The L_∞ Hausdorff Voronoi diagram revisited

Evanthia Papadopoulou*
Faculty of Informatics
University of Lugano, Switzerland
Email: evanthia.papadopoulou@usi.ch

Jinhui Xu
Department of Computer Science and Engineering
State University of New York at Buffalo
Email: jinhui@buffalo.edu

***Abstract*—We revisit the L_∞ Hausdorff Voronoi diagram of clusters of points, equivalently, the L_∞ Hausdorff Voronoi diagram of rectangles, and present a plane sweep algorithm for its construction that generalizes and improves upon previous results. We show that the structural complexity of the L_∞ Hausdorff Voronoi diagram is $\Theta(n+m)$, where n is the number of given clusters and m is the number of *essential* pairs of *crossing* clusters. The algorithm runs in $O((n+M)\log n)$ time and $O(n+M)$ space where M is the number of *potentially essential crossings*; m, M are $O(n^2)$, $m \leq M$, but $m = M$, in the worst case. In practice $m, M << n^2$, as the total number of crossings in the motivating application is typically small. For non-crossing clusters, the algorithm is optimal running in $O(n \log n)$-time and $O(n)$-space. The L_∞ Hausdorff Voronoi diagram finds applications, among others, in the geometric min-cut problem, VLSI critical area analysis for via-blocks and open faults.**

***Keywords*-Voronoi diagram, Hausdorff distance, L_∞ metric, plane sweep, point dominance, VLSI layout.**

I. Introduction

Given a set S of point clusters in the plane, the *Hausdorff Voronoi diagram* of S, denoted *HVD(S)*, is a subdivision of the plane into regions such that the *Hausdorff Voronoi region* of a cluster P, denoted *hreg(P)*, is the locus of points *closer* to P than to any other cluster in S, where distance between a point t and a cluster P is measured as the *farthest distance* between t and any point in P, $d_f(t, P) = \max\{d(t, p), p \in P\}$.

$$hreg(P) = \{x \mid d_f(x, P) < d_f(x, Q), \forall Q \in S\}.$$

hreg(P) is subdivided into finer regions by the *farthest Voronoi diagram* of P, *FVD(P)*. The farthest distance $d_f(t, P)$ is equivalent to the *Hausdorff distance*[1] between t and P. In the L_∞ metric[2], $d_f(t, P)$ is equivalent to $d_f(t, P')$, where P' is the minimum enclosing axis-aligned rectangle of P. Thus, the L_∞ Hausdorff Voronoi diagram of S is equivalent to the L_∞ Hausdorff Voronoi diagram of the set S' of the minimum enclosing rectangles of all clusters in S. In this paper the terms cluster and rectangle are used interchangeably. Once the cluster minimum enclosing rectangles are available, individual cluster points have no further influence.

The Hausdorff Voronoi diagram has appeared in the literature under different names having been motivated by different problems [6], [1], [12], [16], [13], [5], [18]. It first appeared in [6] as the *cluster Voronoi diagram*, where several combinatorial bounds were derived. A tight combinatorial bound on its structural complexity in the Euclidean plane as well as a plane sweep construction was given in [13]. The L_∞ version of the problem was introduced in [12] as a solution to the VLSI *critical area extraction* problem for a defect mechanism on *contact layers*, called *via-block*. It was reduced to an L_∞ Voronoi diagram of additively weighted line-segments and it was computed by plane sweep [12]. A related, reverse type, of Voronoi diagram has been considered in [7], [2], [3].

In this paper we revisit the Hausdorff Voronoi diagram in the L_∞ metric. We provide a tight bound on its structural complexity and a plane sweep algorithm for its construction, generalizing and improving upon [12]. In particular, we show that the structural complexity of the L_∞ Hausdorff Voronoi diagram is $\Theta(n + m)$, where n is the number of input clusters and m is the number of *essential pairs* of *crossing* clusters (see Def. 1). The algorithm consists of an $O((n + M)\log n)$-time preprocessing step, based on *point dominance* in $\mathbb{R}^3$, followed by the main plain sweep algorithm that runs in $O((n + M)\log n)$-time and $O(n+M)$-space, where M reflects special crossings that are *potentially essential* (see Def. 2); m, M are $O(n^2)$, $m \leq M$, but $m = M$, in the worst case. In practice, typically, $m, M << n^2$, as the total number of possible crossings in the motivating application is small. For non-crossing rectangles the algorithm simplifies to optimal $O(n \log n)$-time and $O(n)$-space; no previous algorithm achieves an optimal worse case bound for non-crossing clusters. An $O(n \log n)$-expected time algorithm can be derived in the case of non-crossing rectangles using the randomized incremental construction for abstract Voronoi diagrams [10] (see [1]). For arbitrary rectangles, however, the L_∞ Hausdorff Voronoi diagram does not fall under the umbrella of *abstract Voronoi diagrams* [9] (see e.g. [16]). An output sensitive

*Research supported in part by the Swiss National Science Foundation, grant 200021-127137.

[1]The (directed) Hausdorff distance from a set A to a set B is $h(A, B) = \max_{a \in A} \min_{b \in B}\{d(a, b)\}$. The (undirected) Hausdorff distance between A and B is $d_h(A, B) = \max\{h(A, B), h(B, A)\}$.

[2]The L_∞ distance between two points p, q is $d(p, q) = \max\{|x_p - x_q|, |y_x - y_p|\}$.

