Università della Svizzera italiana

Faculty of Informatics

USI Technical Report Series in Informatics

Randomized incremental construction of the Hausdorff Voronoi diagram of non-crossing clusters

Panagiotis Cheilaris[1], Elena Khramtcova[1], Evanthia Papadopoulou[1]

[1] Faculty of Informatics, Università della Svizzera italiana, Switzerland

Abstract

The Hausdorff Voronoi diagram of a set of clusters of points in the plane is a generalization of the classic Voronoi diagram, where distance between a point t and a cluster P is measured as the maximum distance, or equivalently the Hausdorff distance between t and P. The size of the diagram for non-crossing clusters is $O(n)$, where n is the total number of points in all clusters.

In this paper we present algorithms for efficiently computing the Hausdorff Voronoi diagram of non-crossing point clusters. Our algorithms are incremental and use linear space. If the clusters of points are inserted in a random order, then our best complexity algorithm takes expected time $O(n\log^2 n(\log\log n)^2)$ and worst-case space $O(n)$ to construct the diagram. We also provide a simpler-to-implement algorithm, based on a randomized hierarchical point-location data-structure (the Voronoi hierarchy) that takes expected time $O(n\log^3 n)$ and expected space $O(n)$. Previous (deterministic) algorithms in the Euclidean metric either require time $O(n\log^4 n)$ and $O(n\log^2 n)$ space, or have linear space complexity, but result in at least quadratic time-complexity bounds in the worst case. In order to achieve our bounds, we augment our data structures with the ability to efficiently handle non-standard characteristics of generalized Voronoi diagrams, such as sites of non-constant complexity, sites that are not enclosed in their Voronoi regions, and empty Voronoi regions. To the best of our knowledge these issues have not been addressed simultaneously by randomized incremental constructions for Voronoi diagrams.

The Hausdorff Voronoi diagram finds direct application in VLSI Critical Area Analysis for computing the probability of fault in a VLSI layout due to random manufacturing defects.

Report Info

Published
December 2012

Number
USI-INF-TR-2012-3

Institution
Faculty of Informatics
Università della Svizzera italiana
Lugano, Switzerland

Online Access
www.inf.usi.ch/techreports

1 Introduction

Given a set S of sites (for example, points) contained in some space, the *Voronoi region* of each site $s \in S$ is the geometric locus of points in the space that are closer to s than to any other site. In the classic Voronoi diagram, each site is a point, and closeness is measured according to the Euclidean distance. In this work, we investigate efficient algorithms for constructing the *Hausdorff Voronoi diagram*. The containing space is $\mathbb{R}^2$, each site is a cluster of points (i.e., a set of points), and closeness of a point $t \in \mathbb{R}^2$ to a cluster P of points is measured by the *farthest distance* $d_f(t,P) = \max_{p\in P} d(t,p)$, where d is the usual Euclidean distance between two points. This farthest distance equals the *Hausdorff distance* between t and cluster P, hence the name of the diagram.

The main motivation for investigating Hausdorff Voronoi diagrams comes from Very Large Scale Integration (VLSI) circuit design. The Hausdorff Voronoi diagram can be used to estimate efficiently the *critical area* of a chip design for various open faults [21, 22, 23]. Critical area is a measure reflecting the sensitivity of a

VLSI design to random defects during manufacturing.

1.1 Previous work

We denote by n the total number of points in all clusters. We assume that no two different clusters have a common point.

Hausdorff Voronoi diagrams were first considered in [11], under the name "cluster Voronoi diagram". For arbitrary clusters, the authors proved that the combinatorial complexity of the diagram is $O(n^2\alpha(n))$ and also provided an algorithm of the same time complexity, where $\alpha(n)$ is the inverse Ackermann function. They also proved that when the convex hulls of the clusters are disjoint, the combinatorial complexity of the diagram is linear. In [24] a condition weaker than disjointness of convex hulls was proved to be enough to imply linear combinatorial complexity of the Hausdorff Voronoi diagram. First, we need some definitions. We denote by $\operatorname{conv} P$ the convex hull of cluster P and by $\mathrm{CH}(P)$ the sequence of points of P on the boundary of the convex hull, say in counterclockwise order.

Definition 1.1. We say that two clusters P and Q are *non-crossing* if the convex hull of $P \cup Q$ admits at most two supporting segments with one endpoint in P and one endpoint in Q.

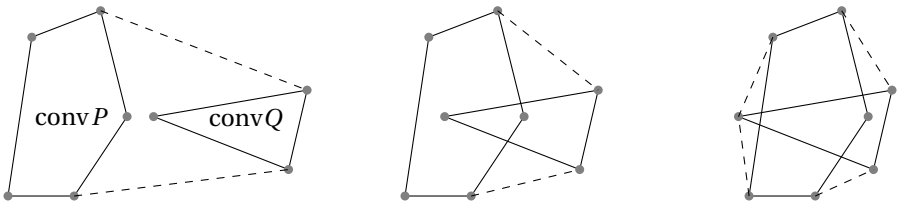

Figure 1: Disjoint convex hull, non-crossing, and crossing clusters

See figure 1, where supporting segments are shown with dashed lines. In [24], it was shown that the Hausdorff Voronoi diagram of a family of *non-crossing clusters* has linear complexity. For non-crossing clusters, the Hausdorff Voronoi diagram is an instance of abstract Voronoi diagrams [17] (see e.g., [1, 24]). Thus, it can be computed in expected $O(b\,n\log n)$ time using the randomized incremental framework of [18] for abstract Voronoi diagrams, where b is the time it takes to construct the bisector between two clusters (see also [8, 20]). We remark that if there are clusters of linear size, then $b = O(n)$ and therefore this approach is interesting for clusters of only small size.

In [24], it was also shown that the combinatorial complexity of the Hausdorff Voronoi diagram is $O(n+m)$, where m is number of supporting segments between pairs of crossing clusters, and this result is tight. In the worst case, m is $O(n^2)$. *Plane sweep* and *divide and conquer* algorithms for constructing the Hausdorff Voronoi diagram of arbitrary clusters were presented in [22, 24]. Both algorithms have a $K\log n$ term in their time complexity, where K is a parameter reflecting the number of pairs of clusters such that one is contained in a specially defined enclosing circle of the other, e.g., the minimum enclosing circle in [24]. However, K can be $\omega(n)$ (superlinear) even in the case of non-crossing clusters.

A recent advancement in the time complexity of constructing the Hausdorff Voronoi diagram of non-crossing clusters was achieved in [9]. The authors presented a parallel algorithm, which solves the problem in $O(p^{-1}n\log^4 n)$ time, where p is the number of available processors. Their parallel algorithm implies a divide and conquer sequential algorithm of time complexity $O(n\log^4 n)$ and space complexity $O(n\log^2 n)$.

Note that the Hausdorff Voronoi diagram is a min-max diagram: Every point t in the plane is assigned to the region of the *closest* cluster with respect to the *farthest* distance. The "dual", i.e., the max-min diagram, has also been studied [14, 2, 7]. In [7] each cluster is a simple polygon (or a polygonal site in general) and the authors provide an $O(n\log^3 n)$ time divide and conquer algorithm for constructing the max-min diagram of disjoint simple polygons, where n is the total description complexity of the polygons.

1.2 Our contribution

We propose a *randomized incremental* approach to build the Hausdorff Voronoi diagram of a family of non-crossing clusters. Our algorithm of best complexity takes $O(n\log^2 n(\log\log n)^2)$ expected time and $O(n)$ worst-case space to construct the diagram. We also provide a simpler-to-implement algorithm, based on

a randomized hierarchical point-location data-structure, called the Voronoi hierarchy, that takes $O(n\log^3 n)$ expected time and $O(n)$ expected space.

The incremental construction approach has been used in computing several Voronoi diagrams; see, e.g., [6]. In short, sites are inserted one by one and the Voronoi diagram is updated at every insertion. The incremental approach is inherently dynamic and it is appropriate for interactive applications where sites are inserted in succession by a user and the diagram is built incrementally. If coupled with a randomization of the insertion order of the sites, this *randomized incremental construction* has usually expected time complexity comparable to that of deterministic algorithms. However, the randomized incremental construction of abstract Voronoi diagrams [18] cannot be directly applied here as clusters have no restriction in size and thus the computation of a bisector can be costly resulting in a high complexity algorithm.

The main technical challenge when using the incremental approach is to find fast some point $t\in\mathbb{R}^2$ that is closer to the new cluster than to any of the already inserted clusters. In the Hausdorff Voronoi diagram, this is made difficult by the following facts: (a) the region of the new cluster might not contain any of the points of the clusters, (b) the region of the new cluster might be empty or make an existing region empty, and (c) sites have non-constant complexity and thus the computation of a bisector or answering an *in-circle test* require non-constant time.

To overcome these difficulties we exploit several interesting properties of Hausdorff Voronoi diagrams and we maintain a *dynamic point location* data structure for the existing diagram, which is also used to perform simple parametric search queries. Divide and conquer algorithms for the Hausdorff Voronoi diagram [9] and for the dual (max-min) diagram [7] also resort to some form of parametric search, when merging the solutions of two subproblems.

Our approach is modular in the sense that it can use any dynamic point location data structure such as [4, 3]. In particular, if we use the one from [4], then we get an algorithm for constructing the Hausdorff Voronoi diagram of non-crossing clusters in expected time $O(n\log^2 n(\log\log n)^2)$. Moreover, our algorithm uses only linear space in contrast with the algorithm of [9] which takes $O(n\log^4 n)$ time and uses $O(n\log^2 n)$ space. An alternative and simpler-to-implement approach can be derived by augmenting the *Voronoi hierarchy* of [15], based on the *Delaunay hierarchy* of [10]. We augment the Voronoi hierarchy with the ability to efficiently handle the aforementioned difficulties (a) to (c) and derive an additional more practical algorithm of expected time complexity $O(n\log^3 n)$ and expected $O(n)$ space. The augmentation of the Voronoi hierarchy may be of interest to incremental constructions of other types of generalized Voronoi diagrams.

2 Definitions and Structural properties

Given is a family $F=\{C_1,\ldots,C_m\}$ of non-crossing clusters of points. We assume that no two different clusters have a common point.

For $s\in C$, the open and closed *farthest region* of s in the *farthest Voronoi diagram* (FVD) of C are

$$\mathrm{freg}_C(s)=\{p\mid\forall s'\neq s\colon d(p,s)>d(p,s')\}\quad\text{and}\quad\overline{\mathrm{freg}}_C(s)=\{p\mid\forall s'\neq s\colon d(p,s)\geq d(p,s')\}.$$

Definition 2.1. The *farthest skeleton* of cluster C, denoted by $\mathrm{fskel}(C)$, is defined for $|C|>1$ as $\mathrm{fskel}(C)=\mathbb{R}^2\setminus\bigcup_{s\in C}\mathrm{freg}_C(s)$ and for $|C|=1$ as $\mathrm{fskel}(C)=C$.

For $|C|>1$, $\mathrm{fskel}(C)$ is a tree corresponding to the graph structure of $\mathrm{FVD}(C)$. Apart from its finite vertices, $\mathrm{fskel}(C)$ has also vertices at infinity each of which corresponds to two consecutive points on $\mathrm{CH}(C)$. See figure 2.

For $C\in F$, the open and closed *Hausdorff region* of C in the *Hausdorff Voronoi diagram* of F are:

$$\mathrm{hreg}_F(C)=\{p\mid\forall C'\neq C\colon d_{\mathrm{f}}(p,C)<d_{\mathrm{f}}(p,C')\}\quad\text{and}\quad\overline{\mathrm{hreg}}_F(C)=\{p\mid\forall C'\neq C\colon d_{\mathrm{f}}(p,C)\leq d_{\mathrm{f}}(p,C')\}.$$

For $s\in C$ and $C\in F$, the open and closed *Hausdorff region* of s in the *Hausdorff Voronoi diagram* of F are

$$\mathrm{hreg}_F(s)=\mathrm{hreg}_F(C)\cap\mathrm{freg}_C(s)\quad\text{and}\quad\overline{\mathrm{hreg}}_F(s)=\overline{\mathrm{hreg}}_F(C)\cap\overline{\mathrm{freg}}_C(s).$$

The Hausdorff Voronoi diagram is *monotone* in the following sense: Adding more clusters to a diagram can only shrink the region of an existing cluster or of a point in it.

The boundary of the Hausdorff region of a point $s\in C$ has structure as shown in figure 3a. It consists of two chains: (1) the *farthest boundary* of s, which belongs to $\mathrm{fskel}(C)$, i.e., $\mathrm{bd}\,\mathrm{hreg}_F(s)\cap\mathrm{bd}\,\mathrm{freg}_C(s)$, and is

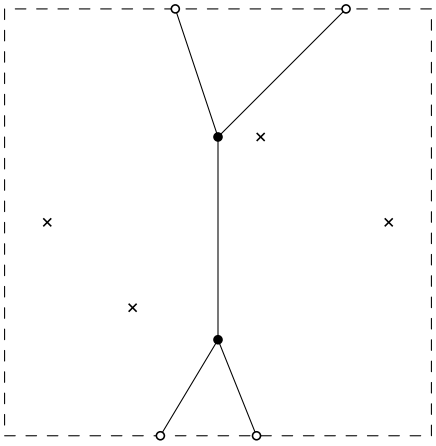

Figure 2: Farthest Voronoi diagram of four input points (denoted by crosses) and its vertices. Filled disk marks denote finite vertices. Unfilled circle marks denote infinite vertices and are on the boundary of a closed curve (shown dashed) which surrounds the finite vertices of the diagram.

