Linear-Time Algorithms for the Farthest-Segment Voronoi Diagram and Related Tree Structures

Elena Khramtcova, Sandeep K. Dey, and Evanthia Papadopoulou

Faculty of Informatics, Università della Svizzera italiana (USI), Lugano, Switzerland
{elena.khramtcova, deys, evanthia.papadopoulou}@usi.ch

Abstract. We present linear-time algorithms to construct tree-structured Voronoi diagrams, after the sequence of their regions at infinity or along a given boundary is known. We focus on Voronoi diagrams of line segments, including the farthest-segment Voronoi diagram, the order-(k+1) subdivision within a given order-k Voronoi region, and deleting a segment from a nearest-neighbor diagram. Although tree-structured, these diagrams illustrate properties surprisingly different from their counterparts for points. The sequence of their faces along the relevant boundary forms a Davenport-Schinzel sequence of order 3. Once this sequence is known, we show how to compute the corresponding Voronoi diagram in linear time, expected or deterministic, augmenting the existing linear frameworks for points in convex position, with the ability to handle non-uniqueness of Voronoi faces. Our techniques contribute towards the question of linear-time construction algorithms for tree-like Voronoi diagrams.

1 Introduction

It is well known that the Voronoi diagram of points in convex position can be computed in linear time, given their convex hull [1]. The same holds for a class of related diagrams such as the farthest-point Voronoi diagram, computing the medial axis of a convex polygon, and deleting a point from the classic nearest-neighbor Voronoi diagram. In an abstract setting, the Hamiltonian Abstract Voronoi diagram can be computed in linear time, given the order of Voronoi regions along an unbounded curve, which visits each region exactly once, and which can intersect each bisector only once [12]. This linear construction has recently been extended to include forest structures [5] under the same conditions of no repetition. The medial-axis of a simple polygon can also be computed in linear time [7] by combining polygon decomposition techniques and the linear algorithm for convex polygons. It is therefore natural to ask what other types of tree-structured Voronoi diagrams can be constructed in linear time.

Classical variants of Voronoi diagrams, such as higher-order Voronoi diagrams, for sites other than points, had been surprisingly ignored in the computational geometry, until recently [4,17,18]. Given a set $S$ of $n$ simple geometric

*Research supported in part by the Swiss National Science Foundation, project 20GG21-134355, under the ESF EUROCORES program EuroGIGA/VORONOI.
objects in the plane, called sites, the order-$k$ Voronoi diagram of $S$ is a partitioning of the plane into regions, such that every point within a region has the same $k$ nearest sites. For $k = 1$, this is the nearest-neighbor Voronoi diagram; for $k = n - 1$ it is the farthest-site Voronoi diagram of $S$, where the region of a site $s \in S$ is the locus of points farther away from $s$ than from any other site. Despite similarities, these diagrams illustrate fundamental structural differences from their counterparts for points, such as the presence of disconnected regions.

In this paper we give linear constructions for tree-structured Voronoi diagrams, whose regions may consist of several disjoint faces, after the sequence of their faces at infinity or along a given boundary is known. In particular, we give expected and deterministic linear-time algorithms for constructing the farthest-segment Voronoi diagram, and the order-$(k+1)$ subdivision within a given order-$k$ Voronoi region of segments, which also applies to updating in linear time the nearest-neighbor segment Voronoi diagram following deletion. A major difference from the classic respective problems for points is that the sequence $G$ of regions along the relevant boundary in each case (infinity for the farthest diagram) allows repetition, forming a Davenport-Schinzel sequence of order 3, unlike points, where no repetition can exist. Repetition introduces several complications, including the fact that a sequence $G'$ of regions, along the relevant boundary, for a subset of the original segments, $S' \subset S$, may be quite unrelated to $G$. Existing linear constructions strongly rely on the fact that $G'$ is a subsequence of $G$, thus, they are not easily generalizable to sequences of order $> 1$.

Differences between point and segment Voronoi diagrams are initially surprising. For example, convex hull properties do not characterise the regions of the farthest-segment Voronoi diagram and a single Voronoi region may consist of $\Theta(n)$ disjoint faces [2, 14]. Nevertheless, a closed polygonal curve, termed the farthest hull, and its Gaussian map, still characterize the faces of this diagram [15], which form a Davenport-Schinzel sequence of order 3. Similar observations involve higher-order Voronoi diagrams [17], including higher order abstract Voronoi diagrams [4]. For more information on Voronoi diagrams see the book of Aurenhammer et al. [3].

The segment counterpart of higher-order Voronoi diagrams is an important variant for a variety of applications, where sites are polygonal objects or embedded planar graphs. For example, line segments may represent wires in a VLSI layer or in a different type of network, while disks represent defects. A defect may cause a fault when it overlaps entire sets of such wires. This problem is modeled by the farthest and order-$k, k > 1$, segment Voronoi diagrams. A concrete example, used in the semiconductor industry, is the VLSI critical area extraction problem, which computes the probability of fail of a VLSI design due to random manufacturing defects, see e.g., [16] and references therein.

In this paper, we first illustrate our techniques for the farthest-segment Voronoi diagram. In Section 3 we give a simple randomized algorithm as inspired by the randomized framework for points in convex position [6]. In Section 4 we augment with our techniques to handle repetition, the more involved deterministic
linear framework [1]. In Section 5 we adapt our method to compute the order-$(k+1)$ Voronoi subdivision within an order-$k$ face. Interestingly, the latter case may initially require the computation of two different tree-structured diagrams, which are later merged into one. In future work, we plan to adapt our methods to the abstract counterparts of these diagrams [4,14].

2 Preliminaries and Definitions

We start with a set $S = \{s_1, \ldots, s_n\}$ of $n$ arbitrary line segments in the plane. Line segments may intersect or touch at a single point. The distance between a point $q$ and a line segment $s_i$ is $d(q, s_i) = \min\{d(q, y) : \forall y \in s_i\}$, where $d(q, y)$ denotes the ordinary distance between two points $q, y$ in the $L_2$ (or the $L_p$) metric.

The farthest Voronoi region of a line segment $s_i$ is $\text{freg}(s_i) = \{x \in \mathbb{R}^2 | d(x, s_i) > d(x, s_j), 1 \leq j \leq n\}$. For disjoint line segments or line segments that intersect but do not touch at endpoints, the order-$k$ Voronoi region of a set $H$, where $H \subset S$, $|H| = k$, and $1 \leq k \leq n - 1$, is $k\text{-reg}(H) = \{x | \forall s \in H, \forall t \in S \setminus H d(x, s) < d(x, t)\}$. For an extension of this definition to line segments that may touch at endpoints, see [15]. Note, $\text{freg}(s_i) = k\text{-reg}(S \setminus \{s_i\})$ for $k = n - 1$. The collection of all farthest (resp., order-$k$) Voronoi regions of segments in $S$, together with their bounding edges and vertices, give the farthest-segment Voronoi diagram $\text{FVD}(S)$ (resp. the order-$k$ Voronoi diagram of $S$), see Fig. 1(a). Any maximally-connected subset of a Voronoi region is called a face.

Fig. 1: [15] (a) $\text{FVD}(S)$, $S = \{s_1, \ldots, s_3\}$; (b) its farthest hull; (c) Gmap($S$)

Given two segments $s_i, s_j \in S$, their bisector $b(s_i, s_j)$ is the locus of points equidistant from $s_i$ and $s_j$. For disjoint line segments $b(s_i, s_j)$ is an unbounded curve that consists of a constant number of pieces, where each piece is portion of an elementary bisector between the endpoints and open portions of $s_i, s_j$. In $L_2$, individual pieces of this curve are line segments, rays, and parabolic arcs. If two segments intersect at point $p$, their bisector consists of two such curves intersecting at $p$. If the segments share a common endpoint, the bisector may
contain two-dimensional regions. To avoid such regions a single line segment is typically treated as three entities: two endpoints and an open line segment, see e.g., [10,13]. In the remaining of this section we concentrate on the farthest-segment Voronoi diagram. We return to the order-$k$ version in Section 5.

