Computational Geometry 52 (2016) 34–43

Contents lists available at ScienceDirect

Computational Geometry: Theory and Applications

www.elsevier.com/locate/comgeo

An almost optimal algorithm for Voronoi diagrams of non-disjoint line segments ☆

Sang Won Bae

Department of Computer Science, Kyonggi University, Suwon, Republic of Korea

ARTICLE INFO

Article history:
Received 27 February 2015
Received in revised form 4 September 2015
Accepted 16 November 2015
Available online 28 November 2015

Keywords:
Voronoi diagram
Line segments
Weakly simple polygon
Polygon with holes
Medial axis

ABSTRACT

This paper presents an almost optimal algorithm that computes the Voronoi diagram of a set S of n line segments that may intersect or cross each other. If there are k intersections among the input segments in S, our algorithm takes $O(n\alpha(n)\log n + k)$ time, where $\alpha(\cdot)$ denotes the inverse of the Ackermann function. The best known running time prior to this work was $O((n+k)\log n)$. Since the lower bound of the problem is shown to be $\Omega(n\log n + k)$ in the worst case, our algorithm is worst-case optimal for $k = \Omega(n\alpha(n)\log n)$, and is only a factor of $\alpha(n)$ away from any optimal-time algorithm, which is still unknown. For the purpose, we also present an improved algorithm that computes the medial axis or the Voronoi diagram of a polygon with holes.

© 2015 Elsevier B.V. All rights reserved.

1. Introduction

There is no doubt that the Voronoi diagram is one of the most fundamental and the most well studied structure in computational geometry. Voronoi diagrams and their variations play an important role not only in computer science but also in many other fields in engineering and sciences, finding a lot of applications. For a comprehensive survey, we refer to Aurenhammer and Klein [2] or to a book by Okabe et al. [18].

In this paper, we are interested in the Voronoi diagram of line segments in the plane. As one of the most popular variants of the ordinary Voronoi diagram, the line segment Voronoi diagram has been extensively studied in the computational geometry community, finding lots of applications in computer graphics, pattern recognition, motion planning, shape representation, and NC machining [11,14,17]. Also, computing line segment Voronoi diagram is used as a frequent subroutine of algorithms for more complex structures [3,8]. For the set of line segments that are disjoint or may intersect only at their endpoints, a variety of optimal $O(n\log n)$-time algorithms that compute the diagram are known. For example, Kirkpatrick [14], Lee [17], and Yap [21] presented divide-and-conquer algorithms, Fortune [10] presented a plane sweep algorithm, and a pure abstract approach to Voronoi diagrams by Klein [15] is also applied to yield an optimal time algorithm [16].

However, only few researches consider line segments that may intersect or cross each other freely. Let S be a set of n arbitrary line segments in the plane and k be the number of intersecting pairs of the segments in S. Karavelas [12] presented an $O((n+k)\log^2 n)$ time algorithm that computes the Voronoi diagram of S in a robust way. In fact, one can easily achieve the time bound $O((n+k)\log n)$ for computing the Voronoi diagram of S as follows: first, specify all the intersection points among the segments in S and consider the set S' of sub-segments obtained by cutting the original segments in S at the

☆ This work was supported by Kyonggi University Research Grant 2014. A preliminary version of this work was presented at WALCOM 2015 [4].
E-mail address: swbae@kgu.ac.kr.

http://dx.doi.org/10.1016/j.comgeo.2015.11.002
0925-7721/© 2015 Elsevier B.V. All rights reserved.

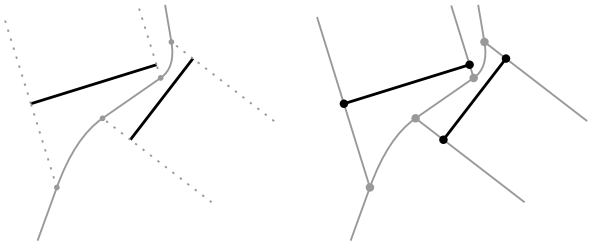

Fig. 1. (a) The Voronoi diagram of two line segments. (b) The Voronoi diagram of six sites: two open line segments and four of their endpoints. Sites are depicted as black segments or dots, and Voronoi edges and vertices are as gray segments and dots.

intersection points. Then, S' consists of at most $n+2k$ line segments that can intersect only at endpoints, so we can apply any of the above existing algorithms. On the other hand, it is not difficult to see that the lower bound of the problem of computing the Voronoi diagram of line segments is $\Omega(n\log n+k)$ in the worst case.

In this paper, we present an almost optimal algorithm that computes the Voronoi diagram of line segments. Our algorithm takes $O(n\alpha(n)\log n+k)$ time, where $\alpha(n)$ denotes the functional inverse of the Ackermann function. Since the lower bound is shown to be $\Omega(n\log n+k)$, our algorithm is only a factor of $\alpha(n)$ away from the optimal running time, and is optimal for large $k=\Omega(n\alpha(n)\log n)$. To our best knowledge, prior to our result, there was no known algorithm better than the simple $O((n+k)\log n)$-time algorithm.

In order to achieve our main result, we also consider an interesting special case where S forms a polygon. In this case, the Voronoi diagram of S is closely related to the *medial axis* of S [9,17]. When S forms a simple polygon, it is known that its Voronoi diagram and medial axis can be computed in linear time by Chin et al. [9]. In this work, we extend their result into more general form of polygons, namely, *weakly simple polygons* and *polygonal domains*. In particular, we devise an $O(m\log(m+t)+t)$-time algorithm that computes the Voronoi diagram or the medial axis of a given polygonal domain, where m denotes the total number of vertices of its holes and t denotes the number of vertices of its outer boundary. Note that our algorithm is strictly faster than any $O(n\log n)$ time algorithm when t is relatively larger than m. We exploit this algorithm for a polygonal domain as a subroutine of the $O(n\alpha(n)\log n+k)$-time algorithm that computes the Voronoi diagram of non-disjoint line segments.

The remaining of the paper is organized as follows: After introducing some preliminaries in Section 2, we present our algorithm that computes the Voronoi diagram of a polygonal domain in Section 3. Then, Section 4 is devoted to describe and analyze our algorithm that computes the Voronoi diagram of line segments.

2. Preliminaries

Throughout the paper, we use the following notations: For a subset $A\subset\mathbb{R}^2$, we denote by ∂A the boundary of A with the standard topology. For any two points $p\in\mathbb{R}^2$ and $q\in\mathbb{R}^2$, let $\overline{pq}$ denote the line segment between p and q.

2.1. Voronoi diagrams of line segments

Let S be a set of n line segments in the plane $\mathbb{R}^2$. The *Voronoi diagram* $\mathrm{VD}(S)$ of S is a subdivision of the plane $\mathbb{R}^2$ into *Voronoi regions* $R(s,S)$, defined to be

$$R(s,S):=\bigcap_{s'\in S\setminus\{s\}}\{x\in\mathbb{R}^2\mid d(x,s)<d(x,s')\},$$

where $d(x,s)$ denotes the Euclidean distance from point x to segment s.

