(y) + d(y,z_2)$), then x_k must lie between z_1 and z_2, i.e., $u = y$. Otherwise $r(u)$ must be after $r(y)$ in $T(d_2|d_s)$; if y is the anchor of the last region in $T(d_2|d_s)$ then s does not induce x_k; update s to the next site in $S(\Delta|d)$.

The time required to determine s, u (i.e., determine x_k) is $O(\log(|T(d_2|d)|)$ times the number of sites in $S(\Delta|d)$ that get visited. If a site $s \in S(\Delta|d)$ is considered during the construction of one component of $\sigma(\Delta)$, then s is not considered again for any other component. Thus, overall, the time to identify an endpoint of every component of $\sigma(\Delta)$ is $O(|S(\Delta|d)|\log(|T(d_2|d)|))$.

The first endpoint of every component of $\tau(\Delta)$ can be identified in the same way as for $\sigma(\Delta)$. First we initialize s at the first site in $S(\Delta|d) \cup S(\Delta|d_2)$. If v_1 is closer to s than the anchor of v_1 in $T(d_1|d)$, then the first component of $\tau(\Delta)$ has an endpoint at v_1. Otherwise, the endpoint of the first component can be determined by binary search similarly to $\sigma(\Delta)$. Thus, the time to identify an endpoint of every component of $\tau(\Delta)$ is $O(|S(\Delta|d) \cup S(\Delta|d_2)|\log(|T(d_1|d)|))$.

Once a point on a component of $\sigma(\Delta)$ (resp. $\tau(\Delta)$) has been identified, the component can be traced in a way similar to tracing the merge curve of the ordinary Voronoi diagram.

When a component of $\sigma(\Delta)$ (resp. $\tau(\Delta)$) hits diagonals d_2 or d_1 (resp. diagonal d_1) it continues along the same diagonal tracing border edges, until it reaches a point on bisector $b_g(S(d_2), S(d_1) \cup S(\Delta))$ (resp. $b_g(S(d_1), S(\Delta))$), in which case it continues in the interior of Δ as $b_g(S(d_2), S(d_1) \cup S(\Delta))$ (resp. $b_g(S(d_1), S(\Delta))$). Note that for a point x on a border edge of $\sigma(\Delta)$ along d_1, $d_g(x, S(d_2)) \leq d_g(x, S(d_1) \cup S(\Delta))$, and for a point x' on a border edge of $\sigma(\Delta)$ along d_2, $d_g(x', S(d_1) \cup S(\Delta)) \leq d_g(x', S(d_2))$. Furthermore, for a point x on a border edge of $\tau(\Delta)$, $d_g(x, S(\Delta)) \leq d_g(x, S(d_1))$. By Lemmas 4–6, when $\sigma(\Delta)$ hits d and when $\tau(\Delta)$ hits $\{d, d_2\}$, the tracing of the component ends.

While computing the merge curves, we also update $T(d_1|d)$ and $T(d_2|d)$ in order to compute $T(d)$. In particular, we compute $T_0(d)$ as follows: first let $T_0(d) = \emptyset$. After the first endpoint x_k of a component of $\tau(\Delta)$ has been identified, we split $T(d_1|d)$ at x_k. Update $T_0(d)$ by adding the part of $T(d_1|d)$ before x_k, and let $T(d_1|d)$ be the part following x_k. While tracing the component of $\tau(\Delta)$, update $T(d_1|d)$ by deleting any region that gets trimmed by $\tau(\Delta)$. When the second endpoint of the component is reached, add to $T_0(d)$ the part of $S(\Delta|d)$ between s_{k_i} and s_{k_j}, where s_{k_i} and s_{k_j} are the sites in $S(\Delta|d)$ that induce the first and the second endpoints of the component, respectively. When all components of $\tau(\Delta)$ have been traced, add to $T_0(d)$ the remainder of $T(d_1|d)$ (if not empty). $T(d)$ can be similarly computed by $T_0(d)$ and $T(d_2|d)$ (resp. (if $S(\Delta) = \emptyset$, $T(d_1|d)$ and $T(d_2|d)$).

LEMMA 9. *Given $Vor_\Delta(S(d_1))$, $Vor_\Delta(S(\Delta))$, and a point on every component of $\tau(\Delta)$, the time to trace $\tau(\Delta)$ is proportional to the size of $\tau(\Delta)$ plus $|S(\Delta)|$.*

PROOF. The total time to trace a component of $\tau(\Delta)$, given an initial point, is proportional to the complexity of the Voronoi cells in $Vor_\Delta(S(d_1))$ and $Vor_\Delta(S(\Delta))$ that the component intersects. As was shown in [1], the planar graph induced by $Vor_\Delta(S(d_1))$ is a *forest* with leaves on d_1 and roots on d and d_2. Since $Vor_\Delta(S(d_1))$ is a *forest*, the total complexity of the Voronoi cells of $Vor_\Delta(S(d_1))$ that $\tau(\Delta)$ intersects is proportional to the number of cells intersected by $\tau(\Delta)$. Thus, the time to trace a component of $\tau(\Delta)$, given an initial point, is proportional to the number of Voronoi cells in $Vor_\Delta(S(d_1))$ that the component intersects, plus the complexity of the cells in $Vor_\Delta(S(\Delta))$ that are intersected by the component. On the other hand, the size of $\tau(\Delta)$ is proportional to the number of Voronoi cells of $Vor_\Delta(S(d_1))$ and $Vor_\Delta(S(\Delta))$ that $\tau(\Delta)$ intersects. Since the total complexity of Voronoi cells in $Vor_\Delta(S(\Delta))$ is $O(|S(\Delta)|)$, the lemma follows. □

LEMMA 10. *Given $\mathcal{SD}_0(d)$, $Vor_\Delta(S(d_2))$, and a point on every component of $\sigma(\Delta)$, the time to trace $\sigma(\Delta)$ is proportional to the size of $\sigma(\Delta)$ plus $|S(\Delta)|$.*

PROOF. The total time to trace a component of $\sigma(\Delta)$, given an initial point, is proportional to the complexity of the Voronoi cells in $\mathcal{SD}(d)$ and $Vor_\Delta(S(d_2))$ that the component intersects. The planar graphs induced by $Vor_\Delta(S(d_2))$ and $Vor_\Delta(S(d_1))$ are *forests* with leaves on d_2 and d_1, respectively, and roots on $d \cup d_1$ and $d \cup d_2$, respectively. Thus, the subgraph induced by the Voronoi cells of $\mathcal{SD}_0(d) \cap \Delta$ that have as owner a site in $S(d_1)$ is also a forest. Hence, the total complexity of the cells of $\mathcal{SD}_0(d) \cap \Delta$ that $\sigma(\Delta)$

intersects is proportional to the number of cells of owner $s_i \in S(d_1)$ that $\sigma(\Delta)$ intersects plus the total complexity of the cells of owner $s_j \in S(\Delta)$ that $\sigma(\Delta)$ intersects. Moreover, the total complexity of the cells of $Vor_\Delta(S(d_2))$ that $\sigma(\Delta)$ intersects is proportional to their number. Since the size of $\sigma(\Delta)$ is proportional to the number of cells it intersects in $\mathcal{SD}_0(d) \cap \Delta$ and $Vor_\Delta(S(d_2))$, and since the time to trace $\sigma(\Delta)$ is proportional to the total complexity of the cells it intersects in both diagrams, the lemma follows. □

Lemmas 9 and 10, the fact that $\sigma(\Delta)$ and $\tau(\Delta)$ may have at most $O(1 + |S(\Delta)|)$ components, and the fact that the size of $\mathcal{SD}(P)$ is $O(n + k)$ imply that the total time spent for tracing merge curves while constructing $\mathcal{SD}(P)$ is $O(n + k)$. Considering also the time to clean up, identify initial points on the components of the merge curves, generate and locate intersection points, update, split, and merge $T(d_2|d)$, $T(d_1|d)$, and $S(\Delta|d)$ we derive the following lemma.

LEMMA 11. *The total time spent for merging while constructing $\mathcal{SD}(P)$ is $O((n + k)\log(n + k))$.*

PROOF. Clearly, the size of $T(d)$ for any diagonal d cannot exceed $(n + k)$ since any region of $T(d)$ has a distinct anchor. When a region of anchor $y \in S \cup V$ gets deleted from $T(d)$, it is never considered again. Thus the number of region deletions from $T(d)$ for any diagonal d throughout the procedure cannot exceed $(n + k)$. To compute $T(d)$ from $T(d_1|d)$, $T(d_2|d)$, and $S(\Delta|d)$ we perform at most one splitting and merging for any endpoint of $\tau(\Delta)$ and $\sigma(\Delta)$. Since $\tau(\Delta)$ and $\sigma(\Delta)$ have at most $O(1 + |S(\Delta)|)$ components, we perform at most $O(1 + |S(\Delta)|)$ splittings and mergings per triangle, i.e., at most $O(n + k)$ splittings and mergings throughout the procedure. Any update of $T(d)$, $T(d_1|d)$, or $T(d_2|d)$ (i.e., deletion, insertion, splitting, or merging) can be done in time logarithmic of its size. Thus, the total time spent for updates of $T(d)$, $T(d_1|d)$, or $T(d_2|d)$ throughout the procedure is $O((n + k)\log(n + k))$.