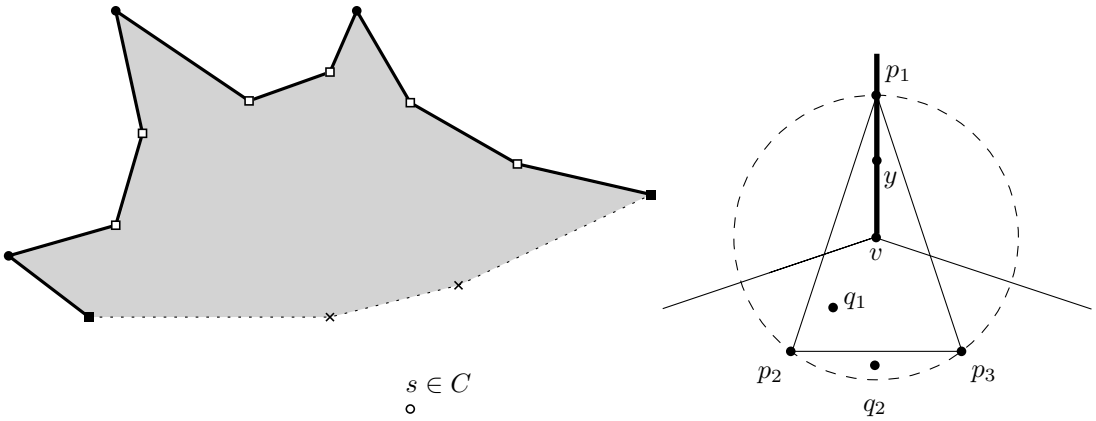

Figure 3: (a) The Hausdorff region of a point s (shown with an unfilled circle mark) of a cluster C. The Hausdorff boundary of s is shown with a heavy line and its farthest boundary with a dashed line. The pure vertices on the boundary of the region are shown with black disk marks. The C-mixed vertices are shown with black square marks. The other mixed vertices are shown with unfilled square marks. The vertices of FVD(C) on the farthest boundary of s are shown with crossmarks. (b) For cluster $P = \{p_1, p_2, p_3\}$, vertex v of fskel(P) is closer to cluster $Q = \{q_1, q_2\}$ than to cluster P. Only one of the three subtrees of fskel(P) that start at v, namely the one denoted with the thick line, might contain points which are closer to P than to Q.

internal to $\text{hreg}_F(C)$; (2) the *Hausdorff boundary* of s, i.e., $\text{bd}\,\text{hreg}_F(s) \cap \text{bd}\,\text{hreg}_F(C)$. Neither chain can be empty if $\text{hreg}_F(C) \neq \emptyset$ and $|C| > 1$.

There are three types of vertices on the boundary of $\text{hreg}_F(s)$ [24] (we use the general position assumption that no four points lie on the same circle). (1) Standard Voronoi vertices, which are equidistant from C and two other clusters, referred in this paper as *pure* vertices (using the terminology of [7]). These vertices appear on the Hausdorff boundary of s. (2) *Mixed* vertices, that are equidistant to three points of two clusters (C and another cluster). The mixed vertices that are equidistant to two points of C and one point of another cluster are called C-*mixed* vertices; there are exactly two of them on the boundary of $\text{hreg}_F(s)$ and they delimit both the farthest boundary of s and the Hausdorff boundary of s. (3) Vertices of $\text{fskel}(C)$ on the farthest boundary of s. See also figure 3a.

Line segment $\overline{ab}$ with $a \neq b$ is a *chord* of cluster C if $a, b \in \text{CH}(C)$. Consider cluster C and a point y on an edge of $\text{fskel}(C)$ induced by chord $\overline{cc^*}$. We assume that $\text{fskel}(C)$ is arbitrarily rooted and we denote this rooted tree with $T(C)$. Let $\overline{y_r y_k}$ be the edge of $\text{fskel}(C)$ containing y oriented so that y_r is the parent of y_k in $T(C)$. Let D_y be the closed disk with center y and whose boundary contains c and c^*. Then, $C \subset D_y$. Point y partitions $\text{fskel}(C)$ into two parts. Let $T(y)$ denote the part containing the descendants of y in the $\text{fskel}(C)$, i.e., the subtree of $T(C)$ that contains segment $\overline{y y_k}$, and let $T^{\sim}(y)$ denote the complement of $T(y)$. Chord $\overline{cc^*}$ partitions D_y in two parts D_y^{r} and D_y^{f}, where D_y^{r} is the *rear part*, containing the portion of $\text{conv}\,C$ inducing $T(y)$, and D_y^{f} is the *forward part*, enclosing the portion of $\text{conv}\,C$ inducing $T^{\sim}(y)$.

Definition 2.2. A cluster $Q \in F$ is called *limiting* with respect to chord $\overline{pp^*}$ of cluster P if there exists a closed disk D_y centered at a point y of $\text{fskel}(P)$ such that p and p^* are on the boundary of D_y and D_y contains both P and Q. Cluster Q is also called *limiting* with respect to point y. Cluster Q is called *forward limiting* if $Q \subset D_y^{\text{f}} \cup \text{conv}\,P$ or *rear limiting* if $Q \subset D_y^{\text{r}} \cup \text{conv}\,P$.

For example, in figure 3b, cluster Q is forward limiting with respect to chord $\overline{p_2 p_3}$ of cluster P, assuming that $T(P)$ is rooted somewhere on the bold portion of $\text{fskel}(P)$. The remainder of this section is a list of properties of the Hausdorff Voronoi diagram of a family of *non-crossing* clusters that can be directly derived from lemma 2 and properties 2 and 3 of [24].

Property 2.1. *If* $\text{hreg}_F(C) \neq \emptyset$, *then* $\text{hreg}_F(C) \cap \text{fskel}(C)$ *consists of exactly one non-empty connected component.*

Property 2.2. *Consider a point* v *of* $\text{fskel}(P)$ *such that* $v \notin \text{hreg}_F(P)$. *Let* Q *be a cluster, which is closer to* v *than* P. *Then, only one of the subtrees of* $\text{fskel}(P)$ *which start at* v, *might contain points which are closer to* P *than to* Q *(see figure 3b). In particular, if* Q *is forward (resp. rear) limiting with respect to* v *then the entire* $T(y)$ *(resp.* $T^{\sim}(y)$*) is closer to* Q *than to* P.

Property 2.3. *Let* $\overline{uv}$ *be an edge of* $\text{fskel}(P)$, *which is part of the bisector between* s *and* t *in* $\text{CH}(P)$. *If both* u *and* v *are closer to* Q *than to* P *then* $\text{hreg}_F(P)$ *cannot intersect* $\overline{uv}$.

A necessary and sufficient condition for an empty Voronoi region is given in [24]:

Property 2.4. *Region* $\text{hreg}_F(P) = \emptyset$ *if and only if either there is a cluster* $Q \subset \text{conv}\,P$, *or there exist a pair of clusters* $\{Q, R\}$ *such that* Q *is rear limiting and* R *is forward limiting with respect to the same point* v *of* $\text{fskel}(P)$. *Pair* $\{Q, R\}$ *is called a* killing pair *for* P.

3 General incremental construction algorithm

Given is a family F of m non-crossing point clusters in the plane. The following is a high-level description of an incremental construction of $\text{HVD}(F)$. We defer discussion on the point location data structures that are used and parametric search to section 5.

We fix a specific order of the m clusters, say $C_1, C_2, \ldots, C_m$. Later, we will prove that a random permutation implies expected efficient construction of the diagram. We denote by F_i the family consisting of the first i clusters according to the aforementioned order. The incremental approach [6] constructs successively the Hausdorff Voronoi diagram of families $F_1, F_2, \ldots, F_m$. Since $F = F_m$, at the end, we have the Hausdorff Voronoi diagram of F, $\text{HVD}(F)$.

For each new cluster C_i that is considered during the incremental construction, we build the farthest Voronoi diagram of C_i, $\text{FVD}(C_i)$, which has complexity linear in $|C_i|$ and can be built in $O(|C_i| \log |C_i|)$ time. We also build a (static) point location data structure for $\text{FVD}(C_i)$ within the same time and space [12, 27].

What remains is to describe the computation of HVD(F_{i+1}), given HVD(F_i) and FVD(C_{i+1}). We first try to find a point $t \in \mathbb{R}^2$ which is closer to C_{i+1} than to any cluster in F_i. In order to find this point fast, we rely on point location data structures for HVD(F_i) and possibly on parametric search. If such a point t exists, we update the existing diagram HVD(F_i) by starting at t and carefully growing the Hausdorff region of C_{i+1} around t, in order to get HVD(F_{i+1}).

We first describe how to find at least one point t, which is closer to C_{i+1} than to any cluster in F_i. Of course, it is also possible that there is no such point and thus we conclude that $\text{hreg}_{F_{i+1}}(C_{i+1}) = \emptyset$. If $C_{i+1} = \{c\}$, then $t = c$. Otherwise (when $|C_i| > 1$), searching for t just in fskel(C_{i+1}) (instead of in $\mathbb{R}^2$) is sufficient (see property 2.1). We propose to search for t as follows. For every vertex w of fskel(C_{i+1}):

- using point location in HVD(F_i), find the nearest to w cluster in F_i; call this cluster C^w;
- if $d_f(w, C_{i+1}) < d_f(w, C^w)$, then w is the point we are looking for, i.e., $t = w$, and we do not have to consider the remaining vertices of fskel(C_{i+1}). Else it may be possible to eliminate from further consideration an entire subtree of fskel(C_{i+1}) incident to v, according to property 2.2.

If there is no vertex w of fskel(C_{i+1}) with $d_f(w, C_{i+1}) < d_f(w, C^w)$, then no vertex of fskel(C_{i+1}) belongs to $\text{hreg}_{F_{i+1}}(C_{i+1})$. However, we have not yet excluded the possibility that $\text{hreg}_{F_{i+1}}(C_{i+1})$ intersects the interior of an edge of fskel(C_{i+1}) (it can be at most one edge, because of property 2.1). See figures 4, 5, and 6.

For any remaining edge uv of fskel(C_{i+1}), we proceed as follows. Remember that we have already computed C^u and C^v such that $u \in \text{hreg}_{F_i}(C^u)$ and $v \in \text{hreg}_{F_i}(C^v)$. It also holds that $d_f(u, C^u) < d_f(u, C_{i+1})$ and $d_f(v, C^v) < d_f(v, C_{i+1})$. If $C^u = C^v$, then, by property 2.3, $\text{hreg}_{F_{i+1}}(C_{i+1})$ can not intersect the interior of uv. If $C^u \neq C^v$, we do two additional point locations: (a) of u in FVD(C^v) and (b) of v in FVD(C^u). By property 2.3, if $d_f(u, C_{i+1}) \geq d_f(u, C^v)$ or $d_f(v, C_{i+1}) \geq d_f(v, C^u)$, then $\text{hreg}_{F_{i+1}}(C_{i+1})$ can not intersect the interior of uv. Therefore, $\text{hreg}_{F_{i+1}}(C_{i+1})$ might intersect the interior of uv only if $d_f(u, C_{i+1}) < d_f(u, C^v)$ and $d_f(v, C_{i+1}) < d_f(v, C^u)$. We summarize as follows.

Definition 3.1. Edge uv is a *candidate* edge if it satisfies the following predicate:

$$\begin{aligned} \text{cand}(uv) = u \in \text{hreg}_{F_i}(C^u) \wedge v \in \text{hreg}_{F_i}(C^v) \wedge C^u \neq C^v \wedge \\ d_f(u, C^u) < d_f(u, C_{i+1}) < d_f(u, C^v) \wedge \\ d_f(v, C^v) < d_f(v, C_{i+1}) < d_f(v, C^u) \end{aligned}$$

If there is no candidate edge, then we can conclude that $\text{hreg}_{F_{i+1}}(C_{i+1}) = \emptyset$. If there is a candidate edge, then we have to resort to parametric search to decide whether $\text{hreg}_{F_{i+1}}(C_{i+1})$ is empty or not and still find a point t in this region in the latter case. We defer this discussion to section 5.

In case we find a point $t \in \mathbb{R}^2$ such that $t \in \text{hreg}_{F_{i+1}}(C_{i+1})$, we grow the region of the new cluster around t and update diagram HVD(F_i) to get diagram HVD(F_{i+1}). The technical details are given in the following.

3.1 A refinement of the diagram

We have already mentioned that the combinatorial complexity of the Hausdorff Voronoi diagram of non-crossing clusters is linear in the total number of points in the clusters [22]. This diagram HVD(F) is stored as a planar subdivision in a doubly connected edge list [5]. It will be beneficial to have a *refinement* of HVD(F) in which each face f has either constant complexity or if the face has higher complexity, then the work performed when updating f (during the insertion of a new cluster) is proportional only to the deleted vertices and edges of the face f. To achieve this, we add some segments to the diagram.

We follow the *visibility decomposition* method of [24]. For every $c \in C$, define the set V_c to contain all vertices of the diagram on the Hausdorff boundary of c. For every point $x \in V_c$, we add to the diagram the segment resulting from intersecting $\text{hreg}_F(c)$ with the ray $\overrightarrow{cx}$, starting at c and having direction from c to x. Such additional segments are shown in figure 7. We remark that these additional segments might create new vertices on the farthest boundary of c.

It is not difficult to prove that the refined HVD(F) diagram has also combinatorial complexity linear in $\sum_{C \in F} |C|$, relying on methods of [22]. We denote this refinement of HVD(F) by HVD*(F).

A face of the refined diagram HVD*(F) which belongs to $\text{hreg}_F(c)$ with $c \in C$ consists of a segment which is part of the Hausdorff boundary of c, one or two visibility decomposition segments and a linear chain which is a portion of the farthest boundary of c, like in figure 8. Observe that each face of HVD*(F) is convex.

We call this chain the fskel-*chain* of the given face. We remark that the size of this chain need not be bounded by a constant, and that it can even be of linear complexity.