A farthest Voronoi region $\text{freg}(s_i)$ is non-empty and unbounded in direction $\phi$ if and only if there exists an open halfplane, normal to $\phi$, which intersects all segments in $S$ but $s_i$ [2]. The line $\ell$, normal to $\phi$, bordering such a halfplane is called a supporting line and $\phi$ is called the unit vector of $\ell$, denoted as $\nu(\ell)$. An unbounded Voronoi edge between two regions $\text{freg}(s_i)$ and $\text{freg}(s_j)$ is the unbounded portion of a bisector $b(p,q)$, where $p,q$ are endpoints of $s_i$ and $s_j$ respectively, such that the line through $pq$ induces an open halfplane that intersects all segments in $S$, except $s_i,s_j$ (and possibly except additional segments incident to $p,q$). Segment $\ell$ is called a supporting segment, and the direction of $b(p,q)$ towards infinity is its unit vector. A segment $s_i \in S$ such that the line through $s_i$ is supporting, is called a hull segment.

Fig. 1(a) and (b) illustrate a farthest-segment Voronoi diagram and its hull respectively. In Fig. 1(b), supporting segments are depicted as dashed segments and hull segments are depicted in bold. Arrows illustrate the unit vectors of all supporting and hull segments.

The Gaussian map of $\text{FVD}(S)$ (for brevity $\text{Gmap}(S)$) provides a correspondence from the faces of $\text{FVD}(S)$ to a circle of directions $K_0$ [15]. Each Voronoi face is mapped to a set of directions (an arc on $K_0$) along which the face is unbounded. Thus, the Gaussian map on $K_0$ can be viewed as a cyclic sequence of arcs, where each arc corresponds to one face of $\text{FVD}(S)$. Two neighboring arcs are separated by the unit vector of an unbounded Voronoi edge. There are two types of arcs: (1) arcs that correspond to a single endpoint of a segment, called a single-vertex arc; and (2) arcs that correspond to an entire segment (a hull segment), called a segment arc. A segment arc consists of two sub-arcs, one for each endpoint of the segment, separated by the segment unit vector. Between any two arcs of the same segment in $\text{Gmap}(S)$, at least one segment arc must always occur (see [15]). Fig. 1(c) illustrates the respective Gaussian map. $\text{Gmap}(S)$ can be computed in $O(n \log n)$ time (or output-sensitive $O(n \log h)$, where $h$ is the number of faces in $\text{FVD}(S)$) by adapting techniques to compute a convex hull [15] (other than Graham’s scan).

The standard point-line duality transformation $T$ offers a correspondence from the faces of $\text{FVD}(S)$ to the upper or lower envelopes of wedges. It maps a point $p = (a,b)$ in the primal plane to a line $T(p) : y = ax - b$ in the dual plane, and vice versa. A segment $s_i = uv$ is sent into two wedges, one below and one above lines $T(u)$ and $T(v)$, referred to as the lower and upper wedge, respectively. Let $E$ (resp., $E'$) be the boundary of the union of the lower (resp., upper) wedges. Then the faces of $\text{FVD}(S)$ in cyclic order, correspond exactly to the edges of $E$ and $E'$ in $x$-order [2]. The boundaries of $E$ and $E'$ form Davenport-Schinzel sequences of order 3 (order 2 if segments are non-crossing); and their complexity is $O(n)$ [20].
Linear-Time Algorithms for Tree-Structured Voronoi Diagrams

There is a clear equivalence between $E$ (resp., $E'$) and the upper (resp., lower) Gmap: the edges of $E$ in increasing $x$-order correspond exactly to the arcs of the upper Gmap in counterclockwise order; the vertices of $E$ are exactly the unit vectors of the upper Gmap; and the apexes of wedges in $E$ are exactly the unit vectors of hull segments, that is, they denote segment arcs. This correspondence is used extensively in the sequel.

Throughout this paper, given an arc $\alpha$, let $s_{\alpha}$ denote the segment in $S$ that induces $\alpha$.

2.1 The Farthest Voronoi Diagram, given a Gaussian Map

To derive a linear-time algorithm to compute $FVD(S)$, given $Gmap(S)$, we need the ability to compute the farthest Voronoi diagram for a valid subsequence of $Gmap(S)$. Recall that the Gaussian map can be viewed as a cyclic sequence of arcs on a circle of directions. There are two major difficulties in this task: (1) for $S' \subset S$, $Gmap(S')$ need not be a subsequence of $Gmap(S)$; and (2) a subsequence of $Gmap(S)$ need not correspond to the Gaussian map of a Voronoi diagram. None of these issues is relevant to points, where the Gaussian map is equivalent to a convex hull.

Let $G'$ be a minimally augmented subsequence of $Gmap(S)$ such that $G'$ forms a Gaussian map. The upper and lower part of $G'$ correspond respectively to an $x$-monotone path in the arrangement of lower and upper wedges of $S$ in dual space.

Let $\alpha$ be an arc in $G'$ associated with a segment $s_{\alpha} \in S$. Given a point $x$, let $r(x, s_{\alpha})$ be the unbounded portion of the ray emanating from $s_{\alpha}$ that realizes the Euclidean distance between $s_{\alpha}$ and $x$, starting at $x$ and extending to infinity away from $s_{\alpha}$. We say that $x$ is attainable from $\alpha$ in $G'$ if the direction of $r(x, s_{\alpha})$ is contained in $\alpha$. Let $d(x, \alpha) = d(x, s_{\alpha})$ if $x$ is attainable from $\alpha$; and let $d(x, \alpha) = 0$ otherwise. The locus of points attainable form $\alpha$ is referred to as the attainable region of $\alpha$, $R(\alpha)$. The farthest Voronoi region of $\alpha$ can be defined in the ordinary way using $d(\cdot, \cdot)$.

$$\text{freg}(\alpha) = \{x \in \mathbb{R}^2 \mid d(x, \alpha) > d(x, \gamma), \forall \text{ arc } \gamma \in G', \gamma \neq \alpha\}.$$  

The farthest Voronoi diagram of $G'$, $FVD(G')$, is the collection of these regions and their boundaries. The same notation is also used to denote the graph structure of the diagram. A bisector between two arcs $\alpha, \gamma (s_{\alpha} \neq s_{\gamma})$ is meaningful within the common intersection of their attainable regions, where $b(\alpha, \gamma) = b(s_{\alpha}, s_{\gamma}) \cap (R(\alpha) \cap R(\gamma))$. Arcs corresponding to the same segment ($s_{\alpha} = s_{\gamma}$) do not normally have a bisector. For consecutive such arcs, we define an artificial bisector as the ray $r$ forming the common boundary of their attainable regions $r = R(\alpha) \cap R(\beta)$ (for more details see Section 3).

Lemma 1. Every point in $\text{freg}(\alpha)$ must be attainable from $\alpha$ ($\text{freg}(\alpha)$ is enclosed in $R(\alpha)$) and the graph structure of $FVD(G')$ is a (connected) tree.
Proof. For any point \( x \) in \( freg(\alpha) \), the unbounded portion of \( r(x, s_i) \) emanating from \( x \) must be entirely contained in it, thus, \( freg(\alpha) \) must be unbounded. Suppose, for the sake of contradiction, that the graph structure of \( FVD(G) \) is disconnected. Then there must be non-consecutive arcs \( \alpha_1, \alpha_2 \) of a segment \( s_i \) such that \( freg(\alpha_1) \) and \( freg(\alpha_2) \) are path-connected. Thus, for any point \( p \) in \( freg(\alpha_1) \) and point \( q \) in \( freg(\alpha_2) \) there is a path \( \pi \) connecting them such that \( \pi \) is entirely contained in \( freg(\alpha_1) \cup freg(\alpha_2) \). But then for any point \( x \) along \( \pi \) ray \( r(s_i, x) \) would be entirely contained in \( freg(\alpha_1) \cup freg(\alpha_2) \), i.e., arcs \( \alpha_1, \alpha_2 \) would be neighboring, a contradiction.

Remark 1. It is easy to see, that for any two arcs \( \alpha_1 \) and \( \alpha_2 \), corresponding to the same line segment \( s_i \), the attainable regions \( R(\alpha_1) \) and \( R(\alpha_2) \) are disjoint.