To identify initial points on the components of $\sigma(\Delta)$ (resp. $\tau(\Delta)$) we spend $O(|S(\Delta|d)|\log|T(d_2|d)|)$ (resp. $O(|S(\Delta|d) \cup S(\Delta|d_2)|\log|T(d_1|d)|)$) time. Thus, the total time spent to identify initial points on components of $\sigma(\Delta)$ and $\tau(\Delta)$ throughout the procedure is $O(k\log(n + k))$.

When $T_0(d)$ and $T(d)$ are computed, intersection points of the new neighboring bisectors are located in the triangulation. As Lemma 12 shows, the total number of intersection points generated is $O(n + k)$. Point location in the triangulation takes $O(\log(n + k))$ time [12]; thus overall $O((n + k)\log(n + k))$ time to locate intersection points.

The time to clean up after the merge curves are traced is proportional to the number of edges to be discarded, which is clearly $O(n+k)$. Thus, tracing merge curves throughout the procedure takes $O(n + k)$ time.

The time bound follows. □

LEMMA 12. *The total number of bisector intersection points generated during the construction of $\mathcal{SD}(P)$ is $O(n + k)$.*

PROOF. Bisector intersection points are generated at the following steps: (1) while

constructing $Vor(S(\Delta))$, (2) during an extension procedure, and (3) at the last step of merging.

Since $Vor(S(\Delta))$ may contain $O(|S(\Delta)|)$ Voronoi vertices, we have at most $O(k)$ intersection points because of (1). During the extension procedure in Δ, bisector intersection points may be generated because of a region deletion or insertion. There are at most two insertions while extending $T(d_1)$ or $T(d_2)$, one for each endpoint of d_1 and d_2. There may be $O(|T(d_1)|)$ deletions while extending $T(d_1)$, and $O(|T(d_2)|)$ deletions while extending $T(d_2)$. Since any region deleted is never considered again, the total number of deletions while extending is $O(n+k)$. Thus, the total number of bisector intersection points generated while extending is $O(n+k)$. At the last step of merging, at most two intersection points may be generated for each component of $\sigma(\Delta)$ and $\tau(\Delta)$. Since there are at most $O(1+|S(\Delta)|)$ components for each Δ, the total number of intersection points generated because of (3) is $O(n+k)$. □

The total time spent for extending Voronoi diagrams from triangle to triangle depends (by Lemma 8) on the total number of intersection points generated throughout the procedure. By Lemmas 8 and 12 we derive that the total time spent for extending during the construction of $\mathcal{SD}(P)$ is $O((n+k)\log(n+k))$. Thus, we have the following lemma.

LEMMA 13. *$\mathcal{SD}(P)$ can be constructed in $O((n+k)\log(n+k))$ time.*

6. Constructing $\mathcal{SD}'(P)$. Consider a polygon diagonal d, and the incident triangles $\Delta = \Delta(d)$ and $\Delta' = \Delta'(d)$. Let $\sigma'(\Delta)$ denote the set of edges of $\mathcal{SD}'(P)$ that are shared by pairs of regions $R(s_i), s_i \in S'(d_4)$, and $R(s_j), s_j \in S(d_3) \cup S(\Delta')$. By the definition of $\mathcal{SD}'(P)$, sites in $S'(d)$ cannot induce border edges on d_1 and d_2. Thus, $\sigma'(\Delta)$ (Figure 12) consists of the part of the bisector $b_g(S'(d_4), S(d_3) \cup S(\Delta'))$ that lies in Δ, union the border edges on d induced by sites in $S(d_3) \cup S(\Delta')$. Under the general position assumption, $\sigma'(\Delta)$ has the following properties.

LEMMA 14. *If $S(\Delta') = \emptyset$, $\sigma'(\Delta)$ is either empty or $\sigma'(\Delta)$ is a simple path with one endpoint at v_2 and the other on $d_1 \cup d_2$.*

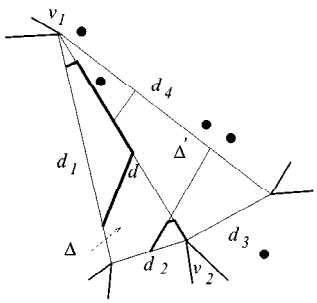

Fig. 12. $\sigma'(\Delta)$.

PROOF. Consider $Vor_{\Delta}(S'(d))$. If the bisector $b_g(S'(d_4), S(d_3))$ does not intersect d, then d belongs entirely to Voronoi cells of either $S'(d_4)$ or $S(d_3)$. If d belongs entirely to $S'(d_4)$, then $\sigma'(\Delta) = \emptyset$. If d belongs entirely to cells of $S(d_3)$, then every cell induces a border edge on d, i.e., $\sigma'(\Delta) = d$. In this case, $\sigma'(\Delta)$ is trivially a simple path with one endpoint on v_2 and the other on d_1.

Following the same reasoning as in Lemma 4, we can show that $b_g(S'(d_4), S(d_3))$ intersects d and $d_1 \cup d_2$ at most once. If $b_g(S'(d_4), S(d_3))$ intersects d, then the part of d between v_2 and the intersection point must be covered by border edges induced by $S(d_3)$ (since this part of d belongs to regions of sites in $S(d_3)$). Thus, in this case, v_2 is an endpoint of $\sigma'(\Delta)$. Since $b_g(S'(d_4), S(d_3))$ intersects d at most once, it must also intersect $d_1 \cup d_2$ exactly once in order to exit Δ. Since there are no border edges induced by $S'(d)$ on d_1, d_2, the lemma follows. □

LEMMA 15. *If $S(\Delta') \neq \emptyset$, $\sigma'(\Delta)$ (if not empty) is a collection of disjoint simple paths with the following properties*:

- *At most one component of $\sigma'(\Delta)$ may contain an edge shared by $R(s_i)$ and $R(s_j)$ for $s_i \in S(d_3)$ and $s_j \in S'(d_4)$. If there is such a component, it is referred to as the* initial *component of $\sigma'(\Delta)$, and has its starting point at vertex v_2 and its endpoint on $d_1 \cup d_2$.*
- *Every component of $\sigma'(\Delta)$, other than the initial one, consists of edges shared by $R(s_i)$ and $R(s_j)$, for $s_i \in S(\Delta'|d)$ and $s_j \in S(d_4)$, and has exactly two endpoints on $d_1 \cup d_2$.*

PROOF. Since $S'(d_4)$ and $S(d_3) \cup S(\Delta')$ are separable by d_4, the bisector $b_g(S'(d_4), S(d_3) \cup S(\Delta'))$ is a simple path with endpoints on the polygon boundary (under Assumptions A and B) [1]. Because $S(\Delta') \neq \emptyset$, $b_g(S'(d_4), S(d_3) \cup S(\Delta'))$ may intersect d more than once. However, there may be at most one component of $\Delta \cap b_g(S'(d_4), S(d_3) \cup S(\Delta'))$ that involves sites in $S(d_3)$ (similar to Lemma 14). If $v_2 \in R(s_i), s_i \in S(d_3) \cup S(\Delta')$, then s_i induces a border edge on d incident to v_2, i.e., $\sigma'(\Delta)$ has an endpoint on v_2. If $v_2 \in R(s_j), s_j \in S'(d_4)$ then for the same reasons as in Lemma 14 no site in $S(d_3)$ may claim a region on d. In that case, $\sigma'(\Delta)$ has no initial component. Note that due to sites in $S(\Delta')$, $\sigma'(\Delta)$ may have no initial component but still have other components. Similarly as in Lemma 14, the initial component of $\sigma'(\Delta)$ (if any) is a simple path with one endpoint at v_2 and the other on $d_1 \cup d_2$.

It can be shown similarly to Lemma 4 that $\sigma'(\Delta)$ cannot have an endpoint on d other than v_2 or v_1; thus the endpoints must lie on $d_1 \cup d_2$. A component of $\sigma'(\Delta)$ other than the initial one consists of edges shared by regions of $s_i \in S'(d_4)$ and $s_j \in S(\Delta')$. However, any such $s_j \in S(\Delta')$ must have a region in $Vor_{\Delta'}S(\Delta')$ that intersects d. In other words, $s_j \in S(\Delta|d)$. □

Let $T'_1(d)$ denote the ordered list of regions of $\mathcal{SD}'(d)$ adjacent to d arranged in a search tree, e.g., a red–black tree [10]. The regions of $T'_1(d)$ belong to sites in $S'(d_4)$. Let $T'_2(d)$ denote the ordered list of regions of $Vor_{\Delta'}(S(d_3))$ adjacent to d. $T'_2(d)$ has been computed while constructing $\mathcal{SD}(P)$. Let $T'(d)$ denote the ordered list of regions in $Vor_{\Delta}(S'(d))$ adjacent to d. Every region of $T'(d)$ that belongs to a site in $S'(d_3) \cup S(\Delta')$

induces a border edge on $\sigma'(\Delta) \cap d$. Lemmas 14 and 15 provide the means for constructing $\sigma'(\Delta) \cap d$ efficiently.