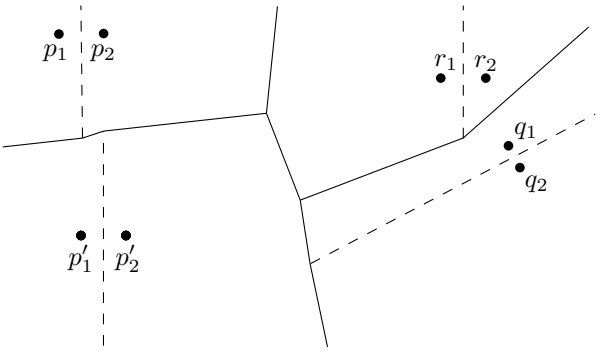

Figure 4: Hausdorff diagram of four two-point clusters $P = \{p_1, p_2\}$, $P' = \{p'_1, p'_2\}$, $Q = \{q_1, q_2\}$, $R = \{r_1, r_2\}$.

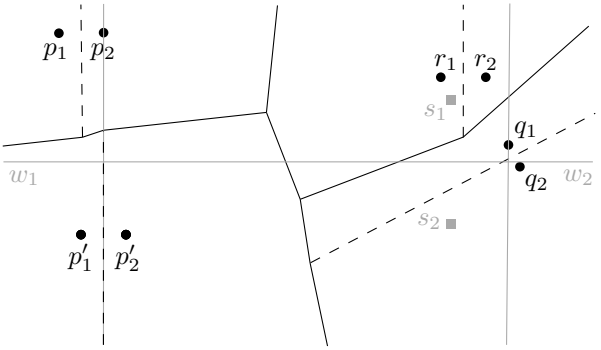

Figure 5: Insertion of a new cluster $S = \{s_1, s_2\}$, with w_1 and w_2 being infinite vertices of fskel(S). Edge $w_1 w_2$ is a candidate edge, because P is closer to w_1 than S and Q is closer to w_2 than S. Minimum "disks" with "centers" at w_1 and w_2 that contain P and Q, respectively, are shown (these "disks" are halfplanes).

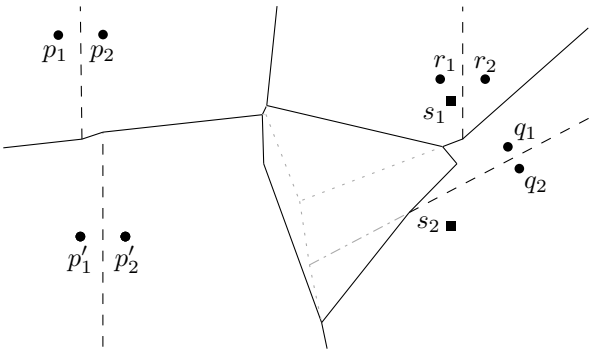

Figure 6: The diagram after the insertion of S. The deleted part is shown in dotted gray lines. We have not refined the region of S by its farthest Voronoi diagram, in order to have a cleaner figure.

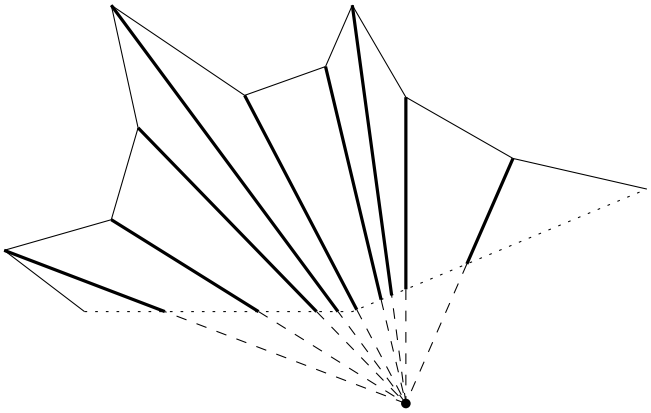

Figure 7: Visibility decomposition of a Hausdorff region of a point

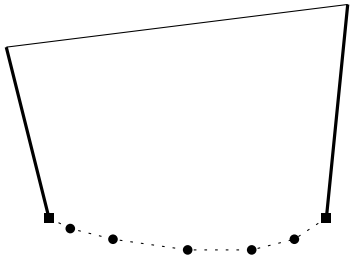

Figure 8: A face of the refined Hausdorff Voronoi diagram

3.2 Computing vertices of fskel(C_{i+1}) closest to C_{i+1}

To get back to our updating process, first we compute which vertices of fskel(C_{i+1}) are in hreg$_{F_{i+1}}(C_{i+1})$. When C_{i+1} consists of a single point, the answer is trivially this single point, so from now on we are concerned with the case $|C_{i+1}| > 1$. For every vertex $v \in$ fskel(C_{i+1}), we do a point location of v in HVD$^*(F_i)$ and we check whether v is closer to C_{i+1} or another cluster $C \in F_i$.

Therefore, we have computed a subset of vertices of fskel(C_{i+1}) which are closer to C_{i+1} than any other cluster $C \in F_i$. Remember that we also have a point $t \in$ hreg$_{F_{i+1}}(C_{i+1})$. If t is not a vertex of fskel(C_{i+1}), we also add it to this computed set. By property 2.1, these vertices are contained in a subtree of fskel(C_{i+1}).

3.3 Edges of fskel(C_{i+1}) through which bd(hreg$_{F_{i+1}}(C_{i+1})$) passes

Moreover, for $|C_{i+1}| > 1$, we can compute a subset of edges of fskel(C_{i+1}) with the following property: each edge vu in the subset has v closer to C_{i+1} than some $C \in F_i$ and u not closer to C_{i+1} than some $C \in F_i$. We refer to these edges as *switch edges*.

We can also easily compute the cyclic order in which the switch edges appear along a traversal of the boundary, say in a counter-clockwise fashion (i.e., a traversal that leaves hreg$_{F_{i+1}}(C_{i+1})$ to the left of its boundary) as follows: We follow successively the *half-edges* of the doubly connected edge list representation of fskel(C_{i+1}) and record the order in which we encounter the switch edges.

Say this cyclic order is $e_0, e_1, \ldots, e_{\kappa-1}$. Then, bd(hreg$_{F_{i+1}}(C_{i+1})$) consists of κ curves, each one connecting two points in consecutive edges in the above cyclic order. These points correspond to mixed vertices in HVD(F_{i+1}) (see [22]).

3.4 Computing the intersection of e_0 with bd(hreg$_{F_{i+1}}(C_{i+1})$)

We next try to find a point on bd(hreg$_{F_{i+1}}(C_{i+1})$). We will find such a point p_0 on switch edge e_0. This p_0 is a mixed vertex of HVD(F_{i+1}).

For switch edge $e_0 = uv$, with $u \in$ hreg$_{F_{i+1}}(C_{i+1})$ and $v \notin$ hreg$_{F_{i+1}}(C_{i+1})$, the method we use is as follows: We trace the movement of a point y on e_0 starting from u and going towards v, until y reaches an element of HVD$^*(F_i)$ which intersects bd(hreg$_{F_{i+1}}(C_{i+1})$). In that case, then we can easily compute point p_0 by taking the

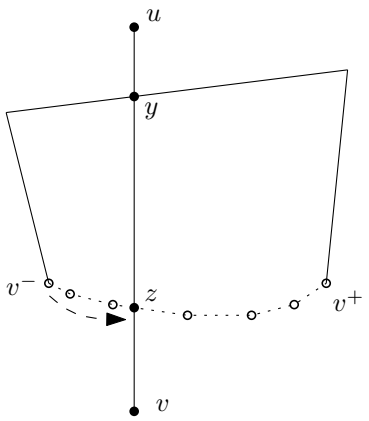

Figure 9: Tracing in order to find a point on the boundary

intersection of yv with a bisector between some specific point of C_{i+1} and another specific point of a cluster in F_i.

This tracing is done by following elements of $\mathrm{HVD}^*(F_i)$ along the switch edge e_0. In order to bound the complexity of the algorithm, it is crucial to only visit a number of elements of $\mathrm{HVD}^*(F_i)$ on the order of the number of elements of $\mathrm{HVD}^*(F_i)$ that have to be updated (in order to get $\mathrm{HVD}^*(F_{i+1})$). This is complicated by the fact that a face f can have a fskel-chain of non-constant size. When we trace on e_0 along such a face f, we have to be careful so that the number of neighboring elements of f that we visit are on the order of the number of updated neighboring elements of f (because of C_{i+1} insertion).

In particular, assume that $y \in \mathrm{cl}\, f$, and $v \notin \mathrm{cl}\, f$. Moreover, $yv \cap f \neq \emptyset$. Since f is convex, there is a single point z of yv which is different from y and lies on the boundary of f (see figure 9). We remark that $z \in \mathrm{hreg}_{F_{i+1}}(C_{i+1})$. We intend to find this point z, without traversing too many edges of the face f that will not be updated.

We first check if yv intersects any of the edges not on the fskel-chain of f (there are at most three of these edges). If there is such an intersection, we found our point z. Otherwise, yv must intersect with the fskel-chain of f. In that case, consider the two extreme vertices v^- and v^+ of the fskel-chain (see figure 9). Since $z \in \mathrm{hreg}_{F_{i+1}}(C_{i+1})$, because of property 2.1, at least one of v^- and v^+ is also in $\mathrm{hreg}_{F_{i+1}}(C_{i+1})$. By comparing distances of v^-, v^+ from C_{i+1} and the closest clusters in F_i, we find (at least) one of v^- and v^+ that is closest to C_{i+1} than any cluster in F_i. Say, without loss of generality, that v^- is closest to C_{i+1}. Then, by property 2.1, the part of the fskel-chain from v^- to z is closest to C_{i+1} and has to be updated. Therefore, we can search for the intersection point z by following the fskel-chain, starting from v^- (see figure 9).

We note that when $C_{i+1} = \{c\}$, we can compute a point in $\mathrm{bd}(\mathrm{hreg}_{F_{i+1}}(C_{i+1}))$ by shooting any ray from c, for example, a vertical ray from c upwards, and tracing along this vertical ray in $\mathrm{HVD}^*(F_i)$, starting from c, until finding a point in $\mathrm{bd}(\mathrm{hreg}_{F_{i+1}}(C_{i+1}))$, as above.

3.5 Computing the boundary of $\mathrm{hreg}_{F_{i+1}}(C_{i+1})$

From section 3.4, we have a point p_0 on the boundary of $\mathrm{hreg}_{F_{i+1}}(C_{i+1})$. We also have the cyclic order of switch edges $e_0, e_1, \ldots, e_{\kappa-1}$, where point $p_0 \in e_0$.

Then, $\mathrm{bd}(\mathrm{hreg}_{F_{i+1}}(C_{i+1}))$ consists of κ polygonal chain curves, each one connecting two points in consecutive edges in the above cyclic order. Let $p_0, p_1, \ldots, p_{\kappa-1}$, be these points. We start from p_0 and we trace $\mathrm{bd}(\mathrm{hreg}_{F_{i+1}}(C_{i+1}))$, so that $\mathrm{hreg}_{F_{i+1}}(C_{i+1})$ lies to the left of the boundary, until we reach (the yet unknown) p_1 on e_1. This tracing is similar to the tracing in section 3.4 and it is done by following parts of bisectors in of $\mathrm{HVD}^*(F_i)$.

In particular, the boundary of $\mathrm{hreg}_{F_{i+1}}(C_{i+1})$ at p_0 consists of a bisector between a point of C_{i+1} and a point in another cluster that can be determined by the face f where p_0 belongs in $\mathrm{HVD}^*(F_i)$. We consider the directed halfline h along this bisector starting from p_0 and having $\mathrm{hreg}_{F_{i+1}}(C_{i+1})$ to its left side. If the intersection point of h with e_1 is contained in $\mathrm{cl}\, f$, then we have found p_1. Otherwise, we have to continue with the next face of $\mathrm{HVD}^*(F_i)$ along halfline h. Again, in order to bound the complexity of the algorithm, it is crucial to only visit a number of elements of $\mathrm{HVD}^*(F_i)$ on the order of the number of elements that have to be updated in $\mathrm{HVD}^*(F_i)$ (in order to get $\mathrm{HVD}^*(F_{i+1})$). This means that if we are currently in a face of $\mathrm{HVD}^*(F_i)$,

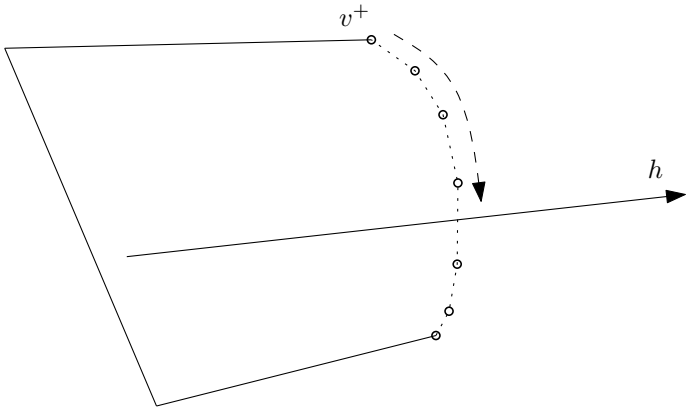

Figure 10: Tracing on the boundary

we find the intersection of h with the boundary of the face by first checking the (at most three) edges not on the fskel-chain. If these checks are unsuccessful and only then, we start looking for the intersection point on the fskel-chain, starting from the extreme vertex v^+, which is to the left side of h and it is guaranteed to be in $\overline{\text{hreg}}_{F_{i+1}}(C_{i+1})$ (again property 2.1 is relevant); see figure 10.

As we move along different faces, we might have to use a different halfline h, having the direction of the relevant bisector in the current face of HVD$^*(F_i)$.

After tracing the portion of bd(hreg$_{F_{i+1}}(C_{i+1})$) between p_0 and p_1, we reach point p_1. We continue and trace the remaining $\kappa - 1$ portions until we get back to p_0, having traced completely bd(hreg$_{F_{i+1}}(C_{i+1})$).

