A Randomized Incremental Approach for the Hausdorff Voronoi Diagram of Non-crossing Clusters *

Panagiotis Cheilaris[1], Elena Khramtcova[1],
Stefan Langerman[2], and Evanthia Papadopoulou[1]

[1] Faculty of Informatics, Università della Svizzera italiana, Lugano, Switzerland
[2] Départment d'Informatique, Université Libre de Bruxelles, Brussels, Belgium

Abstract. In the Hausdorff Voronoi diagram of a set of *point-clusters* in the plane, the distance between a point t and a cluster P is measured as the maximum distance between t and any point in P while the diagram is defined in a nearest sense. This diagram finds direct applications in VLSI computer-aided design. In this paper, we consider "non-crossing" clusters, for which the combinatorial complexity of the diagram is linear in the total number n of points on the convex hulls of all clusters. We present a randomized incremental construction, based on point-location, to compute the diagram in expected $O(n \log^2 n)$ time and expected $O(n)$ space, which considerably improves previous results. Our technique efficiently handles 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.

1 Introduction

Given a set S of sites contained in some space, the *Voronoi region* of a site $s \in S$ is the geometric locus of points in the given 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 consider 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 is measured by the *farthest distance* $d_{\mathrm{f}}(t, P) = \max_{p \in P} d(t, p)$, where $d(\cdot, \cdot)$ is the Euclidean distance between two points. The farthest distance $d_{\mathrm{f}}(t, P)$ equals the *Hausdorff distance* between t and cluster P, hence the name of the diagram.

Our motivation for investigating the Hausdorff Voronoi diagram comes from VLSI circuit design, where this diagram can be used to efficiently estimate the *critical area* of a VLSI layout for various types of open faults [20, 21].

1.1 Previous Work

Let k be the number of clusters in the input family, and n be the total number of points on the convex hulls of all clusters. We denote by $\operatorname{conv} P$ the convex

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

hull of cluster P and by $\mathrm{CH}(P)$ the sequence of points of P on the boundary of the convex hull in counterclockwise order.

Definition 1. *Two clusters P and Q are called* 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, or equivalently convex hulls of P and Q are pseudodisks. See Fig. 1.*

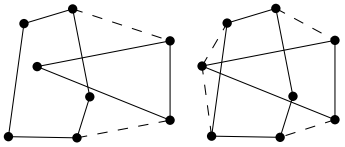

Fig. 1: Non-crossing and crossing clusters with supporting segments (dashed lines)

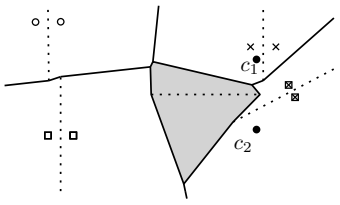

Fig. 2: HVD of five 2-point clusters; region of $C = \{c_1, c_2\}$ (gray)

The combinatorial complexity (size) of the Hausdorff Voronoi diagram is $O(n + m)$, where m is the number of supporting segments reflecting crossings between all pairs of crossing clusters, and this is tight [22]. In the worst case, m is $\Theta(n^2)$. If all clusters are *non-crossing* ($m = 0$) the diagram has linear size. There are *plane sweep* [20] and *divide and conquer* [22] algorithms for constructing the Hausdorff Voronoi diagram of arbitrary clusters. 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, for example, the minimum enclosing circle [22]. However, K can be $\omega(n)$ (superlinear) even in the case of non-crossing clusters. The Hausdorff Voronoi diagram is equivalent to an upper envelope of a family of lower envelopes of an arrangement of hyperplanes in (each envelope corresponds to a cluster) [13]. Edelsbrunner *et al.* give a construction algorithm of $O(n^2)$ time complexity.[3] Although the time complexity is optimal in the worst case, it remains quadratic even for non-crossing clusters, for which the size of the diagram is linear. A more recent parallel algorithm [10] constructs the Hausdorff Voronoi diagram of non-crossing clusters in $O(p^{-1} n \log^4 n)$ time with p processors, which implies a divide and conquer sequential algorithm of time complexity $O(n \log^4 n)$ and space complexity $O(n \log^2 n)$.

The Hausdorff Voronoi diagram of a family of non-crossing clusters is an instance of abstract Voronoi diagrams [16]. Using the randomized incremental framework of Klein *et al.* [17], it can be computed in expected $O(bn \log n)$ time, where b is the time it takes to construct the bisector between two clusters [1]. If there are clusters of linear size, then b can be $\Theta(n)$. The framework was successfully applied to compute the Voronoi diagram of disjoint polygons [18]

[3] The reported $O(n^2 \alpha(n))$ time complexity (where $\alpha(n)$ is the inverse Ackermann function) improves to $O(n^2)$ due to the $O(n^2)$ bound on the size of the diagram.

in $O(k \log n)$ time, where k is the number of the sites, and n is their total size. It is not easy, however, to apply a similar approach to the Hausdorff Voronoi diagram because of a fundamental difference between the farthest and the nearest distance from a point to a convex polygon [12].

The Hausdorff Voronoi diagram is a min-max diagram type of diagram, where every point t in the plane lies in the region of the *closest* cluster with respect to the *farthest* distance. The "dual" max-min diagram is the *farthest color* Voronoi diagram [2, 14]. For disjoint simple polygons, the farthest color Voronoi diagram can be constructed in $O(n \log^3 n)$ time where n is the total size of the sites [8].