Remark 2. Two arc bisectors \( b(\alpha, \beta) \) and \( b(\beta, \gamma) \) may intersect at most once (as opposed to segment bisectors).

Proof. If some of the corresponding segments \( s_\alpha, s_\beta, s_\gamma \) are the same, then at least one of arc bisectors is artificial, and the statement follows from its definition.

Otherwise, if \( s_\alpha, s_\beta, s_\gamma \) are distinct segments, than the arc bisectors are portions of the segment bisectors \( (s_\alpha, s_\beta) \) and \( (s_\beta, s_\gamma) \). Suppose the segment bisectors intersect twice (otherwise, the statement follows trivially). This means that in \( FVD(s_\alpha, s_\beta, s_\gamma) \), the region \( freg(s_\beta) \) has two faces. One face corresponds to the arc \( \beta^* \), which encloses \( \beta \), and thus, \( R(\beta) \subset R(\beta^*) \). The other face is disjoint and it must be enclosed in the corresponding attainable region, by Lemma 1. This attainable region is disjoint from \( R(\beta^*) \) by Remark 1. One of the two intersection points of the curves \( b(s_\alpha, s_\beta) \) and \( b(s_\beta, s_\gamma) \) must be incident to this second face of \( freg(s_\beta) \). Thus, this intersection point is not in \( R(\beta) \), and by definition it is disjoint from both arc bisectors \( b(\alpha, \beta) \) and \( b(\beta, \gamma) \).

3 A Randomized Linear Construction

In this section we give an expected linear time algorithm to compute \( FVD(S) \), given \( Gmap(S) \). It is inspired from the two-phase randomized approach of [6] for points in convex position. Several non-trivial complications arise due to non-uniqueness of elements in the cyclic sequence of the Gaussian map. The tools developed in this section to handle non-uniqueness will also be used in the deterministic framework in Sec. 4.

Let \( \alpha_1, \alpha_2, \ldots, \alpha_h \) be a random permutation of arcs in \( Gmap(S) \), and let \( A_i = \{ \alpha_1, \alpha_2, \ldots, \alpha_i \}, 1 \leq i \leq h \), be the set of the first \( i \) arcs in this order. The arcs in \( A_h \) are referred to as original arcs. Throughout the algorithm, arcs in the Gaussian map may expand or shrink. As a result of shrinkage, new arcs, which do not belong in \( A_h \), may be created. At all times, arcs that are not new must entirely overlap their original version in \( A_h \).

The algorithm proceeds in two phases. Let \( t \) be the first index such that \( s_{\alpha_t} \neq s_{\alpha_{t+1}} \), where \( s_{\alpha_t} \) denotes the segment inducing \( \alpha_t \). Phase 1, decrementally computes \( G_i, t < i < h \), where \( G_h = Gmap(A_h) \), and \( G_i \) is obtained from \( G_{i+1} \).
by removing arc $\alpha_{i+1}$. At the time of removal, the two neighbors of $\alpha_{i+1}$ in $G_{i+1}$ are recorded as a tentative re-entry point for $\alpha_{i+1}$ in phase 2. Note, however, that both neighbors may correspond to the same segment, or the segment of one neighbor may coincide with $s_{n_{i+1}}$. At the end of phase 1, we obtain $G_{t+i+1}$. In phase 2, the algorithm computes incrementally $G'_i$ and FVD($G'_i$), for $t < i \leq h$, starting with FVD($G'_{t+1}$). $G'_{t+1} = G_{t+1}$. $G'_{i+1}$ is the Gaussian map obtained from $G'_i$ by inserting back arc $\alpha_{i+1}$. During the re-entry of $\alpha_{i+1}$, new arcs may be created. As a result, $G'_i \neq G_i$. Because of new arcs, the two recorded neighbors of $\alpha_{i+1}$ from phase 1 need not be neighbors of $\alpha_{i+1}$ in $G_{i+1}$. Despite the new arcs, we are able to show that the complexity of $G'_i$ remains $O(i)$ (see Lemma 3) and the expected time to insert $\alpha_{i+1}$ in FVD($G'_i$) remains $O(1)$ (see Lemma 4).

At the end of phase 2, we obtain FVD($G'_h$) = FVD($S$) ($G'_h = G_h$).

In the following we describe the two phases in detail.

**Phase 1 (deletion phase).** Consider one step of phase 1 that computes $G_i$ by removing arc $\alpha_{i+1}$ from $G_{i+1}$. For brevity, let $\beta = \alpha_{i+1}$ and let $\alpha, \gamma$ be its clockwise and counterclockwise neighbors respectively, in $G_{i+1}$. Let $s_{\alpha}, s_{\beta}, s_{\gamma}$ be the segments inducing $\alpha, \beta$ and $\gamma$, respectively. When removing $\beta$, the neighboring arcs $\alpha$ and $\gamma$ expand. In particular, either both $\alpha$ and $\gamma$ expand (see Fig. 2(a), for segments in the dual space) or one expands while the other shrinks (see Fig. 2(b)).

During the expansion, $\alpha$ and/or $\gamma$ may change from being a single-vertex arc to a segment arc; however, they always remain separated by unit vector $\nu(\alpha, \gamma)$, which indicates their bisector. Along with the deleted arc $\beta$, we record pair $\alpha, \gamma$ as a tentative re-entry point for $\beta$, and a label indicating the position of $\nu(\alpha, \gamma)$ with respect to $\beta$: “within”, “clockwise” or “counterclockwise”.

We now handle complications due to the non-uniqueness of site appearances in $G_i$. If $s_{\alpha} = s_{\gamma}$, we still need to keep $\alpha$ and $\gamma$ separately, although their union $\alpha\gamma$ is single maximal arc of $s_{\alpha}$. For this purpose, let $\nu(\alpha, \gamma) = \nu(s_{\beta})$, the unit vector of $s_{\beta}$. Note that $\beta$ must be a segment arc (see 15), thus, $\nu(s_{\beta})$ must be contained in $\beta$, and thus, it must be contained in the expanded $\alpha\gamma$. As a result, $\nu(s_{\beta})$ can be used to separate $\alpha$ and $\gamma$ in $G_i$. $\nu(s_{\beta})$ corresponds to an artificial bisector $b(\alpha, \gamma)$ as defined in Def. 1. If $s_{\alpha} = s_{\beta}$ then $\alpha$ expands to cover the entire $\beta$ and $\nu(\alpha, \gamma) = \nu(\beta, \gamma)$. The deleted arc $\beta$ stores $\nu(\alpha, \beta)$ so that it knows its separator from $\alpha$ when it will be inserted back in phase 2. Recall that $\nu(\alpha, \beta)$ is the unit vector of an already deleted segment arc. Respectively for $s_{\beta} = s_{\gamma}$.

Consider now the result of inserting back $\beta$ between the expanded arcs $\alpha, \gamma$ (see Figs. 2 and A.2 illustrating the corresponding segments in dual space). The result is $\alpha\beta\gamma$, or $\alpha\gamma\beta\gamma$, or $\alpha\beta\alpha'\gamma$, where $\alpha'$ and $\gamma'$ are new arcs, depending on the label stored along $\beta$ in phase 1. For the new arcs to come into existence, $\beta$ must be a segment arc (see e.g., Fig. 2(b) and Fig. A.2). Without loss of generality, in the following we only consider $\alpha\gamma'\beta\gamma$ and ignore the symmetric case $\alpha\beta\alpha'\gamma$.

**Phase 2 (insertion phase).** Let $G'_{t+1} = G_{t+1}$, $t \geq 1$, where $G_{t+1}$ is obtained at the end of phase 1. Every arc $\alpha_i, i \leq t$, corresponds to the same segment $s_t$, while $\alpha_{t+1}$ corresponds to a different segment $s_{t+1}$. Two consecutive arcs
Fig. 2: Deleting and re-inserting $\beta$ in sequence $\alpha\beta\gamma$ shown in dual space. (a) both $\alpha$ and $\gamma$ enlarge; (b) $\gamma$ enlarges and $\alpha$ shrinks. From left to right: the initial sequence $\alpha\beta\gamma$ (left); the result of removing $\beta$ (middle); the result of re-inserting $\beta$ (right).