The algorithm to construct $\mathcal{SD}'(P)$ performs a preorder traversal of the dual tree of the triangulation. Given a triangle Δ of root diagonal d ($\Delta = \Delta(d)$), let $\mathcal{SD}'(\Delta)$ be the part of $\mathcal{SD}'(P)$ that lies in Δ including the edges on d but excluding any border edges on d_1 and d_2. We initialize d to one of the boundary edges of the root triangle. Let $\Delta = \Delta(d)$ be the root triangle. Let $T_1'(d) = \emptyset$, $T_2'(d) = \emptyset$. Call procedure *Construct-*$\mathcal{SD}'(\Delta(d))$.

Algorithm *Construct-*$\mathcal{SD}'(\Delta)$
Input: $\Delta = \Delta(d)$, $T_1'(d)$, $T_2'(d)$, $S(\Delta|d)$.
Output: $\mathcal{SD}'(\Delta)$, $T_1'(d_1)$, $T_1'(d_2)$.

Begin

1. Compute $\{\sigma'(\Delta) \cap d\}$. At the same time update $T_1'(d)$ by deleting regions that do not expand into Δ. Each component of $\{\sigma'(\Delta) \cap d\}$ partitions $T_1'(d)$ into disjoint parts. Thus, split $T_1'(d)$ at the beginning of each component.
2. Compute $T'(d)$ by merging the remaining components of $T_1'(d)$, with the regions induced by the border edges of $\sigma'(\Delta) \cap d$.
3. Extend $T'(d)$ in Δ using the extension procedure of [1]. While extending, compute $\mathcal{SD}'(\Delta)$.
4. Split the extended $T'(d)$ at $v_{1,2}$. Let $T_1'(d_1)$ be the part adjacent to d_1, and let $T_1'(d_2)$ be the part adjacent to d_2.
5. If d_1 is not a boundary edge, call recursively *Construct-*$\mathcal{SD}'(\Delta(d_1))$.
6. If d_2 is not a boundary edge, call recursively *Construct-*$\mathcal{SD}'(\Delta(d_2))$.

end

Assume that $T_1'(d)$, $T_2'(d)$, and $S(\Delta|d)$ are all ordered from v_2 to v_1. We order the components of $\sigma'(\Delta)$ according to the order they appear on d from v_2 toward v_1. By Lemma 15, the initial component of $\sigma'(\Delta)$ (if any) starts at v_2. We determine whether there is an initial component as follows: Let x and y be the anchors of v_2 in $T_1'(d)$ and $T_2'(d)$, respectively. Let s be the first site in $S(\Delta'|d)$. If $w(x) + d(x, v_2) < \min\{w(y) + d(y, v_2), d(s, v_2)\}$, then v_2 is nearer to the owner of x in $S'(d_4)$ than to any site in $S(\Delta') \cup S(d_3)$; thus, there is no initial component. Otherwise, v_2 belongs in the region of either s or a site in $S(d_3)$ which induces a border edge on d incident to v_2, i.e., there is an initial component.

For the other components of $\sigma'(\Delta)$, we determine their first endpoint on d similarly to determining the first endpoints of components of $\sigma(\Delta)$ in the previous section. Let x_k denote the first endpoint on d of a component of $\sigma'(\Delta)$. x_k is induced by $b(s, u)$ where $s \in S(\Delta|d)$ and u is a vertex or site in $P'(d_4)$. We determine s and u as follows: Initialize s to the first site in $S(\Delta'|d)$ (from v_2 toward v_1) that has not been involved with a prior component of $\sigma'(\Delta) \cap d$. Let $\overline{d_s}$ denote the segment on d that lies in the Voronoi cell of s in $Vor_\Delta(S(\Delta))$, i.e., $Vr(s) \cap d$. Note that since s has not been involved with the prior component of $\sigma'(\Delta) \cap d$, segment $\overline{d_s}$ does not overlap with any prior component. Let $T_1'(d_s)$ denote the part of $T_1'(d)$ intersecting $\overline{d_s}$. Using binary search on $T_1'(d_s)$, we can determine as in the previous section whether x_k is induced by s, and if so determine u. The only difference here is that the ordering is defined from v_2 to v_1 while in the previous

section it was defined from v_1 to v_2. If it turns out that the whole $\overline{d_s}$ is closer to sites in $S'(d_4)$ than to s, then s does not induce a border edge. In that case, mark s as considered, update s to the next site in $S(\Delta'|d)$, and repeat the process.

The time to identify an endpoint of a component of $\sigma'(\Delta) \cap d$ is $O(\log|T_1'(d)|)$ times the number of sites in $S(\Delta|d)$ that get marked during the procedure. A site that gets marked is never considered again. Therefore, the time to identify the endpoints of all components of $\sigma'(\Delta) \cap d$ is $O(|S(\Delta|d)| \log|T_1'(d)|)$.

Once an endpoint x_k of a component of $\sigma'(\Delta) \cap d$ has been determined, we can easily trace the component by walking on d. Before starting to walk, we split $T_1'(d)$ at x_k. While walking on d we delete any regions of $T_1'(d)$ that get dominated by regions in $T_2'(d)$ or by sites in $S(\Delta'|d)$. Every time we cross into a new region of $T_2'(d)$ or $S(\Delta'|d)$ we determine border vertices. Every time we cross into a new region of $T_1'(d)$, we delete the previous region from $T_1'(d)$. Walking along d stops when a point x on bisector $b_g(S'(d_4), S(\Delta) \cup S(d_3))$ is reached. Such a point defines a basic border vertex $\mathcal{SD}'(\Delta)$ and delimits the end of a component of $\sigma'(\Delta) \cap d$. Note that, while walking, we always stay on border edges and stop when we reach a region of $T_1'(d)$ that extends into Δ. The time spent walking on a component of $\sigma'(\Delta) \cap d$ is proportional to its size (i.e., the number of border edges on the component) plus the time spent for deletions. Every region deletion from $T_1'(d)$ takes $O(\log|T_1'(d)|)$ time.

After tracing all components of $\sigma'(\Delta) \cap d$, $T'(d)$ can be easily computed by merging the remaining parts of $T_1'(d)$ (if any) with the regions induced by border edges. (Convert each component of $\sigma'(\Delta) \cap d$ into a search tree. For every border vertex, compute the intersection of the corresponding bisector with the bisectors inducing its neighboring border vertices; if the border vertex is basic, compute the intersection of its bisector with the neighboring bisector in $T_1'(d)$. Locate the intersection points in the triangulation and assign them to corresponding buckets.) Point location in the triangulation can be done in $O(\log n)$ time. Thus, we have the following lemma.

LEMMA 16. *The time to compute $T'(d)$ is $O((1 + b(d)) \log(n + k))$, where $b(d)$ is the number of border vertices on d.*

By Lemma 16, the time to compute $T'(d)$ for every diagonal d throughout the procedure is $O((n+k) \log(n+k))$. Moreover, the size of $T'(d)$ cannot exceed $(n+k)$ for any diagonal d (since every region in $T'(d)$ has a distinct anchor in $S \cup V$). Similarly as for $\mathcal{SD}(P)$, the total number of intersection points generated while constructing $\mathcal{SD}'(P)$ is $O(n + k)$. Thus, the total time for extending in all triangles is $O((n + k) \log(n + k))$. Therefore, $\mathcal{SD}'(P)$ can be computed in $O((n + k) \log(n + k))$ time.