1.2 Our Contribution

In this paper we give a randomized incremental algorithm to compute the Hausdorff Voronoi diagram of a family of k non-crossing clusters, based on point location. Clusters are inserted in random order one by one, while the diagram computed so far is maintained in a dynamic data structure, where generalized point location queries can be answered efficiently. To insert a cluster, a representative point in the new Voronoi region of this cluster is first identified and located, and then the new region is traced while the data structure is updated [6, 11, 15].

In case of the Hausdorff Voronoi diagram, a major technical challenge is to quickly identify a representative point that lies in the new Voronoi region. This is difficult because: (a) the region of the new cluster might not contain any of its points, (b) the region of the new cluster might be empty, and (c) sites have non-constant size and thus the computation of a bisector or answering an *in-circle test* require non-constant time. Furthermore, the addition of a new cluster may make an existing region empty.

The dynamic data structure that we use is a variant of the *Voronoi hierarchy* [15], which in turn is inspired by the Delaunay hierarchy [11], and which we augment with the ability to efficiently handle the difficulties (a) to (c). We also exploit a technique by Aronov *et al.* [4] to efficiently query the static farthest Voronoi diagram of a cluster. The expected running time of our algorithm is $O(n \log n \log k)$ and the expected space complexity is $O(n)$. The augmentation of the Voronoi hierarchy introduced in this paper may be of interest for incremental constructions of other non-standard types of generalized Voronoi diagrams. Our algorithm can also be implemented in *deterministic* $O(n)$ space and $O(n \log^2 n (\log \log n)^2)$ expected running time, using the dynamic point location data structure by Baumgarten *et al.* [5], while applying a simplified type of *parametric search* similarly to Cheong *et al.* [8].

Due to the lack of space some proofs and technical details are omitted in this version of the paper. Please refer to the online full version [7].

2 Preliminaries

Throughout this paper, we consider a family $F = \{C_1, \ldots, C_k\}$ of non-crossing clusters of points. We assume that no two clusters have a common point, and no four points lie on the same circle.

For a point $c \in C$, the *farthest Voronoi region* of c is $\mathrm{freg}_C(c) = \{p \mid \forall c' \in C \setminus \{c\} \colon d(p,c) > d(p,c')\}$. The farthest Voronoi diagram of C is denoted as $\mathrm{FVD}(C)$ and its graph structure as $\mathcal{T}(C)$. If $|C| > 1$, $\mathcal{T}(C)$ is a tree defined as $\mathbb{R}^2 \setminus \bigcup_{c \in C} \mathrm{freg}_C(c)$, and $\mathcal{T}(C) = c$, if $C = \{c\}$. A point at infinity along an arbitrary unbounded edge of $\mathcal{T}(C)$ is treated as the root of $\mathcal{T}(C)$, denoted as $\mathrm{root}(C)$.

For a cluster $C \in F$, the *Hausdorff Voronoi region* of C is

$$\mathrm{hreg}_F(C) = \{p \mid \forall C' \in F \setminus \{C\} \colon d_{\mathrm{f}}(p,C) < d_{\mathrm{f}}(p,C')\}.$$

For a point $c \in C$, $\mathrm{hreg}_F(c) = \mathrm{hreg}_F(C) \cap \mathrm{freg}_C(c)$. The closure of $\mathrm{freg}_C(c)$, $\mathrm{hreg}_F(C)$, and $\mathrm{hreg}_F(c)$ is denoted by $\overline{\mathrm{freg}}_C(c)$, $\overline{\mathrm{hreg}}_F(C)$, and $\overline{\mathrm{hreg}}_F(c)$, respectively. When there is no ambiguity on the set under consideration, we omit the subscript from the above notation. The partitioning of the plane into non-empty Hausdorff Voronoi regions, together with their bounding edges and vertices, is called the *Hausdorff Voronoi diagram* of F, and it is denoted as $\mathrm{HVD}(F)$. Below we review some useful definitions and properties of the Hausdorff Voronoi diagram, which appeared in previous work [22].

The Hausdorff Voronoi diagram is *monotone*, that is, a region of the diagram can only shrink with the insertion of a new cluster. The structure of the Hausdorff Voronoi region of a point $c \in C$ is shown in Fig. 3. Its boundary consists of two chains: (1) the *farthest boundary* that belongs to $\mathcal{T}(C)$ and is internal to $\mathrm{hreg}(C)$, $(\mathrm{bd}\,\mathrm{hreg}(c) \cap \mathrm{bd}\,\mathrm{freg}(c))$; (2) the *Hausdorff boundary* $(\mathrm{bd}\,\mathrm{hreg}(c) \cap \mathrm{bd}\,\mathrm{hreg}(C))$. Neither chain can be empty, if $\mathrm{hreg}(C) \neq \emptyset$ and $|C| > 1$. There are three types of vertices on the boundary of $\mathrm{hreg}(c)$: (1) Standard Voronoi vertices that are equidistant from C and two other clusters, referred in this paper as *pure* vertices. Pure vertices appear on the Hausdorff boundary of $\mathrm{hreg}(c)$. (2) *Mixed* vertices that are equidistant to three points of two clusters (C and another cluster). The mixed vertices which 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 $\mathrm{hreg}(c)$ and they delimit both the farthest boundary of c and the Hausdorff boundary of c. (3) Vertices of $\mathcal{T}(C)$ on the farthest boundary of c.

