International Journal of Computational Geometry & Applications

THE HAUSDORFF VORONOI DIAGRAM OF POLYGONAL OBJECTS: A DIVIDE AND CONQUER APPROACH*

EVANTHIA PAPADOPOULOU
IBM TJ Watson Research Center
P.O. Box 218, Yorktown Heights, NY 10598
e-mail:evanthia@watson.ibm.com

D. T. LEE†
Institute of Information Science, Academia Sinica
Nankang, Taipei, Taiwan
e-mail:dtlee@iis.sinica.edu.tw

Received 19 February 2004
Revised 21 October 2004
Communicated by M. Iri

ABSTRACT

We study the *Hausdorff Voronoi diagram* of a set S of polygonal objects in the plane, a generalization of Voronoi diagrams based on the *maximum* distance of a point from a polygon, and show that it is equivalent to the Voronoi diagram of S under the *Hausdorff distance* function. We investigate the structural and combinatorial properties of the Hausdorff Voronoi diagram and give a divide and conquer algorithm for the construction of this diagram that improves upon previous results. As a byproduct we introduce the *Hausdorff hull*, a structure that relates to the Hausdorff Voronoi diagram in the same way as a convex hull relates to the ordinary Voronoi diagram. The Hausdorff Voronoi diagram finds direct application in the problem of computing the *critical area* of a VLSI Layout, a measure reflecting the sensitivity of a VLSI design to random manufacturing defects, described in a companion paper.[13]

Keywords: Voronoi diagram; Hausdorff distance; Hausdorff-hull; divide and conquer; VLSI yield; VLSI critical area; via-block defects.

1. Introduction

The *Hausdorff Voronoi diagram* of a set S of polygonal objects in the plane is a subdivision of the plane into regions such that the Voronoi region of a polygon

*Preliminary version: "The Min-Max Voronoi diagram of polygonal objects and applications in VLSI manufacturing" appeared in *Proc. 13th International Symposium on Algorithms and Computation – ISAAC'02*, November 2002, Vancouver, Canada.
†Supported in part by the National Science Council under the Grants NSC-93-2213-E-001-013, NSC-93-2422-H-001-0001, and NSC-93-2752-E-002-005-PAE.

$P \in S$ is the locus of points whose maximum distance from P is less than their maximum distance from any other object in S. The Hausdorff Voronoi diagram can be defined equivalently in terms of the *Hausdorff distance*, where the (directed) Hausdorff distance from a set of points P to a set of points Q is the maximum distance from any point in P to its nearest neighbor in Q (see Section 2). The (undirected) Hausdorff distance between P and Q is the maximum between the two directed distances from P to Q and from Q to P. The Hausdorff Voronoi region of a polygon $P \in S$ is subdivided into finer regions by the farthest point Voronoi diagram of the vertex set of P. This structure generalizes both the ordinary Voronoi diagram of points and the farthest-point Voronoi diagram. If shapes degenerate to points, we obtain the ordinary Voronoi diagram, and in case where S consists of a single shape ($|S| = 1$), we have the farthest-point Voronoi diagram. The definition can be given equivalently on a set S of clusters of points instead of polygonal objects. This diagram represents the reverse of the *farthest color* Voronoi diagram of Abellanas et al.[2] that had also been considered in Ref. [6]. This paper is a companion paper to Ref. [13] which provides a plane sweep construction to the Hausdorff Voronoi diagram and a direct application in computing the *critical area* for *via-blocks* of a VLSI Layout.