7. Merging the Two Subdivisions. Once $\mathcal{SD}(P)$ and $\mathcal{SD}'(P)$ are available it is easy to construct the Voronoi diagram of S in P by merging the two diagrams. Let $\Delta = \Delta(d)$ be a triangle of root diagonal d. Let $\varphi(\Delta)$ denote the set of Voronoi edges in $Vor_\Delta(S)$ that are shared by pairs of Voronoi cells $Vr(s_i)$ and $Vr(s_j)$ for $s_i \in S(d)$ and $s_j \in S'(d)$. Let $\beta(\Delta)$ (resp. $\alpha(\Delta)$) denote the set of Voronoi edges in $Vor_\Delta(S)$ that are shared by pairs of Voronoi cells $Vr(s_i)$ and $Vr(s_j)$ for $s_i, s_j \in S(d)$ (resp. $s_i, s_j \in S'(d)$). Let $\varphi(P)$, $\beta(P)$, and $\alpha(P)$ denote the union of $\varphi(\Delta)$, $\beta(\Delta)$, and $\alpha(\Delta)$, respectively, for every triangle

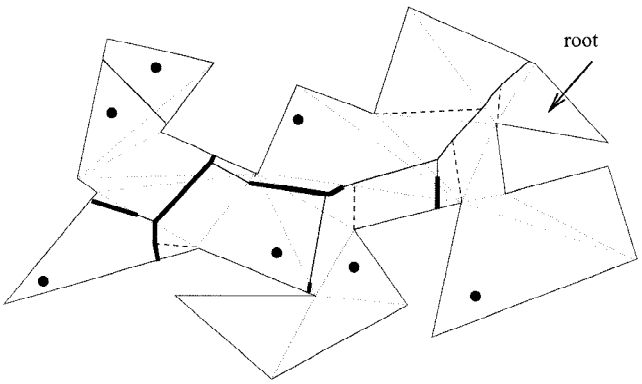

Fig. 13. Merging $\mathcal{SD}(P)$ and $\mathcal{SD}'(P)$. $\varphi(P)$ is shown in thick lines.

Δ in P. Clearly, $\beta(P) \subset \mathcal{SD}(P)$ and $\alpha(P) \subset \mathcal{SD}'(P)$ (see Figure 13). Furthermore, $Vor_P(S) = \varphi(P) \cup \beta(P) \cup \alpha(P)$. The following lemmas show properties of $\varphi(P)$.

LEMMA 17. *$\varphi(\Delta)$ contains no cycles and consists of disjoint simple paths with endpoints on the boundary of Δ.*

PROOF. $S(d)$ and $S'(d)$ are separable by d, thus $\varphi(\Delta)$ contains no cycles. $\varphi(\Delta)$ consists of portions of Voronoi edges of $Vor_P(S)$, thus, the endpoints of a component can only lie on Voronoi vertices or the boundary of Δ.

Let $s_i, s_j, s_k \in S$ induce a Voronoi vertex v in the interior of Δ that is incident to a component of $\varphi(\Delta)$. Let $s_i \in S(d)$ and $s_j \in S'(d)$, i.e, the Voronoi edge shared by $Vr(s_i)$ and $Vr(s_j)$ belongs to $\varphi(\Delta)$. If $s_k \in P(d)$, then the edge shared by $Vr(s_k)$ and $Vr(s_j)$ belongs to $\varphi(\Delta)$. If $s_k \in P'(d)$, then the edge shared by $Vr(s_k)$ and $Vr(s_i)$ belongs to $\varphi(\Delta)$. Thus, v must be internal to $\varphi(\Delta)$, i.e., v cannot be an endpoint of $\varphi(\Delta)$. Furthermore, only two of the edges incident to v may belong to $\varphi(\Delta)$. Note that if $s_k \in P(d)$ (resp. $s_k \in P'(d)$), then the edge shared by $Vr(s_i)$ (resp. $Vr(s_j)$) and $Vr(s_k)$ does not belong to $\varphi(\Delta)$. In other words, a component of $\varphi(\Delta)$ must be a simple path with endpoints on the boundary of Δ. □

LEMMA 18. *The endpoints of any component of $\varphi(P)$ lie on the polygon boundary or at a basic border vertex of $\mathcal{SD}(P)$ or $\mathcal{SD}'(P)$.*

PROOF. Since the endpoints of $\varphi(\Delta)$ are always on the boundary of Δ, the endpoints of a component of $\varphi(P)$ must be on the polygon boundary or on diagonals of the triangulation. Consider an endpoint x of a component of $\varphi(P)$ on a diagonal d, such that the edge of $\varphi(P)$ incident to x, denoted as e_φ, lies in $\Delta(d)$ (see Figure 14(a)). e_φ must be shared by $Vr(s_i)$ and $Vr(s_j)$, where $s_i \in S(d)$ and $s_j \in S'(d)$. Suppose that x is not a Voronoi vertex. Let e denote the Voronoi edge in $Vor_P(S)$ that contains x. There is a part of e in $\Delta(d)$ ($e \cap \Delta(d) = e_\varphi$) and a part in $\Delta'(d)$. Clearly, $s_j \notin S'(d_4)$, because otherwise $e \cap \Delta'(d)$ would belong to $\varphi(P)$, i.e., x would not be an endpoint of $\varphi(P)$.

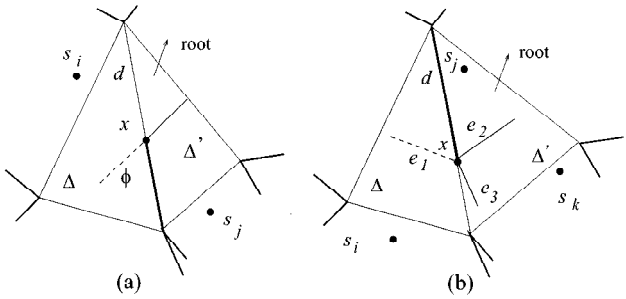

Fig. 14. Any endpoint of $\varphi(P)$, x, that is not on the polygon boundary coincides with a basic border vertex.

Since $s_j \in S'(d) - S'(d_4)$, $e \cap \Delta'(d)$ appears in $\mathcal{SD}(P)$ but $e \cap \Delta(d)$ does not. However, then $e \cap \Delta'(d)$ induces a basic border vertex on d that coincides with x.

Suppose that x coincides with a Voronoi vertex of $Vor_P(S)$, and let $Vr(s_i)$, $Vr(s_j)$, and $Vr(s_k)$ be the incident Voronoi cells, where $s_i \in S(d)$ and $s_j \in S'(d)$ (see Figure 14(b)). Let e_1 denote the Voronoi edge shared by $Vr(s_i)$ and $Vr(s_j)$ that is incident to x ($e_1 \in \varphi(P)$). Let e_2 (resp. e_3) denote the Voronoi edge shared by $Vr(s_j)$ and $Vr(s_k)$ (resp. $Vr(s_k)$ and $Vr(s_i)$) that is incident to x. Clearly, e_2 cannot be in $\Delta(d)$ because otherwise all e_1, e_2, e_3 would be in $\Delta(d)$ which contradicts properties of Voronoi edges, i.e., $e_2 \cap \Delta'(d) \neq \emptyset$. If $s_k \in P'(d_4)$, then e_3 would be part of $\varphi(P)$ no matter whether e_3 lies in $\Delta(d)$ or $\Delta'(d)$, i.e., x would not be an endpoint of $\varphi(P)$. Thus, $s_k \in P(d_4)$. Then $s_j \in P'(d) \cap P(d_4)$ because otherwise e_2 would be part of $\varphi(P)$. However, then e_2 appears in $\mathcal{SD}(P)$ while e_1 does not. Thus, in $\mathcal{SD}(P)$, s_j induces a border edge on d, incident to a basic border vertex that coincides with x. (Note that, depending on whether $s_k \in P(d)$ or $s_k \in P'(d) \cap P(d_4)$, e_3 lies in $\Delta(d)$ or $\Delta'(d)$, respectively. However, in any case e_3 appears in $\mathcal{SD}(P)$. Figure 14(b) only shows the case where $s_k \in P'(d) \cap P(d_4)$.)

Consider now an endpoint x of a component of $\varphi(P)$ on a diagonal d, such that the edge of $\varphi(P)$ incident to x lies in $\Delta'(d)$. Assuming that x is not a Voronoi vertex, let e denote the Voronoi edge in $Vor_P(S)$ that contains x. e is shared by $Vr(s_i)$, $s_i \in S(d_4)$, and $Vr(s_j)$, $s_j \in S'(d_4)$. More precisely, $s_i \in P'(d) \cap P(d_4)$ because otherwise $e \cap \Delta(d) \in \varphi(P)$, i.e., x would not be an endpoint of $\varphi(P)$. However, then $e \cap \Delta(d)$ appears in $\mathcal{SD}'(P)$ while $e \cap \Delta'(d)$ does not. This implies that $e \cap \Delta(d)$ induces a basic border vertex in $\mathcal{SD}'(P)$ that coincides with x.