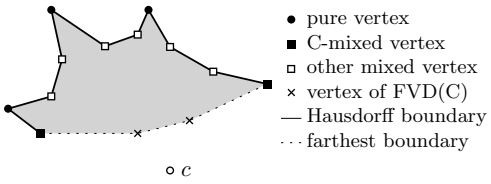

Fig. 3: Features of the Hausdorff Voronoi region of a point $c \in C$

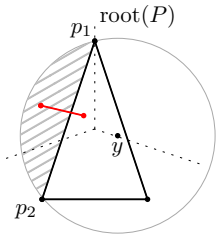

Fig. 4: The 2-point cluster Q (red), forward limiting w.r.t. the 3-point cluster P (black) with P-circle $\mathcal{K}_y$; portion $\mathcal{K}_y^f$ (shaded)

A line segment $\overline{c_1c_2}$ is a *chord* of cluster C if $c_1, c_2 \in \mathrm{CH}(C)$ and $c_1 \neq c_2$. In Fig. 4, $\overline{p_1p_2}$ is a chord of cluster P.

Definition 2 (C-circle $\mathcal{K}_y$; $\mathcal{K}_y^f$, $\mathcal{K}_y^r$ [22]). *Let uv be an edge of $\mathcal{T}(C)$ bisecting a chord c_1c_2 of $C \in F$. A circle centered at $y \in uv$ of radius $d(y, c_1) = d_{\mathrm{f}}(y, C)$ is called the* C-circle *of y and is denoted as $\mathcal{K}_y$. The chord c_1c_2 partitions $\mathcal{K}_y$ in two parts: $\mathcal{K}_y^f$ and $\mathcal{K}_y^r$, where $\mathcal{K}_y^f$ is the part that encloses the two points of C that define* $\mathrm{root}(C)$. *In case y and* $\mathrm{root}(C)$ *are on the same edge of $\mathcal{T}(C)$, $\mathcal{K}_y^f$ is the portion of $\mathcal{K}_y$ that is enclosed in the halfplane bounded by c_1c_2 which does not contain* $\mathrm{root}(C)$.

Definition 3 (Rear/forward limiting cluster [22]). *A cluster $P \in F \setminus \{C\}$ is* rear limiting *with respect to C, if there is a C-circle $\mathcal{K}_y$ such that P is enclosed in $\mathcal{K}_y^r \cup \mathrm{conv}\, C$. Similarly, P is* forward limiting *with respect to C, if there is a C-circle K_y such that P is enclosed in $\mathcal{K}_y^f \cup \mathrm{conv}\, C$. See Fig. 4.*

Properties. (They can be directly derived from Lemma 2, Properties 2, 3 [22].)

1. If $\mathrm{hreg}(C) \neq \emptyset$, then $\mathrm{hreg}(C) \cap \mathcal{T}(C)$ consists of exactly one non-empty connected component.
2. Consider a point v of $\mathcal{T}(P)$, such that $v \notin \mathrm{hreg}(P)$. Let Q be a cluster, which is closer to v than P. Then, only one of the subtrees of $\mathcal{T}(P)$ rooted at v, might contain points which are closer to P than to Q.
3. Let uv be an edge of $\mathcal{T}(P)$. If both u and v are closer to Q than to P then $\mathrm{hreg}_F(P)$ cannot intersect uv.
4. Region $\mathrm{hreg}_F(P) = \emptyset$, if and only if there is a cluster $Q \subset \mathrm{conv}\, P$, or there is a pair of clusters $\{Q, R\}$ such that Q is rear limiting and R is forward limiting with the same P-circle. Pair $\{Q, R\}$ is called a *killing pair* for P.

3 A Randomized Incremental Algorithm

Let $C_1, \ldots, C_k$ be a random permutation of the clusters in family F, and let $F_i = \{C_1, \ldots, C_i\}$ for $1 \leq i \leq k$. The algorithm iteratively constructs $\mathrm{HVD}(F_1), \ldots, \mathrm{HVD}(F_k) = \mathrm{HVD}(F)$. The cluster C_i is inserted in $\mathrm{HVD}(F_{i-1})$ as follows:

1. Identify a point t that is closer to C_i than to any cluster in F_{i-1} (i.e., $t \in \mathrm{hreg}_{F_i}(C_i)$) or determine that no such point exists (i.e., $\mathrm{hreg}_{F_i}(C_i) = \emptyset$).
2. If t exists, grow $\mathrm{hreg}_{F_i}(C_i)$ starting from t and update $\mathrm{HVD}(F_{i-1})$ to derive $\mathrm{HVD}(F_i)$; otherwise, $\mathrm{HVD}(F_i) = \mathrm{HVD}(F_{i-1})$.

The main challenge is to perform Step 1 efficiently. Step 2 can be performed in linear time [22]. Throughout this section, we skip the subscript F_i and let $\mathrm{hreg}(C_i)$ stand for $\mathrm{hreg}_{F_i}(C_i)$.