The Hausdorff Voronoi diagram was first considered in Ref. [5], where it was termed the *Voronoi diagram of point clusters*, and later in Ref. [1] where it was termed the *closest covered set diagram*. In Ref. [14], a simpler L_∞ version of the problem for non-crossing rectangles was termed the *min-max* Voronoi diagram. The min-max Voronoi diagram was formulated to address the *critical area* computation problem for *via-blocks* in VLSI designs. The term *min-max* Voronoi diagram was also followed in a preliminary version of this paper.[15] In Ref. [5] the size of the Hausdorff Voronoi diagram was shown to be $O(n^2\alpha(n))$ for any arbitrary S, and $O(n)$ for clusters of points with disjoint convex hulls, where n is the number of points on the convex hulls of shapes in S, and $\alpha(n)$ is the inverse of Ackermann's function. The $O(n)$ bound was also shown in Ref. [1] for disjoint convex shapes and arbitrary convex distance functions. Using a powerful geometric transformation in three dimensions,[5] and a divide and conquer algorithm for computing the upper envelope of piecewise linear functions in three dimensions, the Hausdorff Voronoi diagram was shown in Ref. [5] to be constructed in $O(n^2\alpha(n))$ time, with $O(n^2)$ time being sufficient if S consisted of disjoint segments. In Ref. [1] the problem for disjoint convex sets was reduced to abstract Voronoi diagrams and the randomized incremental construction of Ref. [9] was proposed for its computation, resulting in an $O(kn \log n)$-time algorithm, where k is the time to construct the Hausdorff bisector between two disjoint convex polygons. In Ref. [14] a plane sweep algorithm was given for the simpler L_∞ *non-crossing*[a] version of the problem of time complexity

[a]Two polygons P, Q are called *non-crossing* if their convex hulls admit at most two *supporting segments* appearing on the convex hull of $P \cup Q$ (see Def. 4 and Def. 6).

$O((n+K)\log n)$, where K was the number of *interacting*[b] pairs of shapes. The plane sweep approach was generalized in Ref. [13] for arbitrary clusters of points and the Euclidean metric with time complexity as given below.

In this paper we list the structural properties of the Hausdorff Voronoi diagram and provide tighter combinatorial bounds and algorithms. Specifically, we show that for arbitrary polygons or arbitrary clusters of points, the size of the Hausdorff Voronoi diagram is $O(n+m)$, where m is $O(n^2)$ and reflects the number of *crossings* among shapes in S (see Theorem 1 for a precise definition of m). In the case of *non-crossing* polygons, not necessarily disjoint, Hausdorff Voronoi regions are shown to remain connected and thus, the size of the Hausdorff Voronoi diagram is shown to be $O(n)$. That is, the connectivity and linearity of the diagram is established for a more general class of polygonal objects than the ones shown in Refs. [1, 5]. This bound automatically improves the time complexity of the algorithm of Ref. [5] to $O(n^2)$. We present a divide and conquer algorithm to construct the Hausdorff Voronoi diagram of time complexity $O(M + n\log^2 n + (m+K)\log n)$, where n is the number of points on the convex hulls of shapes in S, $K = \Sigma_{P\in S}K(P)$, $M = \Sigma_{P\in S}M(P)$, where $K(P)$ is the number of shapes enclosed in the minimum enclosing circle of P, and $M(P)$ is the number of convex hull points $q \in Q$ that are *interacting* with P that is, q is enclosed in the minimum enclosing circle of P and either Q is entirely enclosed in the minimum enclosing circle of P or Q is crossing with P. The algorithm assumes an $O(|S|\log n)$ preprocessing time to compute the convex hulls of input shapes. Note that this is an improvement over the bound given in a preliminary version of this paper.[15] In addition we introduce the *Hausdorff hull*, a structure that relates to the Hausdorff Voronoi diagram in the same way as a convex hull relates to the ordinary Voronoi diagram (see Def. 8), and show that it can be computed in $O(n\log n)$ time. In the companion paper,[13] we refine the $O(n+m)$ bound on the size of the Hausdorff Voronoi digram and show that it is tight in the worst case. We also present a simple plane sweep algorithm of comparable time complexity $O(M_a + (n+m+K_a)\log m)$, where M_a and K_a are defined similarly to M and K with the difference that they are defined over the *anchor circle* of P, a specially defined enclosing circle, generally different from the minimum enclosing circle of P.