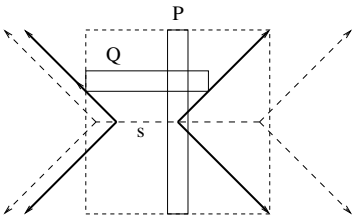

Figure 1. The L_∞ Hausdorff bisector of crossing rectangles.

version of the plane sweep construction, at the expense, however, of advanced data structures that require increased space consumption, is given in [18].

The L_∞ Hausdorff Voronoi diagram finds applications in VLSI design for manufacturability as explained in [12], [14], and in its early form it has been integrated in [19]. Due to the simplicity of the plain sweep approach and the absence, in L_∞, of numerical issues, the approach is very well suited for use in applications, especially in VLSI applications, where the majority of geometries under consideration are axis parallel or in constant orientation.

II. Definitions and structural complexity

Let S be a set of n rectangles, or a set of n point clusters in the plane, where each cluster has been substituted by its minimum enclosing rectangle. A pair of rectangles (P, Q) is called *crossing* if P and Q intersect in the shape of a cross. Given a crossing pair (P, Q), P is assumed to be at least as long as Q. For a rectangle P, let P^n, P^s, P^e, and P^w denote the north, south, east, and west edge of P respectively. P is called a horizontal (resp. vertical) if P^n is longer (resp. shorter) then P^e. The axis parallel line through edge P^i, $i = n, s, e, w$, is denoted as $l(P^i)$. The term P^i is also used to denote the main coordinate of edge P^i. The *core segment* of P is the locus of centers of all minimum enclosing squares of P, and it is given by the axis-parallel line segment of the L_∞ *farthest Voronoi diagram* of P. It can be viewed as an ordinary line segment s additively weighted with $w(s) = d_f(s, P)$. In Fig. 1, *FVD(P)* is illustrated in dashed lines and the core segment is indicated by s. The L_∞ Hausdorff Voronoi diagram of S is equivalent to the (weighted) Voronoi diagram of the set of *core segments* of all clusters in S (see [12]).

The Hausdorff bisector between two clusters P, Q is $b_h(P, Q) = \{y \mid d_f(y, P) = d_f(y, Q)\}$. As shown in [16], $b_h(P, Q)$ is a subgraph of *FVD*$(P \cup Q)$. For a rectangle Q strictly enclosed in the interior of a minimum enclosing square of P, $b_h(P, Q)$ consists of either one (if P and Q are non-crossing) or two (if P and Q are crossing) chains, each one forming a V-shape out of the ± 1-slope rays of *FVD*$(P \cup Q)$; the apex of each chain is called a *V-vertex*. A V-vertex v is incident to the core segment of P and its $90°$-angle faces the portion of the plane closer to P. It is characterized as *up*, *down, right*, or *left*, depending on whether its $90°$-angle is facing north, south, east, or west respectively. In addition, it is characterized as *crossing*, if Q is crossing P, and *non-crossing*, otherwise. The minimum enclosing square of P centered at V-vertex v is also enclosing Q and it is denoted as $square(P, v)$. It is also denoted as $square(P, Q^i)$, where Q^i, is the non-crossing edge of Q that delimits one of its edges. Fig. 1 illustrates $b_h(P, Q)$ consisting of two crossing V-vertices, one right and one left; $square(P, Q^w)$ is illustrated dashed. $square(P, Q^i)$ is referred to as an *extremal minimum enclosing square* of P and Q. The V-vertices of *HVD(S)* are referred to as *Voronoi V-vertices*.

Definition 1: A pair of crossing rectangles (P, Q) is called *essential* if there is an extremal minimum enclosing square of P and Q, $square(P, Q^i)$, that is empty of any other rectangle.

The following lemma is easy to see.

Lemma 1: A pair of crossing rectangles (P, Q) induces a Voronoi V-vertex v in *HVD(S)* if and only if (P, Q) is an essential crossing. Assuming that P is a vertical rectangle, v is a right (resp. left) V-vertex if and only if $square(P, Q_i^w)$ (resp. $square(P, Q_i^e)$) is empty of other rectangles. Similarly for a horizontal rectangle.

Combining Lemma 1 with the structural complexity results of [16] we derive the following bound.

Theorem 1: The structural complexity of the L_∞ Hausdorff Voronoi diagram of a set S of n point clusters, equivalently n rectangles, is $\Theta(n+m)$, where m is the total number of essential crossings.

Proof: In [16] it was shown that, in L_2, the structural complexity of the Hausdorff Voronoi diagram is $O(n+m')$, where m' is the total number of *crossing mixed vertices* on any Hausdorff bisector between a pair of *crossing clusters*. The upper bound trivially applies to the L_∞ metric as well, where m' simplifies to the total number of V-vertices between pairs of crossing clusters. Then the theorem follows from Lemma 1. ■

Definition 2: A collection of crossings for a vertical rectangle P, (P, Q_i), $i = 1, \ldots, k$, is called a *staircase*, if $Q_i^w < Q_{i+1}^w$ and $Q_i^e < Q_{i+1}^e$, $i = 1, \ldots, k$. If in addition, $square(P, Q_i^w)$ is empty of $Q_j \neq Q_i$, the staircase and its crossings are called *potentially essential*. The maximum size of a potentially essential staircase for P is the *number of potentially essential crossings* for P. Let M denote the total number of potentially essential crossings for all vertical rectangles in S, plus the number of essential crossings for all horizontal rectangles in S.