$\alpha_i, \alpha_{i+1}, i < t$, are separated in $G_{t+1}$ by a unit vector $\nu(\alpha_i, \alpha_{i+1})$, corresponding to some arc $\beta$ whose deletion in phase 1 made them consecutive. This unit vector indicates an artificial bisector between the two arcs as defined above.

We first compute $\text{FVD}(G_{t+1})$ in time $O(t)$. In detail, let $\alpha$ denote the maximal arc in $G_{t+1}$ consisting of all arcs in $A_t$. Then $\text{FVD}(\{\alpha, \alpha_{t+1}\})$ consists of exactly two faces, $\text{freg}(\alpha)$ and $\text{freg}(\alpha_{t+1})$, separated by a single bisector curve $b$, and thus, it can be readily available in $O(1)$ time. In particular, curve $b = b(s_t, s_{t+1})$, if $s_t, s_{t+1}$ are disjoint; and $b \subset b(s_t, s_{t+1})$, otherwise, consisting of the two semi-curves emanating from the segment intersection point, such that $\alpha$ and $\alpha_{t+1}$ are at opposite sides of $b$. Given $\text{FVD}(\alpha, \alpha_{t+1})$, we can construct $\text{FVD}(G_{t+1})$ in $O(t)$ time, by simply inserting all artificial bisectors between the consecutive sub-arcs of $\alpha$ and splitting along them in $\text{freg}(\alpha)$, to derive $\text{freg}(\alpha_i), i \leq t$.

Then, we iteratively compute $\text{FVD}(G'_{t+1})$, for $t < i \leq h$. At one step of phase 2, $G'_{t+1}$ is obtained from $G'_t$ by inserting back arc $\alpha_{t+1}$ in $G'_t$, and $\text{FVD}(G'_{t+1})$ is obtained from $\text{FVD}(G'_t)$ by inserting $\text{freg}(\alpha_{t+1})$. Let $\beta = \alpha_{t+1}$, and let $\alpha, \gamma$ be the pair of neighbors stored with $\beta$ at phase 1.

Let us first assume that $s_\beta \neq s_\alpha, s_\gamma$. If $\alpha$ and $\gamma$ are still neighbors in $G'_t$, then the unbounded Voronoi edge of $b(\alpha, \gamma)$ in $\text{FVD}(G'_t)$ provides an entry point for $\text{freg}(\beta)$. Else a number of new arcs in $G'_t$ may need to be traced to identify an entry point, which is either a pair of consecutive new arcs $\delta', \epsilon'$ such that $\nu(\delta', \epsilon') \in \beta$, or a single (new) arc $\alpha$ that entirely contains $\beta$. In the former case $b(\delta', \epsilon')$ provides an entry point for $\text{freg}(\beta)$ as before. In the latter case, $\text{freg}(\beta)$ splits $\text{freg}(\sigma)$ into $\text{freg}(\sigma_1)$ and $\text{freg}(\sigma_2)$ in $\text{FVD}(G'_{t+1})$. The latter operation can be performed in time proportional to the complexity of $\text{freg}(\beta)$ and $\text{freg}(\sigma_1)$. Inserting $\text{freg}(\beta)$ in $\text{FVD}(G'_t)$, after an unbounded Voronoi edge enclosed within has been identified, can be performed in time proportional to the complexity of $\text{freg}(\beta)$, (see, e.g., [8]). Note that if $s_\alpha = s_\gamma$, $\nu(\alpha, \gamma)$ corresponds to an artificial bisector, which is already present in the diagram as an unbounded Voronoi edge.

If $s_\alpha = s_\beta$ (resp. $s_\gamma = s_\beta$), then we insert $\beta$, by simply splitting arc $\alpha$ in $G'_t$ into two parts separated by $\nu(\alpha, \beta)$, which is the unit vector that had been stored with $\beta$ in phase 1. Recall that $\nu(\alpha, \beta)$ must be the unit vector of some $\alpha_k, k > i + 1$, which will be inserted later. In addition, $\text{freg}(\alpha)$ in $\text{FVD}(G'_t)$ is split into $\text{freg}(\alpha)$ and $\text{freg}(\beta)$ in $\text{FVD}(G'_{t+1})$, by inserting the artificial bisector.
implied by \( \nu(\alpha, \beta) \) (see Def. 1). This can be performed in time proportional to the complexity of \( \text{freg}(\beta) \). Region \( \text{freg}(\gamma) \) remains intact. An example is shown in Fig. A.1 in the Appendix.

The following definition and lemma show that artificial bisectors have been correctly defined and their presence does not affect the resulting diagram.

**Definition 1.** Let \( \alpha, \gamma \) be consecutive arcs in \( G_i \) such that \( s_\alpha = s_\gamma \). Let \( \beta \) be the arc in \( G_j, j > i \), whose removal made \( \alpha \) and \( \gamma \) consecutive. Let the artificial bisector \( b(\alpha, \gamma) \) be a ray in the direction of \( \nu(s_\beta) \), emanating from the endpoint \( p_\alpha \) of \( s_\alpha \), which induces the point of \( \nu(s_\beta) \) along \( \alpha \gamma \) (see Fig. 3(d)).

**Lemma 2.** Let \( \alpha, \beta, \gamma \) be consecutive elements of \( G_{i+1} \), where \( s_\alpha = s_\gamma \) and \( \beta = \alpha_{i+1} \). In \( G_i \), arcs \( \alpha \) and \( \gamma \) expand over the deleted \( \beta \) and get delimited by \( \nu(\alpha, \gamma) = \nu(s_\beta) \). The artificial bisector \( b(\alpha, \gamma) \) in the direction of \( \nu(s_\beta) \) splits \( \text{freg}(\alpha \gamma) \) into \( \text{freg}(\alpha) \) and \( \text{freg}(\gamma) \) in \( \text{FVD}(G_{i+1}) \). When \( \beta \) is inserted back in \( G_{i+1} \), any remaining portion of this artificial bisector must be entirely contained in \( \text{freg}(\beta) \), and thus, it is deleted from \( \text{FVD}(G_{i+1}) \).

**Proof.** It is easy to see, using the dual wedges, that for any two occurrences of arcs of the same segment (that are not already consecutive), a different segment arc must lie between them (see also [15]). Thus, \( \beta \) must be a segment arc and \( \nu(s_\beta) \) must be contained in \( \beta \), i.e., it must also be contained in the expanded \( \alpha \gamma \) after the deletion of \( \beta \). As a result, \( \text{freg}(\alpha \gamma) \) must be unbounded in the direction of \( r \). Furthermore, by the definition of farthest Voronoi regions, \( p_\alpha \) cannot lie in \( \text{freg}(\alpha \gamma) \). Thus, ray \( r \) intersects the boundary of \( \text{freg}(\alpha \gamma) \) exactly once and splits \( \text{freg}(\alpha \gamma) \) in two parts such that each part is attainable form \( \alpha \) and \( \gamma \) respectively, where \( \alpha \) and \( \gamma \) are delimited by \( \nu(s_\beta) \). Note that \( r \) is exactly the boundary of the attainable regions \( R(\alpha) \) and \( R(\gamma) \).

Let \( H(\beta) \) denote the portion of the plane delimited by the bisector \( b(s_\alpha, s_\beta) \) that contains \( \beta \). By definition of a bisector, \( H(\beta) \) must contain at least one endpoint of \( s_\alpha \), thus, \( p_\alpha \in H(\beta) \), and hence, the entire ray \( r \) is contained in \( H(\beta) \). As a result, any portion of \( r \) remaining in \( \text{FVD}(G_{i+1}) \) must be deleted from \( \text{FVD}(G_{i+1}) \) when \( \beta \) is inserted, because it must be entirely contained in \( \text{freg}(\beta) \).

**Lemma 3.** The complexity of \( G_i \) and \( \text{FVD}(G_i) \) is \( O(i) \). For \( G_i \) this is \( \leq 2i \).
Proof. All elements of $G_i$ correspond to original arcs of $A_i$ that may have expanded (or later shrunk) by the deletion process, thus, their number is $i$. At each step of phase 2, we create at most one new arc by the insertion of an original arc $\alpha_i$. The resulting arc $\alpha_i$ in $G'_i$ must entirely contain the portion of its original counterpart in $A_i$; thus, it must be connected. Hence, the insertion of $\alpha_i$ can only split an existing arc of $G'_{i-1}$ in two (see, e.g., Fig. 2(b)), in which case a new arc is created, or it can trim two existing arcs and eliminate all arcs in-between them, in which case no new arc is created. Using induction, the number of elements in $G_i'$ is at most $2^i$.