Our motivation for studying the Hausdorff Voronoi diagram comes from an application in VLSI manufacturing, namely VLSI *yield prediction*, as explained in our companion paper.[13] The problem statement is repeated here for completeness. The main computational bottleneck in predicting the *yield* of a VLSI chip is the extraction of *critical area*, a measure reflecting the sensitivity of the design to random defects during manufacturing (see e.g. Refs. [10, 12, 19, 20, 11, 14, 16]). In Refs. [14, 16] the critical area computation problem for the three main types of defect mechanisms: *shorts*, *opens*, and *via-blocks*, was reduced to variations of L_∞ Voronoi

[b]In Ref. [14], a shape Q is called *interacting* with P if Q is enclosed in *a* minimum enclosing square of P (see also Def. 12).

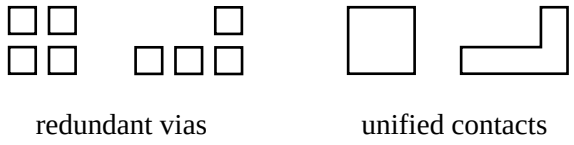

Fig. 1. Contacts as groups of redundant vias.

diagrams of segments. The L_∞ metric reflected a natural model of square manufacturing defects and considerably simplified the construction of Voronoi diagrams.[17] However, the square defect model has been criticized in the case of via-blocks (see e.g. Ref. [12]), and thus the Euclidean version of the problem needs also to be addressed in this case. The construction of the Euclidean Hausdorff Voronoi diagram is realistic as robustness issues are similar to the construction of Voronoi diagrams of points and not to those of segments.

The critical area problem for via-blocks is as follows. In a VLSI layout contacts between different layers are realized by square shapes called vias. To achieve a desired resistance or to reduce the probability of missing contacts, designers often use *redundant vias*, a group of multiple vias that connect the same shapes on different layers. Redundant vias are usually grouped together side by side and they are regarded as a single contact of larger size (see Figure 1). After preprocessing to identify redundant vias, a via layer consists of rectilinear shapes. The majority of these shapes are disjoint but some may be crossing in case of redundant vias located further apart. A *via-block* is a defect that completely covers an entire contact.[11,20] A defect of size r is a circle of radius r. The *critical radius* of a point t on a via layer is the radius of the smallest defect centered at t causing a via-block. The problem is to compute the critical radius for via-blocks of every point on a via layer, as we need to integrate to compute the total *critical area* for all possible defect sizes following a given defect distribution. That is, the problem is to compute the Hausdorff Voronoi diagram of the unified contact shapes on a via layer. Once this diagram is available the critical area integral can be computed as shown in Refs. [14, 16, 19].

This paper is organized as follows. Section 2 provides preliminary definitions. In Section 3 we list the structural and combinatorial properties of the Hausdorff Voronoi diagram and introduce the concept of the Hausdorff hull. In Section 4 we give a divide and conquer algorithm for the Hausdorff Voronoi diagram. The main difficulty of the divide and conquer scheme is that the standard merge curve contains multiple connected components including cycles. Tracing the multiple components in linear time is shown in Section 4. In sub-Section 4.1 we show how to compute starting points for the unbounded portions of merge curve and in sub-Section 4.2 we show how to compute starting points for cycles. Section 5 provides concluding remarks.

2. Preliminaries

Let $d(p,q)$ denote the ordinary distance between two points p, q. The ordinary bisector between p and q, denoted as $b(p,q)$, is the locus of points equidistant from p and q. The bisector $b(p,q)$ partitions the plane into two *half-planes*, one associated with p and the other with q. The *farthest distance* of a point p from a shape Q is $d_{\rm f}(p,Q) = \max\{d(p,q), \forall q \in Q\}$. It is well known that $d_{\rm f}(p,Q) = d(p,q)$ for some vertex q on the convex hull of Q. It is also well known that $d_{\rm f}(p,Q)$ can be determined by the *farthest point* Voronoi diagram of the vertex set of Q, denoted as *f-Vor(Q)* (see e.g. Ref. [18]). Any region of *f-Vor(Q)* is denoted by *freg*(q), for any point q on the convex hull of Q. The convex hull of Q is denoted as $CH(Q)$. The (directed) Hausdorff distance from a shape P to Q is $h(P,Q) = \max_{p\in P} \min_{q\in Q} d(p,q)$. The (undirected) Hausdorff distance between P and Q is $d_{\rm h}(P,Q) = \max\{h(P,Q), h(Q,P)\}$.