Fig. 2 shows a potentially essential staircase for a vertical rectangle P. In the absence of additional rectangles, all crossings are essential i.e., they all induce Voronoi V-vertices in the Hausdorff Voronoi diagram. The shaded regions in Fig. 2 belong to $reg(P)$.

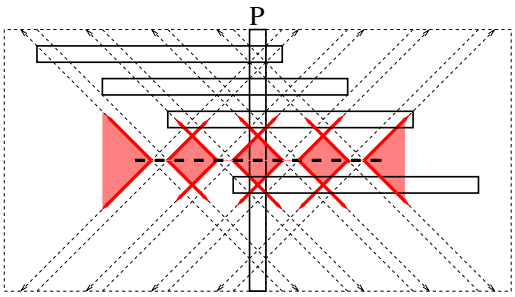

Figure 2. Collection of essential crossings. Shaded regions belong to P

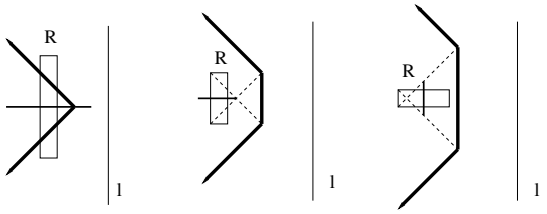

Figure 3. The wavecurve of a rectangle R.

III. A REFINED PLANE SWEEP CONSTRUCTION

In this section we revisit the plane sweep construction of the L_∞ Hausdorff Voronoi diagram given in [12], generalize and improve its time complexity to optimal in the non-crossing case and to worst case asymptotically optimal in general. It is based on the standard plane sweep paradigm for Voronoi diagrams [8], [4], its adaptation for line-segments in L_∞ [15], and the generalization to handle the special features of Hausdorff Voronoi diagrams introduced in [12], [13]. Note that in the Hausdorff Voronoi diagram sites need not be enclosed in their Voronoi regions and Voronoi regions may be disconnected, which are features not addressed by the standard plane sweep paradigm for Voronoi diagrams.

Assume a vertical sweep-line l_t sweeping the entire plane from left to right. At any instant t of the sweeping process we compute $HVD(S_t \cup l_t)$, for $S_t = \{P \in S \mid l(P^e) < t\}$. The boundary of the Voronoi region of l_t is the *wavefront* at time t. Voronoi edges and core segments incident to the wavefront are called *spike bisectors* and *spike core segments* respectively. The combinatorial structure of the wavefront changes at specific *events* organized in a priority queue. We have four types of *site events*: *start-vertical-rectangle*, *end-vertical-rectangle*, *V-vertex* events (for brevity *V-events*), and *horizontal-rectangle* events. Site events are partially similar to those for ordinary line-segments [15] enhanced with additional functions to handle V-vertices and disconnected Voronoi regions. *Spike events* are caused by the intersection of incident spike bisectors, and they remain the same as in the ordinary plane sweep paradigm.

The *wave-curve* of a rectangle R is the bisector between R and the sweep line l_t, at time t, $b(R, l_t) = \{y \mid d_f(y, R) = d(y, l_t)\}$, where $d(y, l_t)$ is the ordinary distance between y and l_t. In L_∞, it consists of two or three *waves*: a ray of slope -1, corresponding to $b(R^s, l_t)$, a ray of slope $+1$, corresponding to $b(R^n, l_t)$, and possibly a vertical line-segment corresponding to $b(R^w, l_t)$, if appropriate. The wave-curve of R can be seen equivalently as the (weighted) bisector between the core segment s of R and l_t, i.e., $b(R, l_t) = \{y \mid d_w(y, s) = d(y, s) + w(s) = d(y, l_t)\}$. In Fig. 3, the wavecurve of several instances of rectangles is illustrated. The bold axis-aligned segment illustrates the core segment of R. The *wavefront*, at time t, is the lower envelope, with respect to the sweep-line, of the *wave-curves* of all rectangles in S_t. In L_∞, the wavefront is monotone with respect to any line of slope ± 1.

The wavefront is typically maintained as a height balanced binary tree, $\mathcal{T}$, ordered from bottom to top. Leaf nodes correspond to waves, while internal nodes correspond to spike bisectors and spike core segments revealing *break-points* between incident waves.

In this paper we augment $\mathcal{T}$ with additional information in order to efficiently answer queries regarding V-vertices. Each node x is augmented with a *w-max* value representing the rightmost west edge of all rectangles contributing a wave to the portion of the wavefront rooted at x, and two *x-min* values (*x-min-I* and *x-min-II*) that in combination represent the minimum x-coordinate of the portion of the wavefront rooted at x. In particular, for a leaf node representing a wave of rectangle R, the *w-max* value is R^w and both *x-min* values are $+\infty$. For an internal node x, *w-max* is the maximum between the *w-max* values of its children. If node x corresponds to a horizontal (resp. ± 1-slope) bisector then *x-min-I* (resp. *x-min-II*) points to the breakpoint of minimum x-coordinate among its own breakpoint and the *x-min-I* (resp. *x-min-II*) values of its children; otherwise *x-min-I* (resp. *x-min-II*) points to the breakpoint of minimum x-coordinate among its children only. The minimum x-coordinate of the portion of the wavefront rooted at node x is *x-min* $= \min\{$*x-min-I*, *x-min-II*$\}$. The augmentation values *w-max, x-min-I, x-min-II* remain the same unless a combinatorial change in the wavefront (i.e., an event) takes place. This is the reason for keeping two separate x-min values instead of one, one for each type of spike bisector, to ensure that no updates are needed unless an event takes place. For brevity, in the following, we assume only two augmentation values, *w-max* and *x-min*, where *x-min* is readily available from *x-min-I* and *x-min-II*.