To identify a representative point t in $\mathrm{hreg}(C_i)$ (Step 1) it is enough to search along $\mathcal{T}(C_i)$, by Property 1. However, $\mathrm{hreg}(C_i) \cap \mathcal{T}(C_i)$ might not contain a vertex of $\mathcal{T}(C_i)$, see e.g., the gray region in Fig. 2. In this case, $\mathrm{hreg}(C_i)$ is either empty, or intersects exactly one edge of $\mathcal{T}(C_i)$, which is called a *candidate edge*.

Definition 4. *Let uv be an edge of $\mathcal{T}(C_i)$. Let the clusters $Q^u, Q^v \in F_{i-1}$ be the clusters closest to u and v respectively. We call uv a* candidate *edge if $Q^u \neq Q^v$ and uv satisfies the following predicate:*
$\text{cand}(uv) = d_f(u, Q^u) < d_f(u, C_i) < d_f(u, Q^v) \wedge d_f(v, Q^v) < d_f(v, C_i) < d_f(v, Q^u)$.

By Properties 2 and 3 we derive the following.

Lemma 1. *Suppose* $\text{hreg}(C_i) \cap \mathcal{T}(C_i)$ *does not contain any vertex of* $\mathcal{T}(C_i)$. *Then at most one edge uv of $\mathcal{T}(C_i)$ can be a candidate edge, in which case* $\text{hreg}(C_i) \cap \mathcal{T}(C_i) \subset uv$. *Otherwise* $\text{hreg}(C_i) = \emptyset$.

A high-level description of Step 1 is as follows: We traverse $\mathcal{T}(C_i)$ starting at $\text{root}(C_i)$, checking its vertices and pruning if possible appropriate subtrees according to Property 3. In this process we either determine t as a vertex of $\mathcal{T}(C_i)$, or we determine a candidate edge uv, or $\text{hreg}(C_i) = \emptyset$. Pseudocode is given as Procedure 1 below, which should be run with $u = \text{root}(C_i)$.

In more detail, to check if a vertex w suits as t, determine the cluster $Q^w \in F_{i-1}$, which is nearest to w by point location in $\text{HVD}(F_{i-1})$. If $d_f(w, C_i) < d_f(w, Q^w)$, then $t = w$. To compute $d_f(w, P)$ for a cluster P, do point location in $\text{FVD}(P)$. If Procedure 1 identifies a candidate edge, the representative point t is determined by performing *parametric point location* along the candidate edge in $\text{HVD}(F_{i-1})$.

Procedure 1 Tracing the subtree of $\mathcal{T}(C_i)$ rooted at u (within Step 1)

Require: $d_f(u, C_i) > d_f(u, Q^u)$.
  Let v and w be children of u.
  Locate v and w in $\text{HVD}(F_{i-1})$ to obtain Q^v and Q^w respectively.
  if $d_f(v, C_i) < d_f(v, Q^v)$ **or** $d_f(w, C_i) < d_f(w, Q^w)$ **then return** v or w respectively.
  if either uv or uw is a candidate edge **then**
    return the uv or uw respectively.
  if $d_f(v, C_i) < d_f(v, Q^u)$ **then** ▷ Otherwise, prune the subtree of w
    Set $u = w$ and recurse.
  if $d_f(w, C_i) < d_f(w, Q^u)$ **then** ▷ Otherwise, prune the subtree of v
    Set $u = v$ and recurse.

Definition 5 (Parametric point location). *Given* $\text{HVD}(F_{i-1})$ *and a candidate edge $uv \subset \mathcal{T}(C_i)$ determine the cluster $P_j \in F_{i-1}$ and the point $t \in uv$ such that $d_f(t, C_i) = d_f(t, P_j) = \min_{P \in F_{i-1}} d_f(t, P)$. If such point p does not exist, return* nil.

Parametric point location in the Hausdorff Voronoi diagram is performed using the data structure that stores the diagram. Its performance determines the time complexity of our algorithm. In Sections 4 and 5, we describe the data structures and the algorithms used to answer the necessary queries.

4 Separator Decomposition

In this section we describe a data structure to efficiently perform point location and answer so-called *segment queries* in a *tree-type* of planar subdivision such as a farthest Voronoi diagram.

It is well-known [19] that any tree with h vertices has a vertex called *centroid*, removal of which decomposes the tree into subtrees of at most $h/2$ vertices each. The centroid can be found in $O(h)$ time [19]. Thus, the farthest Voronoi diagram of a cluster P can be organized as a balanced tree, whose nodes correspond to vertices of the diagram. This representation is called the *separator decomposition*, it is denoted as $\mathrm{SD}(P)$, and can be built as follows:

- Find a centroid c of $\mathcal{T}(P)$. Create a node for c and assign it as the root node.
- Remove c from $\mathcal{T}(P)$. Recursively build the trees for the remaining three connected components, and link them as subtrees of the root.