As done in the literature [1,21], we regard each segment $s\in S$ as three distinct *sites*: the two endpoints of s and the relative interior of s. We thus assume that the set S is implicitly the set of points and open line segments that form n closed line segments. See Fig. 1. Note that each Voronoi edge of $\mathrm{VD}(S)$ is then either a straight or parabolic segment. The Voronoi vertices of $\mathrm{VD}(S)$ are distinguished into two kinds: those of one kind are proper vertices which are equidistant points from three distinct sites, while those of the other kind are simply the intersection points of S at which at least three Voronoi edges meet. Fig. 2 illustrates the Voronoi diagram of an example set of line segments.

When we are interested in the diagram $\mathrm{VD}(S)$ inside a compact region $A\subseteq\mathbb{R}^2$, we shall write $\mathrm{VD}_A(S)$ to denote the subdivision of A induced by $\mathrm{VD}(S)$. In other words, $\mathrm{VD}_A(S)$ is obtained by intersecting the diagram $\mathrm{VD}(S)$ itself with A.

The following fact is well known as the *star-shape* or *weak star-shape property* of Voronoi regions of a point or an open line segment.

Lemma 1. *If $s\in S$ is a point, then $R(s,S)$ is star-shaped with respect to s. If $s\in S$ is an open line segment, then $R(s,S)$ is weakly star-shaped in the sense that for any $x\in R(s,S)$, the segment $\overline{xs_x}$ is totally contained in $R(s,S)$, where s_x is the perpendicular foot from x to segment s.* □

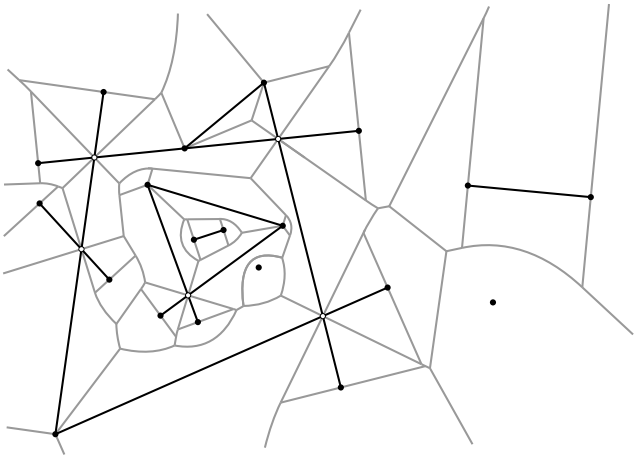

Fig. 2. The Voronoi diagram of non-disjoint line segments.

Note that this fundamental observation implies the monotonicity of each Voronoi region $R(s, S)$ of any site $s \in S$ with respect to the site s; that is, $R(s, S)$ is rotationally monotone around s if s is a point, or is monotone in the direction of s if s is a segment in the sense that every line vertical to s intersects $R(s, S)$ in at most one connected subset.

As pointed out in Section 1, cutting segments in S at every intersection point and applying any existing optimal algorithm, including Yap [21], Klein [15], and Fortune [10], yields the following immediate upper bound.

Lemma 2. *Let S be a set of n line segments with k pairwise intersections. Then, the Voronoi diagram $\mathsf{VD}(S)$ of S has $\Theta(n+k)$ combinatorial complexity and can be computed in $O((n+k)\log n)$ time.* □

On the other hand, one can show a lower bound of $\Omega(n \log n + k)$ for computing $\mathsf{VD}(S)$ by a reduction from the problem of finding the intersection points of line segments, of which the lower bound is known to be $\Omega(n \log n + k)$ in the worst case in the algebraic decision tree model [7].

Lemma 3. *Any algorithm that correctly computes the Voronoi diagram $\mathsf{VD}(S)$ of S takes $\Omega(n \log n + k)$ time in the worst case in the algebraic decision tree model.* □

2.2. Weakly simple polygons

A *polygon* in the plane is a closed curve that is piecewise linear. A polygon can be defined by a cyclic sequence of points in $\mathbb{R}^2$, called *vertices*, that are connected by line segments, called *edges*. A polygon is called *simple* if and only if all of its vertices are distinct and its edges intersect only at their common endpoints. Note that if a polygon is not simple, then its vertices are not necessarily distinct and its edges may overlap or cross each other.

A *weakly simple polygon* is, informally, a polygon without "self-crossing". Intuitively, one may think that a weakly simple polygon is allowed to have vertices and edges "touched". Although the definition using the term "self-crossing" is common in the literature, it is turned to be imprecise as a formal definition. See Chang et al. [5] for discussions on this subject. In this work, we adopt their definition of weakly simple polygons.

Definition 1. (See Chang et al. [5].) A polygon P is *weakly simple* if and only if P has at most two vertices or P can be made simple by an arbitrarily small perturbation of its vertices.

By definition, we can find a simple polygon that is arbitrarily close to any given weakly simple polygon by a perturbation of its vertices; see Fig. 3. Note that a line segment or a point itself is considered a degenerate form of weakly simple polygons, and that the interior of a weakly simple polygon may be an empty set. By convention, any (weakly) simple polygon is also considered as a compact subset containing its interior.

3. Voronoi diagram of a polygon

In this section, we consider some special cases of the set S of line segments, namely, when S forms a weakly simple polygon or a polygon having holes.

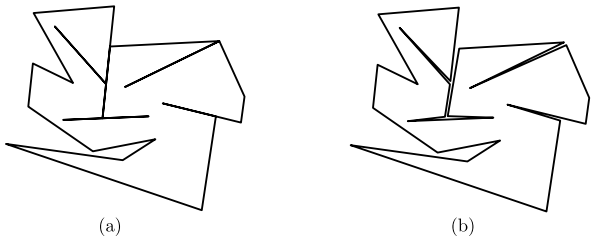

Fig. 3. (a) A weakly simple polygon and (b) a simple polygon obtained by a small perturbation.

3.1. Voronoi diagram of a weakly simple polygon

Let P be a weakly simple polygon with n vertices. The *Voronoi diagram of a (weakly) simple polygon* P is the Voronoi diagram $\mathsf{VD}_P(P)$ of the vertices and edges of P inside the polygon P. We denote by $\mathsf{VD}^-(P) = \mathsf{VD}_P(P)$ the Voronoi diagram of a polygon P.

If P is a simple polygon, $\mathsf{VD}^-(P)$ can be computed in linear time by Chin et al. [9]. Chin et al.'s algorithm indeed extends to weakly simple polygons as well since their approach is based on decomposing a given polygon P into histograms and does not rely on the strict simplicity of P. Also, by the definition of weakly simple polygons, it is not difficult to show that our case can also be handled by the algorithm of Chin et al. [9] after a perturbation of its vertices as in Fig. 3.