Any Voronoi point in *HVD(S)* enters the wavefront at the time of its *priority*. The *priority* of a point v is the rightmost x-coordinate of the smallest square centered at v that is entirely enclosing the rectangle P that induces v. The priority of a rectangle P is denoted as *priority(P)* and corresponds to the x-coordinate of P^e.

Let us now discuss the handling of various types of events of a rectangle P of core segment s (see Fig. 4). At time t, let

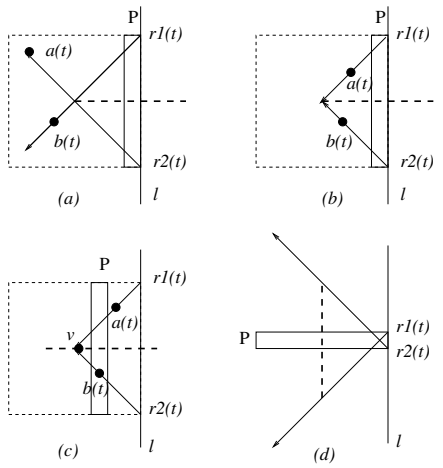

Figure 4. (a) At $t = priority(P)$ the wavefront has not reached s. (b) At $t = priority(P)$ the wavefront has covered portion of s. (c) An invalid event at time $t = priority(v)$. (d) A horizontal rectangle event.

$r_1(t)$ and $r_2(t)$ be the rays of slope $+1$ and -1 respectively, emanating from $l(P^n) \cap l_t$, and $l(P^s) \cap l_t$ respectively, extending towards the left of l_t. Let $a(t)$ and $b(t)$ be the intersection points of $r_1(t)$ and $r_2(t)$ with the wavefront, respectively, at time t. Since the wavefront is ± 1 monotone, $a(t)$ and $b(t)$ can be determined by binary search in $O(logn)$ time. In case of a wave collinear with $r_1(t)$ or $r_2(t)$, the rightmost endpoint is assigned to $a(t), b(t)$, adopting the convention that an equidistant region is assigned to the rectangle preceding P. Because the wavefront is monotone with respect to any line of slope ± 1, in case of a vertical rectangle, the entire portion of the wavefront between $a(t)$ and $b(t)$ must be either to the left (Fig. 4a) or the right (Fig. 4b,c) of the intersection point of $r_1(t)$ and $r_2(t)$, and thus it may intersect $r_1(t), r_2(t)$, or the core segment of s at most once. For a horizontal rectangle (Fig. 4d), the wavefront can intersect the vertical core segment of P a number of times.

Consider time $t = priority(P)$. There are three cases: 1. The wavefront has not reached core segment s yet (either at a start-vertical-rectangle or a horizontal-rectangle event), 2. the wavefront has already covered portion of s, where s is horizontal (start-vertical-rectangle event), 3. the wavefront has already covered portion of s but s is not horizontal (horizontal-rectangle event). Case 3 will be discussed later at a horizontal-rectangle-event.

In case 1, the handling of the corresponding event (a start-vertical-rectangle or a horizontal-rectangle event) is similar to processing an ordinary line-segment event [15], [12]: The portion of the wavefront between $a(t)$ and $b(t)$ is finalized and gets substituted by the wave-curve of P. There is one new action to take: For any crossing V-vertex on the finalized portion of the wavefront, induced by a rectangle Q, generate a V-event for the right V-vertex of $b_h(P, Q)$ and insert it to the event queue.

In case 2 (start-vertical rectangle event), a V-event is generated to predict the first right V-vertex along s (if any). Given the wavefront, perform a binary search in the augmented wavefront to determine the wave between $a(t)$ and $b(t)$ with the rightmost w-max value as induced by a rectangle Q. If no other information is available, generate a V-event for the right V-vertex of $b_h(P, Q)$. Note that at a start-rectangle event, Q must be non-crossing with P.

A V-event v is processed similarly. If at time $t = priority(v)$ the event is *invalid* i.e., the wavefront has already covered v, generate a new V-event, given the wavefront, as described above. In particular, let Q be the rectangle inducing the wave with the rightmost w-max value among the waves between $a(t)$ and $b(t)$. If no additional information is available and assuming that Q is crossing P, generate a V-event for the right V-vertex of $b_h(P, Q)$.

End-rectangle events are similar to right-events of [12].

Correctness in handling vertical-rectangle events follows from the following lemma.

Lemma 2: Let P be a vertical rectangle and let $square(P, v)$ be a minimum enclosing square of P centered along a point v of its core segment s such that $square(P, v)$ contains at least one rectangle in addition to P. At time $t = priority(v)$, let Q be the rectangle with the wave of maximum value *w-max* among all waves between $r_1(t)$ and $r_2(t)$. If Q^w is to the left of P^w, let u be the right V-vertex of $b_f(P, Q)$, otherwise let u be the right endpoint of s. Then $reg(P) \cap \overline{vu} = \emptyset$, and no Voronoi V-vertex can occur on the portion of the core segment $\overline{vu}$.