Definition 1. The farthest (resp. Hausdorff) bisector, denoted $b_{\rm f}(P,Q)$ (resp. $b_{\rm h}(P,Q)$, between P and Q is the locus of points equidistant from P and Q according to $d_{\rm f}(P,Q)$ (resp. $d_{\rm h}(P,Q)$). That is, $b_{\rm f}(P,Q) = \{y \mid d_{\rm f}(y,P) = d_{\rm f}(y,Q)\}$ and $b_{\rm h}(P,Q) = \{y \mid d_{\rm h}(y,P) = d_{\rm h}(y,Q)\}$.

Lemma 1. *$b_{\rm f}(P,Q)$ and $b_{\rm h}(P,Q)$ are equivalent. That is, for any point y, $d_{\rm f}(y,P) < d_{\rm f}(y,Q)$ iff $d_{\rm h}(y,P) < d_{\rm h}(y,Q))$.*

Proof. For any point y, $h(y,P) = d(y,P)$, where $d(y,P) = \min_{p\in P}\{d(y,p)\}$, and $h(P,y) = d_{\rm f}(y,P)$. But $d_{\rm f}(y,P) \geq d(y,P)$. Thus, $d_{\rm h}(y,P) = \max\{h(y,P), h(P,y)\} = d_{\rm f}(y,P)$. □

Definition 2. The *Hausdorff* Voronoi diagram of S, denoted as *H-Vor(S)*, is a subdivision of the plane into regions such that the Hausdorff Voronoi region of a polygon P, denoted as *hreg*(P), is the locus of points closer to P, according to $d_{\rm f}$ (equivalently $d_{\rm h}$), than to any other shape in S i.e., *hreg*$(P) = \{y \mid d_{\rm f}(y,P) \leq d_{\rm f}(y,Q), \forall Q \in S, Q \neq P\}$. The Hausdorff Voronoi region of P is subdivided into finer regions by the farthest point Voronoi diagram of the vertex set of P. That is, for $p \in CH(P)$, *hreg*(p) = *hreg*$(P) \cap$ *freg*$(p) = \{y \mid d(y,p) = d_{\rm f}(y,P) \leq d_{\rm f}(y,Q), \forall Q \in S, Q \neq P\}$. The Voronoi edges on the boundary of *hreg*(P) are portions of Hausdorff bisectors between P and other objects in S, referred to as *inter-bisectors*. The bisectors in the interior of *hreg*(P) are portions of bisectors in *f-Vor(P)* and are called *intra-bisectors*.

Figure 2 illustrates the Hausdorff Voronoi diagram of $S = \{P_1, P_2, P_3\}$. The shaded regions depict *hreg*(P_1) and *hreg*(P_3); the unshaded portion corresponds to *hreg*(P_2). Inter-bisectors are shown in solid lines and intra-bisectors are depicted in dashed lines. Figure 3 illustrates the Hausdorff Voronoi diagram of two intersecting segments P and Q, where *hreg*(P) is illustrated shaded. Any portion of an intra- or inter-bisector segment correspond to a portion of an ordinary bisector $b(p,q)$

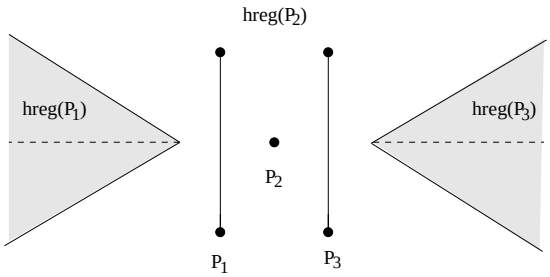

Fig. 2. The Hausdorff Voronoi diagram of $S = \{P_1, P_2, P_3\}$.