3.6 Updating the diagram

Now that we have computed bd(hreg$_{F_{i+1}}(C_{i+1})$), the boundary of the region of the new cluster, we superimpose it on the existing diagram of HVD$^*(F_i)$. Some edges of HVD$^*(F_i)$ will have to be subdivided by some vertices on this boundary. Each such subdivision operation corresponds to a deletion operation of an edge and two insertion operations (of the two edges into which the original edge is subdivided).

Then, we delete everything from HVD$^*(F_i)$ that is in the interior of the new cluster's region hreg$_{F_{i+1}}(C_{i+1})$. This is done by choosing an element of HVD$^*(F_i)$ inside hreg$_{F_{i+1}}(C_{i+1})$ and deleting everything in a graph traversal of the diagram from there, until we reach elements of bd(hreg$_{F_{i+1}}(C_{i+1})$).

Then, we add hreg$_{F_{i+1}}(C_{i+1}) \cap$ fskel(C_{i+1}) to the diagram, so that the Hausdorff region of C_{i+1} is partitioned by the Hausdorff regions of its constituent points. Finally, we also add to the diagram the visibility decomposition segments that refine the Hausdorff regions of each point in order to get the full HVD$^*(F_{i+1})$ (see section 3.1).

It is crucial that during the above process we also update the point location data structure, so that at the end we have both a valid diagram for HVD$^*(F_{i+1})$ and a point location data structure on it. We discuss relevant details in section 5.

We note again that, with some care, the amount of elements of the arrangement of HVD$^*(F_i)$ that we visit is of the order of the number of changes that have to be done in HVD$^*(F_i)$, in order to get HVD$^*(F_{i+1})$.

4 Complexity analysis

It is evident that the running time of our algorithm depends on the number of update operations (insertions and deletions) during the incremental construction of the diagram. In the worst case, this number can be $\Omega(n^2)$. However, when clusters are inserted in a uniformly random order, the expected number is linear. Our bound is independent of the individual cluster sizes. The possible non-uniformity of cluster size requires some care when applying the Clarkson-Shor technique [8, 28] in order to prove the bound. The technical details of the proof are in section 4.1. We use this result to bound the overall complexity of the algorithm in section 4.2.

4.1 Expected number of operations

We will associate the operations with features of the diagrams. Each feature (vertex, edge, face) of the diagram that appears during the incremental algorithm has been inserted by an operation. If a feature is deleted, then it can not be inserted again in the future, because of the monotonicity of the Hausdorff Voronoi diagram. As a result, the number of deletion operations is bounded by the number of insertion operations.

So, we intend to prove that the expected number of features that appear during the construction of the diagram is $O(n)$. To that end, we can ignore features associated only with the farthest Voronoi diagrams of each cluster, because even their total worst case combinatorial complexity is $O(n)$.

4.1.1 Configurations

We define some notions that are related with some features of the diagram. We follow the terminology of [22, 28].

Definition 4.1. A *configuration* is a triple of points (p,q,r) such that p,q,r lie on the boundary of a disk D and q is contained in the interior of the counterclockwise arc from p to r. We call D the *disk of the configuration*, its center the *center* of the configuration, and the counterclockwise arc pr the *arc of the configuration*.

Given is a family F of clusters.

Definition 4.2. A configuration is *pure* if its three points belong to *three* different clusters of F and no other point of these three clusters belongs to the arc of the configuration.

Definition 4.3. A configuration is *mixed* if its three points belong to *two* different clusters of F and no other point of these two clusters belongs to the arc of the configuration.

From now on, configurations of our interest will be either pure or mixed. Therefore, each configuration is either associated with three (a pure one) or two (a mixed one) clusters.

Definition 4.4. A cluster C is in *conflict with a configuration* if (a) C does not contain any of the points in the configuration and (b) C is contained in the union of the interior of the disk of the configuration and the arc of the configuration.

Remark 4.1. A cluster can be in conflict with a configuration only if it is different from the clusters associated with the configuration.

Definition 4.5. The *weight* of a configuration is the number of clusters in conflict with it.

Lemma 4.1. *The number of zero weight configurations of F is of the same order as the combinatorial complexity of the Hausdorff Voronoi diagram of F.*

Proof. Each zero weight configuration is associated with a vertex of the Hausdorff Voronoi diagram. Indeed the center of this configuration is at the vertex and the disk of the configuration contains the clusters associated with the configuration. Consider a vertex v of the Hausdorff Voronoi diagram. The degree of v in the arrangement equals the number of configurations with center v plus the number of some features that are associated just with farthest Voronoi diagrams (that we have claimed before that we can ignore). As a result, zero weight configurations estimate well the combinatorial complexity of the Hausdorff Voronoi diagram. □

4.1.2 Configurations of weight at most k

Let $K_0^{\text{pure}}(F)$, $K_k^{\text{pure}}(F)$, $K_{\leq k}^{\text{pure}}(F)$ denote the sets of pure configurations of zero weight, weight equal to k, and weight at most k, of a family F of non-crossing clusters, respectively. Let $N_0^{\text{pure}}(F)$, $N_k^{\text{pure}}(F)$, $N_{\leq k}^{\text{pure}}(F)$ denote the cardinality of the aforementioned sets, respectively. Define analogously the sets of mixed configurations $K_0^{\text{mix}}(F)$, $K_k^{\text{mix}}(F)$, $K_{\leq k}^{\text{mix}}(F)$ and their cardinalities $N_0^{\text{mix}}(F)$, $N_k^{\text{mix}}(F)$, $N_{\leq k}^{\text{mix}}(F)$, respectively.

By results of [22] and the discussion in section 4.1.1, we know that both $N_0^{\text{pure}}(F)$ and $N_0^{\text{mix}}(F)$ are

$$O\Big(\sum_{C\in F}|C|\Big)=O(n).$$

We are ready to estimate an upper bound on configurations of weight at most k. We use the Clarkson-Shor technique [8] and in particular the following version that we adapt from [28].

Theorem 4.1.

$$N^{\diamond}_{\leq k}(F) \leq \frac{\mathbb{E}[N^{\diamond}_0(R_p)]}{p^d(1-pk)},$$

for $k > 0$ and $p \in (0, k^{-1})$, where $\diamond \in \{\text{pure}, \text{mix}\}$, R_p is a random sample of clusters from family F, where each cluster is chosen independently with probability p, and d is the number of clusters associated with each configuration ($d = 3$ for pure configurations and $d = 2$ for mixed configurations).

If $\sum_{C \in F} |C| = n$, then $\mathbb{E}[N^{\text{pure}}_0(R_p)] = O(pn)$ and $\mathbb{E}[N^{\text{mix}}_0(R_p)] = O(pn)$. By choosing the appropriate values for p to minimize the upper bound, we obtain:

$$N^{\text{pure}}_{\leq k}(F) \leq c^{\text{pure}} \cdot nk^2 \text{ and } N^{\text{mix}}_{\leq k}(F) =\leq c^{\text{mix}} \cdot nk, \tag{1}$$

for $k > 0$ and some constants c^{pure} and c^{mix}.

4.1.3 Appearance of a feature

Consider a configuration c of weight k in family F with m clusters. Assume the Hausdorff Voronoi diagram of F is constructed with the incremental algorithm and the clusters are inserted according to permutation π. Our analysis has some similarities with that of [13]. The feature corresponding to c appears at some stage of the incremental algorithm if and only if the clusters associated with c occur in π *before* the k clusters that conflict with configuration c. This event happens with probability

$$\Pr[\text{pure } c \text{ feature appears}] = \frac{3!k!}{(k+3)!} = \frac{6}{(k+1)(k+2)(k+3)}$$

for pure configurations and with probability

$$\Pr[\text{mixed } c \text{ feature appears}] = \frac{2!k!}{(k+2)!} = \frac{2}{(k+1)(k+2)}$$

for mixed configurations.

The expected number of appearances of features corresponding to a pure configuration is therefore:

$$\begin{aligned}
&\sum_{k=0}^{m-3} \sum_{c \in K^{\text{pure}}_k(F)} \Pr[\text{pure } c \text{ feature appears}] = \sum_{k=0}^{m-3} \sum_{c \in K^{\text{pure}}_k(F)} \frac{6}{(k+1)(k+2)(k+3)} \\
&= 6 \sum_{k=0}^{m-3} \frac{N^{\text{pure}}_k(F)}{(k+1)(k+2)(k+3)} \\
&= N^{\text{pure}}_0(F) + 6 \sum_{k=1}^{m-3} \frac{N^{\text{pure}}_{\leq k}(F) - N^{\text{pure}}_{\leq k-1}(F)}{(k+1)(k+2)(k+3)} \\
&= \frac{3}{4} N^{\text{pure}}_0(F) + 18 \sum_{k=1}^{m-4} \frac{N^{\text{pure}}_{\leq k}(F)}{(k+1)(k+2)(k+3)(k+4)} + \frac{N^{\text{pure}}_{\leq m-3}(F)}{(m-2)(m-1)m} \\
&\leq \frac{3}{4} N^{\text{pure}}_0(F) + 18 \sum_{k=1}^{m-4} \frac{c^{\text{pure}} \cdot nk^2}{(k+1)(k+2)(k+3)(k+4)} + \frac{c^{\text{pure}} \cdot n(m-3)^2}{(m-2)(m-1)m} \\
&\leq \frac{3}{4} N^{\text{pure}}_0(F) + 18 \cdot c^{\text{pure}} \cdot n \sum_{k=1}^{m-4} \frac{1}{k^2} + \frac{c^{\text{pure}} \cdot n}{m} = O(n)
\end{aligned}$$

Similarly, the expected number of appearances of features corresponding to a mixed configuration is:

$$\begin{aligned}
&\sum_{k=0}^{m-2} \sum_{c \in K^{\text{mix}}_k(F)} \Pr[\text{mixed } c \text{ feature appears}] = \sum_{k=0}^{m-2} \sum_{c \in K^{\text{mix}}_k(F)} \frac{2}{(k+1)(k+2)} \\
&= 2 \sum_{k=0}^{m-2} \frac{N^{\text{mix}}_k(F)}{(k+1)(k+2)}
\end{aligned}$$

$$
\begin{aligned}
&= N_0^{\text{mix}}(F) + 2\sum_{k=1}^{m-2} \frac{N_{\le k}^{\text{mix}}(F) - N_{\le k-1}^{\text{mix}}(F)}{(k+1)(k+2)} \\
&= \frac{1}{2} N_0^{\text{mix}}(F) + 4\sum_{k=1}^{m-3} \frac{N_{\le k}^{\text{mix}}(F)}{(k+1)(k+2)(k+3)} + \frac{N_{\le m-2}^{\text{mix}}(F)}{(m-1)m} \\
&\le \frac{1}{2} N_0^{\text{mix}}(F) + 4\sum_{k=1}^{m-3} \frac{c^{\text{mix}} \cdot nk}{(k+1)(k+2)(k+3)} + \frac{c^{\text{mix}} \cdot n(m-2)}{(m-1)m} \\
&\le \frac{1}{2} N_0^{\text{mix}}(F) + 4 \cdot c^{\text{mix}} \cdot n \sum_{k=1}^{m-3} \frac{1}{k^2} + \frac{c^{\text{mix}} \cdot n}{m} = O(n)
\end{aligned}
$$

Therefore, we have proved the following.

Theorem 4.2. *The expected number of features that appear during the incremental construction is $O(n)$.*

Corollary 4.1. *The expected number of operations is $O(n)$.*

4.2 Overall complexity

The total time for the construction of farthest Voronoi diagrams for all clusters is $O(n \log n)$. For each cluster C_i, we perform $O(|C_i|)$ point location queries and (possibly) one parametric search in the Hausdorff Voronoi diagram of all previous clusters. Therefore, the total number of point location queries is $O(n)$ and the total number of parametric searches is bounded by m. Finally, we have the insertions and deletions of features of the diagram, which are both expected $O(n)$, as corollary 4.1 suggests. We summarize in the following.

Lemma 4.2. *The expected time complexity of the randomized algorithm can be bounded by*

$$O(n \log n) + O(n)t_{\text{q}}(n) + m \cdot t_{\text{p}}(n) + O(n)t_{\text{i}}(n) + O(n)t_{\text{d}}(n),$$

where $t_{\text{q}}(n)$, $t_{\text{p}}(n)$, $t_{\text{i}}(n)$, $t_{\text{d}}(n)$ are the times for a query, a parametric search, an insertion, and a deletion in a point location data structure for a planar subdivision of complexity $O(n)$, respectively. The time for insertion and deletion can be amortized.

In section 5, we will explain how to do a parametric search in a point location data structure in time $t_{\text{p}}(n) = (t_{\text{q}}(n))^2$. The point location data structure from [4] has $t_{\text{q}}(n) = O(\log n \log\log n)$, $t_{\text{i}}(n) = O(\log n \log\log n)$, $t_{\text{d}}(n) = O(\log^2 n)$, whereas the one from [3] has $t_{\text{q}}(n) = O(\log n)$, $t_{\text{i}}(n) = O(\log^{1+\varepsilon} n)$, $t_{\text{d}}(n) = O(\log^{2+\varepsilon} n)$, where $\varepsilon > 0$ can be chosen by the user of the algorithm. Both data structures use linear space and for both of them the insertion and deletion times are amortized. Then, from lemma 4.2, we get expected time complexity $O(n \log^2 n (\log\log n)^2)$ and $O(n \log^{2+\varepsilon} n)$, when using the data structures from [4] and [3], respectively. Thus, our best complexity result is the following.