By Lemma 1, the graph structure of $\text{FVD}(G_i')$ is a tree, having vertices of degree at least three, and exactly one face for every element of $G_i'$. Thus, the complexity of $\text{FVD}(G_i')$ is $O(i)$.

**Lemma 4.** The expected number of new arcs traced at any step of phase 2 is constant (at most 1).

Proof. By construction, $G_i'$ contains $G_i$ as a subsequence, having a number of new arcs in-between each pair of consecutive elements (original arcs) in $G_i$. By Lemma 3, the total number of new arcs in $G_i'$ is at most $i$. To insert $\alpha_{i+1}$, the pair $\alpha, \gamma$ of consecutive original arcs (that has been stored with $\beta = \alpha_{i+1}$) is picked, and new arcs which lie in-between $\alpha$ and $\gamma$ may need to be traced. Since every element of $A_{i+1}$ is equally likely to be the one inserted at step $i$, each pair of consecutive original arcs in $G_i'$ has probability $1/i$ to be considered during step $i$. Let $n_j$ be the number of new arcs in-between the $j$th pair of original arcs in $G_i'$; $\sum_{j=1}^i n_j \leq i$. The expected number of new arcs that are traced is then $\sum_{j=1}^i (n_j/i) \leq 1$.

**Theorem 1.** Once the farthest hull of a set $S$ of $n$ line segments in the plane is available, their farthest Voronoi diagram can be computed in additional $O(h)$ expected time, where $h$ is the number of faces in $\text{FVD}(S)$.

Proof. We use backwards analysis, going from $\text{FVD}(G_{i+1}')$ to $\text{FVD}(G_i')$. Recall that inserting $\text{freg}(\alpha_{i+1})$ in $G_i'$ takes time proportional to the complexity of its boundary, plus occasionally, the complexity of its neighboring Voronoi cell. In addition, it may require the tracing of a number of new arcs between the neighbors of $\alpha_{i+1}$ identified in phase 1. From Lemma 3, the latter time is expected constant. $A_h$ is a random permutation of the arcs in $G_h$, thus, the expected time complexity of inserting $\text{freg}(\alpha_{i+1})$ in $\text{FVD}(G_i')$ is equivalent to the expected complexity of a randomly selected face in $\text{FVD}(G_{i+1})$, plus the expected complexity of its immediate neighbor. Since $\text{FVD}(G_i')$ has size $O(i)$ and it consists of $O(i)$ faces, the expected complexity of a randomly selected region is constant. The same holds for one neighbor of the randomly selected region. Thus, the expected time spent to insert $\text{freg}(\alpha_{i+1})$ in $\text{FVD}(G_i')$ is constant. Since the total number of arcs is $h$, the claim follows.
4 A Deterministic Linear Divide-and-Conquer Algorithm

We now augment the linear framework of Aggarwal et al. [1] for points in convex position with techniques developed in Sec. 3, and derive a linear-time algorithm to compute the farthest Voronoi diagram, given Gmap(S).

Let $A_h$ be the set of arcs in Gmap(S), referred to as original arcs. Let $A$ be an arbitrary, but ordered, subset of $A_h$. Let $G = G(A)$ be the sequence of arcs derived from Gmap(S) by deleting the arcs in $A_h \setminus A$, following the deletion process described in Sec. 3. Note that the arcs in $G$ are possibly expanded versions of the arcs in $A$ forming a cyclic sequence. Let $G' = G'(A)$ be the Gaussian map, which is obtained from $G$ by following the deletion and re-insertion process of its arcs, in the order of $A$, as defined in phases 1 and 2 of Sec. 3. The deletion process ends at $G_t$, when $s(G_t) = 2$. $G'_t = G_t$. The Gaussian map $G'$ is the result of inserting back the arcs in $A_h \setminus A_t$ in reverse order, starting at $G'_t$. It consists of original arcs plus new arcs as obtained by the re-insertion process. The order of $A$ is determined by the flow of the algorithm. The sequence $G$ and the order of $A$ determine the corresponding Gaussian map $G'$.

In the following, we summarize the algorithm, given a sequence of original arcs $G$. The flow follows [12], which in turn follows [1]. Let $s(G)$ denote the set of segments corresponding to arcs in $G$.

1. Color each element of $G$ as red or blue, dividing $G$ into two sets, $R$ (red) and $B$ (blue), by applying consecutively the following two rules on $G$ (see Lemma 5 for details). This augments the respective procedure of [12].
   (a) For each 5-tuple $F$ of consecutive arcs $\{\alpha, \beta, \gamma, \delta, \epsilon\}$ in $G$, such that $|s(F)| \geq 2$, let $F'$ be the Gaussian map of $F$ for the order $A = (\gamma, \delta, \beta, \epsilon, \alpha)$. Color $\gamma$ red, if $freg(\gamma)$ in FVD($F'$) neighbors the regions of only two original arcs, in particular arcs $\beta$ and $\delta$, which are the left and the right neighbor of $\gamma$ in $F$. We use the specific order $A$ in order to minimize the number of new arcs surrounding $\gamma$ in $F'$.
   (b) After applying [12] for each maximal sequence of consecutive blue arcs, color red every other arc without including the last arc.

2. Delete from $G$ all elements of $R$ in any order, following the deletion process described in phase 1 of Sec. 3, producing the sequence $B$.

3. Recursively apply the algorithm to $B$, resulting in FVD($B'$).

4. Divide $R$ into two sets $C$ (crimson) and $Gr$ (garnet), such that: (1) the size of $C$ is at least a constant fraction of the size of $R$; and (2) for any two elements $\alpha$ and $\beta$ in $C$, their Voronoi regions in FVD($(B' \cup \{\alpha, \beta\})'$) are disjoint (see Lemma 3).

5. Insert the elements of $C$ one by one into FVD($B'$), resulting in FVD($(B' \cup C')$). Insertions are done as described in phase 2 of Sec. 3.

6. Recursively apply the algorithm to $Gr$, resulting in FVD($Gr'$).

7. Merge FVD($(B' \cup C')$) and FVD($Gr'$) into FVD($G'$) so that that necessary new arcs are only created.

The recursion ends when $|s(G)| \leq 2$. If $|s(G)| = 1$ then no diagram is generated; instead we return $G$ as a list of arcs to be used later in Step 5 and/or
Step 7. If \(|s(G)| = 2\), then \(G' = G\) and \(\text{FVD}(G')\) can be computed in \(O(|G|)\) time, similarly to Sec. 3. In particular, let us temporarily unite into maximal arcs all consecutive arcs of one segment, and let the result be denoted as \(G_u\), which can consist of 2 or 4 maximal arcs. We first compute \(\text{FVD}(G_u)\) in \(O(1)\) time. Note that when \(|G_u| = 4\), \(\text{FVD}(G_u) = \text{FVD}(s(G_u))\). Then we split its faces along artificial bisectors of individual arcs as described in Sec. 3 and derive \(\text{FVD}(G')\).

In the following, we give details, prove correctness and analyze the time complexity of our algorithm.

**Lemma 5.** (a) No two consecutive elements of \(G\) are in \(R\) and no three consecutive elements of \(G\) are in \(B\). (b) For any two consecutive elements \(\alpha, \beta\) in \(R\), \(\text{freg}(\alpha)\) and \(\text{freg}(\beta)\) in \(\text{FVD}((B' \cup \{\alpha, \beta\})')\) are not adjacent.

**Proof.** The proofs of items (a) and (b) follow the spirit of the proofs of [12, Lemmas 8 and 9] respectively. The possible appearance of new arcs in our setting complicates the process.