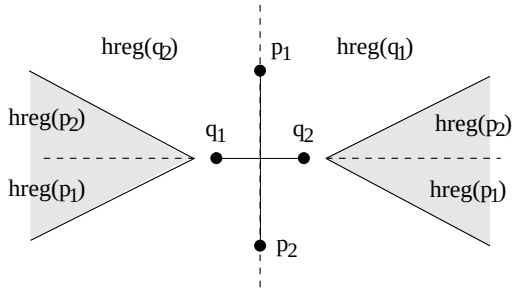

Fig. 3. The Hausdorff Voronoi diagram of two intersecting segments.

between two points p, q. For an inter-bisector, the *half-plane* associated with p is the locus of points closer to p than q, and p, q are points of different shapes. For an intra-bisector, the *half-plane* associated with p is the locus of points farther from p than q, and p, q are points of the same shape. We also distinguish between three types of vertices: *inter-vertices* where at least three inter-bisectors meet, *intra-vertices* where at least three intra-bisectors meet, and *mixed-vertices* where one intra-bisector and two inter-bisectors meet.

Definition 3. The circle $\mathcal{K}_y$, centered at an intra-bisector point y of radius $d_{\mathrm{f}}(y, P)$ is called a *P-circle*. A P-circle that encloses no shape other than P is called *empty*.

Definition 4. A *supporting line* of a convex polygon P is a straight line l passing through a vertex v of P such that the interior of P lies entirely on one side of l. Vertex v is called a *supporting vertex*. Directed line l and vertex v are called *left (resp. right) supporting* if P lies to the right (resp. left) of l. The portion of the supporting line l between the supporting vertices of two convex polygons such that both polygons lie on the same side of l is called a *supporting segment*.

Definition 5. Any segment $\overline{p_i p_j}$ for $p_i, p_j \in CH(P)$ is called a *chord*. Two intersecting chords $\overline{p_i p_j} \in P$ and $\overline{q_i q_j} \in Q$ are called *crossing* if all their endpoints p_i, p_j, q_i, q_j appear on the convex hull of $P \cup Q$. Otherwise $\overline{p_i p_j}$ and $\overline{q_i q_j}$ are called

non-crossing.

Definition 6. A polygon Q is called *crossing* with chord $\overline{p_ip_j} \in P$ if there exists a chord $\overline{q_iq_j} \in Q$ that is crossing with $\overline{p_ip_j}$. Otherwise Q is called *non-crossing* with $\overline{p_ip_j}$. Two polygons P and Q are called *crossing* if they admit a pair of crossing chords $\overline{p_ip_j} \in P$ and $\overline{q_iq_j} \in Q$. Otherwise they are called *non-crossing.*

Note that two polygons are non-crossing if and only if their convex hulls admit at most two supporting segments. An example of non-crossing and crossing polygons can be seen in Figure 5. In Figures 5(a) and 5(b) all polygons are non-crossing; in Figure 5(c) polygons C and D are crossing.

Definition 7. A chain is a simple polygonal line. A chain C is monotone with respect to a line l if every line orthogonal to l intersects C in at most one point.

3. Structure and Properties

In this section we list the structural properties of the Hausdorff Voronoi diagram. These properties are then used throughout the divide and conquer construction of our algorithm. Unless explicitly mentioned otherwise, these properties have not been identified in Refs. [1, 5].

Property 1. *The farthest bisector $b_{\mathrm{f}}(P,Q)$, and equivalently the Hausdorff bisector $b_{\mathrm{h}}(P,Q)$, is a subgraph of f-Vor($P \cup Q$) consisting of edge disjoint monotone chains. If a chain has just one edge, this is a straight line; otherwise its two extreme edges are semi-infinite rays.*