Proof: Let R be the rectangle in the wavefront that is enclosed in $square(P, v)$ and it has the rightmost west edge among all such rectangles (see Fig 5). If $R^w > P^w$ i.e., R is non-crossing with P, to the right of P, then R must be entirely enclosed in $square(P, x)$ for any point x of s to the right of v. Otherwise, let u be the center of $square(P, R^w)$. Then for any point x on $\overline{vu}$, R must be entirely enclosed in $square(P, x)$. Thus, in both cases the lemma is true for R. It remains to show that $Q = R$.

Since $square(P, v)$ is not empty of other rectangles, the wavefront must have already covered v at time t. Thus, $r_1(t)$ and $r_2(t)$ intersect the wavefront as shown in Fig. 4b,c. Note that $r_1(t)$ and $r_2(t)$ correspond to the diagonals of $square(P, v)$. Since the wavefront is monotone with respect to any line of slope ± 1, the portion of the wavefront between $a(t)$ and $b(t)$ must be entirely included within the triangle Δ formed by $r_1(t), r_2(t)$, their intersection point v, and l_t.

We first argue that R must have a wave within Δ. This is because the +1-slope (resp. -1-slope) ray of wave-curve of R must be below $r_1(t)$ (resp. above $r_2(t)$). In addition, since R is entirely enclosed in $square(P, v)$, the vertical portion of its wavecurve (if it exists) or the apex of its V-curve (if the vertical portion does not yet exist) must be to the right of the vertical line through v.

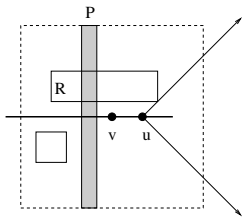

Figure 5. Proof of Lemma 2.

We now argue that only rectangles enclosed within $square(P, v)$ may have a wave within Δ. Let T be a rectangle in the wavefront that is not enclosed in $square(P, v)$ ($priority(T) < t$). If $T^n > P^n$, then the $+1$-slope ray of the wavecurve of T must be above $r_1(t)$. Thus, the entire wavecurve of T must be outside Δ in this case. Similarly if $T^s < P^s$, then the -1-slope ray of the wavecurve of T must be below $r_2(t)$ and thus, the entire wavecurve of T must be outside Δ. If $T^w < square(P, v)^w$, then the vertical (and thus the entire) portion of the wavecurve of T must be to the left of the vertical line through v. Thus, in all cases, no portion of the wavecurve of T can be within Δ at time t.

By the above arguments, the wave of R must have the maximum *w-max* value among all waves in Δ, which corresponds to Q. Thus, $R = Q$. ■

A horizontal-rectangle event is processed at time $t = priority(P)$. The problem is to identify the intersections (V-vertices) of the wavefront with the vertical core segment s (if any). Note that, at time t, the wavefront may intersect s a number of times, each intersection corresponding to a V-vertex, where only the first and the last may be non-crossing V-vertices. If there are no intersections because the wavefront has not reached any portion of s yet, then we have the case 1 of $t = priority(P)$ discussed earlier. Otherwise this is case 3. Any portion of the wavefront to the left of s is finalized and gets substituted by the wave-curve of P as fragmented by the V-vertices and their incident spike bisectors.

To identify V-vertices efficiently we use the *x*-min value of the augmentation. Let r be the breakpoint of minimum *x-min* value between $a(t)$ and $b(t)$. If r is to the right of s then s must be entirely covered by the wavefront and there can be no intersections, i.e., $reg(P) = \emptyset$. Otherwise, trace the wavefront sequentially, starting at r, until the first intersections above and below r are determined. The intersection above (resp. below) corresponds to a *down* (resp. *up*) V-vertex v (resp. u). Repeat the process for the portions of the wavefront above v and below u until all intersections are determined. Any time the *x-min* value of a portion of the wavefront is to the right of s, this portion can be eliminated as we it contains no intersections with s. Correctness in the handling of a horizontal-rectangle event follows easily. Note that a horizontal rectangle event covers also squares.

The time complexity of the plane sweep algorithm, as presented, is $O((n+m+E)\log n)$, where E is the number of invalid V-events. There are two reasons for invalid V-events: 1. Potentially essential staircases of vertical rectangles whose crossings are not all essential. 2. Sequences of *strongly dominated* vertical rectangles, even in the case of non-crossing rectangles. Given a pair of vertical rectangles (P, Q), P is said to *dominate* Q if $Q^w < P^w$ and $Q^n < P^n$, $Q^s > P^s$. If in addition, there is a minimum enclosing square of Q that is crossing P, P is said to *strongly dominate* Q. Fig. 6 shows configurations of vertical rectangles, strongly dominated by P, that may all induce (invalid) V-events on the core segment of P.

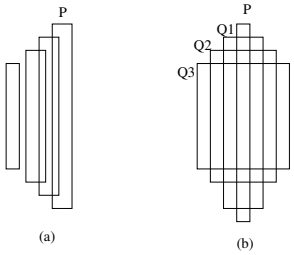

Figure 6. Sequence of vertical rectangles strongly dominated by P. (a) Sequence of non-crossing vertical rectangles. (b) Sequence of vertical pairwise crossing rectangles.