![Fig. 4](image)

(a) a 5-tuple of arcs in \(G\); scheme of (b) \(\text{FVD}(G'_2)\), (c) \(\text{FVD}(G'_3)\)

a. Suppose, for a contradiction, that two red arcs are consecutive, i.e., there is a 6-tuple \(\{\alpha, \beta, \gamma, \delta, \epsilon, \sigma\}\) in \(G\) such that \(\gamma\) and \(\delta\) are in \(R\). This may happen only if they both are colored red by rule 1a. Consider applying rule 1 to \(\gamma\), i.e., consider the 5-tuple \(F = \{\alpha, \beta, \gamma, \delta, \epsilon\}\), the order of \(A = (\gamma, \delta, \beta, \epsilon, \alpha)\) and \(\text{FVD}(F')\). The order of \(A\) implies, that there is no new arc between \(\gamma\) and \(\delta\). There might be a new arc \(\delta'\) between \(\beta\) and \(\gamma\) (only if \(s_\gamma \neq s_\delta\)). See Fig. 3. Consider the diagram \(\text{FVD}(G'_3)\). By Lemma 1, the regions of \(\delta'\) and \(\delta\) must be disjoint. Thus, \(\text{freg}(\gamma)\) must be bounded by a portion of bisector \(b(\beta, \gamma)\), followed by a portion of \(b(\gamma, \delta)\). Since \(\gamma\) satisfies rule 1, the boundary of \(\text{freg}(\gamma)\) remains the same in \(\text{FVD}(F')\), i.e., after inserting \(\epsilon\) and \(\alpha\). This as well means, that the regions of \(\delta, \gamma\) and \(\beta\) have a common Voronoi vertex (vertex \(w\) on Fig. 4), caused by the intersection of bisectors \(b(\beta, \gamma)\) and \(b(\gamma, \delta)\) (see Remark 2). By repeating the same argument for the 5-tuple \(D = \{\beta, \gamma, \delta, \epsilon, \sigma\}\), bisector \(b(\gamma, \delta)\) must first meet \(b(\delta, \epsilon)\) giving rise to a Voronoi vertex. Bisector \(b(\gamma, \delta)\) on both \(\text{FVD}(F')\) and \(\text{FVD}(D')\) are portions of the same curve (see Sec. 2.1). By construction, arc \(\delta\) in \(D'\) may be
shorter than in $F'$. However, in order for $\delta$ to satisfy rule 1, its region should neighbor the region of $\gamma$. Thus, both vertices induced by the intersection of $b(\gamma, \delta)$ with $b(\delta, \epsilon)$ and with $b(\beta, \gamma)$ must be incident to $f_{reg}(\delta)$, and this region has three original neighbors $\gamma, \beta$ and $\epsilon$ in the diagram $FVD(D')$, which is a contradiction. Note that the 2nd rule prevents three consecutive blue arcs.

b. Suppose there is a 5-tuple $F = \{\alpha, \beta, \gamma, \delta, \epsilon\}$ of the arcs in $G$, such that the arcs $\beta$ and $\delta$ are colored red, and the regions of $\beta$ and $\delta$ in $FVD((B' \cup \{\beta, \delta\})')$ are adjacent. Then the regions of $\beta$ and $\delta$ must be adjacent in $FVD(F')$, where $F'$ is obtained from $F$ as described in rule 1. Then $\gamma$ must satisfy rule 1 and must have been colored red during Step 1. A contradiction. If there are two blue arcs between the red arcs in question, same argument works for one of the two corresponding 5-tuples.

**Lemma 6.** Step 4 can be performed in $O(|G'|)$ time.

**Proof.** We perform Step 4 following the principles in [112], however, several differences are present in our setting. We apply the combinatorial lemma [1], to the tree structure $T$ of $FVD(B')$. To this goal, every leaf of $T$ should be associated with a neighborhood. For an arc $\beta \in R$, consider the leaf of $T$, which serves as an entry point for $\beta$ into $FVD(B')$, and can be found as described in phase 2 of the randomized algorithm. The neighborhood for such a leaf is the subtree of $T$ intersected by $f_{reg}(\beta)$ in $FVD((B' \cup \{\beta\})')$. However, unlike [112], $\beta$ may not have a leaf of $T$ as an entry point. This happens when $\beta$ splits an arc of $B'$ into two, in which case we insert the corresponding artificial bisector between these two arcs into $FVD(B')$, and consider this bisector as the entry point for $\beta$. Leaves of $T$, that are not associated with any arc of $R$, are contracted. Thus, we obtain a tree $T'$ of size $O(|B|)$ with $|R|$ leaves. By Lemma 5, neighborhoods of consecutive leaves are disjoint. By the combinatorial lemma, the set $C$ can be found in time $O(|B'| + |R|) = O(|G'|)$. Finding all entry points, and transforming the tree also take the same time $O(|G'|)$.

**Details of Step 7.** In Step 7 we merge $FVD((B' \cup C)')$ and $FVD(Gr')$ to obtain $FVD(G')$. Merging is performed by identifying a starting point, i.e., an unbounded Voronoi edge, on each merge curve, followed by tracing of this merge curve. In our setting $(B' \cup C)'$ and $Gr'$ may contain new arcs, whose number may be large. To maintain the linear time complexity of our algorithm, we must ensure that $|G'|$, and thus the size of $FVD(G')$, remain $O(|G|)$. To this goal, we merge only areas which contain original arcs.

In detail, consider the merged diagram as a collection of connected components, where each component belongs to one of the input diagrams. If such a component contains no original arc then it is of no use to the merged diagram and we can drop it. For such a component we simply ignore the bounding merge curves, thus, all its arcs are dropped from $G'$. (Recall that the entire component will be deleted later, and no portion of it can remain in the final diagram).
Because of the recursion ending condition, one of the diagrams (FVD((B' \cup C)'), FVD(Gr')) may be just a list of arcs (when the recursion ends with only one segment). In this case, we simply insert the arcs from this list into the other diagram. Note that not both FVD((B' \cup C)') and FVD(Gr') can be lists of arcs, because otherwise, s(G) \leq 2, and G should not have been subdivided during Step 1. Note as well that since all the arcs we are inserting correspond to the same line segment, their faces in the resulting FVD(G') are disjoint, thus, the insertion procedure takes time \(O(|G'|)\).

From now on, let \(m\) denote the size of \(G\).

**Lemma 7.** The number of arcs in \(G'\), which are neither in \((B' \cup C)\) nor in \(Gr'\), is \(O(m)\).

**Proof.** Let FVD(\(G'\)) be obtained by merging FVD((B' \cup C)') and FVD(Gr') using the modified merging process described above. Arcs in \(G'\), which are neither in \((B' \cup C)\) nor in \(Gr'\), appear as a result of splitting an arc in one Gmap with an arc of the other. These arcs must be separated by an unbounded edge of a merge curve. Since each merge curve has two unbounded edges, the number of arcs which are the subject of this lemma is at most 4 times the number of the merge curves.

Since the merge curves are disjoint, the number of merge curves equals the number of connected components of the input diagrams comprising FVD(\(G'\)). Since we only keep the components containing at least one original arc, the number of merge curves is at most \(m\). The claim follows.

**Lemma 8.** The total number of arcs in \(G'\) is \(O(m)\).

**Proof.** Let us denote the maximum number of arcs in \(G'\) as \(S(m)\). Suppose that FVD(B') and FVD(Gr') are already computed. To obtain FVD(\(G'\)), we first insert arcs from set \(C\) into FVD(B'), and then merge the obtained diagram with FVD(Gr'). By construction, \(|B| \leq q_1 m\) and \(|Gr| \leq q_2 m\), where \(q_1 + q_2 < 1\). Recall also, that \(|C| = O(m)\). Insertion of a single arc may cause at most one new arc, thus, the number of arcs in \((B' \cup C)'\) is at most \(S(q_1 m) + 2|C| = S(q_1 m) + O(m)\). By Lemma 7, merging the two diagrams will create \(O(m)\) new arcs. Thus, \(S(m) \leq S(q_1 m) + S(q_2 m) + O(m)\). Thus, \(S(m) = O(m)\). (Note that the equation and the coefficients \(q_1\) and \(q_2\) are the same as in the proof of Theorem 2). The claim follows.

**Theorem 2.** The farthest-segment Voronoi diagram can be computed in \(O(h)\) time, after its Gaussian map has been computed.