Suppose now that the edge of $\varphi(P)$ incident to x lies in $\Delta'(d)$ and that x coincides with a Voronoi vertex. Let $Vr(s_i)$, $Vr(s_j)$, and $Vr(s_k)$ be the incident Voronoi cells, where $s_i \in S(d_4)$ and $s_j \in S'(d_4)$. Let e_1, e_2, and e_3 denote the Voronoi edges that are incident to x and are shared by $Vr(s_i)$ and $Vr(s_j)$, $Vr(s_j)$ and $Vr(s_k)$, and $Vr(s_k)$ and $Vr(s_i)$, respectively. $e_1 \in \varphi(P)$. Clearly, e_3 must lie in $\Delta(d)$ because otherwise all e_1, e_2, e_3 would be in $\Delta'(d)$ which contradicts properties of Voronoi edges. Moreover, $s_k \in P'(d_4)$ because otherwise e_2 would be part of $\varphi(P)$ (whether $e_2 \in \Delta(d)$ or $e_2 \in \Delta'(d)$). Thus, e_2 must lie in $\Delta'(d)$. Then $s_i \in P'(d) \cap P(d_4)$ because otherwise e_3 would be part of $\varphi(P)$. However, then e_2 and e_3 appear in $\mathcal{SD}'(P)$ while e_1 does not. Thus, in $\mathcal{SD}'(P)$, s_i induces a border edge on d incident to a basic border vertex on d that coincides with x. □

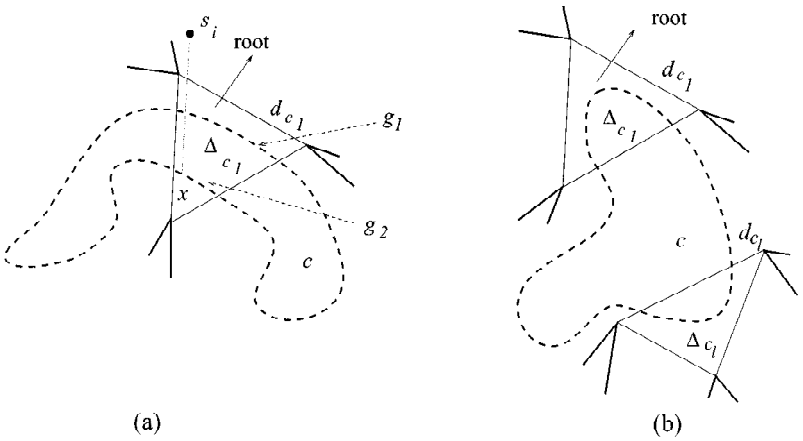

Fig. 15. c cannot expand into both subpolygons rooted at Δ_{c_1}.

LEMMA 19. *If $\varphi(P)$ contains a cycle, the cycle must pass through a basic border vertex which is common in both $\mathcal{SD}(P)$ and $\mathcal{SD}'(P)$.*

PROOF. Suppose that $\varphi(P)$ contains a cycle c. Since $\varphi(\Delta)$ is cycle free for any triangle Δ, c must pass through a number of triangles. Consider the dual subtree formed by the triangles intersected by c. Let Δ_{c_1} be the triangle corresponding to the root of this subtree and let Δ_{c_l} be the triangle corresponding to the rightmost leaf of the dual subtree. Let $\Delta_{c_i}, 1 \leq i \leq l$, denote the sequence of triangles between Δ_{c_1} and Δ_{c_l}, and let $d_{c_i}, 1 \leq i \leq l$, denote their root diagonals.

Suppose that c expands into both subpolygons rooted at Δ_{c_1} (see Figure 15(a)). Then $c \cap \Delta_{c_1}$ consists of two components, g_1 and g_2, where g_1 denotes the component toward d_{c_1}. Any edge along the two components is shared by the Voronoi cells of a site s_i "above" d_{c_1} (i.e., $s_i \in S'(d_{c_1})$) and a site s_j "below" d_{c_1} (i.e., $s_j \in S(d_{c_1})$). However, for any point x along g_2, $\pi(s_i, x)$ intersects g_1 which contradicts the star-shape property of Voronoi cells. Thus, c cannot expand into both subpolygons rooted at Δ_{c_1}. In other words, $c \cap \Delta_{c_1}$ must consist of only one component with both endpoints on the same nonroot diagonal (see Figure 15(b)).

Consider a pair of sites that induce an edge of c. Clearly, one must lie in the area enclosed by c and the other must lie out of that area. For brevity we say that one is enclosed by c and the other is out of c. Moreover, one of the two sites inducing an edge in $c \cap \Delta_{c_i}$ must lie in $P'(d_{c_i})$ while the other must lie in $P(d_{c_i})$. An edge in $c \cap \Delta_{c_1}$ must be induced by $s_i \in P'(d_{c_1})$ and $s_j \in P(d_{c_1})$. Since $s_i \in P'(d_{c_1})$, s_i must be out of c while s_j must be enclosed by c. In other words, an edge of $c \cap \Delta_{c_1}$ is induced by a site that lies out of c and above the root diagonal, and a site that is enclosed by c that lies below the root diagonal. On the other hand, an edge in $c \cap \Delta_{c_l}$ is induced by $s_k \in S'(d_{c_l})$ and $s_r \in S(d_{c_l})$. It is not hard to see that there must be an edge in $c \cap \Delta_{c_l}$ such that s_k is enclosed by c and s_r is out of c. (Suppose for a contradiction otherwise, i.e., for every edge in $c \cap \Delta_{c_l}$, s_k is out of c and s_r is enclosed by c. Then there must be a Voronoi

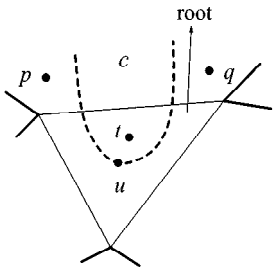

Fig. 16. u cannot be (geodesically) equidistant from p, q, t.

vertex u induced by $p, q \in S'(d_{c_l})$ and $t \in S(d_{c_l})$ such that t is enclosed by c, and p, q lie out of c on opposite sides of the components of c derived by removing $c \cap \Delta_{c_l}$ and $c \cap \Delta_{c_1}$ (see Figure 16). Consider the triangle formed by u, p, and q (the triangle may intersect the polygon boundary). Clearly, t lies in the interior of the triangle. Thus $d_g(u, t) = d(u, t) < \max\{d(u, p), d(u, q)\} \leq \max\{d_g(u, p), d_g(u, q)\}$, i.e., u cannot be a Voronoi vertex.) Thus, there is an edge in $c \cap \Delta_{c_l}$ that is induced by a site enclosed by c which lies above the root diagonal, and a site that lies out of c and "below" the root diagonal.

By the above discussion, there must be two consecutive Voronoi edges in c, e_1 and e_2 (ordered in a traversal of c from Δ_{c_1} to Δ_{c_l}), such that e_1 is induced by a site x "above" and a site y "below" the root diagonals of all the triangles that e_1 traverses, where x is out of c and y is enclosed by c, and e_2 is induced by a site t "above" and a site z "below" the root diagonals of all the triangles that e_2 traverses, where t is enclosed by c and z is out of c. Clearly, e_1 and e_2 share a Voronoi vertex v that is equidistant from x, y, t, and z. Since we have assumed that no four sites lie on the same geodesic circle, at least two of the sites must be identical. However, x and z must be distinct because x lies "above" and z lies below the same diagonal. Moreover, x must be different from t and z must be different from y because x and z lie out of c while y and t are enclosed by c. Thus, y and t must be identical, but then v cannot lie in the interior of a triangle Δ_{c_i} because in that case $y \equiv t$ would have to lie both "above" and "below" d_{c_i}. Thus, v must lie on a diagonal d_{c_i} (i.e., $e_1 \in \Delta_{c_{i-1}}$ and $e_2 \in \Delta_{c_i}$). Then $x \in S'(d_{c_{i-1}}), z \in S(d_{c_i})$, and $y \equiv t \in S(d_{c_{i-1}}) \cap S'(d_{c_i})$. However, then y induces a basic border vertex on d_{c_i} that coincides with v in both $\mathcal{SD}(P)$ and $\mathcal{SD}'(P)$. Figures 17(a) and (b) illustrate the basic border vertex induced by y in $\mathcal{SD}(P)$ and $\mathcal{SD}'(P)$, respectively; dashed edges indicate the Voronoi edges incident to v in the final Voronoi diagram. (If Assumption B did not hold, y and t might be different but $y, t \in S(d_{c_{i-1}}) \cap S'(d_{c_i})$ and thus the lemma would still hold.) □

Note that $\varphi(P)$ may branch in three ways at some Voronoi vertex. For example, in Figure 17 all three Voronoi edges incident to v belong to $\varphi(P)$. It is not hard to show that in such a case v would be a basic border vertex in both $\mathcal{SD}(P)$ and $\mathcal{SD}'(P)$. Clearly, any intersection of $\varphi(P)$ with a Voronoi edge of $\mathcal{SD}(P)$ and $\mathcal{SD}'(P)$ implies a Voronoi vertex of the final Voronoi diagram. Such a Voronoi vertex is an endpoint of $\beta(P)$ or

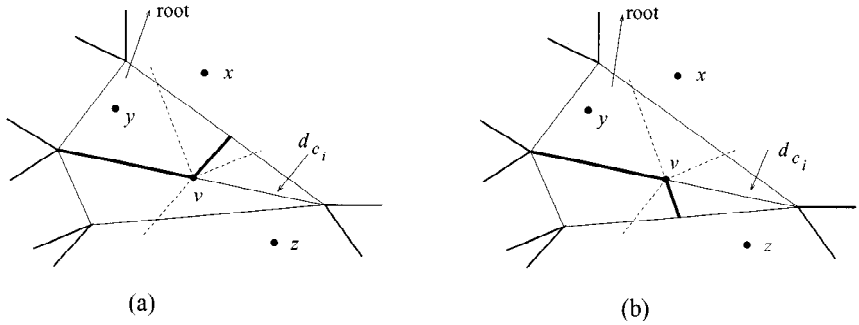

Fig. 17. v induces a basic border vertex in both $\mathcal{SD}(P)$ and $\mathcal{SD}'(P)$.