Point location in $\mathrm{SD}(P)$ for a query point q, is performed as follows. Starting from the root of $\mathrm{SD}(P)$, perform a constant-time test of the query point q against a node of $\mathrm{SD}(P)$, to decide in which of the node's subtrees to continue. When a leaf of $\mathrm{SD}(P)$ is reached, choose p among the owners of the three regions that are adjacent to the corresponding vertex of $\mathrm{FVD}(P)$. The test of q against a node α of $\mathrm{SD}(P)$ is due to Aronov *et al.* [4]. In more detail, let the node α correspond to a vertex w of $\mathrm{FVD}(P)$. Let the points $p_1, p_2, p_3 \in P$ be the owners of the three regions of $\mathrm{FVD}(P)$, incident to w. Consider the rays $r_i, i = 1, 2, 3$ with origin at w and direction $\overrightarrow{p_i w}$ respectively. Each ray r_i lies entirely inside $\mathrm{freg}_P(p_i)$, and thus r_1, r_2 and r_3 subdivide the plane into three sectors with exactly one connected component of $\mathcal{T}(P) \setminus \{w\}$ in each sector. Choose the sector that contains q, and pick the corresponding subtree of α.

A *segment query* in a farthest Voronoi diagram is as follows. Let $C, P \in F$. Given $\mathrm{FVD}(P)$ and a segment $uv \subset \mathcal{T}(C)$ such that $d_{\mathrm{f}}(u, C) < d_{\mathrm{f}}(u, P)$ and $d_{\mathrm{f}}(v, C) > d_{\mathrm{f}}(v, P)$, find the point $x \in uv$, that is equidistant from both C and P ($d_{\mathrm{f}}(x, C) = d_{\mathrm{f}}(x, P)$).

If $\mathrm{FVD}(P)$ is represented as a separator decomposition, the segment query can be performed efficiently similarly to a point location query with the difference that we test a segment against a node of $\mathrm{SD}(P)$. In particular, consider a node of $\mathrm{SD}(P)$ corresponding to a vertex w of $\mathrm{FVD}(P)$. Let rays $r_i, i = 1, 2, 3$, be defined as above. Consider the (at most two) intersection points of uv with the rays r_i. If any of these points is equidistant to C and P, return it. Otherwise, since P and C are non-crossing, there is exactly one subsegment $u'v' \subset uv$ such that $d_{\mathrm{f}}(u', C) < d_{\mathrm{f}}(u', P)$ and $d_{\mathrm{f}}(v', C) > d_{\mathrm{f}}(v', P)$, where u', v' can be any of u, v, or the intersection points. The subsegment $u'v'$ can be computed in constant time, together with one of the three sectors where $u'v'$ is contained.

If we reached a leaf of $\mathrm{SD}(P)$, we are left with a single edge e of $\mathcal{T}(P)$. Suppose e bisects the chord p_1p_2 of P, and the current $u'v'$ bisects the chord c_1c_2 of C. Then, return as point x the center of the circle passing through p, c_1, c_2, where p is the point among p_1, p_2 farthest from x.

Lemma 2. *The separator decomposition* $\mathrm{SD}(P)$ *of a cluster* $P \in F$ *can be built in* $O(n_p \log n_p)$ *time, where* n_p *is the number of vertices of* $\mathrm{FVD}(P)$*. Both the point location and the segment query in* $\mathrm{SD}(P)$ *require* $O(\log n_p)$ *time.*

5 Voronoi Hierarchy for the Hausdorff Voronoi Diagram

Consider a set S of k sites. The *Voronoi hierarchy* of S is a sequence of *levels* $S = S^{(0)} \supseteq \ldots \supseteq S^{(h)}$. For $\ell \in \{1, \ldots, h\}$, level $S^{(\ell)}$ is a random sample of $S^{(\ell-1)}$ according to a Bernoulli distribution with parameter $\beta \in (0, 1)$. For each level $S^{(\ell)}$ the data structure stores the Voronoi diagram of $S^{(\ell)}$. The Voronoi hierarchy is inspired by the Delaunay hierarchy given by Devillers [11].

In the Hausdorff Voronoi diagram sites are clusters of non-constant size each. We first adapt some known properties of the hierarchy to be valid in such an environment. Then, we consider several enhancements of the hierarchy to handle efficiently the Hausdorff Voronoi diagram and its queries, such as point location through walks, dynamic updates, including the handling of empty Voronoi regions, and parametric point location along a segment.

Lemma 3. *Let the underlying Voronoi diagram have size* $O(n)$*, where* n *is the total size of the sites. Then for any set* S *of* k *sites of total size* n*, the Voronoi hierarchy of* S *has* $O(n)$ *expected size and* $O(\log k)$ *expected number of levels.*

To perform *point location* in the Voronoi hierarchy for a query point q, we start at level h, and for each level ℓ, we determine the site $s^\ell \in S^{(\ell)}$ that is closest to q, by performing a *walk*. Each step of the walk moves from a site $s \in S^{(\ell)}$ to a neighbor of s, such that the distance to q is reduced. A walk at level $\ell - 1$ starts from s^ℓ. The answer to the query is s^0.

Lemma 4. *Let* $s_0^\ell, \ldots, s_r^\ell = s^\ell$ *be the sequence of sites visited at level* ℓ *during the point location of a query point* q*. Assuming that* $d_{\mathrm{f}}(q, s_i^\ell) < d(q, s_{i-1}^\ell)$*, for* $i \in \{1, \ldots, r\}$*, and either* $s^{\ell+1} = s_0^\ell$*, or* $d_{\mathrm{f}}(q, s_0^\ell) < d_{\mathrm{f}}(q, s^{\ell+1})$*, the expectation of the length* r *of the walk at level* ℓ *is constant.*

In the original Voronoi hierarchy for a set of disjoint convex objects [15], one step of the walk to determine the correct neighboring site consists of a binary search among the neighbors of the site. For a Hausdorff Voronoi diagram, however, there is no natural ordering for the set of neighbors of a site. In addition, the subset of points in a cluster that contribute to the diagram reduces over time.