Lemma 4. *(See Chin et al. [9].) The Voronoi diagram* $\mathsf{VD}^-(P)$ *of a weakly simple polygon* P *with* n *vertices can be computed in* $O(n)$ *time.*

The *medial axis* $\mathsf{MA}(P)$ of a polygon P is the set of points x in P such that x has at least two distinct points that are closest from x among all points on ∂P. Remark that the medial axis $\mathsf{MA}(P)$ of a weakly simple polygon P can also be computed in linear time since $\mathsf{MA}(P)$ is a subset of $\mathsf{VD}^-(P)$ [9].

Next, we show that an optimal point location structure on $\mathsf{VD}^-(P)$ can be constructed in linear time if P is a weakly simple polygon. For the purpose, we adopt the hierarchical subdivision method for planar triangulations by Kirkpatrick [13]. Note that for a planar triangulation Kirkpatrick's point location structure can be constructed in linear time and supports logarithmic query time. It is not trivial to apply Kirkpatrick's method to our case since some edges of $\mathsf{VD}^-(P)$ can be parabolic, while it turns to be possible due to the star-shape property of the Voronoi regions, as stated in Lemma 1.

Lemma 5. *For any weakly simple polygon* P*, one can build in linear time a data structure that supports a point location query on* $\mathsf{VD}^-(P)$ *in logarithmic query time.*

Proof. Let $\mathsf{VD} = \mathsf{VD}^-(P)$. Compute VD in linear time by Lemma 4. For any parabolic edge e of VD, we consider its extension γ_e that is a whole parabola. Let v_e be the *vertex of the parabola* γ_e, defined to be the intersection point between γ_e and its axis of symmetry. Recall that any parabolic edge e of VD is determined by a segment site $s \in P$ and a point site $s' \in P$: that is, e lies on the common portion of the boundaries of $R(s, P)$ and $R(s', P)$. More specifically, the supporting line of s is the directrix of γ_e and s' is the focus of γ_e. We will call such s and s' the *base* and *focus* of e, respectively. By Lemma 1, we make a basic observation:

(*) For any parabolic edge e of VD, any line segment between two points on e is contained in $R(s', P)$, where s' is the focus of e.

This also implies that any line segment between two points on e does not intersect $R(s, P)$.

First, we refine our original planar subdivision VD into VD^* by performing the following operations in order: (i) for each parabolic edge e of VD, if e contains v_e, then introduce a new vertex at v_e and cut e at v_e into two parabolic edges, and then (ii) for each parabolic edge e after step (i), introduce a new vertex at the perpendicular foot of each endpoint p of e onto the segment site s defining e and a new edge between p and its perpendicular foot onto s. Certainly, this refined subdivision VD^* can be built in linear time from VD. If a face f of VD^* is incident to a parabolic edge e with its base s and focus s', then, obviously, we have either $f \subseteq R(s, P)$ or $f \subseteq R(s', P)$. We then observe the following by our construction of VD^*:

(**) If $f \subseteq R(s, P)$, then the boundary of f consists of exactly four edges: e, a sub-segment of s, and the two perpendicular segments from the endpoints of e.

In addition, we *mark* all faces f of VD^* that are incident to at least one parabolic edge.

Second, we linearize VD^* by making all parabolic edges of VD^* *straight*. For each edge e of VD^*, let $\overline{e}$ be the line segment joining the two endpoints of e. We replace every edge e of VD^* by $\overline{e}$, resulting in a straight-line planar subdivision $\overline{\mathsf{VD}^*}$.

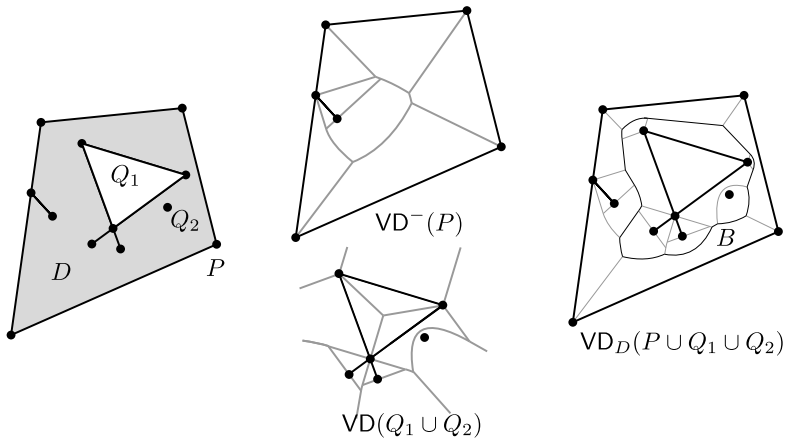

Fig. 4. How to merge $\mathsf{VD}^-(P)$ and $\mathsf{VD}(Q)$ to obtain $\mathsf{VD}_D(P \cup Q)$, for an example polygonal domain D having its outer boundary P and two holes Q_1, Q_2. The outer boundary P and the holes Q_1, Q_2 are weakly simple polygons with 7 vertices, 7 vertices, and one vertex, respectively. The merge curve $\beta = B$ is depicted as black solid line.

Note that $\overline{\mathsf{VD}^*}$ is a proper planar map since this replacement does not introduce a new crossing with the original edges of VD^* by Observation (*). We associate each edge $\overline{e}$ of $\overline{\mathsf{VD}^*}$ with its original edge e of VD^* and the base s and focus s' of e. Also, associate each face $\overline{f}$ of $\overline{\mathsf{VD}^*}$ with its original face f of VD^*.

We claim that the point location on VD^* can be reduced from that on $\overline{\mathsf{VD}^*}$ in $O(1)$ time. Suppose that we know the face $\overline{f}$ of $\overline{\mathsf{VD}^*}$ that contains any query point q. If the face f in VD^* associated with $\overline{f}$ is not marked (so, incident to no parabolic edge), then we are done since $f = \overline{f}$. Or, if $f \subseteq R(s', P)$ for some point site $s' \in P$, then we have $q \in f$ by Observation (*). Otherwise, f is marked, so is incident to a parabolic edge e, and $f \subseteq R(s, P)$, where s is the base of e. Then f is incident to exactly four edges by Observation (**) and the region $\overline{f} \setminus f$ is a subset of $R(s', P)$, where s' is the focus of e, by Observation (*). Hence, by testing in which side of e q lies, we specify whether $q \in f$ or $q \in R(s', P)$.