$\alpha(P)$, respectively. A basic border vertex b of $\mathcal{SD}(P)$ (resp. $\mathcal{SD}'(P)$) that is an endpoint of a component of $\varphi(P)$ is clearly an endpoint of a component of $\beta(P)$ (resp. $\alpha(P)$) as well. More specifically, the edges of $\varphi(P)$ and $\beta(P)$ (resp. $\alpha(P)$) incident to b belong to the same Voronoi edge of the final Voronoi diagram. On the other hand, by definition, $\beta(P)$ and $\alpha(P)$ cannot have common edges. However, they may have common points at nonbasic border vertices which are common to both $\mathcal{SD}(P)$ and $\mathcal{SD}'(P)$.

To compute the final Voronoi diagram we basically have to compute $\varphi(P)$. Once $\varphi(P)$ is determined, it is easy to compute the final diagram by identifying $\beta(P)$ and $\alpha(P)$ in $\mathcal{SD}(P)$ and $\mathcal{SD}'(P)$, respectively. This can be done by simple scans starting at possible endpoints of $\beta(P)$ and $\alpha(P)$. Lemmas 18 and 19 provide the means to compute $\varphi(P)$. The algorithm can be briefly stated as follows:

Trace the polygon boundary to identify points which are equidistant from their region owner in $\mathcal{SD}(P)$ and their region owner in $\mathcal{SD}'(P)$. For every such point, trace the corresponding component of $\varphi(P)$ in the ordinary way [1], [18]. After all components starting at boundary points have been traced, trace the components (if any) with both endpoints at basic border vertices. Note that if $\varphi(P)$ reaches a border vertex common to both subdivisions, it may branch into two branches. When $\varphi(P)$ intersects an edge e of $\mathcal{SD}(P)$ (resp. $\mathcal{SD}'(P)$), a Voronoi vertex v of the final Voronoi diagram is determined which is incident to an edge of $\beta(P)$ (resp. $\alpha(P)$). In that case, break e at v, add v to $\mathcal{SD}(P)$, and mark v; mark the part of e that belongs in $\beta(P)$ (resp. $\alpha(P)$) as useful and the other part as useless. When an endpoint of $\varphi(P)$ at a basic border vertex is reached, mark that border vertex.

Tracing $\beta(P)$ and $\alpha(P)$ in $\mathcal{SD}(P)$ and $\mathcal{SD}'(P)$, respectively, can start at marked vertices and at the polygon boundary. At the end, delete all border edges and vertices. It is not hard to see that merging can be done in $O(n+k)$ time.

8. Other Polygonal Domains. In this section we consider the geodesic Voronoi diagram of points in some other restricted polygonal domains. We first consider polygonal domains where the shortest path between two points is *monotone* with respect to a constant number of directions. A polygonal chain $C = (q_1, q_2, \ldots, q_k)$ is said to be *monotone* with respect to a straight line l (i.e., a direction l) if the perpendicular projec-

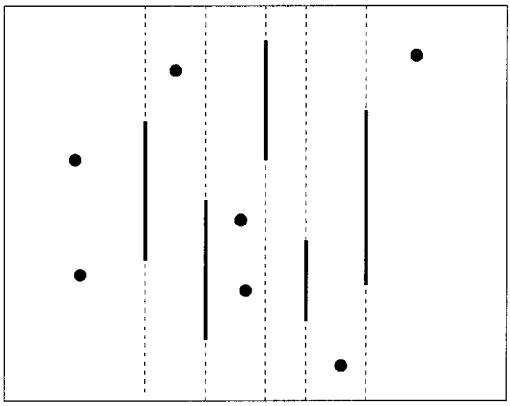

Fig. 18. A polygonal domain of vertical line segments and a set of point sites.

tions $l(q_1), l(q_2), \ldots, l(q_k)$ on l of the vertices in C are ordered on l as $(q_1, q_2, \ldots, q_k)$. Consider a polygonal domain where obstacles are n disjoint and parallel line segments. Without loss of generality, we can assume that segments are vertical, i.e., parallel to the y axis. As was shown in [14], in this case the shortest path between any two points is monotone with respect to the x axis.

Consider the subdivision of the plane obtained by drawing a vertical line through every obstacle (see Figure 18). Let l_i, $1 \leq i \leq n$, denote the vertical line through the ith obstacle, where obstacles are ordered from left to right. Without loss of generality, we may assume that the polygonal domain is enclosed in a rectangle and thus the lines extend up to the boundary of that rectangle (see Figure 18). Let $r(l_i)$ denote the neighboring region to the left of l_i. The dual graph of this subdivision is a simple path. Due to the monotonicity property of shortest paths with respect to the x axis, we can compute the geodesic Voronoi diagram by plane sweep in two passes: first left-to-right and then right-to-left. In the left-to-right pass we construct a subdivision equivalent to $\mathcal{SD}(P)$, such that, for every l_i, the part of the subdivision in $r(l_i)$ contains the geodesic Voronoi diagram of all sites to the left of l_i. In the right-to-left pass we construct a subdivision equivalent to $\mathcal{SD}'(\mathcal{P})$ such that, for every l_i, the part of the subdivision in $r(l_i)$ contains the geodesic Voronoi diagram of all sites to the right of l_i. At the end, we merge the two diagrams to derive the geodesic Voronoi diagram of all sites.

Consider now a polygonal domain of n isothetic rectangles, where distance between two points is the length of the shortest *rectilinear* path (i.e., consisting of axis-parallel segments) between the points. As was shown in [6], in this case the shortest path between any two points is monotone with respect to either the x or the y axis.

Consider the subdivision obtained by drawing the vertical lines passing through the vertical edges of the rectangles until they hit an obstacle or the boundary of the polygonal domain (we assume without loss of generality that the polygonal domain is enclosed in a rectangle). This subdivision is referred to as the *vertical* subdivision and partitions the polygonal domain into rectangular regions. The additional line segments that are not obstacle edges are referred to as *chords*. In the dual graph a vertex corresponds to

a region (other than an obstacle) and an edge joins two vertices if the corresponding regions are adjacent. We can make the dual graph a directed acyclic graph by assigning a $(+x)$ direction along the edges (i.e., assign to edge (v, u) a direction from v to u if the region corresponding to v is to the left of u). The dual of any x-monotone path between two points s and t, where the region of s is to the left of the region of t, is obviously embedded into the dual directed graph. (The nodes of a dual path correspond to the regions intersected by the path.) Consider also the *horizontal* subdivision of the polygonal domain which is obtained by drawing the horizontal lines passing through the horizontal edges of the rectangles until they hit an obstacle or the boundary of the polygonal domain. We can make the dual graph of the horizontal subdivision an acyclic directed graph by assigning a $(+y)$ direction along the edges, (i.e., let edge (v, u) be directed from v to u if the region corresponding to v is below the region of u). Obviously, any y-monotone path between two points s and t, where the region of s is below the region of t, is embedded into the directed dual graph. Note that for every node in both dual graphs the in-degree and out-degree are at most two.

We can compute the geodesic Voronoi diagram of a set of sites S in P in two phases. In phase 1 we perform a plane sweep based on the topological sorting of the directed dual graph of the *vertical* subdivision, first to the right and next to the left. In phase 2 we perform a plane sweep based on the topological sorting of the directed dual graph of the *horizontal* subdivision, first toward the top and next toward the bottom. More specifically, in phase 1 we compute x-monotone shortest paths from S to all vertices of P. (The shortest path from S to a point t is the shortest path from the site $s \in S$ nearest to t.) We assign the length of these paths as weight to the corresponding vertices of P. In phase 2 we treat the vertices of P as sites weighted with nonnegative weights, and compute the geodesic Voronoi diagram of $S \cup V$. There is actually not much difference between weighted and unweighted sites. The only difference is that in the former case, a site s_j with a large enough weight ($w(s_j)$) need not have a Voronoi cell of its own. Instead s_j would belong in the Voronoi cell of a site s_i such that $w(s_j) > w(s_i) + d_g(s_i, s_j)$.