A single step of the walk for the Hausdorff Voronoi diagram. Consider point location in the Voronoi hierarchy for a family $\mathcal{F}$ of non-crossing clusters and a query point q. Let $C \in \mathcal{F}^{(\ell)}$ be the current cluster being considered at level ℓ. We need to determine a cluster Q at level ℓ whose region neighbors the region of C and whose distance from q gets reduced. Let $\hat{C} \subset C$ denote the set of all *active* points $c \in C$ that contribute a face to $\mathrm{hreg}_{\mathcal{F}^{(\ell)}}(C)$ at the current level ℓ ($\mathrm{hreg}_{\mathcal{F}^{(\ell)}}(c) \neq \emptyset$). Let $\mathrm{hreg}_{\mathcal{F}}^{(\ell)}(\cdot)$ denote $\mathrm{hreg}_{\mathcal{F}^{(\ell)}}(\cdot)$.

The cluster Q is determined as follows. Let $c \in \hat{C}$ be the active point that is farthest from query point q ($q \in \overline{\mathrm{freg}}_{\hat{C}}(c)$). To determine point c it is enough to draw the tangents from q to $\mathrm{CH}(\hat{C})$. Let $v_1, \ldots, v_j$ be the pure vertices in $\mathrm{hreg}^{(\ell)}_{\mathcal{F}}(c)$ (see Fig. 5) in counterclockwise order, and let $Q^0, \ldots, Q^j, Q^{j+1}$ be their respective adjacent clusters. The rays $\overrightarrow{cv_1}, \ldots, \overrightarrow{cv_j}$ partition $\overline{\mathrm{freg}}_{\hat{C}}(c)$ into $j+1$ unbounded regions. The walk should move from C to Q^i such that the ray $\overrightarrow{cq}$ immediately follows $\overrightarrow{cv_i}$ or immediately precedes $\overrightarrow{cv_{i+1}}$. For example, in Fig. 5, $c \in \hat{C}^{(\ell)}$ is the farthest active point from q ($d_{\mathrm{f}}(q, \hat{C}^{(\ell)}) = d(q, c)$). Region $\mathrm{freg}_{\hat{C}}(c)$ is shown gray and its boundary is drawn bold. The step in the walk should move from C to $Q = Q^2$. We organize $\hat{C}$ as a sorted list of its points and for each point $c \in \hat{C}$ we maintain a sorted list of all Voronoi vertices adjacent to $\mathrm{hreg}^{(\ell)}_{\mathcal{F}}(c)$. It can be shown that $d_{\mathrm{f}}(q, \hat{Q}) \le d_{\mathrm{f}}(q, \hat{C})$, thus, the above procedure is correct. Note that $d_{\mathrm{f}}(q, Q)$ may be greater than $d_{\mathrm{f}}(q, C)$ because $d_{\mathrm{f}}(q, \hat{C})$ may be different from $d_{\mathrm{f}}(q, C)$ if $q \notin \mathrm{hreg}(C)$.

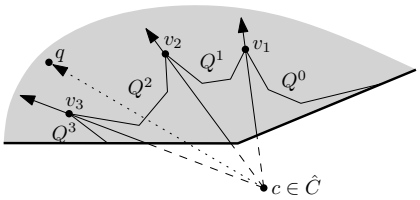

Fig. 5: The step of a walk from the cluster C

Parametric point location in the Voronoi hierarchy. We are given $\mathrm{HVD}(F_{i-1})$, stored as a Voronoi hierarchy, and the candidate edge $uv \in \mathcal{T}(C_i)$. For each level ℓ of the Voronoi hierarchy, starting from the last level h, we search for the cluster $Q^\ell \in F^{(\ell)}_{i-1}$ and a point $u^\ell \in uv$ such that $u^\ell \in \mathrm{hreg}_{F^{(\ell)}_{i-1}}(Q^\ell)$ and $d_{\mathrm{f}}(u^\ell, C_i) = d_{\mathrm{f}}(u^\ell, Q^\ell)$. If at some level there is no such point, return *nil*. Else return the cluster C^0 and the point u^0 determined at level 0.

In more detail, suppose that $u^{\ell+1}$ and $Q^{\ell+1}$ have been computed, for some $\ell \in \{0, \ldots, h-1\}$. To compute u^ℓ and Q^ℓ, we determine a sequence $u^{\ell+1} = a_0, a_1, \ldots, a_r = u^\ell$ of points on uv. Let Q^{a_j} be the cluster in $F^{(\ell)}_{i-1}$ nearest to a_j. It is determined by a walk at level ℓ starting with $Q^{a_{j-1}}$. Then point a_{j+1} is the point on uv, equidistant from C_i and Q^{a_j} ($d_{\mathrm{f}}(a_{j+1}, C_i) = d_{\mathrm{f}}(a_{j+1}, Q^{a_j})$). If a_j is equidistant from C_i and Q^{a_j}, we are done at level ℓ; continue to level $\ell - 1$ with $u^\ell = a_j$ and $Q^\ell = Q^{a_j}$. Else, if $d_{\mathrm{f}}(v, Q^{a_j}) > d_{\mathrm{f}}(v, C_i)$, perform a segment query to determine a_{j+1}. Otherwise, report that a point t does not exist.