Thus, we are done by constructing a point location structure for $\overline{\mathsf{VD}^*}$ in linear time. We triangulate each face of $\overline{\mathsf{VD}^*}$, which forms a simple polygon, in linear time [6], and then build the hierarchical structure of Kirkpatrick [13] on the triangulation. It is known that the Kirkpatrick's structure can be constructed in linear time and supports logarithmic query time for point location. Our point location structure thus consists of all the intermediate maps VD^*, $\overline{\mathsf{VD}^*}$, and the Kirkpatrick's location structure on the triangulation of $\overline{\mathsf{VD}^*}$. All the cost of construction is shown to be linear and the query time for point location on VD is logarithmic, so the lemma follows. □

3.2. Voronoi diagram of a polygon with holes

We now extend our interests to a polygon with polygonal holes, called a *polygonal domain*, also known as a *multiply-connected polygon* [20] or a *pocket* [11] in the literature. A polygonal domain $D \subset \mathbb{R}^2$ is a polygonal region that may have one or more *holes*. If D has h holes, then its boundary ∂D consists of $h+1$ pairwise disjoint (weakly) simple polygons $P, Q_1, Q_2, \ldots, Q_h$ such that P contains all the other $Q_1, \ldots, Q_h$ in its interior. We call P the *outer boundary* of D, and $Q_1, \ldots, Q_h$ the *holes* of D. In the literature, the outer boundary and the holes of polygonal domains are often supposed to be simple polygons [11,20]. However, in this paper, we relax the condition for P and the Q_i to be weakly simple polygons.

Let $\mathsf{VD}^-(D) := \mathsf{VD}_D(P \cup Q_1 \cup \cdots \cup Q_h)$ be the Voronoi diagram of a given polygonal domain D. Since the edges of D intersect only at their endpoints, which are the vertices of D, we can compute $\mathsf{VD}^-(D)$ in $O(n \log n)$ time using existing algorithms [10,15,21], where n denotes the total number vertices of D. In the following, we achieve an improved time bound by introducing more parameters: let m denote the total number of vertices of the holes $Q_1, \ldots, Q_h$ of D and t denote the number of vertices of the outer boundary P of D. Note that $n = m + t$.

Our strategy is as follows: Let $Q := Q_1 \cup \cdots \cup Q_h$ be the set of all the vertices and the edges that form the holes $Q_1, \ldots, Q_h$. We first compute the Voronoi diagram $\mathsf{VD}^-(P)$ of the outer boundary P, and the Voronoi diagram $\mathsf{VD}(Q)$ of the edges of the holes. We then merge the two diagrams into our target $\mathsf{VD}^-(D)$. Recall that $\mathsf{VD}^-(P)$ can be computed in $O(t)$ time by Lemma 4 and $\mathsf{VD}(Q)$ in $O(m \log m)$ time since P is a weakly simple polygon and Q is the set of line segments that may intersect only at their endpoints.

Consider the vertices and the edges of $\mathsf{VD}^-(D) = \mathsf{VD}_D(P \cup Q)$ to be computed. We make a general position assumption that no point $x \in D$ is equidistant from four distinct sites in $P \cup Q$; one can remove this assumption by a standard perturbation technique [9]. Some of them come from $\mathsf{VD}^-(P)$ or from $\mathsf{VD}(Q)$, and the others cannot be found in either of the two diagrams $\mathsf{VD}^-(P)$ and $\mathsf{VD}(Q)$, which are the set of equidistant points from two closest sites $s \in P$ and $s' \in Q$. We denote the union of all such edges and vertices of $\mathsf{VD}_D(P \cup Q)$ by B. The set $B \subset D$ can be described as the set of points x such that $d(x, P) = d(x, Q)$, where $d(x, P) = \min_{s \in P} d(x, s)$ and $d(x, Q) = \min_{s' \in Q} d(x, s')$. By definition, B properly divides the interior of D into two regions, one closer to P and the other closer to Q. The following lemma describes how B looks

topologically. For a simple closed curve $C \subset \mathbb{R}^2$, we denote by C^- the *interior* of C, that is, the region bounded by C, and by $C^+ := \mathbb{R}^2 \setminus (C^- \cup C)$ the *exterior* of C.

Lemma 6. *Let $\beta_1, \ldots, \beta_b$ be the connected components of B. We then have:*

(i) *Each component β_i for $i = 1, \ldots, b$ is a simple closed curve in D.*
(ii) *Every hole of D is contained in the interior β_i^- of some β_i.*
(iii) *Each β_i contains at least one hole of D in its interior β_i^-.*
(iv) *No two distinct curves β_i and β_j are nested, that is, $\beta_i \not\subset \beta_j^-$ and $\beta_j \not\subset \beta_i^-$.*

Proof. Pick any point $x \in B$, and take an arbitrarily small open disk U_x centered at x. Consider the intersection between ∂U_x and B. The intersection $\partial U_x \cap B$ is a finite set of points by our choice of U_x.

Observe that $|\partial U_x \cap B|$ is never an odd number larger than two since B properly divides D; the sections around x divided by the edges incident to x should alternate a region of $s \in P$ and that of $s' \in Q$. If $|\partial U_x \cap B| = 1$, then x must lie on some site in $P \cup Q$, again because B properly divides D. Then, $d(x, P) = \min_{s \in P} d(x, s) = 0$. By the definition of B, we must have that $d(x, Q) = 0$, too. This implies that P intersects Q at x, a contradiction, since the outer boundary and each hole is disjoint. Thus, we have that $|\partial U_x \cap B|$ is even. Moreover, observe that if $|\partial U_x \cap B| \geq 4$, then x is a Voronoi vertex of $\mathsf{VD}_D(P \cup Q)$ such that there are at least four distinct sites in $P \cup Q$ that are equidistant from x. This contradicts to our general position assumption, so there is no such point x. This implies that for all points $x \in B$ $|\partial U_x \cap B| = 2$, and hence each connected component of B should be a simple closed curves. This proves property (i).

Since B properly divides D, the union of regions $R(s, P \cup Q)$ for all $s \in P$ coincides with the intersection of the exteriors of the closed curves β_i for all $i = 1, \ldots, b$, that is,

$$\bigcup_{s \in P} R(s, P \cup Q) = \bigcap_{i=1,\ldots,b} \beta_i^+.$$

This also means that the Voronoi region $R(s', P \cup Q)$ for any $s' \in Q$ must lie inside the interior β_i^- of some curve β_i. Thus, for any hole Q_j of D and any site $s' \in Q_j$ of Q_j, $R(s', P \cup Q) \subseteq \beta_i^-$ for some β_i. Also, note that B does not intersect hole Q_j as discussed above, so Q_j lies completely in the interior β_i^- of β_i or completely in its exterior β_i^+. Suppose that $Q_j \subset \beta_i^+$. Then, $s' \in Q_j$ must lie in the exterior β_i^+ of β_i. However, the region $R(s', P \cup Q)$ has the star-shape property of Lemma 1, so s' lies in the interior β_i^- of β_i since $R(s', P \cup Q) \subseteq \beta_i^-$. So, we have a contradiction, proving property (ii).