Theorem 4.3. *There is a randomized algorithm that constructs the Hausdorff Voronoi diagram of a family of non-crossing clusters in linear space and in expected time $O(n \log^2 n (\log\log n)^2)$.*

5 Point location data structures and parametric search

The performance of our randomized incremental algorithm heavily depends on the point location data structure used. Neither static nor semi-dynamic data structures are useful, because elements (vertices, edges, and faces) are added and deleted from the diagram during its incremental construction. For dynamic point location we consider two possibilities: deterministic data structures [4, 3] and a randomized data structure, called *Voronoi hierarchy* [15], inspired from the Delaunay hierarchy of [10].

5.1 Deterministic data structures

Data structures from [4] or [3] are designed for a general planar subdivision and therefore can be used for the Hausdorff Voronoi diagram without any adaptation. It only remains to clarify parametric search on candidate edge $uv \in \text{fskel}(C_{i+1})$.

ParametricSearch(uv):

$s_1 \leftarrow u$; $s_2 \leftarrow v$

run the point location algorithm, but when a call to the SIDE primitive predicate is reached, instead call the SimulateSide function below

SimulateSide(L):

$x_1 \leftarrow \text{SIDE}(L, s_1)$; $x_2 \leftarrow \text{SIDE}(L, s_2)$

if $x_1 = x_2$ then
  return x_1
else if $x_1 = 0$ or $x_2 = 0$ then
  return $x_1 + x_2$
else if $x_1 \neq x_2$ and $x_1 \neq 0$ and $x_2 \neq 0$ then
  let t be the intersection of L with segment $s_1 s_2$
  find C such that $t \in \text{hreg}_{F_i}(C)$
  if $d_f(t, C_{i+1}) < d_f(t, C)$ then
    stop parametric search with result t
  else
    $c_1 \leftarrow \text{cand}(s_1, t)$; $c_2 \leftarrow \text{cand}(t, s_2)$
    if $c_1 = c_2$ then
      stop parametric search with result "not found"
    else if c_1 then
      $s_2 \leftarrow t$; return x_1
    else if c_2 then
      $s_1 \leftarrow t$; return x_2
    end if
  end if
end if

Figure 11: Parametric search

We will rely on the point location algorithm to implement parametric search. It can be verified that that each geometric predicate that involves query point q in the algorithms of [4, 3] can be written as a finite function of primitive geometric predicates of the following form: "On which side of a given line L does point q lie?", called the SIDE predicate. The parametric search for $t \in uv$ is in fact a simulation of a point location query for the unknown point t. It is a simplified form of the parametric search of [19] and similar to the method of [7]. During this simulation, we keep track of a segment $s_1 s_2$ that (possibly) contains t. Initially, $s_1 s_2 = uv$. Whenever, the primitive predicate SIDE is called in the simulation (with some line L as an argument), we check the SIDE predicate for both endpoints of the segment $s_1 s_2$. If the answer for both endpoints is the same or at least one of the answers is zero (i.e., one endpoint is on line L), then we know the answer of the geometric predicate for the possible t and we continue with the simulation. Otherwise $s_1 s_2$ intersects in its interior line L at one point; call this point w. We find out the Hausdorff region in which w lies in $\text{HVD}(F_i)$ by performing the normal point location query (it costs at most $t_q(n)$). Say $w \in \text{hreg}_{F_i}(C^w)$. If $d_f(w, C_{i+1}) < d_f(w, C^w)$, then w is the point we were looking for, so we can stop the parametric search and return $t = w$. Otherwise, we compute the predicates $\text{cand}(s_1 w)$ and $\text{cand}(w s_2)$. If the predicate is true for exactly one of $s_1 w$ and $w s_2$, we update $s_1 s_2$ to the candidate edge among $s_1 w$ and $w s_2$. Moreover, we know the answer of the geometric predicate for the unknown t (because t is possibly contained in the interior of the candidate edge) and we continue with the simulation. Otherwise, we stop the parametric search knowing that there is no $t \in \text{hreg}_{F_{i+1}}(C_{i+1})$. See figure 11 for pseudocode.

Theorem 5.1. *The parametric search can be implemented in $t_p(n) = O((t_q(n))^2)$ time.*

Proof. The point location query operation has $t_q(n)$ elementary steps. Some of them are calls to the side geometric predicate. In the parametric search, the calls to the side predicate are substituted by simulation calls. Each simulation call takes time $O(t_q(n))$, because in it we have a constant number of point location queries. □

5.2 The Voronoi hierarchy for the Hausdorff Voronoi diagram

Our randomized point location data structure is an augmentation of the *Voronoi hierarchy* [15, 6]. A randomized hierarchical point location data structure called *Delaunay hierarchy* [10] was proposed for the incremental construction of the Delaunay triangulation of a set of points. The Delaunay hierarchy is analogous to a skip list [26] for 2D data and yields an optimal randomized incremental construction of the Delaunay triangulation. A similar point location data structure, called *Voronoi hierarchy*, is presented in [15]. The current Voronoi hierarchy does not perform efficiently for sites with empty regions and its performance has not been analyzed for sites of non-constant complexity. However, empty regions and sites of non-constant complexity are serious issues that are inherent to the Hausdorff Voronoi diagram. In the following, we define this hierarchy for a family F of general sites and we explain how we address these issues, by augmenting the Voronoi hierarchy.

Every level ℓ of the hierarchy corresponds to a subset $F^{(\ell)}$ of F and also stores the Voronoi diagram of $F^{(\ell)}$. Level 0 corresponds to F itself. A *Voronoi hierarchy* of *height* k is then: $F = F^{(0)} \supseteq F^{(1)} \supseteq \ldots \supseteq F^{(k)}$. For all $\ell \in \{1,\ldots,k\}$, $F^{(\ell)}$ is a random sample of $F^{(\ell-1)}$ according to a Bernoulli distribution with parameter $\beta \in (0,1)$. It is not difficult to show that the expected height of the hierarchy is $O(\log m)$, where m is the number of sites.

Point location in the Voronoi hierarchy works as follows: Starting from the topmost level k, for each level ℓ, find the site in $F^{(\ell)}$ which is nearest to the query point q, by performing a *walk*. Each step of the walk reduces the distance of q from the current site S by moving to a site, neighboring to S. When continuing to the lower level $\ell-1$, start from the site that was found in the previous level ℓ. Answer the query with the closest site found at level 0.

For Hausdorff Voronoi diagrams several complications with the hierarchy arise. In particular, (a) performing the walk efficiently for sites of non-constant complexity, (b) the need of performing parametric search on the hierarchy and performing walks for an unknown point along a candidate edge, (c) the existence of empty Voronoi regions, i.e., some cluster (site) P can have empty region in level ℓ, but non-empty region in level $\ell+1$; if such a cluster P is the last cluster visited in level $\ell+1$ during a walk, we will not know with which cluster to continue in level ℓ. In the following we augment the hierarchy with the ability to handle these issues efficiently.

The following results show that the expected length of a walk at level ℓ is constant, and that the total size of the hierarchy is $O(n)$.

Lemma 5.1. *Let q be a point in $\mathbb{R}^2$ and let $S^{(\ell+1)} \in F^{\ell+1}$ be the site nearest to q at level $\ell+1$. The expected number of sites in $F^{(\ell)}$, which are strictly closer to q than is $S^{(\ell+1)}$, is constant.*

Proof. The proof is an adaptation of the proof of lemma 9 in [15].

The probability that $S^{(\ell+1)}$ is the k-th nearest site to q, among all sites in $F^{(\ell)}$, is $\beta(1-\beta)^{k-1}$. This is because the probability that a site among the $k-1$ sites, nearest to q in $F^{(\ell)}$, does not belong in $F^{\ell+1}$ is $(1-\beta)$, the probability that the k-th nearest site to q in $F^{(\ell)}$ belongs in $F^{\ell+1}$ is β, and all these events are independent. Therefore, the expected number of sites in $F^{(\ell)}$, which are strictly closer than $S^{(\ell+1)}$ to q, is

$$N_\ell < \sum_{k=1}^{|F^{(\ell)}|} k(1-\beta)^{k-1}\beta < \beta \sum_{k=1}^{\infty} k(1-\beta)^{k-1} = \frac{1}{\beta},$$

which is constant. □

Lemma 5.2. *Let $S_0^\ell, S_1^\ell, \ldots S_r^\ell \in F^{(\ell)}$ be the sequence of sites visited during a walk at level ℓ to locate the site in $F^{(\ell)}$ nearest to query point q. Let $S^{\ell+1} \in F^{(\ell+1)}$ be the site nearest to q at level $\ell+1$. Assume that either $S^{\ell+1} = S_0^\ell$, or $d_f(q,S_0^\ell) < d_f(q,S^{\ell+1})$, and $d_f(q,S_i^\ell) < d(q,S_{i-1}^\ell)$, for $i \in \{1,\ldots,r\}$. Then, the expected length of the walk at level ℓ is constant.*

Proof. Distance to q during the walk is monotonically decreasing, and the walk starts either at $S^{\ell+1}$ or at a site strictly closer to q than $S^{\ell+1}$ is. Therefore, each of the sites $S_1^\ell, \ldots, S_r^\ell$ is strictly closer to q than $S^{\ell+1}$. By lemma 5.1 the expected number of such sites is constant. □

Lemma 5.3. *Let the time complexity of one step of the walk be bounded by t_s with $t_s = \Omega(\log n)$. Then, a point location query is answered in expected $O(t_s \log m)$ time, where m is the number of sites.*

Proof. The expected length of the walk at level ℓ is constant. The expected time spent in level ℓ is thus $O(t_s)$. Going to an appropriate cluster in the level below takes time $O(\log n) = O(t_s)$. The expected height of the hierarchy is $O(\log m)$ Thus, the point location query is answered in expected $O(t_s \log m)$ time. □

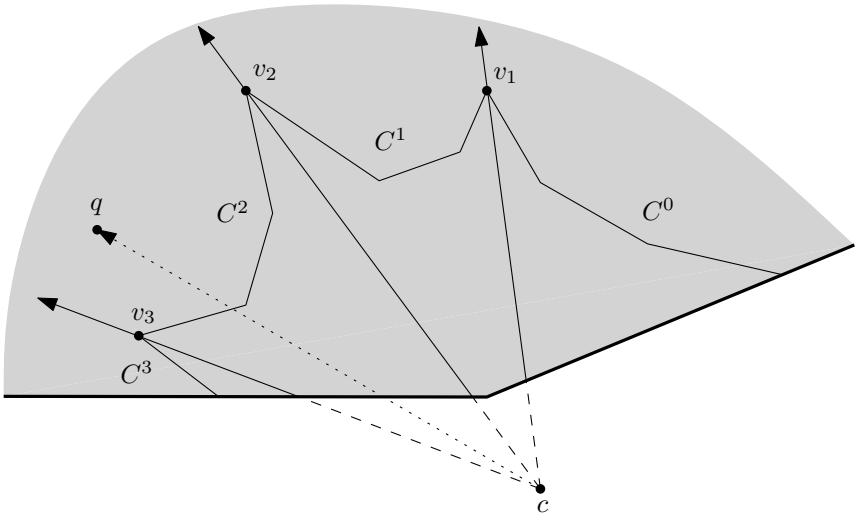

Figure 12: Vertices v_1, v_2, v_3 are pure vertices on the Hausdorff boundary of active point c. Region $\text{freg}_{\hat{C}^{(\ell)}}(c)$ is shown gray and its boundary is drawn heavy. In this example, for the given q, the neighboring cluster computed is $C' = C^2$.

Lemma 5.4. *Let n be the sum of the sizes of all sites in a family of sites F. Assuming that the underlying type of Voronoi diagram for F is of size $O(n)$, then the expected size of the Voronoi hierarchy for F is also $O(n)$.*

Proof. For each site $S_i \in F$ the expected number of levels where S_i appears is $\frac{1}{1-\beta}$. At each level ℓ, the size of the Voronoi diagram is $O(|F^\ell|)$, where $|F^\ell|$ is the sum of the sizes of all sites in F^ℓ. Since sampling is performed independently, the expected sum of sizes of all sites at all levels of the hierarchy is

$$N = \sum_{i=1}^{m} \frac{1}{1-\beta}\text{size}(S_i) < \frac{1}{1-\beta}\sum_{i=1}^{\infty}\text{size}(S_i) = O(n). \qquad \square$$

5.2.1 A step in the walk

We now describe how to perform a step of the walk in a level ℓ of the hierarchy. To perform this step efficiently we augment the hierarchy with a list of points for each cluster at every level that have non-empty region at the diagram of that level. We will use the notation $\text{hreg}^{(\ell)}_{\dots}(\cdot)$ for the Hausdorff region of a cluster or a point at level ℓ of the hierarchy. Consider a cluster C at some level ℓ with non-empty Hausdorff region $\text{hreg}^{(\ell)}_F(C) = \text{hreg}_{F^{(\ell)}}(C)$.

Definition 5.1. We say that point $c \in C$ is *active* at level ℓ, if $\text{hreg}^{(\ell)}_F(c) \neq \emptyset$; otherwise c is called *inactive* at level ℓ. Denote the subset of active points of cluster C at level ℓ by $\hat{C}^{(\ell)}$. We call $\hat{C}^{(\ell)}$ the *active set* of C at level ℓ. We also define the shorthand notation $d^{(\ell)}_{\text{f}}(t, C) = d_{\text{f}}(t, \hat{C}^{(\ell)})$.

The following additional data are stored for each cluster C in each level ℓ such that $C \in F^{(\ell)}$:

- The active set $\hat{C}^{(\ell)}$ of C at level ℓ is stored in counterclockwise order of $\text{CH}(\hat{C}^{(\ell)})$ in a balanced binary tree; $\text{CH}(\hat{C}^{(\ell)})$ is a subsequence of $\text{CH}(C)$.
- For each active point $c \in \hat{C}^{(\ell)}$, we keep a list of all pure vertices adjacent to $\text{hreg}^{(\ell)}_F(c)$ (see figure 12), sorted in counterclockwise order along the boundary of the region and stored in a balanced binary tree.