**Proof.** By construction, there exist constants \(q_1\) and \(q_2\) such that the size of \(B\) is at most \(q_1 m\), and the size of \(Gr\) is at most \(q_2 m\) and \(q_1 + q_2 < 1\). Thus, the time complexity of the described algorithm is \(T(m) \leq T(q_1 m) + T(q_2 m) + O(m)\), and \(T(m) = O(m)\), since \(q_1 + q_2 < 1\). Indeed, we drop out a constant fraction of elements (the set \(C\)) and recursively apply the algorithm to the sequences \(B\) and \(Gr\). Note that the recursive calls are done only on sets of original arcs. That is, the appearance of new arcs does not affect the dividing phase of the algorithm,
but only the conquer phase. The number of faces of the two diagrams remains linear in the number of their original arcs, despite the appearance of new arcs (see Lemma 7), thus, merging the two diagrams takes linear time, as well as choosing the set $C$ (see Lemma 4) and inserting all its elements into $\text{FVD}(B')$ (all the entry points are computed at Step 4).

5 Computing the Order-$(k+1)$ Subdivision Within an Order-$k$ Voronoi Region

Let $F$ be a face of the order-$k$ Voronoi region $k\text{-reg}(H)$, $H \subset S$, $|H| = k$, and let $S_F$ denote the set of segments in $S \setminus H$ inducing the boundary $\partial F$. Consider the order-1 Voronoi diagram, within $\partial F$, $\mathcal{V}_1(S_F)$. As shown recently [18], $\mathcal{V}_1(S_F)$ is a tree structure, and the sequence of its faces along $\partial F$ forms a Davenport-Schinzel sequence of order 3 (order 2 for non-intersecting segments). We can compute $\mathcal{V}_1(S_F)$ in time linear in the complexity of $\partial F$ by slightly adapting the algorithms in the previous sections. This directly implies that the order-$k$ Voronoi diagram of $S$ can be computed in $O(k^2 n + n \log n)$ time, improved from $O(k^2 n \log n)$, by iteratively computing higher-order diagrams, starting at $\mathcal{V}_1(S)$.

There is an additional complication to resolve, which does not appear for point sites, and which is not related to the farthest diagram. Namely, the boundaries of intermediate diagrams may be disconnected. Lemma 9 shows that the number of connected components of such diagrams is constant, at most 2, and this enables the linear construction. In the following we present the necessary adaptations to obtain a linear construction algorithm for this diagram.

Let $A$ be the sequence of appearances along $\partial F$ of segments in $S_F$. Each appearance of segment $s$ corresponds to an arc $\alpha$ of $\partial F$, as delimited by two consecutive Voronoi vertices. Arc $\alpha$ is visible from $a$, in the sense that for any point $x$ on $\alpha$, the segment realizing $d(s,x)$ does not intersect $\partial F$. In the following we do not differentiate between elements in $A$ and their arcs on $\partial F$, and we use the same notation for both.

Let us first define the bisector of two elements $\alpha, \gamma$ in $A$, $b(\alpha, \gamma)$, as an oriented semi-curve. If $s_\alpha \neq s_\gamma$, $b(\alpha, \gamma)$ is a branch of $b(s_\alpha, s_\gamma)$ starting at the point of minimum distance from the two segments, extending to infinity so that $\alpha$ and $\gamma$ lie at its opposite sides. If $s_\alpha = s_\gamma$, $b(\alpha, \gamma)$ can be defined artificially using a point $x$ on an arc $\beta$ between $\alpha$ and $\gamma$, which is visible from $s_\alpha$. In particular, $b(\alpha, \gamma)$ is the ray $r$ emanating from $s_\beta$, realizing $d(s_\beta, x)$, extending towards infinity away from $s_\beta$.

Let $A_i$, $1 \leq i \leq h$, where $h = |A|$, consist of $i$ elements in $A$ such that $A_i \subset A_{i+1}$. For the randomized algorithm, $A_h$ is a random permutation and $A_i$ consists of the first $i$ elements of $A_h$. For the deterministic algorithm, $A_i$ is an arbitrary collection of $i$ elements from $A$, which follow an arbitrary but fixed order as determined by the execution of the recursive algorithm. Both algorithms are based on the deletion and re-insertion process, described in Sec. 3.

Let $F_h = F$. $F_h$ can be regarded as the Voronoi region of $H$ in a Hausdorff Voronoi diagram (also called cluster Voronoi diagram [3]) of the family
of clusters $C_h = \{H, \{\alpha_i\}, \alpha_i \in A_h\}$, where $|H| = k$ and all other clusters are singletons. The deletion process computes $\partial F_i$, from $\partial F_{i+1}$ by deleting $\alpha_{i+1}$ and recomputing the Hausdorff region of $H$ for $C_i = \{H, \{\alpha_j\}, \alpha_j \in A_i\}$. In particular, let $\alpha, \beta, \gamma$ be arcs in counterclockwise order along $\partial F_{i+1}$, where $\beta = \alpha_{i+1}$. To delete $\beta$ we remove its incident Voronoi vertices and extend the incident Voronoi edges on $\partial F_{i+1}$ inducing $\alpha$ and $\gamma$, until they intersect, creating a new Voronoi vertex, or until it is determined that the incident Voronoi edges become unbounded. We remark that $F_i$ entirely encloses $F_{i+1}$ and it may become unbounded. The number of connected components of $\partial F_i$ can be shown to be at most two (see Lemma 9). If during the deletion of an arc, $\partial F_i$ breaks in two components, then during the re-insertion phase we are initially computing two different tree-structured diagrams as defined in the sequel.

The re-insertion process inserts back the elements of $A_i$ in reverse order and computes incrementally $F'_i$, $A'_i$, and the corresponding Voronoi diagram $V_1(A'_i)$. The insertion of element $\alpha_{i+1}$ in $F'_i$ may cause the appearance of a new element in $A'_{i+1}$ and a new arc in $F'_{i+1}$ similarly to Sections 3, 4.

![Fig. 5: Two line segments and (a) their 2-order Voronoi region; (b) the first opening of the boundary after removing two elements (c) the disconnection of the boundary into two connected components; (d) removing a segment does not reduce the number of connected components of the Hausdorff Voronoi region](image)

**Lemma 9.** $\partial F_i$ may consist of at most two connected components.

**Proof.** (By induction on $k$, the number of line segments in cluster $H$.)
If $H$ is a singleton $(k = 1)$, the claim follows from the geometric properties of parabolas and halfplanes induced by the same segment. Suppose the claim is true for a cluster $H$ of $k$ segments. Consider a cluster $H$ of $k + 1$ line segments, for which there exist a set of singleton-neighbors $N$ such that the boundary of the Hausdorff region of $H$ has at least three connected components. Each of the connected components separates $H$ from at least one element of $N$. Now, if we remove a line segment from $H$, the Hausdorff region of $H$ can only grow (see Figure 5c, d), thus no two connected components of its boundary may unite. No connected component may disappear, since the singletons in $N$, which are separated from $H$ by these connected components, remain. Thus, the number of connected components remains at least three for a cluster of size $k$, a contradiction.

Let $F_1^i$ (resp. $F_2^i$) denote the portion of the plane as delimited by the first (resp. second) component of $\partial F_i^i$, which contains $F_i^i$. If $\partial F_i^i$ consists of one component then $F_2^i = \emptyset$. Let $F_1^* \text{ or } F_2^*$ denote any of $F_i^1$ or $F_i^2$ (if not empty). $F_i^1$ and $F_i^2$ are independent and they clearly overlap. Let $A_i^*$ (where $* \text{ corresponds to } 1, 2$) denote the elements of $A_i^*$ in one component of $F_i^i$.