The above argument also shows that for each curve β_i, its interior is closer to Q than to P and its exterior is closer to P than to Q, locally. This implies that β_i^- must contain at least one hole of D, proving property (iii). Property (iv) is also implied since B properly divides D. □

We call such a closed curve $\beta \subseteq B$, described in Lemma 6, a *merge curve*. Lemma 6 tells us that B consists of one or more merge curves and each hole Q_i is contained in a unique merge curve. Once all the merge curves in B are specified, one can easily compute the target diagram $\mathsf{VD}_D(P \cup Q)$ by cutting and gluing $\mathsf{VD}^-(P)$ and $\mathsf{VD}(Q)$ along the merge curves B in time linear to the combinatorial complexity of the final output $\mathsf{VD}_D(P \cup Q)$, which is bounded by $O(n)$. Hence, we now focus on how to efficiently find every merge curve $\beta \subseteq B$.

The following lemma will be helpful for further discussions.

Lemma 7. *Let $\beta \subseteq B$ be any merge curve, and $s \in P$ and $s' \in Q$ be any two sites. Then, $\beta^+ \cap R(s, P)$ and $\beta^- \cap R(s', Q)$ are monotone with respect to s and s', respectively. Also, if $\beta^+ \cap R(s, P)$ is nonempty, then s lies in β^+; if $\beta^- \cap R(s', Q)$ is nonempty, then s' lies in β^-.*

Proof. Recall that β is the union of some Voronoi edges of $\mathsf{VD}_D(P \cup Q)$. Thus, $\beta \cap R(s, P)$ is a portion of the boundary of $R(s, P \cup Q)$ and $\beta \cap R(s', Q)$ is a portion of the boundary of $R(s', P \cup Q)$. Since $R(s, P \cup Q)$ and $R(s', P \cup Q)$ are monotone with respect to s and s', respectively, by Lemma 1, so are $\beta \cap R(s, P)$ and $\beta \cap R(s', Q)$. Moreover, $R(s, P \cup Q) \subset \beta^+$ while $R(s', P \cup Q) \subset \beta^-$. Hence, we prove the first statement of the lemma. Further, if they are not empty, then s and s' must lie in β^+ and in β^-, respectively, again by Lemma 1. □

Now, suppose that $\mathsf{VD}^-(P)$ and $\mathsf{VD}(Q)$ have been computed correctly, and an optimal point location structure on $\mathsf{VD}^-(P)$ has been built. Such a structure supports a point location query on $\mathsf{VD}^-(P)$ in $O(\log t)$ time, and can be constructed in $O(t)$ time by Lemma 5. Our merge algorithm is then described in Fig. 5.

Note that after a merge curve β is traced and identified, one can easily find out which hole lies in its interior β^- by traversing all the regions of $\mathsf{VD}(Q)$ intersected by β^- by Lemmas 6 and 7. From now on, we thus describe and analyze (1) how to find a point $z \in \beta$ and (2) how to trace the merge curve β in more details.

Algorithm MergeTwoVDs
1: **while** there is Q_i that is not bounded by a merge curve traced before **do**
2: Let $\beta \subseteq B$ be the merge curve such that $Q_i \subset \beta^-$.
3: Find a point $z \in \beta$.
4: Trace β from z.
5: Identify all Q_j bounded by β, that is, $Q_j \subset \beta^-$.
6: **end while**
7: Cut and glue $\mathsf{VD}^-(P)$ and $\mathsf{VD}(Q)$ along all merge curves.
8: **return** the resulting diagram as $\mathsf{VD}_D(P \cup Q) = \mathsf{VD}^-(D)$.

Fig. 5. Algorithm for merging $\mathsf{VD}^-(P)$ and $\mathsf{VD}(Q)$ into $\mathsf{VD}_D(P \cup Q)$.

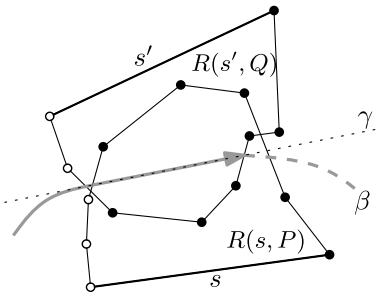

Fig. 6. Tracing a merge curve β in $R(s, P) \cap R(s', Q)$. To find the point on the boundary hit by β, test vertices of $R(s, P)$ and $R(s', Q)$ in the order along β which side of γ it lies in. Vertices marked with black dots will get tested in order and those marked with white dots will not because they are passed already by β.

3.2.1. Finding a point $z \in \beta$

Assume that we are to find a point z on a merge curve β, which contains Q_i in its interior β^-. For the purpose, we first pick any vertex q of a specific Q_i and find a point $p \in P$ that is the closest from q among all points in P. This can be done by a point location query on $\mathsf{VD}^-(P)$. We then walk along $\overline{qp}$ from q towards p until we meet a point z on β.

Our strategy is promising due to the following lemma.

Lemma 8. *The segment $\overline{qp}$ intersects β exactly in a single point.*

Proof. By Lemma 7, β intersects $\overline{qp}$ no more than once. In the following, we show that β intersects $\overline{qp}$ at least once.

Let $f(x) := d(x, Q) = \min_{s' \in Q} d(x, s')$ and $g(x) := d(x, P) = d(x, p)$. Observe that $f(x)$ and $g(x)$ are continuous over $x \in \overline{qp}$ and it holds that $f(q) - g(q) < 0$ and $f(p) - g(p) > 0$. Therefore, there exists a point $z \in \overline{qp}$ such that $f(z) = g(z)$ by Bolzano's Theorem. Obviously, z lies on B, and Lemmas 6 and 7 imply that z lies on the merge curve that contains Q_i, that is, β. □

More precisely, we walk on the diagram $\mathsf{VD}(Q)$ along the segment $\overline{qp}$ from q towards p. While walking in a Voronoi region $R(s', Q)$, the function $f(x) := d(x, Q) = d(x, s')$ can be explicitly described for $x \in \overline{qp} \cap R(s', Q)$. We check whether the equation $f(x) = d(x, p)$ has a solution $z \in \overline{qp} \cap R(s', Q)$. If so, we are done; otherwise, we proceed to the next region of $\mathsf{VD}(Q)$ along $\overline{qp}$. Lemma 8 guarantees that this procedure will terminate with a point $z \in \beta$.

3.2.2. Tracing β from $z \in \beta$

Let $s \in P$ and $s' \in Q$ be the sites such that $z \in R(s, P)$ and $z \in R(s', Q)$. This is automatically identified once $z = \overline{qp} \cap \beta$ is found in the above step. Then, z must lie on the *bisecting curve* γ between s and s'. Note that γ is either a straight line or a parabola according to the types of s and s'.