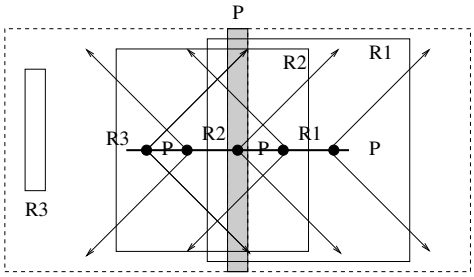

Figure 7. The dominating sequence of rectangle P consisting of rectangles R_1, R_2, R_3

.

To eliminate source-2 of invalid V-vertex events, an (optional) preprocessing step may be added to the algorithm that pre-computes *dominating sequences* of vertical rectangles, as described in Section IV. In the remaining of this section we show that, given the preprocessing information, the number of all V-events generated is bounded by $O(n + M)$.

A. Bounding the number of V-events

Definition 3: The *dominating-sequence* of a vertical rectangle P, denoted *ds(P)*, is a maximal collection of vertical rectangles $R_i, i = 1, \ldots, k$, such that every R_i is entirely enclosed in a minimum enclosing square of P, R_1 is the rectangle with the rightmost west edge among all rectangles dominated by P, and if R_i is crossing P, $i \geq 1$, then R_{i+1} is the vertical rectangle with the rightmost west edge dominated by $square(P, R^e_{i-1})$.

Every R_i in the *dominating-sequence* of P, except possibly the last rectangle R_k, is crossing P. Rectangle R_1 in the dominating sequence of P is referred to as the *rightmost* rectangle dominated by P. In the case of non-crossing rectangles the dominating sequence of P is R_1 (if not empty). Figure 7 shows the dominating sequence of a vertical rectangle P consisting of three rectangles R_1, R_2, R_3. Considering only those four rectangles, the region of P is alternating with regions of R_i on the core segment of P as shown in Fig. 7.

Lemma 3: The dominating sequence of P (except the last rectangle R_k, if R_k is non-crossing with P) forms a maximal staircase of vertical rectangles crossing P that is potentially essential. No vertical rectangle, other than the dominating sequence of P, may induce a right Voronoi V-vertex on the core segment of P, even when only $ds(P)$ is considered.

Proof: By definition, $ds(P)$ forms a potentially essential staircase with P. Let Q be a vertical rectangle enclosed in some minimum enclosing square of P such that Q is not included in the dominating-sequence of P. Recall that only rectangles enclosed in a minimum enclosing square of P may induce a V-vertex on the core segment of P. Let R_i be the rectangle in $ds(P)$ of maximum index such that $Q^w < R_i^w$ (i.e., $Q^w > R_{i+1}^w$ if $i < k$). Since $Q \neq R_{i-1}$, Q^w must be enclosed in $square(P, R_i^e)$. But the right branch of $b_f(P, Q)$ must be in between the two branches of $b_f(P, R_i)$ and thus, the right V-vertex of $b_f(P, Q)$ is subsumed within the region of R_i. ∎

Given the dominating sequences of vertical rectangles, source-2 of invalid V-events can be completely eliminated by considering the corresponding dominating sequence every time a V-event is generated. Let P be a vertical rectangle processed at event v at time $t = priority(v)$. The generation of a V-event for P is described in detail as follows.

Algorithm Generate V-Event(P, ds(P), wavefront, t)
begin

- Let Q be the rectangle in the wavefront at time t, with the rightmost west edge, dominated by P. Q is identified as before by binary search in the augmented wavefront using the w-max value of the waves between rays $r_1(t)$ and $r_2(t)$.
- Let R be the last rectangle in *ds(P)*.
- If R is non-crossing with P then
  - If $Q^w < R^w$, generate a V-event for (P, R);
  - if $R^w \leq Q^w$, generate a V-event for (P, Q);
  - Delete R from *ds(P)*; return.
- While *ds(P)*$\neq \emptyset$ do // all rectangles in *ds(P)* are crossing P
  - Let R be the last rectangle in *ds(P)* (R must be crossing P).
  - If $Q^w < square(P, R^e)^w$, generate a V-event for (P, Q).
  - return.
  - If $Q^w \leq R^w$, generate a V-event for (P, R); delete R from *ds(P)*; return.
  - Else delete R from *ds(P)*.
- If *ds(P)* $= \emptyset$ and no event has been generated so far, generate a V-event for (P, Q).

end

Lemma 4: Using the information of dominating sequences of vertical rectangles, the total number of V-events that may be produced throughout the algorithm is $O(n+M)$.

Proof: Let P be a vertical rectangle of core segment s. By Lemma 3 the only vertical rectangles that may generate V-events on s are the rectangles of the dominating-sequence of P. Thus, any other rectangle must be horizontal (or square). Let $Q_1, \ldots, Q_k$ be a list of non-vertical rectangles that induce right V-events of s. Among them only Q_1 can be non-crossing with P. Since any vertical rectangle with a non-empty Voronoi region is guaranteed to be inserted in the wavefront at the time of its priority and since we always select the rectangle of the rightmost west edge among all rectangles in the wavefront dominated by R, rectangle Q_i cannot be enclosed in $square(P, Q_{i-2}^w)$ for any $i > 2$. Thus, the list of odd indexed rectangles $(Q_1, Q_3, \ldots)$, except possibly Q_1, and the list of even indexed rectangles $(Q_2, \ldots)$ must each form a potentially essential staircase of P. The dominating sequence of P is also potentially essential. Thus, each sequence cannot exceed the number of potentially essential crossings for P. Thus, the total number of V-events that can be generated is $O(n + M)$. ∎

The problem of computing the dominating sequence of every vertical rectangle can be transformed into a *point dominance* problem in $\mathbb{R}^3$ which can be solved by plane sweep in $O((n+M)\log n)$-time, as described in Section IV. Combining with Lemma 4 we conclude.