Now, we describe how to perform one step of the walk. Let C be the current cluster visited during the walk in level ℓ, and q be the query point. We first describe which is the next cluster C' in the walk. Let c be a point of $\hat{C}^{(\ell)}$ such that $d(q,c) = d^{(\ell)}_{\text{f}}(q, C)$, i.e., among points in $\hat{C}^{(\ell)}$, c is farthest from q ($q \in \overline{\text{freg}}_{\hat{C}^{(\ell)}}(c)$). Consider a counterclockwise traversal of the Hausdorff boundary of $\text{hreg}^{(\ell)}_F(c)$. If the pure vertices on this Hausdorff boundary are encountered in the order $v_1, \dots, v_j$, then the rays $\overrightarrow{cv_1}, \dots, \overrightarrow{cv_j}$ partition $\overline{\text{freg}}_{\hat{C}^{(\ell)}}(c)$ into $j+1$ unbounded regions (see figure 12). If $\overrightarrow{cq}$ is just after ray $\overrightarrow{cv_i}$ or just before ray $\overrightarrow{cv_{i+1}}$, then set $C' = C^i$, where $C^0, \dots, C^j, C^{j+1}$ are the clusters in the counterclockwise order inducing the Hausdorff boundary of c.

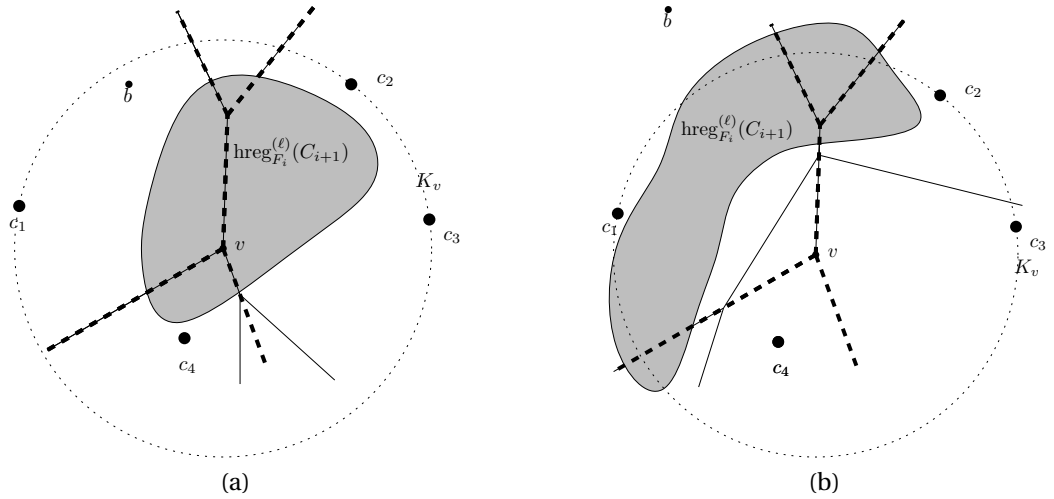

Figure 13: For cluster $C = \{b, c_1, \ldots, c_4\} \in F^{(\ell)}$, point $b \notin \hat{C}^{(\ell)}$. Say c_1 and c_2 lie on the two tangents from b to $\hat{C}^{(\ell)}$. In subfigure (a), the statement of lemma 5.5 is true and we have $v \notin \mathrm{freg}_{\hat{C}^{(\ell)}\cup\{b\}}(b)$. In subfigure (b), $v \in \mathrm{freg}_{\hat{C}^{(\ell)}\cup\{b\}}(b)$, which gives a contradiction. Region $\mathrm{hreg}_{F^{(\ell)}}(C)$ is shown as a gray area and a minimum enclosing circle centered at v is shown dotted.

We first describe how to find the farthest from q point c of $\hat{C}^{(\ell)}$. Initially, we find the farthest from q point b of C. If b is active, we set $c = b$. Otherwise, b is inactive and in that case we take the (at most) two tangents from b to $\mathrm{conv}\,\hat{C}^{(\ell)}$, which touch $\mathrm{conv}\,\hat{C}^{(\ell)}$ at points $c_1, c_2 \in \mathrm{CH}(\hat{C}^{(\ell)})$. Finally, we set c to be the farthest from q among c_1 and c_2. The tangents can be computed with binary search in $\mathrm{CH}(\hat{C}^{(\ell)})$. The correctness of the procedure to compute c is proved in the following.

Lemma 5.5. *Let C be a cluster at level ℓ and b an inactive point of cluster C at level ℓ. Then,* $\mathrm{freg}_C(b) \subset \mathrm{freg}_{\hat{C}^{(\ell)}}(c_1) \cup \mathrm{freg}_{\hat{C}^{(\ell)}}(c_2)$*, where $c_1, c_2 \in \hat{C}^{(\ell)}$ lie on the two tangents from b to* $\mathrm{conv}\,\hat{C}^{(\ell)}$.

Proof. Set $C' = \hat{C}^{(\ell)} \cup \{b\}$. Since $C' \subseteq C$, we have $\mathrm{freg}_C(b) \subseteq \mathrm{freg}_{C'}(b)$ and thus it is enough to prove that $\mathrm{freg}_{C'}(b) \subset \mathrm{freg}_{\hat{C}^{(\ell)}}(c_1) \cup \mathrm{freg}_{\hat{C}^{(\ell)}}(c_2)$.

First, it is not difficult to see that c_1 and c_2 are consecutive points in $\mathrm{CH}(\hat{C}^{(\ell)})$. Assume for the sake of contradiction that $\mathrm{freg}_{C'}(b) \cap \mathrm{freg}_{\hat{C}^{(\ell)}}(c_3) \neq \emptyset$ for some $c_3 \in \hat{C}^{(\ell)}$ such that c_3 is different from c_1 and c_2. Since both $\mathrm{fskel}(\hat{C}^{(\ell)})$ and $\mathrm{fskel}(C')$ have a tree-like structure and $\mathrm{freg}_{C'}(b)$ includes elements from at least three regions of $\mathrm{FVD}(\hat{C}^{(\ell)})$, $\mathrm{freg}_{C'}(b)$ has to contain at least one vertex v of $\mathrm{FVD}(\hat{C}^{(\ell)})$. Since $v \in \mathrm{freg}_{C'}(b)$, we have $d(v,b) < d(v,c)$ for every $c \in \hat{C}^{(\ell)}$. But v also belongs to $\mathrm{fskel}(C) \cap \mathrm{hreg}^{(\ell)}_F(C)$, which implies $d(v,c) > d(v,b)$ for some $c \in \hat{C}^{(\ell)}$, since b is not active at level ℓ; a contradiction. □

As soon as we have c, then C' can be computed by binary search of the slope of $\overrightarrow{cq}$ in the slopes of $\overrightarrow{cv_1}$, $\ldots$, $\overrightarrow{cv_j}$ (see figure 12). The balanced binary tree of pure vertices $v_1, \ldots, v_j$ allows as to do it in $O(\log j)$ time. Therefore, computing C', given C and q, takes $O(\log n)$ time in total.

Let $D(x_0, R)$ denote the *closed disk* with center x_0 and radius R. We now argue about correctness of computation of C'.

Lemma 5.6. *Let C' be computed from C and q as above and among clusters in $F^{(\ell)}$, C is not closest to q. Then* $d_{\mathrm{f}}(q, C') < d^{(\ell)}_{\mathrm{f}}(q, C) \leq d_{\mathrm{f}}(q, C)$.

Proof. Let q be furthest from c among points in $\hat{C}^{(\ell)}$ and define the closed disk $D_q = D(q, |cq|)$. Then, $D_q \supseteq \hat{C}^{(\ell)}$. We treat separately the following two cases:

1. Ray $\overrightarrow{cq}$ intersects the Hausdorff boundary of c. In that case, segment cq intersects the Hausdorff boundary of c at a single point x in the interior of segment cq, i.e., $|cx| < |cq|$ (see figure 14). Point x is equidistant from C and C'. For the closed disk $D_x = D(x, |cx|)$, we have $D_x \supset C'$ and $D_q \supset D_x$. Moreover, the only point on the boundary of D_q which lies also in D_x is c and $c \notin C'$ (because $c \in C$). Therefore, C' is contained in the interior of D_q which implies $d_{\mathrm{f}}(q, C') < d_{\mathrm{f}}(q, \hat{C}^{(\ell)}) \leq d_{\mathrm{f}}(q, C)$.

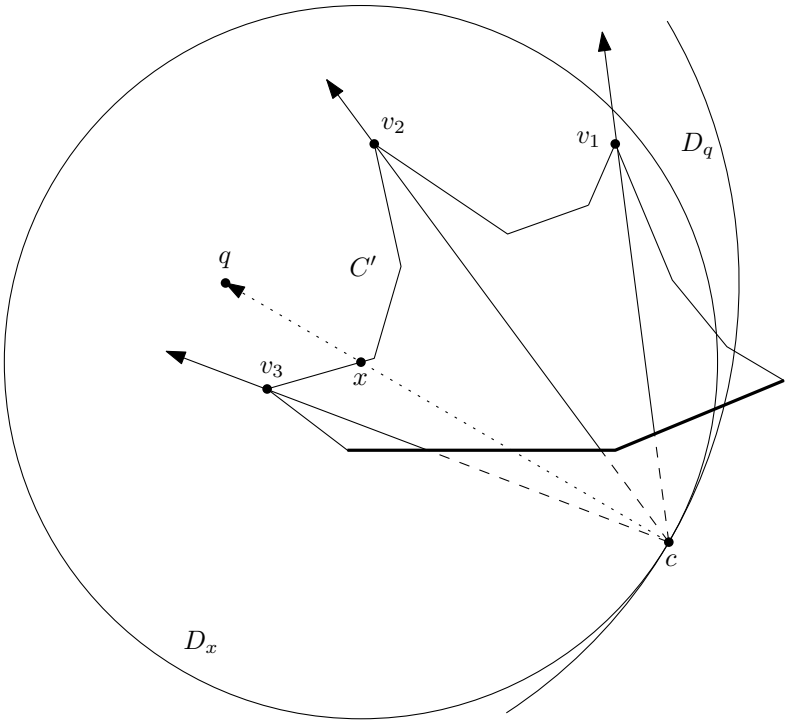

Figure 14: Ray $\overrightarrow{cq}$ intersects the Hausdorff boundary of c at point x. Closed disk D_x contains both C and C'. Moreover, $D_x \subset D_q$.

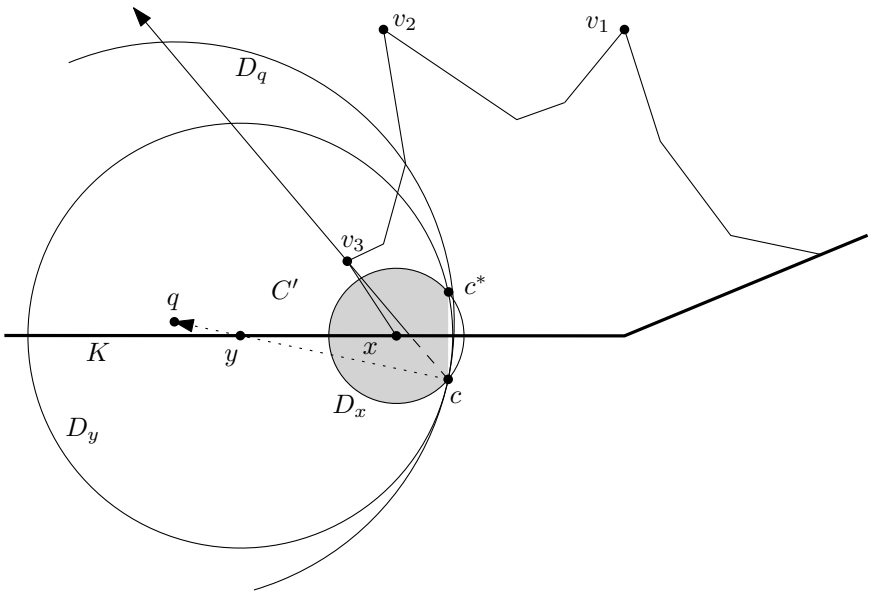

Figure 15: Ray $\overrightarrow{cq}$ does not intersect the Hausdorff boundary of c, but it intersects K, the boundary of $\text{freg}_{\tilde{C}^{(\ell)}}(c)$, at a point y not closest to C among clusters in $F^{(\ell)}$. The mixed vertex x is closer to y on curve K and is equidistant to C and C'. D_x^{f} is depicted in gray.

2. Ray $\overrightarrow{cq}$ does not intersect the Hausdorff boundary of c. In that case, let $K = \operatorname{bd}\operatorname{freg}_{\hat{C}^{(\ell)}}(c)$. Then, K intersects with segment cq at a single point y and thus $|cy| \leq |cq|$ (see figure 15). Consider the closed disk $D_y = D(y, |cy|)$. We have $\hat{C}^{(\ell)} \subset \operatorname{conv}\hat{C}^{(\ell)} \subset D_y \subseteq D_q$. Moreover, since y lies on the interior of an edge of $\mathrm{FVD}(\hat{C}^{(\ell)})$, the boundary of D_y contains exactly two points c and c^* of $\hat{C}^{(\ell)}$ and all other points of $\operatorname{conv}\hat{C}^{(\ell)}$ are contained in the interior of D_y. Now, walk on K from y until you reach the Hausdorff boundary of c, say at point x. Then, x is a mixed vertex of C and thus it is equidistant from c, c^*, and the neighboring cluster C'. Consider the closed disk $D_x = D(x, |cx|)$. Cluster C' is limiting w.r.t. chord $\overline{cc^*}$ (see definition 2.2). Assume without loss of generality that C' is forward limiting. Then, $C' \subset D_x^{\mathrm{f}} \cup \operatorname{conv}\hat{C}^{(\ell)}$. Since y is not closest to C among clusters in $F^{(\ell)}$, we have $y \in T(x)$ and $D_x^{\mathrm{f}} \subset D_y \subseteq D_q$ (see figure 15, where D_x^{f} is shown with gray color). Except c and c^* every point in $D_x^{\mathrm{f}} \cup \operatorname{conv}\hat{C}^{(\ell)}$ is contained in the interior of D_q. Thus, C' (which contains neither c nor c^*) is contained in the interior of D_q and this implies $d_{\mathrm{f}}(q, C') < d_{\mathrm{f}}(q, \hat{C}^{(\ell)}) \leq d_{\mathrm{f}}(q, C)$. □

As soon as C' is computed from C and q, we compare $d_{\mathrm{f}}(q, C')$ and $d_{\mathrm{f}}(q, C)$. If $d_{\mathrm{f}}(q, C') < d_{\mathrm{f}}(q, C)$, then we continue the walk with C', otherwise the walk at level ℓ stops.