In order to trace β from z, we walk along locally γ in the direction that $s \in P$ lies to the right and $s' \in Q$ lies to the left of γ. Note that γ coincides with β in $R(s, P) \cap R(s', Q)$. We thus trace γ until it hits the boundary $\partial R(s, P)$ or $\partial R(s', Q)$. Assume that γ hits $\partial R(s, P)$ before $\partial R(s', Q)$. Then, we proceed to the next region $R(s'', P)$ in $\mathsf{VD}^-(P)$, replacing s by s'' and repeating the above procedure. Since β is a closed curve, we are done when getting back to z.

In order to find the point on the boundary $\partial R(s, P)$ or $\partial R(s', Q)$ hit by γ, we test each vertex on $\partial R(s, P)$ and $\partial R(s', Q)$ in a certain order whether it lies to the left or to the right of the bisecting curve γ between s and s'. This test can be performed in $O(1)$ time per tested vertex by examining a point and a line or a parabola γ. The order of vertices is induced by the tracing direction of β. Since $\beta \cap R(s, P) \cap R(s', Q)$ is monotone with respect to s and s', simultaneously, by Lemma 7, one can find the next vertex from a vertex in $O(1)$ time along the boundary of each region $R(s, P)$ and $R(s', Q)$. See Fig. 6. We thus test the vertices in this order. If the tested vertex v is of $R(s, P)$ and it is the first that lies to the right of γ, then γ crosses $R(s, P)$ between v and its preceding vertex on $\partial R(s, P)$; if v is of $R(s', Q)$ and it is the first that lies to the left of γ,

then γ crosses $R(s', Q)$ between v and its preceding vertex on $\partial R(s', Q)$. Hence, as soon as we find a vertex of $R(s, P)$ lying to the right of γ or one of $R(s', Q)$ lying to the left of γ, γ hits the boundary $\partial R(s, P)$ or $\partial R(s', Q)$, respectively.

Assume without loss of generality that γ hits $\partial R(s, P)$ before $\partial R(s', Q)$. Then, β is traced up to the hitting point, and is about to enter the next region $R(s'', P)$ from $R(s, P)$ for some $s'' \in P$. Since we know the entering point $y \in \gamma$ at $\partial R(s'', P)$, we know the next vertex along $\partial R(s'', P)$ from y. Note that we do not have to test vertices on $\partial R(s', Q)$ that have been tested before by Lemma 7. Hence, we substitute s by s'' and proceed on testing next vertices on $\partial R(s'', P)$ and $\partial R(s', Q)$ as above without repetition.

3.2.3. The time complexity

We now analyze the cost of finding a merge curve β, which corresponds to a single **while**-loop in the algorithm described in Fig. 5.

Lemma 9. *Specifying a merge curve $\beta \subseteq B$ can be done in time $O(\log t + M_\beta + T_\beta + |\beta|)$, where M_β denotes the total complexity of Voronoi regions $R(s', Q)$ of $\mathsf{VD}(Q)$ that are intersected by β^-, T_β denotes the number of Voronoi vertices of $\mathsf{VD}^-(P)$ lying inside β^-, and $|\beta|$ denotes the number of Voronoi edges of $\mathsf{VD}_D(P \cup Q)$ along β.*

Proof. Finding a point $z \in \beta$ involves a point location on $\mathsf{VD}^-(P)$ and a walk along the segment $\overline{qp}$ up to $z \in \beta$ on $\mathsf{VD}(Q)$. The time spent for the former is bounded by $O(\log t)$ by Lemma 5 and that for the latter is bounded by $O(M_\beta)$ by Lemmas 6 and 7.

The cost of tracing β is proportional to (i) the number of tested vertices of $\mathsf{VD}^-(P)$ and $\mathsf{VD}(Q)$ plus (ii) the number of regions of $\mathsf{VD}^-(P)$ and $\mathsf{VD}(Q)$ that are crossed by β. Observe that the number of tested vertices of $\mathsf{VD}(Q)$ is easily bounded by $O(M_\beta)$ and the number of regions that are crossed by β is exactly the same as $|\beta|$, the number of Voronoi edges in β.

The tested vertices of $\mathsf{VD}^-(P)$ are either those lying in β^- or the others lying in β^+. The number of those lying in β^- is exactly T_β, while the others in β^+ have been tested only when β hits the boundary of the corresponding Voronoi region. Therefore, the number of tested vertices of $\mathsf{VD}^-(P)$ is bounded by $O(T_\beta + |\beta|)$, so the lemma follows. □

We then bound the total running time of our merging algorithm.

Lemma 10. *Merging two Voronoi diagrams $\mathsf{VD}^-(P)$ and $\mathsf{VD}(Q)$ into $\mathsf{VD}_D(P \cup Q)$ can be done in $O(h \log t + m + t)$ time.*

Proof. The total time bound is obtained by summing $O(\log t + M_\beta + T_\beta + |\beta|)$ over all merge curves $\beta \subseteq B$ by Lemma 9. We show that (i) $\sum_\beta M_\beta = O(m)$, (ii) $\sum_\beta T_\beta = O(t)$, and (iii) $\sum_\beta |\beta| = O(m + t)$.

(i) The value M_β counts the total complexity of $R(s', Q)$ for all $s' \in Q_j$ and $Q_j \subset \beta^-$, and each hole Q_j is contained in a unique merge curve by Lemma 6. This implies that $\sum_\beta M_\beta$ does not exceed the complexity of the Voronoi diagram $\mathsf{VD}(Q)$, bounded by $O(m)$.

(ii) T_β counts the number of vertices of $\mathsf{VD}^-(P)$ lying in β^-. Since the merge curves are not nested by Lemma 6, $\sum_\beta T_\beta$ does not exceed the total number of vertices of $\mathsf{VD}^-(P)$, bounded by $O(t)$.

(iii) The merge curves $\beta \subseteq B$ are the union of Voronoi edges of $\mathsf{VD}_D(P \cup Q)$. Thus, its total complexity $\sum_\beta |\beta|$ does not exceed the complexity of the merged diagram $\mathsf{VD}_D(P \cup Q)$, bounded by $O(m + t)$.

Therefore, we have

$$\sum_\beta O(\log t + M_\beta + T_\beta + |\beta|) = \sum_\beta O(\log t) + O(m + t) = O(h \log t + m + t),$$

since the number of merge curves is at most h by Lemma 6. □

We are finally ready to conclude the main theorem of this section.