Let r be a region of in-degree and out-degree two in the *vertical* subdivision, and let d_1, d_2, d_3, d_4 be the chords of r, where d_1, d_2 are to the left of d_3, d_4, and d_1, d_3 are above d_2, d_4 (see Figure 19). Given a chord d, let $r(d)$ (resp. $r'(d)$) be the region to the left (resp. right) of d (i.e., $r = r(d_3) = r(d_4)$ and $r = r'(d_1) = r'(d_2)$). Let $P(d)$ (resp. $P'(d)$) be the set of regions preceding (and including) (resp. following) $r(d)$ in the order of the topological sorting of the dual graph. Furthermore, let $T(d)$ (resp. $T'(d)$) be the list of regions in the Voronoi diagram of sites in $P(d)$ (resp. $P'(d)$). Let $S(r(d)|d)$ denote the list of sites in $r(d)$ whose Voronoi regions intersect d in the Voronoi diagram of $S \cap r(d)$. Given $T(d_1)$, $T(d_2)$, $S(r|d_3)$, and $S(r|d_4)$ we can compute $T(d_3)$ and $T(d_4)$ similarly to the simple polygon case. Namely, extend $T(d_1)$ and $T(d_2)$ in r, merge them, and split at the obstacle vertices between d_3 and d_4. Merge the part intersecting d_3 with $S(r|d_3)$ to derive $T(d_3)$ and the part intersecting d_4 with $S(r|d_4)$ to derive $T(d_4)$. Intersections of new neighboring bisectors are located in the subdivision and assigned into buckets (every region has a corresponding bucket to hold bisector intersection points). Clearly, $T(d_3)$ and $T(d_4)$ reveal the x-monotone shortest paths from $S \cap P(d_3)$ to every point along d_3 and d_4 and hence to their endpoints. In the second pass we compute $T'(d_3)$ and $T'(d_4)$ and thus we compute x-monotone shortest paths from $S \cap P'(d_3)$ to the endpoints of d_3

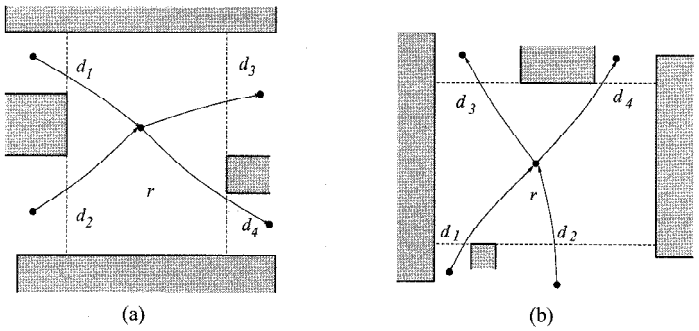

Fig. 19. (a) A region in the *vertical* subdivision. (b) A region in the *horizontal* subdivision.

and d_4. At the end of the first phase all vertices of the polygonal domain are weighted with the length of the shortest x-monotone path from S.

Now let r be a region of in-degree and out-degree two in the *horizontal* subdivision, and let d_1, d_2, d_3, d_4 be the chords of d where d_1, d_2 are below d_3, d_4, and d_1, d_3 are to the right of d_2, d_4. Given a chord d, let $r(d)$ (resp. $r'(d)$) be the region below (resp. above) d (i.e., $r = r(d_3) = r(d_4)$ and $r = r'(d_1) = r'(d_2)$), and $P(d)$ (resp. $P'(d)$) be the set of regions preceding (and including) (resp. following) $r(d)$ in the order of the topological sorting of the horizontal dual graph. In the first pass of phase 2 we compute a subdivision $\mathcal{SD}(P)$ such that, for every region r, $\mathcal{SD}(P)$ reveals the geodesic Voronoi diagram in r of all sites (including obstacle vertices) in $P(d)$ $((S \cup V) \cap P(d))$. In the second pass we compute a subdivision $\mathcal{SD}'(P)$ such that, for every region r, $\mathcal{SD}'(P)$ reveals the geodesic Voronoi diagram in r of all sites (including obstacle vertices) in $P'(d)$. At the end we can merge the two subdivisions similarly to the simple polygon case.

The same technique can be applied to any polygonal domain where shortest paths are restricted to being monotone with respect to a constant number of directions. Path monotonicity can be a property of the polygonal domain or simply a restriction induced by the application. For a constant number of directions c, the problem can be solved in c phases, i.e., in $O(c(n+k)\log(n+k))$ time. For every direction θ_i, $1 \leq i \leq c$, we construct an (η_i)-subdivision, where η_i is the direction perpendicular to θ_i, by drawing at each obstacle vertex a line in the η_i direction until it hits an obstacle or the boundary of the polygonal domain. At phase i we use the (η_i)-subdivision, and compute shortest θ_i-monotone paths from S to all vertices of the polygonal domain. We assign to each vertex the minimum among the weights computed so far. At phase c we use the (η_c)-subdivision, and compute the geodesic Voronoi diagram of $S \cup V$, where vertices are weighted with the length of the shortest $\{\theta_1 \cdots \theta_{c-1}\}$-monotone paths. At each phase we do two plane sweeps based on the topological sorting of the dual graph of the (η_i)-subdivision.

Our algorithm is also generalizable to an $O((n+k)\log(n+k))$-time algorithm for computing the geodesic Voronoi diagram of points in polygons with holes such that there exist *bridges* that partition the polygon into parts of at most one hole (see Figure 21). A *bridge* is a line segment with endpoints on the polygon boundary that is entirely contained in the polygon free space. We first generalize our algorithm for computing the

geodesic Voronoi diagram of points in polygons with only one hole. Given a polygon P with one hole, we can transform it into a simple polygon Q by drawing a chord e with one endpoint on the polygon boundary and the other on the boundary of the hole. Treating e as an ordinary boundary, we can compute the Voronoi diagram of points in Q. Let $T(e)$ and $T'(e)$ be the list of Voronoi regions intersecting e from opposite sides. Then we can remove the chord and modify the Voronoi diagram, by extending $T(e)$ and $T'(e)$ in opposite directions. Given a polygon with more than one hole such that holes are separated by bridges, we triangulate it using the bridges as triangulation diagonals. The dual graph is now a *tree of cycles*, where each cycle corresponds to hole. We call the dual graph a *tree of cycles* because it becomes a tree (referred to as the dual *collapsed* tree) if each cycle is collapsed into a single node (referred to as a *collapsed* node). We can now apply our algorithm using the dual *collapsed* tree as a guide, and treating collapsed nodes as polygons with one hole.

In a general polygonal domain, our method is not directly applicable. The reason is that the dual graph of a potential subdivision contains cycles and thus there is no clear order in which the regions of the subdivision could be processed. Whether there is a subdivision that would allow a Dijkstra type access of regions is an open problem.

9. Answering Proximity Questions. Once the geodesic Voronoi diagram of a set of point sites S in a polygonal domain P is available, several proximity questions can be answered. Examples include the minimum spanning tree, finding the closest pair of sites, and finding the nearest neighbor for every site. However, this is not a straightforward generalization of the ordinary case.

Let $G(S)$ denote the dual graph of the geodesic Voronoi diagram whose vertices are the given sites and an edge joins two vertices if and only if the Voronoi cells of the corresponding sites are adjacent. Every edge is weighted with the geodesic distance between the sites corresponding to its endpoints. The minimum spanning tree of S, denoted as $MST(S)$, is the minimum spanning tree of graph $G(S)$ where an edge between any two sites s and t represents $\pi(s,t)$. Note that if the shortest paths between two pairs of sites share a common part, the length of this common part is counted twice in the cost of $MST(S)$.

Consider now two sites $s, t \in S$ with adjacent Voronoi cells and let e be the Voronoi edge separating $Vr(s)$ and $Vr(t)$. If $\pi(s,t)$ intersects the Voronoi edge e, then the geodesic distance between s and t is not hard to compute. We only need to locate the point x along e where $d_g(s,x) + d_g(t,x)$ is minimized. Then clearly $d_g(s,t) = d_g(s,x) + d_g(t,x)$ (see Figure 20(a)). Point x can be easily identified since for every breakpoint b of edge e we can readily compute $d_g(s,b)$ and $d_g(t,b)$. However, if $\pi(s,t)$ does not intersect e (i.e., $\pi(s,t)$ passes through a third Voronoi cell), then it is not easy to determine $d_g(s,t)$ (see Figure 20(b)). In other words, unlike the ordinary case, given $Vor_P(S)$ we cannot readily obtain the geodesic distance between every pair of sites with adjacent Voronoi cells. The following lemma allows us to avoid computing the geodesic distance for every pair of sites with adjacent Voronoi cells.