Theorem 2: HVD(S) can be computed by plane sweep in $O((n + M)\log n)$ time. $O(n + M)$-space. In the case of non-crossing rectangles this results in optimal $O(n \log n)$-time and $O(n)$-space.

IV. Preprocessing step – Point dominance

For any vertical rectangle P, map P^w into point $p = (P^n, -P^s, P^w)$ in $\mathbb{R}^3$. A point $p = (p_x, p_y, p_z)$ is said to *dominate* a point $q = (q_x, q_y, q_z)$ if $p_i \geq q_i$ for all $i = x, y, z$. Given point p, let $R(p)$ denote the rectangle that is mapped into point p. In case two different rectangles map into the same point p, let $R(p)$ be the one with the leftmost east edge. Note that any rectangle, other than $R(p)$, that maps onto the same point p must have an empty Hausdorff Voronoi region, and thus it can be discarded.

To determine the dominating sequence of every rectangle P we first determine R_1, the rectangle dominated by P with the rightmost west edge. In terms of point dominance, the

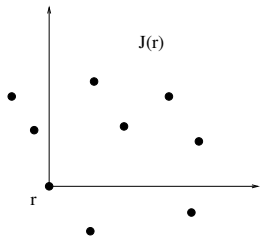

Figure 8. The range $J(r)$ for a point r.

problem is equivalent to determining for every point p, the topmost point q dominated by p. This is is a variant of standard *point dominance* problem in R^3 (see e.g. [17]). This variant can be solved in $O(n \log n)$ time and $O(n)$-space, where n is the number of points, by combining a simple sweeping algorithm and the use of an auxiliary *priority search tree* of [11]. In particular, we sweep points in decreasing z-coordinate, while maintaining their *minima*, $M(t)$, where $M(t)$ is the set of points above the sweeping plane at $z = t$, that do not dominate anything so far, projected on the xy-plane. The set of minima $M(t)$ is organized in a priority search tree T. Let $r = (r_x, r_y, r_z)$ be the point to be considered at time $t = r_z$. Perform a range query on T for the NE quadrant of (r_x, r_y), i.e., range $J(r) = [r_x, \infty] \times [r_y, \infty]$ as shown in Fig. 8. For any point p within $J(r)$, report (p, r). Delete all points in $J(r)$ from T and insert r in T. Point r has an *expiration time* corresponding to the west edge of its leftmost minimum enclosing square, after which r can be permanently deleted from T. Since priority search trees are fully dynamic this is an $O(n \log n)$-time $O(n)$-space algorithm.

In case of non-crossing rectangles the above algorithm is sufficient as a dominating sequence consists of at most one rectangle. In the presence of crossings, however, the remaining rectangles of a dominating sequence must be determined. Let R be a rectangle in $ds(P)$ such that $point(P)$ gets deleted from T because of R. Then a point representing P can get re-inserted into $M(t)$ at time $t = square(P, R^e)^w$, so that the rectangle in $ds(P)$ following R can be determined. This process repeats until either a rectangle non-crossing with P, dominated by P, is reached (the last rectangle in $ds(P)$) or the expiration time of P is reached at which time P is permanently deleted from T. The algorithm is summarized as follows.

Compute dominating sequences of vertical rectangles
begin

- Sort all vertical rectangles and their leftmost minimum enclosing square in decreasing order of their west edge into a list L.
- For each original rectangle P, create $p = point(P) = (p_x, p_y, p_z)$, where $p_x = P^n$, $p_y = -P^s$, $p_z = P^w$; let $R(p) = P$; initialize *ds(P)*$= \emptyset$; let the west edge of $square(P, P^e)$ (the leftmost minimum enclosing square of P) represent the expiration time of P.
- Initialize a dynamic priority search tree T to $\emptyset$.
- Initialize a max-heap H to $\emptyset$. H essentially keeps pairs of crossing rectangles (P, R) such that the expiration time of P has not been reached yet, and R is the last rectangle in $ds(P)$ discovered so far. Pair (P, R) corresponds to $square(P, R^e)$ and H is organized according to maximum west edge of $square(P, R^e)$.
- Process rectangles in $L \cup H$ in decreasing order of west edge. For each rectangle R do
  - If R is an original rectangle, query T for range $J(r)$, $r = point(R)$.
    For each point p in $J(r)$ do
    - Insert R in *ds(P)*, where $P = R(p)$.
    - If R is crossing P, consider $Q = square(P, R^e)$, and insert Q in H. Let $point(Q) = (p_x, p_y, p'_z)$, where $p'_z = Q^w$. Let $R((p_x, p_y, p'_z)) = P$, and set the expiration time of $point(Q)$ to the one of P.
    - Delete p from T.
  - If R is either an original rectangle or an enclosing square in H insert $point(R)$ into T.
  - If R simply marks the expiration time of a rectangle P (i.e., R is the leftmost minimum enclosing square of P), delete the corresponding point of P from T.

end

Correctness follows easily by the definition of a dominating-sequence for vertical rectangles. The time complexity is clearly $O((n + d) \log n)$, where d is the size of dominating sequences of all vertical rectangles, and space is $O(n+d)$, as $O(d)$ space is needed for keeping the resulting dominating sequences. By Lemma 3, d is $O(M)$, thus the total space complexity is $O(n + M)$.