Therefore, we have the following.

Lemma 5.7. *A step of the walk takes $O(\log n)$ time.*

Corollary 5.1. *A point location query is answered in expected $O(\log^2 n)$ time.*

5.2.2 Parametric search in the Voronoi hierarchy

We now explain how to do parametric search on candidate edge $uv \in \mathrm{fskel}(C_{i+1})$. For $\ell \in \{0, \ldots, k\}$, let I^ℓ be the interval of points on uv which are closer to C_{i+1} than any cluster in $F_i^{(\ell)}$. (By convention, $I^{k+1} = uv$.) Because of property 2.1, I^ℓ is an open segment (or ray or line) or empty and we have $uv = I^{k+1} \supseteq I^k \supseteq I^{k-1} \supseteq \cdots \supseteq I^1 \supseteq I^0$. If $I^\ell \neq \emptyset$, our algorithm will compute successively the leftmost endpoint u^ℓ of interval I^ℓ, i.e., the endpoint closer to u. If $I^0 \neq \emptyset$, then u^0 is a point on the boundary of the Hausdorff region of C_{i+1} from which we can start tracing this region.

The computation of u^ℓ happens entirely in level ℓ of the hierarchy. From the previous level, we already have $u^{\ell+1}$ and the cluster of $F_i^{(\ell+1)}$ closest to $u^{\ell+1}$. Within level ℓ, the algorithm computes a sequence of points $u^{\ell+1} = a_0, a_1, \ldots, a_r = u^\ell$. At all times, we keep track of the cluster in F_i^ℓ that is closest to each of these points. We compute a_{j+1} from a_j as follows. Let C^{a_j} be the cluster closest to a_j. Observe that $d_{\mathrm{f}}(x, C_{i+1}) = d(x, c) = d(x, c^*)$, for every $x \in uv$, where $\overline{cc^*}$ is the chord of C_{i+1} corresponding to uv, and thus the computation of $d_{\mathrm{f}}(x, C_{i+1})$ as above takes constant time.

- If $d_{\mathrm{f}}(a_j, C_{i+1}) \leq d_{\mathrm{f}}(a_j, C^{a_j})$, we set $u^\ell = a_j$ and continue to the next level.
- Else, if $d_{\mathrm{f}}(v, C^{a_j}) \leq d_{\mathrm{f}}(v, C_{i+1})$, we stop and report that t does not exist, and that $\mathrm{hreg}_{F_{i+1}}(C_{i+1})$ is empty.
- Otherwise, the Hausdorff bisector of C_{i+1} and C^{a_j} crosses the interior of $a_j v$ at a single point, a_{j+1}, for which $d(a_{j+1}, C_{i+1}) = d(a_{j+1}, C^{a_j})$. To determine a_{j+1} we do a parametric search in $\mathrm{FVD}(C^{a_j})$ with segment $a_j v$. Then, we perform a walk (in level ℓ) from cluster C^{a_j} to determine the cluster $C^{a_{j+1}}$ closest to a_{j+1}. If $C^{a_{j+1}} = C^{a_j}$, we set $u^\ell = a_j$ and continue to the next level.

Pseudocode for the parametric search in the Voronoi hierarchy is given in figure 16. Pseudocode for the parametric search in $\mathrm{FVD}(C)$ is given in figure 17 and it is similar to the parametric search in section 5.1. Correctness of the algorithm is implied by the following: If parametric search stops at level ℓ with answer "not found", then there exist clusters C', $C'' \in F_i^{(\ell)}$ such that $\mathrm{hreg}_{\{C_{i+1}, C', C''\}}(C_{i+1}) = \emptyset$. Otherwise, interval $u^{\ell+1}u^\ell \notin \mathrm{hreg}_{F_{i+1}}(C_{i+1})$.

We now argue about time complexity. We start with the following.

Lemma 5.8. *A parametric search query in $\mathrm{FVD}(C)$ can be performed in time $O(\log^2|C|)$.*

Proof. The static point location data structure of [16] has size $O(|C|)$, construction time $O(|C|\log|C|)$, and point location query time $t_{\mathrm{q}} = O(\log|C|)$. Arguing as in section 5.1, the parametric search query can be implemented in $O(t_q^2) = O(\log^2|C|)$ time. □

Lemma 5.9. *The expected number of clusters visited during the parametric search at level ℓ is constant.*

ParametricSearchVH($u\,v$):

$a' \leftarrow u$
walk in level k to cluster $C^{a'}$ closest to a'
for $\ell \leftarrow k$ downto 0 do:
  repeat:
    $a \leftarrow a'$; $C^a \leftarrow C^{a'}$
    if $d_{\mathrm{f}}(a, C_{i+1}) \leq d_{\mathrm{f}}(a, C^a)$ then
      $a' \leftarrow a$
    else if $d_{\mathrm{f}}(v, C_a) \leq d_{\mathrm{f}}(v, C_{i+1})$ then
      stop parametric search with result "not found"
    else
      $a' \leftarrow$ ParametricSearchFVD(C^a, a, v)
      walk in level ℓ from cluster C^a to $C^{a'}$ closest to a'
    end if
  until $a' = a$
  if $\ell > 0$ then
    $C^{a'} \leftarrow$ cluster of level $\ell - 1$ linked from $C^{a'}$ and closer to a'
  end if
end for
return a'

Figure 16: Parametric search in the Voronoi hierarchy

ParametricSearchFVD(C, x, y):

$s_1 \leftarrow x$; $s_2 \leftarrow y$
run the point location algorithm FVD(C), but when a call to the SIDE primitive predicate is reached, instead call the SimulateSide function below

SimulateSide(L):

$x_1 \leftarrow \mathrm{SIDE}(L, s_1)$; $x_2 \leftarrow \mathrm{SIDE}(L, s_2)$
if $x_1 = x_2$ then
  return x_1
else if $x_1 = 0$ or $x_2 = 0$ then
  return $x_1 + x_2$
else if $x_1 \neq x_2$ and $x_1 \neq 0$ and $x_2 \neq 0$ then
  let z be the intersection of L with segment $s_1 s_2$
  find $p \in C$ such that $z \in \mathrm{freg}_p(C)$
  if $d_{\mathrm{f}}(z, C_{i+1}) = d(z, p)$ then
    stop ParametricSearchFVD and return point z
  else if $d_{\mathrm{f}}(z, C_{i+1}) < d(z, p)$ then
    $s_1 \leftarrow t$; return x_2
  else if $d_{\mathrm{f}}(z, C_{i+1}) > d(z, p)$ then
    $s_2 \leftarrow t$; return x_1
  end if
end if

Figure 17: Parametric search in the farthest Voronoi diagram FVD(C)

Proof. The procedure of parametric search in a candidate edge uv at level ℓ consists of a successive search for points $u^{\ell+1} = a_0, a_1, \ldots, a_r = u^\ell$ as described before. For each such point a_j, a walk from $C^{a_{j-1}}$ to C^{a_j} is performed. We first claim that the expected length r of the sequence $a_0, a_1, \ldots, a_r$ is constant.

Each point a_j, for $j \in \{1, \ldots, r\}$, is by construction equidistant from C_{i+1} and $C^{a_{j-1}}$. Cluster $C^{a_{j-1}}$ must be enclosed in disk D_{a_j}, centered at a_j and with radius $d_f(a_j, C_{i+1})$. Furthermore, since $d_f(v, C^{a_{j-1}}) > d_f(v, C_{i+1})$, $C^{a_{j-1}}$ must be enclosed in $D^r_{a_j} \cup \operatorname{conv} C_{i+1}$ (assuming that u is an ancestor of v in a rooted tree, corresponding to fskel(C_{i+1})). In other words, $C^{a_{j-1}}$ is a rear limiting cluster with respect to $\overline{cc^*}$. By lemma 1 from [22], $C^{a_{j-1}} \subset D_{a_0}$. Note, that $C^{a_r} = C^{a_{r-1}}$. Thus, $C^{a_j} \subset D_{a_0}$ for all $j \in \{0, \ldots, r\}$.

Let $C^{\ell+1} \in F_i^{(\ell+1)}$ be the cluster nearest to $a_0 = u^{\ell+1}$ in level $\ell+1$. By lemma 5.1, the expected number of clusters in $F_i^{(\ell)}$ that are enclosed in D_{a_0} is constant. Since all C^{a_j}, for $j \in \{0, \ldots, r\}$, are enclosed in disk D_{a_0}, r is expected to be constant.

Now, note that for each $j \in \{1, \ldots, r\}$, $d_f(a_j, C^{a_{j-1}}) = d_f(a_j, C_{i+1})$, and a_j is closer to C_{i+1} than to any cluster in $F_i^{(\ell+1)}$. Therefore, by lemma 5.2, the expected length of the walk from $C^{a_{j-1}}$ to C^{a_j} is constant.

To summarize, during the parametric search at level ℓ an expected constant number of walks is performed, of expected constant length each. Thus, the claim follows. □

Finally, we prove expected $O(\log^3 n)$ time for the parametric search operation in the Voronoi hierarchy:

Lemma 5.10. *A parametric search query in the Voronoi hierarchy takes expected time* $O(\log^3 n)$.

Proof. The expected number of levels in the hierarchy is $O(\log n)$. In each level we visit an expected constant number of clusters (lemma 5.9). For each cluster C that we visit we make a parametric search in time $O(\log^2 |C|)$ (lemma 5.8) and we compute its neighboring cluster in time $t_s = O(\log n)$. Therefore, the parametric search query in the Voronoi hierarchy takes expected time $O(\log^3 n)$. □

5.2.3 Updating the hierarchy when a cluster disappears

During the incremental construction of the Hausdorff Voronoi diagram, it can happen that the initially non-empty region of a cluster P becomes empty because of the insertion of subsequent clusters. We describe how to update the hierarchy when an initially non-empty region of a cluster P becomes empty.

Definition 5.2. We say that a cluster $P \in F_i$ *disappears* at level ℓ as a result of insertion of cluster C_{i+1} if $\operatorname{hreg}^{(\ell)}_{F_i}(P) \neq \emptyset$ and $\operatorname{hreg}^{(\ell)}_{F_{i+1}}(P) = \emptyset$.

Cluster P can disappear from many levels where it appeared because of the insertion of C_{i+1}, but not necessarily all of them.

Definition 5.3. We say that a cluster $P \in F_i$ is *critical at level* $\ell+1$ if it disappears at level ℓ but not at level $\ell+1$.

Obviously, either a disappearing cluster is critical at a single level or it is not critical (it disappears from all levels where it appeared). We must pay special attention to a cluster P as soon as it becomes critical at level $\ell+1$, because P is an obstacle to correct point location. Indeed, for query point $q \in \operatorname{hreg}^{(\ell+1)}_{F_{i+1}}(P)$, we do not know in which cluster to continue the point location in level ℓ. One straightforward way to fix the problem would be to remove criticality by erasing P from all levels, like in [15], but we can not afford to do it in the case of Hausdorff Voronoi diagrams, because it is computationally expensive. Instead, we resort to linking cluster P at level $\ell+1$ to at most two other clusters C', C'' in level ℓ with the following property: Every $q \in \operatorname{hreg}^{(\ell+1)}_{F_{i+1}}(P)$ is closer to at least one of C', C'' than P. By property 2.4, such C' and C'' are guaranteed to exist in a family of non-crossing clusters and we explain how to find them in the following.

Suppose P is critical at level $\ell+1$, (i.e., $\operatorname{hreg}^{(\ell+1)}_{F_{i+1}}(P) \neq \emptyset$, but $\operatorname{hreg}^{(\ell)}_{F_{i+1}}(P) = \emptyset$). While inserting C_{i+1} at level ℓ we temporarily keep track of the list V of all the (deleted) P-mixed vertices at level ℓ.

At level $\ell+1$, for every P-mixed vertex v of $\operatorname{hreg}^{(\ell+1)}_{F_{i+1}}(P)$, we check if v is closer to C_{i+1} or to P (this can be done in $O(\log n)$ time per vertex). If $d_f(v, C_{i+1}) \geq d_f(v, P)$, then we automatically know the point $c \in C_{i+1}$ for which $d_f(v, C_{i+1}) = d(v, c)$. Point $c \notin \operatorname{conv} P$ and it will be useful at level ℓ. The linking is performed as follows.

1. If all P-mixed vertices of $\operatorname{hreg}^{(\ell+1)}_{F_{i+1}}(P)$ are closer to C_{i+1} than to P, we link only to C_{i+1}.
2. Else we identify cluster K at level ℓ such that $\{K, C_{i+1}\}$ is a killing pair for P.

   If C_{i+1} is a cluster at level $\ell+1$ (in this case all P-mixed vertices must be closer to P) we link only to cluster K.