LEMMA 20. *Given two sites $s, t \in S$, if $\pi(s,q)$ intersects $Vr(q)$, $q \neq s, t$, then edge (s,t) cannot be in the minimum spanning tree of $G(S)$.*

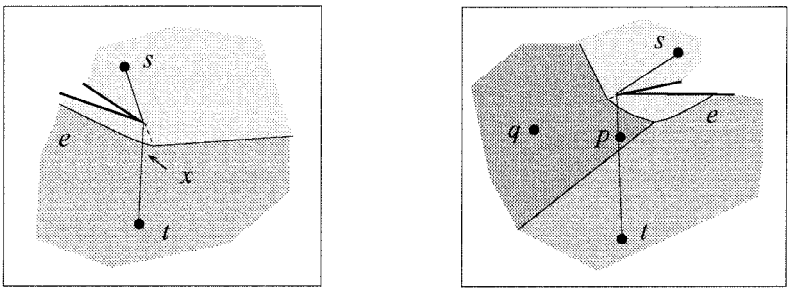

Fig. 20. $\pi(s, t)$ may or may not intersect the Voronoi edge e.

PROOF. Let T denote the minimum spanning tree of $G(S)$ and let $s, t \in S$ be two sites such that edge (s, t) belongs to T. Suppose for a contradiction that $\pi(s, t)$ passes through $Vr(q)$, $q \neq s, t$. Assume, without loss of generality, that the path in T between q and s does not pass through t. Then, since $(s, t) \in T$, the path in T connecting q and t passes through s because otherwise there would be a cycle.

Let p be a point along $\pi(s, t)$ in the interior of $Vr(q)$. Then $d_g(q, p) < d_g(s, p)$ and $d_g(q, p) < d_g(t, p)$. Let T' be the spanning tree derived from T by deleting edge (s, t) and adding edge (q, t). (Note that no cycle is introduced by the transformation and thus T' is a spanning tree.) However, then $d_g(q, t) \leq d_g(q, p) + d_g(p, t) < d_g(s, t)$, i.e., the cost of T' is less than the cost of T which is a contradiction. Thus, edge (s, t) cannot be in the minimum spanning tree. □

The above lemma allows us to compute $MST(S)$ as follows. Consider the dual graph $G(S)$. Instead of assigning to an edge (s, t) of $G(S)$ the geodesic distance $d_g(s, t)$ as weight, we assign the quantity $w(s, t) = \min_{x \in e} d_g(s, x) + d_g(t, x)$, where e is the Voronoi edge between $Vr(s)$ and $Vr(t)$. Note that if $\pi(s, t)$ intersects e, then $d_g(s, t) = w(s, t)$ while otherwise $d_g(s, t) < w(s, t)$. By Lemma 20 if $\pi(s, t)$ does not intersect e, edge (s, t) should not be part of the minimum spanning tree. Thus, increasing the weight of edge (s, t) in the latter case should not affect the computation of $MST(S)$.

The same observations hold for other proximity problems. For example, it can be similarly shown that the nearest neighbor of a site s must be a site t such that $\pi(s, t)$ is totally contained within $Vr(s)$ and $Vr(t)$. Thus, the dual graph of the geodesic Voronoi diagram with modified edge weights as explained above, can answer proximity questions in the same manner as in the ordinary case.

10. Conclusion and Open Problems. In this paper we presented an $O((n+k)\log(n+k))$-time algorithm for computing the geodesic Voronoi diagram of points in a simple polygon. Whether this time complexity can be improved is an open problem since the only lower bound known is the trivial $\Omega(n+k\log k)$ [1]. Our approach can be extended to compute the geodesic Voronoi diagram of point sites in polygonal domains where shortest

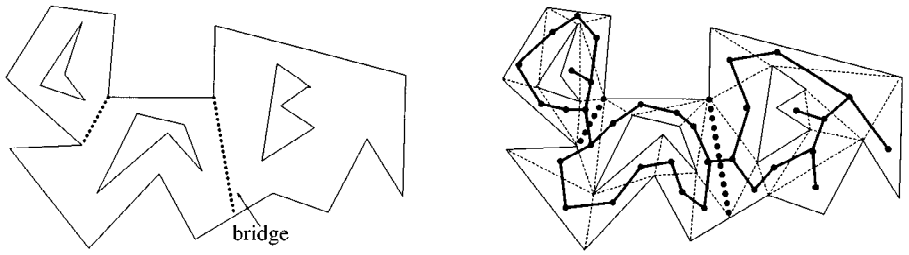

Fig. 21. A polygon with holes that can be partitioned into disjoint parts by *bridges*. The dual graph of the triangulation is a *tree of cycles*.

paths must be monotone with respect to a constant number of directions. Examples include a polygonal domain of n parallel line segments and a polygonal domain of rectangular obstacles and rectilinear paths. For these two cases the number of obstacles is proportional to the number of vertices, and thus we can derive an $\Omega(n \log n)$ lower bound by a reduction from sorting [14]. $\Omega(k \log k)$ is a trivial lower bound since in a polygonal domain of no obstacles the geodesic Voronoi diagram of k sites is identical to the ordinary Voronoi diagram. Thus, $\Omega((n+k)\log(n+k)) = \Omega(n \log n + k \log k)$ is also a lower bound, i.e., our algorithm is asymptotically optimal in these cases. Furthermore, our algorithm is generalizable to an $O((n+k)\log(n+k))$-time algorithm for computing the geodesic Voronoi diagram of points in polygons with holes such that there exist *bridges* that partition the polygon into parts of at most one hole (see Figure 21). The questions of determining if bridges exist and how to compute them, if they do, are of interest. Whether there is a subdivision that would allow a sweep of a general polygonal domain in a Dijkstra-like fashion and therefore would allow this approach to be generalized for the geodesic Voronoi diagram of points in a general polygonal domain, is an open problem.

Acknowledgment. We would like to thank one of the referees for comments that helped improve the readability of this paper.

References

[1] B. Aronov. On the geodesic Voronoi diagram of point sites in a simple polygon, *Algorithmica*, 4 (1989), 109–140.
[2] Ta. Asano and Te. Asano, Voronoi diagrams for points in a polygon, in *Discrete Algorithms & Complexity*: *Perspective in Computing*, D. S. Johnson, ed., Academic Press, New York, 1987, pp. 51–64.
[3] F. Aurenhammer, Voronoi diagrams: a survey of a fundamental geometric data structure, *ACM Comput. Surv.*, 23 (1991), 345–405.
[4] B. Chazelle, Triangulating a simple polygon in linear time, *Discrete Comput. Geom.*, 6 (1991), 485–524.
[5] L. P. Chew, Constrained Delaunay triangulations, *Algorithmica*, 4 (1989), 97–108.
[6] P. J. deRezende, D. T. Lee, and Y. F. Wu, Rectilinear shortest paths with rectangular barriers, *Discrete Comput. Geom.*, 4 (1989), 41–53.
[7] S. Fortune, A sweepline algorithm for Voronoi diagrams, *Algorithmica*, 2 (1987), 153–174.

[8] S. Guha and I. Suzuki, Proximity problems for points on a rectilinear plane with rectangular obstacles, *Algorithmica*, 17 (1997), 281–307.
[9] L. J. Guibas, J. Hershberger, D. Leven, M. Sharir, and R. E. Tarjan, Linear-time algorithms for visibility and shortest path problems inside triangulated simple polygons, *Algorithmica*, 2 (1987), 209–233.
[10] L. J. Guibas and R. Sedgewick, A dichromatic framework for balanced trees, *Proc*. 19*th IEEE Symp*. *on Foundations of Computer Science*, 1978, pp. 8–21.
[11] J. Hershberger and S. Suri, Efficient computation of Euclidean shortest paths in the plane, *Proc*. 34*th Symp*. *on Foundations of Computer Science*, 1993, pp. 508–517.
[12] D. Kirkpatrick, Optimal search in planar subdivisions, *SIAM J. Comput*., 12(1) (1983), 28–35.
[13] D. T. Lee and A. K. Lin, Generalized Delaunay triangulations for planar graphs, *Discrete Comput*. *Geom*., 1 (1986), 201–217.
[14] D. T. Lee and F. P. Preparata, Euclidean shortest paths in the presence of rectilinear barriers, *Networks*, 14 (1984), 393–410.
[15] J. S. B. Mitchell, L_1 shortest paths among polygonal obstacles in the plane, *Algorithmica*, 8 (1992), 55–88.
[16] J. S. B. Mitchell, Shortest paths among obstacles in the plane, *Proc*. 9*th ACM Symp*. *on Computational Geometry*, May 1993, pp. 308–317.
[17] J. S. B. Mitchell, Shortest paths among obstacles in the plane, *Internat. J. Comput. Geom. Appl*., 6.3 (Sept. 1996), 309–332.
[18] F. P. Preparata and M. I. Shamos, *Computational Geometry*: *an Introduction*, Springer-Verlag, New York, 1985.
[19] C. A. Wang and F. Chin, *Finding the Constrained Delaunay Triangulation and Constrained Voronoi Diagram of a Simple Polygon in Linear Time*, Lecture Notes in Computer Science, Vol. 979, Springer-Verlag, Berlin, 1995, pp. 280–294.
[20] C. Wang and L. Schubert, An optimal algorithm for constructing the Delaunay triangulation of a set of line segments, *Proc*. 3*rd ACM Symp*. *on Computational Geometry*, 1987, pp. 223–232.