Lemma 5. *The expected number of visits of clusters at level ℓ during the parametric point location is $O(1)$.*

Handling the empty regions. After inserting C_i at level 0 of the hierarchy for $\mathrm{HVD}(F_{i-1})$, we insert C_i into the series of higher levels. When C_i is inserted at a given level, however, a region of another cluster may become empty.

We call a cluster P *critical at level* ℓ if $\mathrm{hreg}^{(\ell-1)}_{F_{i-1}}(P) \neq \emptyset$, $\mathrm{hreg}^{(\ell-1)}_{F_i}(P) = \emptyset$, and $\mathrm{hreg}^{(\ell)}_{F_i}(P) \neq \emptyset$. Such a cluster P becomes an obstacle to correct point location in the Voronoi hierarchy for $\mathrm{HVD}(F_i)$. Indeed, if a query point lies in $\mathrm{hreg}^{(\ell)}_{F_i}(P)$, we do not know where to continue the point location at level $\ell - 1$.

To fix the problem, P can be deleted from all levels, but this is computationally expensive. Instead, we link P to at most two other clusters $Q, R \in F_i^{(\ell-1)}$, such that every point $q \in \mathbb{R}^2$ is closer to either Q or to R than to P. Property 4 guarantees that such a cluster or two clusters exist (a cluster contained in conv P or a killing pair for P, respectively).

We now describe how to find a killing pair for P. While inserting C_i at level $\ell - 1$, we store all (deleted) P-mixed vertices of $\mathrm{hreg}^{(\ell-1)}_{F_{i-1}}(P)$ in a list V. At level ℓ, for each P-mixed vertex v of $\mathrm{hreg}^{(\ell)}_{F_i}(P)$, we check if v is closer to C_i or to P. If $d_{\mathrm{f}}(v, C_i) \geq d_{\mathrm{f}}(v, P)$, let c be the point in C_i for which $d_{\mathrm{f}}(v, C_i) = d(v, c)$. Note that $c \notin \mathrm{conv}\, P$, which will be useful. The linking is performed as follows:

- If all P-mixed vertices of $\mathrm{hreg}^{(\ell)}_{F_{i-1}}(P)$ are closer to C_i than to P, link only to C_i. (This happens only if $C_i \notin F^{(\ell)}$.)
- Else find the cluster $K \in F^{(\ell-1)}$ such that $\{K, C_i\}$ is a killing pair for P. If $C_i \in F^{(\ell)}$, link only to cluster K. Otherwise, link to both K and C_i.

What remains is to determine cluster K. To this aim, we use list V and point c. Each vertex $u \in V$ is equidistant from two points $p_1, p_2 \in P$, and one point $q \in Q$, for some $Q \in F^{(\ell-1)}$. We simply check whether c and q are on different sides of the chord $\overline{p_1 p_2}$. If they are, then we set $K = Q$ and we stop. By Property 4, $\{K, C_i\}$ is a killing pair for P, and thus the linking is correct.

We summarize the result on the Voronoi hierarchy in the following lemma, which is easily derived from Lemmas 3 to 5 and the discussion in Section 5.

Theorem 1. *The Voronoi hierarchy for the Hausdorff Voronoi diagram of a family of k clusters of total complexity n has expected size $O(n)$. Both the point location query and the parametric point location take expected $O(\log n \log k)$ time. Insertion of a cluster takes amortized $O((N/k) \log n)$ time, where N is the total number of update operations in all levels during the insertion of all k clusters.*

6 Complexity Analysis

The running time of our algorithm depends on the number of update operations (insertions and deletions) during the construction of the diagram. Using the Clarkson-Shor technique [9], we prove that the expectation of this number is linear, when clusters are inserted in random order. Note that in contrast to the standard probabilistic argument, our proof does not assume sites (clusters) to have constant size.

Theorem 2. *The expected number of update operations is* $O(n)$.

Theorem 2 can be easily extended to all levels of the Voronoi hierarchy. The total time for the construction of the separator decomposition for all clusters is $O(n \log n)$ (see Lemma 2). For each cluster $C \in F$, we perform $O(|C|)$ point location queries and at most one parametric point location in Voronoi hierarchy. By Lemma 2 and Theorems 1 and 2, we conclude.

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

Deterministic $O(n)$ space could be achieved by using a dynamic point location data structure for a planar subdivision [3, 5]. On this data structure, the parametric point location can be performed as a simplified form of the parametric search, as described by Cheong *et al.* [8]. The time complexity of such a query is t_q^2, where t_q is the time complexity of point location in the chosen data structure. In particular, the data structure by Baumgarten *et al.* [5] has $t_q \in O(\log n \log \log n)$, which leads to the construction of the Hausdorff Voronoi diagram with expected running time $O(n \log^2 n (\log \log n)^2)$ and deterministic space $O(n)$.