V. Conclusion

We revisited the Hausdorff Voronoi diagram in the L_∞ metric and the plane sweep approach for its construction introduced in [12]. We gave new insight that led into a time complexity bound that is optimal in the worst case, while maintaining the simplicity and practicality of the algorithm. This was achieved in two ways: 1. Augmentation of the main *wavefront* data structure of the plane sweep construction that resulted in the ability to efficiently answer queries on the wavefront for the non-standard features of the Hausdorff Voronoi diagram, and 2. the addition of an (optional) preprocessing step based on a reduction to point dominance in $\mathbb{R}^3$. For non-crossing rectangles this achieves an optimal $O(n \log n)$-time $O(n)$-space algorithm.

The L_∞ Hausdorff Voronoi diagram and its variants have offered the basis for an industrial VLSI CAD tool that

computes the probability of fault (POF) for open faults and via blocks in VLSI designs due to random defects occurring during the manufacturing process [19], [20]. It also offers a solution to the geometric min-cut problem [14], [18]. For information on the Voronoi diagram approach to VLSI critical area extraction see [14], [12], [15] and references within.

REFERENCES

[1] M. Abellanas, G. Hernandez, R. Klein, V. Neumann-Lara, and J. Urrutia. A combinatorial property of convex sets. *Discrete Computat. Geometry*, 17:307–318, 1997.

[2] M. Abellanas, F. Hurtado, C. Icking, R. Klein, E. Langetepe, L. Ma, B. Palop, and V. Sacristán. The farthest color Voronoi diagram and related problems. In *Abstracts 17th European Workshop Comput. Geom. 2001*, 113–116.

[3] O. Cheong, H. Everett, M Glisse, J Gudmundsson, S. Hornus, S. Lazard, M. Lee, and H.-S. Na. Farthest-Polygon Voronoi Diagrams. arXiv:1001.3593v1 [cs.CG], 2010.

[4] F. Dehne and R. Klein. The big sweep": On the power of the wavefront approach to Voronoi diagrams. *Algorithmica*, 17:19–32, 1997.

[5] F. Dehne, A. Maheshwari, and R. Taylor. A coarse grained parallel algorithm for Hausdorff Voronoi diagrams. In *Proc. 2006 Int. Conf. on Parallel Processing*, 497–504, 2006.

[6] H. Edelsbrunner, L. Guibas, and M. Sharir. The upper envelope of piecewise linear functions: Algorithms and applications. *Discrete Computat. Geometry*, 4:311–336, 1989.

[7] D. P.Huttenlocher, K. Kedem, and M. Sharir. The upper envelope of Voronoi surfaces and its applications. *Discrete Computat. Geometry*, 267–291, 1993.

[8] S. J. Fortune. A sweepline algorithm for Voronoi diagrams. *Algorithmica*, 2:153–174, 1987.

[9] R. Klein, "Concrete and Abstract Voronoi Diagrams", vol. 400, *Lecture Notes in Computer Science*, Springer-Verlag, 1989.

[10] R. Klein, K. Mehlhorn, S. Meiser, "Randomized incremental construction of abstract Voronoi diagrams", *Computational Geometry: Theory and Applications*, 3, 157-184, 1993.

[11] E. M. McCreight. Priority Search Trees. *SIAM J. Comput.*, 14:257–276, 1985.

[12] E. Papadopoulou. Critical area computation for missing material defects in VLSI circuits. *IEEE Trans. Comp.-Aided Design*, 20(5):583–597, 2001.

[13] E. Papadopoulou. The Hausdorff Voronoi diagram of point clusters in the plane. *Algorithmica*, 40:63–82, 2004.

[14] E. Papadopoulou. Net-aware critical area extraction for opens in VLSI circuits via higher-order Voronoi diagrams. *IEEE Trans. on Comp.-Aided Design*, 30(5), 704–716, 2011. Preliminary version in *LNCS* vol 4835, 716–727, 2007.

[15] E. Papadopoulou and D. T. Lee. The L_∞ Voronoi diagram of segments and VLSI applications. *Int. Journal of Computat. Geometry and Applications*, 11(5):503–528, 2001.

[16] E. Papadopoulou and D. T. Lee. The Hausdorff Voronoi diagram of polygonal objects: A divide and conquer approach. *Int. J. of Computational Geometry and Applications*, 14(6):421–452, 2004.

[17] Preparata, F. P. and M. I. Shamos, *Computational Geometry: an Introduction*, Springer-Verlag, New York, NY 1985.

[18] J. Xu, L. Xu, and E. Papadopoulou. Computing the map of geometric minimal cuts. In *Proc. 20th Int. Symp. on Algorithms and Computation*, vol 5878 of *LNCS*, 244–254, 2009.

[19] "Voronoi CAA: Voronoi Critical Area Analysis", IBM CAD Tool, Dept. Electronic Design Automation, IBM Microelectronics, Burlington, VT. Initial patents: US6178539, US6317859. Distributed by Cadence.

[20] S. C. Braasch, J. Hibbeler, D. Maynard, M. Koshy, R. Ruehl, and D. White, "Model-based verification and analysis for 65/45nm physical design," CDNLive!, September 2007.