Theorem 1. *Let D be a polygonal domain whose outer boundary and holes may be weakly simple polygons. The Voronoi diagram $\mathsf{VD}^-(D)$ can be computed in $O(m \log(m + t) + t)$ time, where m is the total complexity of the holes of D and t is the complexity of the outer boundary of D.*

Proof. Our algorithm first computes $\mathsf{VD}^-(P)$ and $\mathsf{VD}(Q)$ in $O(m \log m + t)$ time and builds the point location structure on $\mathsf{VD}^-(P)$ in additional $O(t)$ time as described in Lemma 5. Then, the two diagrams $\mathsf{VD}^-(P)$ and $\mathsf{VD}(Q)$ are merged into $\mathsf{VD}^-(D) = \mathsf{VD}_D(P \cup Q)$, as described above, in time $O(h \log t + m + t)$ by Lemma 10. The total time spent is then bounded by $O(m \log m + t) + O(h \log t + m + t) = O(m \log(m + t) + t)$. □

Since the medial axis $\mathsf{MA}(D)$ of a polygonal domain D is a subset of its Voronoi diagram $\mathsf{VD}^-(D)$, Theorem 1 automatically implies an improved algorithm that computes the medial axis of a polygonal domain.

Corollary 1. *Given a polygonal domain D, the medial axis $\mathsf{MA}(D)$ can be computed in $O(m\log(m+t)+t)$ time, where m and t are defined as above.*

4. Voronoi diagram of line segments

Let S be a set of n line segments in the plane $\mathbb{R}^2$ with k pairwise intersections. In this section, we describe our algorithm that computes the Voronoi diagram $\mathsf{VD}(S)$ of line segments.

Our algorithm runs based on the decomposition of the plane $\mathbb{R}^2$ into faces of the arrangement of S. The *arrangement* $\mathcal{A}(S)$ of S is a decomposition of $\mathbb{R}^2$ into vertices, edges, and faces induced by the line segments in S. We regard edges and faces of $\mathcal{A}(S)$ relatively open sets: each edge as a set does not contain its end vertices and each face does not contain its incident edges and vertices.

For any face σ of $\mathcal{A}(S)$, we consider its boundary $\partial\sigma$, defined to be the union of vertices and edges that are incident to σ. Since a face in the line segment arrangement can have holes, the boundary $\partial\sigma$ may consist of several connected components, and each of them indeed forms a weakly simple polygon. This means that σ forms a polygonal domain whose outer boundary and holes can be weakly simple polygons, while the unbounded face of $\mathcal{A}(S)$ is regarded as a polygonal domain with outer boundary at infinity. Note that the complexity of the boundary $\partial\sigma$ of a face is bounded by $O(n\alpha(n))$, where $\alpha(n)$ denotes the functional inverse of the Ackermann function, and this bound can be realized [19].

We observe the following.

Lemma 11. *Let σ be any face of the arrangement $\mathcal{A}(S)$ and $x\in\sigma$ be any point. The Voronoi region $R(s,S)$ of site $s\in S$ intersects σ if and only if the site s appears on the boundary $\partial\sigma$ of σ.*

Proof. If the site $s\in S$ appears in $\partial\sigma$, then it is obvious that $R(s,S)$ intersects σ. We thus consider the other direction, and assume that $\mathsf{VD}(s,S)\cap\sigma\neq\emptyset$. Suppose to the contrary that the site $s\in S$ does not appear in the boundary $\partial\sigma$. This implies that for any point $x\in s$ on s and any point $y\in\sigma$, the line segment $\overline{xy}$ must cross the boundary $\partial\sigma$.

We then pick any point $p\in R(s,S)\cap\sigma$ and the point $s_p\in s$ on s such that $d(p,s)=d(p,s_p)$. By the above observation, the segment $\overline{ps_p}$ crosses the boundary $\partial\sigma$; let $s'\in S$ be the site appearing on $\partial\sigma$ intersected by $\overline{ps_p}$. By Lemma 1, it must hold that $\overline{ps_p}\subset R(s,S)$, but it is impossible because at least the intersection point $z=\overline{ps_p}\cap s'$ is closer to s' than to s. We thus obtain a contradiction, and hence s must appear on the boundary $\partial\sigma$. □

Lemma 11 implies that the Voronoi diagram $\mathsf{VD}(S)$ cropped by a face σ coincides with the Voronoi diagram of a polygonal domain σ, that is, $\mathsf{VD}_\sigma(S)=\mathsf{VD}^-(\sigma)$. Hence, we compute the diagram $\mathsf{VD}(S)$ in a face-by-face fashion. For example, the polygonal domain D in Fig. 4 is identical to a face σ of the arrangement of the line segments S in Fig. 2, and it holds that $\mathsf{VD}_\sigma(S)=\mathsf{VD}^-(D)$.

Our algorithm is described in three steps: (1) Compute all the intersection points in S, (2) compute the arrangement $\mathcal{A}(S)$ and the description of its faces, and then (3) for every face σ of $\mathcal{A}(S)$, compute the Voronoi diagram $\mathsf{VD}^-(\sigma)$ of σ. Then, the diagram $\mathsf{VD}(S)$ of the original n line segments can be obtained by taking the union of the face diagrams $\mathsf{VD}^-(\sigma)$ over all faces σ of $\mathcal{A}(S)$.

The first and second steps can be finished in $O(n\log n+k)$ time by Chazelle and Edelsbrunner [7]. The third step is handled by applying Theorem 1 to every bounded face σ of $\mathcal{A}(S)$. Let $m=m_\sigma$ be the total complexity of the holes of σ, and $t=t_\sigma$ be the complexity of the outer boundary of σ. Also, let n_σ be the number of segments in S that appear in the boundary of a hole of σ. Since the boundary of σ forms a polygonal domain whose outer boundary and holes are weakly simple polygons, we apply Theorem 1 to compute $\mathsf{VD}^-(\sigma)$ in time $O(m\log(m+t)+t)$. The unique unbounded face σ_0 of $\mathcal{A}(S)$ can be handled in $O(m_0\log m_0)$ time, where m_0 denotes the complexity of the boundary of the unbounded face, by running any optimal algorithm that computes the Voronoi diagram of disjoint line segments.

We are then ready to conclude our main result.

Theorem 2. *Let S be a set of n line segment with k pairwise intersections. Then, the Voronoi diagram $\mathsf{VD}(S)$ of S can be computed in $O(n\alpha(n)\log n+k)$ time, where $\alpha(\cdot)$ denotes the functional inverse of the Ackermann function.*

Proof. As described above, once the arrangement $\mathcal{A}(S)$ is computed, we apply Theorem 1 to every face σ of the arrangement $\mathcal{A}(S)$. The correctness of our algorithm simply follows from Lemma 11.

We now analyze the time complexity of our algorithm. We spend $O(n\log n+k)$ time for Steps (1) and (2). The total time spent in Step (3) of our algorithm is the sum of $O(m_\sigma\log(m_\sigma+t_\sigma)+t_\sigma)$ over all faces σ of $\mathcal{A}(S)$, plus the effort $O(m_0\log m_0)$ for the unbounded face σ_0. Since $m_0=O(n\alpha(n))$ [19], we have $O(m_0\log m_0)=O(n\alpha(n)\log n)$, which is not more than the claimed bound.