We can now define the Voronoi diagram of $A_i^*$ in $F_i^i$, denoted $V_i(A_i^*)$, similarly to Sec. 2.1. A point $x$ in $F_i^i$ is said to be attainable from $\alpha$ in $A_i^*$, if $d(s_\alpha, x)$ is realized as a segment passing through $\alpha$. Let $d(\alpha, x) = d(s_\alpha, x)$, if $x$ is attainable from $a$, and $d(\alpha, x) = \infty$, otherwise. $V_i(A_i^*)$ is a partitioning of $F_i^i$ into regions defined for $\alpha$ in $A_i^*$:

$$\text{reg}(\alpha) = \{ x \in F_i^i \mid d(x, \alpha) < d(x, \gamma), \forall \gamma \in A_i^*, \gamma \neq \alpha \}.$$  

The graph structure of $V_i(A_i^*)$ can be shown to be a tree, similarly to Lemma 1, where all unbounded bisectors, including the unbounded sides of $\partial F_i^i$, are assumed to be incident to a point at infinity.

Both the randomized and the deterministic algorithm can now be directly applied obtaining a linear construction algorithm for $V_i(F)$. There are two important arc removals and re-insertions. A simple one that converts $\partial F_i^i$ into an open curve; and a crucial one that splits the open $\partial F_i^i$ into two components (if any). The re-insertion of this arc, requires time linear in the size of $A_i^i$ and $A_i^j$. In the randomized algorithm this operation is performed once, thus, it does not affect the overall time complexity. In the deterministic algorithm, it is performed a number of times, as determined by the tree of the recursion. However, any arc can participate to such an operation at most once, thus, the overall time complexity is not affected. We conclude with the following theorem.

Note that $\partial F_i^i$ and $\partial F_i^j$ need not be explicitly computed. Any leaf bisector incident to $\partial F_i^i$ intersects $\partial F_i^i$ after its origin, and no two neighboring such bisectors can intersect before $\partial F_i^i$. Thus, it is possible to create the tree structure of $\partial F_i^i$ without its enclosing boundary.

**Theorem 3.** The order-$(k + 1)$ subdivision in a face $F$ of the order-$k$ Voronoi diagram can be computed in time proportional to complexity of $\partial F$.  

---

**Linear-Time Algorithms for Tree-Structured Voronoi Diagrams**

---

---

---
Proof. The deterministic algorithm of Sec. 4 is directly applicable as argued above. There is only one additional complication, that removing an arc may make $\partial F_i$ disconnected. This obviously may happen only during the divide phase. Suppose at some moment, during the divide phase (Step 2 of the algorithm), $\partial F_i$ became disconnected. The algorithm computes recursively the two diagrams of these two connected components separately. In the specific moment of the merge phase, when the two connected components are supposed to unite, during Step 7 the diagram is rebuilt from two (recursively computed) diagrams, corresponding to the two connected components. This is performed in time, linear in the number of arcs in the resulting diagram, that is, linear in the number of original arcs participating in the rebuilding procedure.

In the recursion tree of the deterministic algorithm, each path from root to leaf corresponds to a specific permutation $A_h$ of the set $A$. By Lemma 9 for any permutation of $A$ (and thus for any path in the recursion tree) there is at most one disconnection into two connected components. Each original arc corresponds to only one leaf of the recursion tree, and thus to only one path. Thus it participates in at most one of the expensive rebuilding operations. Therefore, in total during the whole execution of the algorithm, the rebuilding operations take time $O(h)$. All the other operations take $O(h)$ time in total, by exactly same argument as in Theorem 2.

6 Concluding remarks

We presented linear-time algorithms for the farthest-segment Voronoi diagram, after the sequence of its regions at infinity is known, and for computing the order-$(k+1)$ subdivision within an order-$k$ Voronoi region of line segments. Although tree-structured, these diagrams have disconnected regions and the cyclic sequence of their faces, along their boundary or infinity, forms a Davenport-Schinzel sequence of order 3. Our techniques augment the existing linear frameworks with the ability to handle non-uniqueness of Voronoi faces. The presentation focuses on line segments, however, our techniques are not specific to them. In future work, we plan to extend our techniques to the abstract counterparts of these diagrams, see e.g., [4,14]. In the abstract version, the corresponding cyclic sequence of Voronoi faces is in fact simpler, forming a Davenport-Schinzel sequence of order 2.

References

1. Aggarwal, A., Guibas, L., Saxe, J., Shor, P.: A linear-time algorithm for computing the Voronoi diagram of a convex polygon. Discrete Comput. Geom. 4, 591–604 (1989)
2. Aurenhammer, F., Drysdale, R., Krasser, H.: Farthest line segment Voronoi diagrams. Inform Process. Lett. 100, 220–225 (2006)
3. Aurenhammer, F., Klein, R., Lee, D.T.: Voronoi Diagrams and Delaunay Triangulations. World Scientific (2013)
4. Bohler, C., Cheilaris, P., Klein, R., Liu, C.H., Papadopoulo, E., Zavershynskyi, M.: On the complexity of higher order abstract Voronoi diagrams. In: Proc. 40th ICALP. LNCS, vol. 7965, pp. 208–219 (2013)
5. Bohler, C., Klein, R., Liu, C.H.: Forest-like abstract Voronoi diagrams in linear time. In: EuroCG (2014)
6. Chew, L.P.: Building Voronoi diagrams for convex polygons in linear expected time. Tech. rep., Dartmouth College, Hanover, USA (1990)
7. Chin, F., Snoeyink, J., Wang, C. A.: Finding the medial axis of a simple polygon in linear time. Discrete and Computational Geometry, 21(3), pp. 405–420 (1999)
8. de Berg, M., Schwarzkopf, O., van Kreveld, M., Overmars, M.: Computational Geometry: Algorithms and Applications. Springer-Verlag, 3rd edn. (2008)
9. Edelsbrunner, H., Maurer, H.A., Preparata, F.P., Rosenberg, A.L., Welzl, E., Wood, D.: Stabbing line segments. BIT 22(3), 274–281 (1982)
10. Kirkpatrick, D.: Efficient computation of continuous skeletons. In: Proc. 20th FOCS pp. 18–27 (1979)
11. Klein, R.: Concrete and Abstract Voronoi Diagrams, LNCS, vol. 400. Springer-Verlag (1989)
12. Klein, R., Lingas, A.: Hamiltonian abstract Voronoi diagrams in linear time. In: Proc. 9th ISAAC. pp. 11–19 (1994)
13. Lee, D.T., Drysdale, R.: Generalization of Voronoi diagrams in the plane. SIAM J. Comput. 10, 73–87 (1981)
14. Mehlhorn, K., Meiser, S., Rasch, R.: Furthest site abstract Voronoi diagrams. Int. J. Comput. Geom. Ap. 11(6), 583–616 (2001)
15. Papadopoulo, E., Dey, S.: On the farthest line segment Voronoi diagram. In: Proc. 23rd ISAAC. LNCS, vol. 7676, pp. 187–196 (2012). Full version IJCGA, vol. 23, no. 6, 443–459 (2013)
16. Papadopoulo, E.: Net-aware critical area extraction for opens in VLSI circuits via higher-order Voronoi diagrams. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 30(5), pp. 704–717 (2011).
17. Papadopoulo, E., Zavershynskyi, M.: On higher-order Voronoi diagrams of line segments. In: Proc. 23rd ISAAC. LNCS, vol. 7676, pp. 177–186 (2012)
18. Papadopoulo, E., Zavershynskyi, M.: The Higher-Order Voronoi Diagram of Line Segments. Algorithmica (to appear). Preliminary version arXiv:1405.3806 [cs.CG]
A Additional Figures

A.1 Farthest-Segment Voronoi Diagram With Artificial Bisector

![Diagram showing the addition of a face of segment $s_3$ within a face of segment $s_5$. (a) $FVD(G_i)$ and (b) $FVD(G_{i+1})$.](image)

Fig. A.1: The addition of a face of segment $s_3$ within a face of segment $s_5$. (a) $FVD(G_i)$ and (b) $FVD(G_{i+1})$.

A.2 Examples of Deletion and Re-Insertion of an Arc in Gmap

![Diagram showing examples of deletion and re-insertion of an arc in Gmap](image)

Fig. A.2: Deleting and re-inserting $\beta$ in sequence $\alpha\beta\gamma$. case 1(a),(b): both $\alpha$ and $\gamma$ enlarge; case 2(a)-(c): $\alpha$ enlarges and $\gamma$ shrinks. From left to right: the initial sequence $\alpha\beta\gamma$ (left); the result of removing $\beta$ (middle); the result of re-inserting $\beta$ (right).