7 Discussion and Open Problems

We have provided improved complexity algorithms for constructing the Hausdorff Voronoi diagram of a family of non-crossing point clusters based on randomized incremental construction and point location. There is still a gap in the complexity of constructing the Hausdorff Voronoi diagram between our best $O(n \log^2 n)$ expected time algorithm and the well-known $\Omega(n \log n)$ time lower bound. An open problem is to close or reduce this gap. It is interesting that in the L_∞ metric, a simple $O(n \log n)$-time $O(n)$-space algorithm, based on plane sweep, is known [23]. In parallel, we are considering the application of the randomized incremental construction paradigm through history graphs. In future research we plan to consider families of arbitrary point clusters that may be crossing. In this case, the size of the diagram can vary from linear to quadratic, and therefore, an output-sensitive algorithm is most desirable. Another direction for research is to study the problem for clusters of segments, clusters of convex polygons or other shapes, rather than clusters of points.

References

1. Abellanas, M., Hernandez, G., Klein, R., Neumann-Lara, V., Urrutia, J.: A combinatorial property of convex sets. Discrete Comput. Geom. 17(3), 307–318 (1997)
2. Abellanas, M., Hurtado, F., Icking, C., Klein, R., Langetepe, E., Ma, L., Palop, B., Sacristán, V.: The farthest color Voronoi diagram and related problems. In: 17th Eur. Workshop on Comput. Geom. (EWCG). pp. 113–116 (2001)
3. Arge, L., Brodal, G.S., Georgiadis, L.: Improved dynamic planar point location. In: 47th Ann. IEEE Symp. Found. Comput. Sci. (FOCS). pp. 305–314 (2006)

4. Aronov, B., Bose, P., Demaine, E.D., Gudmundsson, J., Iacono, J., Langerman, S., Smid, M.: Data structures for halfplane proximity queries and incremental Voronoi diagrams. In: LATIN 2006: Theor. Inf. pp. 80–92 (2006)
5. Baumgarten, H., Jung, H., Mehlhorn, K.: Dynamic point location in general subdivisions. J. Algorithm. 17(3), 342–380 (1994)
6. Boissonnat, J.D., Wormser, C., Yvinec, M.: Curved Voronoi diagrams. In: Boissonnat, J.D., Teillaud, M. (eds.) Effective Computational Geometry for Curves and Surfaces, pp. 67–116. Springer Berlin Heidelberg (2006)
7. Cheilaris, P., Khramtcova, E., Papadopoulou, E.: Randomized incremental construction of the hausdorff voronoi diagram of non-crossing clusters. CoRR abs/1306.5838 (2013)
8. Cheong, O., Everett, H., Glisse, M., Gudmundsson, J., Hornus, S., Lazard, S., Lee, M., Na, H.S.: Farthest-polygon Voronoi diagrams. Comput. Geom. 44(4), 234–247 (2011)
9. Clarkson, K., Shor, P.: Applications of random sampling in computational geometry, II. Discrete Comput. Geom. 4, 387–421 (1989)
10. Dehne, F., Maheshwari, A., Taylor, R.: A coarse grained parallel algorithm for Hausdorff Voronoi diagrams. In: 35th Int. Conf. on Parallel Processing (ICPP). pp. 497–504 (2006)
11. Devillers, O.: The Delaunay Hierarchy. Int. J. Found. Comput. S. 13, 163–180 (2002)
12. Edelsbrunner, H.: Computing the extreme distances between two convex polygons. J. Algorithm. 6(2), 213–224 (1985)
13. Edelsbrunner, H., Guibas, L.J., Sharir, M.: The upper envelope of piecewise linear functions: algorithms and applications. Discrete Comput. Geom. 4, 311–336 (1989)
14. Huttenlocher, D.P., Kedem, K., Sharir, M.: The upper envelope of Voronoi surfaces and its applications. Discrete Comput. Geom. 9, 267–291 (1993)
15. Karavelas, M., Yvinec, M.: The Voronoi diagram of convex objects in the plane. Technical report RR-5023, INRIA (2003)
16. Klein, R.: Concrete and abstract Voronoi diagrams, Lecture Notes in Computer Science, vol. 400. Springer (1989)
17. Klein, R., Mehlhorn, K., Meiser, S.: Randomized incremental construction of abstract Voronoi diagrams. Comput. Geom. 3(3), 157–184 (1993)
18. McAllister, M., Kirkpatrick, D., Snoeyink, J.: A compact piecewise-linear Voronoi diagram for convex sites in the plane. Discrete Comput. Geom. 15(1), 73–105 (1996)
19. Megiddo, N., Tamir, A., Zemel, E., Chandrasekaran, R.: An $O(n \log^2 n)$ algorithm for the kth longest path in a tree with applications to location problems. SIAM J. Comput. 10(2), 328–337 (1981)
20. Papadopoulou, E.: The Hausdorff Voronoi diagram of point clusters in the plane. Algorithmica 40(2), 63–82 (2004)
21. Papadopoulou, E.: Net-aware critical area extraction for opens in VLSI circuits via higher-order Voronoi diagrams. IEEE T. Comput. Aid D. 30(5), 704–716 (2011)
22. Papadopoulou, E., Lee, D.T.: The Hausdorff Voronoi diagram of polygonal objects: a divide and conquer approach. Int. J. Comput. Geom. Ap. 14(6), 421–452 (2004)
23. Papadopoulou, E., Xu, J.: The L_∞ Hausdorff Voronoi diagram revisited. In: 8th Int. Symp. on Voronoi Diagr. in Sci. and Eng. (ISVD). pp. 67–74 (2011)