We claim that (i) $\sum_\sigma t_\sigma\le 2n+4k$ and (ii) $\sum_\sigma n_\sigma\le n$. Since $m_\sigma=O(n_\sigma\alpha(n_\sigma))=O(n_\sigma\alpha(n))$ [19], if our claim is true, then we have

$$\sum_{\sigma} O(m_\sigma \log(m_\sigma + t_\sigma) + t_\sigma) = \sum_{\sigma} O(n_\sigma \alpha(n) \log(n_\sigma \alpha(n) + t_\sigma) + t_\sigma)$$
$$= O(n\alpha(n) \log(n\alpha(n) + k) + k)$$
$$= O(n\alpha(n) \log n + k),$$

as desired.

Therefore, we are done by showing that our claim is true. Consider any sub-segment s' after cutting segments in S at every intersection point. Observe that s' can appear in the outer boundary of a face σ of $\mathcal{A}(S)$ at most twice, since s' is incident to at most two faces of the arrangement $\mathcal{A}(S)$. Thus, the sum of t_σ over all faces σ is at most twice the number of such sub-segments, that is, $2n + 4k$, so claim (i) is shown. Claim (ii) easily follows from the fact that each segment $s \in S$ cannot participate in two different holes in different faces of $\mathcal{A}(S)$; otherwise, s must be placed inside two different faces, which is impossible. □

5. Concluding remarks

We presented an almost optimal algorithm that computes the Voronoi diagram $\mathrm{VD}(S)$ of line segments that may intersect or cross each other. By Lemma 3, our algorithm is only an $\alpha(n)$ factor away from any optimal-time algorithm, which is still unknown. Note that for any large $k = \Omega(n\alpha(n) \log n)$, the term k dominates the other term $n\alpha(n) \log n$, so our algorithm turns to be optimal. If $k = O(n)$, then the preliminary algorithm of Lemma 2 guarantees the optimal time bound $O((n+k) \log n) = O(n \log n)$. Consequently, we are now unaware of any optimal algorithm for k in a very narrow range between $\omega(n)$ and $o(n\alpha(n) \log n)$.

Remark that after spending $O(n \log n + k)$ time, we are aware of the number k of intersections in S. Thus, in fact, one can select one of the two algorithms described in Lemma 2 and Theorem 2 according to the number k of intersections.

Though we describe our problem and algorithms under the Euclidean metric, all the results still hold if one adopts any L_p metric or even convex distance function instead of the Euclidean metric. Recall that every argument in this paper has been built up on the basic properties of Voronoi diagrams such as the star-shape property described in Lemma 1 which remains true for any convex distance function, and the algorithm by Chin et al. also works properly for convex distance functions as the authors remarked [9].

References

[1] H. Alt, O. Cheong, A. Vigneron, The Voronoi diagram of curved objects, Discrete Comput. Geom. 34 (3) (2005) 439–453.
[2] F. Aurenhammer, R. Klein, Voronoi diagrams, in: J.-R. Sack, J. Urrutia (Eds.), Handbook of Computational Geometry, Elsevier, 2000.
[3] S.W. Bae, Tight bound and improved algorithm for farthest-color Voronoi diagrams of line segments, Comput. Geom. Theory Appl. 47 (8) (2014) 779–788.
[4] S.W. Bae, An almost optimal algorithm for Voronoi diagrams of non-disjoint line segments, in: Proc. 9th Workshop Algo. Comput., WALCOM 2015, in: Lecture Notes in Comput. Sci., vol. 8973, 2015, pp. 125–136.
[5] H. Chang, J. Erickson, C. Xu, Detecting weakly simple polygons, in: Proc. 26th ACM–SIAM Sympos. Discrete Algo., SODA'15, 2015, pp. 1655–1670.
[6] B. Chazelle, Triangulating a simple polygon in linear time, Discrete Comput. Geom. 6 (1991) 485–524.
[7] B. Chazelle, H. Edelsbrunner, An optimal algorithm for intersecting line segments in the plane, J. ACM 39 (1992) 1–54.
[8] O. Cheong, H. Everett, M. Glisse, J. Gudmundsson, S. Hornus, S. Lazard, M. Lee, H.-S. Na, Farthest-polygon Voronoi diagrams, Comput. Geom. Theory Appl. 44 (4) (2011) 234–247.
[9] F. Chin, J. Snoeyink, C.A. Wang, Finding the medial axis of a simple polygon in linear time, Discrete Comput. Geom. 21 (3) (1999) 405–420.
[10] S.J. Fortune, A sweepline algorithm for Voronoi diagrams, Algorithmica 2 (1987) 153–174.
[11] M. Held, On the Computational Geometry of Pocket Machining, Lecture Notes in Comput. Sci., vol. 500, Springer-Verlag, Berlin, Germany, 1991.
[12] M.I. Karavelas, A robust and efficient implementation for the segment Voronoi diagram, in: Proc. 1st Int. Sympos. Voronoi Diagrams Sci. Engineer., ISVD'04, 2004, pp. 51–62.
[13] D. Kirkpatrick, Optimal search in planar subdivisions, SIAM J. Comput. 12 (1) (1983) 28–35.
[14] D.G. Kirkpatrick, Efficient computation of continuous skeleton, in: Proc. 20th Annu. IEEE Sympos. Found. Comput. Sci., 1979, pp. 18–27.
[15] R. Klein, Concrete and Abstract Voronoi Diagrams, Lecture Notes in Comput. Sci., vol. 400, Springer-Verlag, Berlin, Germany, 1989.
[16] R. Klein, K. Mehlhorn, S. Meiser, Randomized incremental construction of abstract Voronoi diagrams, Comput. Geom. Theory Appl. 3 (3) (1993) 157–184.
[17] D.T. Lee, Medial axis transformation of a planar shape, IEEE Trans. Pattern Anal. Mach. Intell. PAMI-4 (4) (1982) 363–369.
[18] A. Okabe, B. Boots, K. Sugihara, S.N. Chiu, Spatial Tessellations: Concepts and Applications of Voronoi Diagrams, 2nd ed., John Wiley and Sons, New York, 2000.
[19] M. Sharir, P.K. Agarwal, Davenport–Schinzel Sequences and Their Geometric Applications, Cambridge University Press, New York, 1995.
[20] V. Srinivasan, L.R. Nackman, Voronoi diagram for multiply-connected polygonal domains I: algorithm, IBM J. Res. Dev. 31 (3) (1987) 361–372.
[21] C.K. Yap, An $O(n \log n)$ algorithm for the Voronoi diagram of a set of simple curve segments, Discrete Comput. Geom. 2 (1) (1987) 365–393.