   Otherwise, we link to both C_{i+1} and K.

```
AllCloserToNew ← TRUE
for each v in the set of P-mixed vertices of hreg^(ℓ+1)_{F_{i+1}}(P) do:
   result ← d_f(v, C_{i+1}) − d_f(v, P)
   if result > 0 then
      AllCloserToNew ← FALSE
      let c ∈ C_{i+1} be such that d_f(v, C_{i+1}) = d(v, c)
      break out of for loop
   end if
end for
if AllCloserToNew then
   link P at level ℓ+1 only to C_{i+1}
else
   K ← ComputeKinLevelBelow(c)
   if C_{i+1} ∈ F^(ℓ+1)_{i+1} then
      link P at level ℓ+1 only to K
   else
      link P at level ℓ+1 to both C_{i+1} and K
   end if
end if
```

Figure 18: Linking critical cluster P at level $\ell+1$

Pseudocode for the linking procedure is given in figure 18.

We determine cluster K of the killing pair using the list V of the P-mixed vertices of level ℓ, and the point $c \in C_{i+1}$, $c \notin \operatorname{conv} P$, that was computed at level $\ell+1$. Each such vertex $v \in V$ is equidistant from points p, $p^* \in P$, and $q \in Q$, for some $Q \in F_i^{(\ell)}$. We simply check whether c and q are on different sides of the chord $\overline{pp^*}$. If yes, then we set $K = Q$ and we stop. Note that in this case Q and C_{i+1} are limiting clusters for $\overline{pp^*}$ of opposite type (one rear and one forward limiting) and they form a killing pair for P (see definition 2.2 and property 2.4). It remains to argue about correctness of the linking.

Lemma 5.11. *If P at level $\ell+1$ is linked only to C_{i+1} (case 1), then all points of* $\operatorname{hreg}^{(\ell+1)}_{F_{i+1}}(P)$ *are closer to C_{i+1} than P.*

Proof. In this case, all P-mixed vertices of $\operatorname{hreg}^{(\ell+1)}_{F_{i+1}}(P)$ are closer to C_{i+1} than P. Then, by property 2.2, no portion of $\operatorname{fskel}(P)$ in $\operatorname{hreg}^{(\ell+1)}_{F_{i+1}}(P)$ can be closer to P than C_{i+1}. This implies that no point in $\operatorname{hreg}^{(\ell+1)}_{F_{i+1}}(P)$ can be closer to P than C_{i+1}. □

Lemma 5.12. *If P at level $\ell+1$ is linked to cluster K (case 2), then K is a unique cluster at level ℓ that constitutes together with C_{i+1} a killing pair of P.*

Proof. Let u be a P-mixed vertex of $\operatorname{hreg}^{(\ell+1)}_{F_{i+1}}(P)$ closer to P than C_{i+1}, which reveals a point $c \in C_{i+1}$ for which $c \notin \operatorname{conv}(P)$. Since we are in case 2, at least one such mixed vertex exists. By property 2.2, the portion of $\operatorname{fskel}(P)$ in $\operatorname{hreg}^{(\ell)}_{F_i}(P)$ is a connected subtree that can be regarded without loss of generality as a descendant of u in $T(P)$. Then, C_{i+1} is a forward limiting cluster for any P-mixed vertex v of $\operatorname{hreg}^{(\ell+1)}_{F_{i+1}}(P)$, that is, $C_{i+1} \subset D_v$ and in particular, $C_{i+1} \subset D^{\mathrm{f}}_y \cup \operatorname{conv} P$; thus, $c \in D^{\mathrm{f}}_v$. Let w be the first P-mixed vertex of $\operatorname{hreg}^{(\ell)}_{F_i}(P)$ encountered as we traverse $\operatorname{fskel}(P)$ from u to its portion enclosed in $\operatorname{hreg}^{(\ell+1)}_{F_{i+1}}(P)$. Let Q be the cluster inducing w and let q be the point in Q for which $d(w,q) = d_{\mathrm{f}}(w,Q)$. By definition of w, cluster Q must be rear limiting with respect to w and thus $q \in D^{\mathrm{r}}_w$. Thus, $\{Q, C_{i+1}\}$ is a killing pair for P and q, c lie at opposite sides of the chord of P inducing w. It is easy to see (see lemma 2 of [22]) that all other mixed vertices v_i of $\operatorname{hreg}^{(\ell)}_{F_i}(P)$, where $v_i \neq r$, must be induced by clusters Q_i that are forward limiting with respect to v_i. Thus, any point q_i, inducing a P-mixed vertex v_i, considered during our algorithm, other than q, must lie on the same side of D_{v_i} as c. Thus, our algorithm correctly sets $K = Q$. Furthermore, there is no other cluster on level ℓ that can form a killing pair with C_{i+1} for P. □

Corollary 5.2. *In case 2 of the linking algorithm, all points of* $\operatorname{hreg}^{(\ell+1)}_{F_{i+1}}(P)$ *are closer to either C_{i+1} or K than P. If in addition $C_{i+1} \in F^{(\ell+1)}_{i+1}$, then all points of* $\operatorname{hreg}^{(\ell+1)}_{F_{i+1}}(P)$ *are closer to K than P.*

5.2.4 Complexity analysis for the Voronoi hierarchy

The remaining details of the complexity analysis for constructing HVD(F) with the help of the Voronoi hierarchy are given here.

We first argue about structural changes during the incremental construction in all Hausdorff Voronoi diagrams which are maintained (one per level) within the hierarchy. Recall from section 4, that the expected number of structural changes is proportional to the expected number of appearing features, where features correspond to pure and mixed configurations (see definitions 4.2 and 4.3).

Lemma 5.13. *The expected number of features that appear at any level of the Hausdorff Voronoi hierarchy during the incremental construction is $O(n)$.*

Proof. The expected total number of points in $F^{(\ell)}$ is $\beta^\ell n$. Using theorem 4.2, the expected number of features that appear during the incremental construction of HVD($F^{(\ell)}$) at level ℓ is $O(\beta^\ell n)$. Therefore, the expected number of features that appear at any level is $\sum_{\ell=0}^{\infty} O(\beta^\ell n) = O(n)$. □

We now argue about space complexity. The expected total sizes of the diagrams are $O(n)$ at any time (this is a consequence of lemma 5.13). We augment each level ℓ of the Voronoi hierarchy with lists of active points, one for each cluster C with non-empty region at level ℓ. For each point c in such a list we store a list of all pure vertices, adjacent to its region $\text{hreg}_{F_i^{(\ell)}}(c)$. Each region has a link to one or two clusters in the level below. Therefore, the expected space taken by the hierarchy at any time is equal to the $O(n)$.

We remark that the two additional data structures (namely, the lists of active points for each cluster, and the list of pure vertices for each active point) need to be updated if and only if the region of the corresponding cluster or active point changes. Since the overall expected number of such feature changes is linear and modification of each data structure takes $O(\log n)$ time, the overall expected time needed for maintenance of these additional data structures during the construction algorithm is $O(n \log n)$. We also do some work on the P-mixed vertices of a critical cluster P at level $\ell+1$, but this happens only one time during the whole incremental construction. We have at most $O(n)$ point locations each of which takes time $O(\log^2 n)$. We also have at most m parametric searches which take time $O(\log^3 n)$ each. All these imply the following.

Theorem 5.2. *The Hausdorff Voronoi diagram of non-crossing clusters can be constructed in $O(n \log^3 n)$ expected time and $O(n)$ expected space, using the Voronoi hierarchy.*

6 Discussion and open problems

We have provided improved complexity algorithms for constructing the Hausdorff Voronoi diagram of a family of non-crossing clusters of points. In future work we will also investigate families of arbitrary point clusters (i.e., possibly crossing). The complexity of the diagram in this case can vary from linear to quadratic and therefore an output-sensitive algorithm is most desirable. The randomized incremental construction in this case will have to be augmented with the ability to handle disconnected Voronoi regions. There is still a gap in the complexity of constructing the Hausdorff Voronoi diagram between our best $O(n \log^2 n (\log\log n)^2)$ expected time algorithm and the trivial $\Omega(n \log n)$ time lower bound. An open problem is to close this gap. It is interesting that for the L_∞ metric, an $O(n \log n)$ time algorithm is known [25]. Another direction for research is the study the problem for clusters of other shapes, such as segments or convex polygons. Finally, it would be interesting to apply our incremental approach to the farthest polygon Voronoi diagram, as this could possibly improve on the best known $O(n \log^3 n)$ algorithm of [7].

Acknowledgment

Supported in part by the Swiss National Science Foundation grant 134355, under the auspices of the ESF EUROCORES program EuroGIGA/VORONOI.

References

[1] Manuel Abellanas, Gregorio Hernandez, Rolf Klein, Victor Neumann-Lara, and Jorge Urrutia. A combinatorial property of convex sets. *Discrete and Computational Geometry*, 17(3):307–318, 1997.

[2] Manuel Abellanas, Ferran Hurtado, Christian Icking, Rolf Klein, Elmar Langetepe, Lihong Ma, Belén Palop, and Vera Sacristán. The farthest color Voronoi diagram and related problems. In *Proceedings of the 17th European Workshop on Computational Geometry (EWCG)*, pages 113–116, 2001.

[3] Lars Arge, Gerth Stolting Brodal, and Loukas Georgiadis. Improved dynamic planar point location. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 305–314, 2006.

[4] Hanna Baumgarten, Hermann Jung, and Kurt Mehlhorn. Dynamic point location in general subdivisions. In *Proceedings of the 3rd annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 250–258, 1992.

[5] Mark de Berg, Otfried Cheong, Marc van Kreveld, and Mark Overmars. *Computational Geometry. Algorithms and Applications*. Springer-Verlag, 3rd edition, 2008.

[6] Jean-Daniel Boissonnat, Camille Wormser, and Mariette Yvinec. Curved Voronoi diagrams. In Jean-Daniel Boissonnat and Monique Teillaud, editors, *Effective Computational Geometry for Curves and Surfaces*, pages 67–116. Springer Berlin Heidelberg, 2006.

[7] Otfried Cheong, Hazel Everett, Marc Glisse, Joachim Gudmundsson, Samuel Hornus, Sylvain Lazard, Mira Lee, and Hyeon-Suk Na. Farthest-polygon Voronoi diagrams. *Computational Geometry*, 44(4):234–247, 2011.

[8] Kenneth Clarkson and Peter Shor. Applications of random sampling in computational geometry, II. *Discrete and Computational Geometry*, 4:387–421, 1989.

[9] Frank Dehne, Anil Maheshwari, and Ryan Taylor. A coarse grained parallel algorithm for Hausdorff Voronoi diagrams. In *Proceedings of the International Conference on Parallel Processing (ICPP)*, pages 497–504, 2006.

[10] Olivier Devillers. The Delaunay Hierarchy. *International Journal of Foundations of Computer Science*, 13:163–180, 2002.

[11] Herbert Edelsbrunner, Leonidas J. Guibas, and Micha Sharir. The upper envelope of piecewise linear functions: algorithms and applications. *Discrete and Computational Geometry*, 4:311–336, 1989.

[12] Herbert Edelsbrunner, Leonidas J. Guibas, and Jorge Stolfi. Optimal point location in a monotone subdivision. *SIAM Journal on Computing*, 15(2):317–340, 1986.

[13] Leonidas J. Guibas, Donald E. Knuth, and Micha Sharir. Randomized incremental construction of Delaunay and Voronoi diagrams. *Algorithmica*, 7:381–413, 1992.

[14] Daniel P. Huttenlocher, Klara Kedem, and Micha Sharir. The upper envelope of Voronoi surfaces and its applications. *Discrete and Computational Geometry*, 9:267–291, 1993.

[15] Menelaos Karavelas and Mariette Yvinec. The Voronoi diagram of convex objects in the plane. Technical report RR-5023, INRIA, 2003.

[16] David Kirkpatrick. Optimal search in planar subdivisions. *SIAM Journal on Computing*, 12(1):28–35, 1983.

[17] Rolf Klein. *Concrete and abstract Voronoi diagrams*, volume 400 of *Lecture Notes in Computer Science*. Springer, 1989.

[18] Rolf Klein, Kurt Mehlhorn, and Stefan Meiser. Randomized incremental construction of abstract Voronoi diagrams. *Computational Geometry*, 3(3):157–184, 1993.

[19] Nimrod Megiddo. Applying parallel computation algorithms in the design of serial algorithms. *Journal of the ACM*, 30(4):852–865, 1983.

[20] Ketan Mulmuley. *Computational Geometry: An introduction through randomized algorithms*. Prentice Hall, 1993.

[21] Evanthia Papadopoulou. Critical area computation for missing material defects in VLSI circuits. *IEEE Transactions on CAD of Integrated Circuits and Systems*, 20(5):583–597, 2001.

[22] Evanthia Papadopoulou. The Hausdorff Voronoi diagram of point clusters in the plane. *Algorithmica*, 40(2):63–82, 2004.

[23] Evanthia Papadopoulou. Net-aware critical area extraction for opens in VLSI circuits via higher-order Voronoi diagrams. *IEEE Transactions on CAD of Integrated Circuits and Systems*, 30(5):704–716, 2011.

[24] Evanthia Papadopoulou and D. T. Lee. The Hausdorff Voronoi diagram of polygonal objects: a divide and conquer approach. *International Journal of Computational Geometry and Applications*, 14(6):421–452, 2004.

[25] Evanthia Papadopoulou and Jinhui Xu. The L_∞ Hausdorff Voronoi diagram revisited. In *Proceedings of the 8th International Symposium on Voronoi Diagrams in Science and Engineering (ISVD)*, pages 67–74, 2011.

[26] William Pugh. Skip lists: a probabilistic alternative to balanced trees. *Commun. ACM*, 33(6):668–676, 1990.

[27] Neil Sarnak and Robert Endre Tarjan. Planar point location using persistent search trees. *Communications of the ACM*, 29(7):669–679, 1986.

[28] Micha Sharir. The Clarkson-Shor technique revisited and extended. *Combinatorics, Probability and Computing*, 12(2):191–201, 2003.