Proof. Let t be a point along $b_{\mathrm{f}}(P,Q)$. Then there is a vertex $p \in P$ and a vertex $q \in Q$ such that $d_{\mathrm{f}}(t,P) = d(t,p) = d(t,q) = d_{\mathrm{f}}(t,Q)$. But then $d_{\mathrm{f}}(t,P \cup Q) = d(t,p) = d(t,q)$. Thus, t must be on the border of *freg*(p) and *freg*(q) in *f-Vor*$(P \cup Q)$. Thus, $b_{\mathrm{f}}(P,Q)$ must be a subgraph of *f-Vor*$(P \cup Q)$. By assigning two different colors to the regions of P and Q in *f-Vor*$(P \cup Q)$, $b_{\mathrm{f}}(P,Q)$ consists of boundaries between regions of different colors. Since *f-Vor*$(P \cup Q)$ is a two colorable map consisting only of unbounded regions, $b_{\mathrm{f}}(P,Q)$ must consist of edge-disjoint chains extending to infinity (similarly to the ordinary Voronoi diagram[18]). If points are assumed to be in general position i.e., there are no four co-circular points, the chains in $b_{\mathrm{f}}(P,Q)$ must also be vertex-disjoint.

Let's now show that the chains constituting $b_{\mathrm{f}}(P,Q)$ are monotone. Any semi-infinite ray of *f-Vor*$(P \cup Q)$ corresponds to the perpendicular bisector of a distinct convex hull edge of $CH(P \cup Q)$. Thus, any semi-infinite edge of $b_{\mathrm{f}}(P,Q)$ is the perpendicular bisector of a distinct supporting segment between $CH(P)$ and $CH(Q)$. Let C be a chain of $b_{\mathrm{f}}(P,Q)$ and let the two corresponding pairs of supporting segments be $\overline{p_iq_i}$ and $\overline{q_jp_j}$, where $p_i, p_j \in P$, $q_i, q_j \in Q$, and p_i, q_i, q_j, and p_j appear in counterclockwise order on $CH(P \cup Q)$. Let L and R denote the set of vertices on $CH(P \cup Q)$ that lie between vertices q_i and q_j, and between vertices p_i and

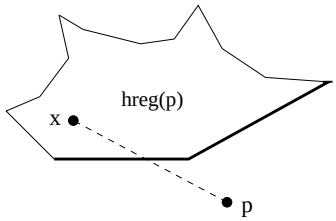

Fig. 4. The structure of $hreg(p)$

p_j respectively in counterclockwise order. Consider *f-Vor*(R) and *f-Vor*(L). Then C coincides with the polygonal dividing line σ which is obtained by merging *f-Vor*(R) and *f-Vor*(L). Since R and L are linearly separable it is well known (see e.g. Ref. [18]) that σ, and therefore C, is monotone. □

Corollary 1. *Any semi-infinite ray of* $b_{\mathrm{f}}(P,Q)$ *corresponds to the perpendicular bisector induced by a distinct supporting segment between* $CH(P)$ *and* $CH(Q)$. *The number of chains constituting* $b_{\mathrm{f}}(P,Q)$ *is derived by the number of supporting segments between* $CH(P)$ *and* $CH(Q)$.

Property 2. *For any point* $x \in hreg(p)$ *the segment* $\overline{px} \cap freg(p)$ *lies entirely in* $hreg(p)$.

Proof. Suppose to the contrary that there is a point y on $\overline{px} \cap freg(p)$ such that $y \in hreg(q)$, where $p \in P$ and $q \in Q \neq P$. By definition of x, we have $d(x,p) = d_{\mathrm{f}}(x,P) < d_{\mathrm{f}}(x,Q)$, and the circle $\mathcal{K}_x$ centered at x of radius $d(x,p)$ contains P totally in its interior, but not Q. Since $y \in hreg(q)$, $d(y,q) < d(y,p) < d(x,p)$. Circle $\mathcal{K}_y$ centered at y of radius $d(y,q)$ contains Q totally in its interior. However, circle $\mathcal{K}'_y$ centered at y of radius $d(y,p)$, which contains $\mathcal{K}_y$, is contained in $\mathcal{K}_x$. In other words, $\mathcal{K}_x$ contains Q in its interior, which is a contradiction. □

Property 3. *The boundary of any connected component of* $hreg(p)$, $p \in P$, $p \neq P$, *consists of a sequence of outward convex chains, each one corresponding to an inter-bisector* $b_{\mathrm{f}}(p,Q_i)$, $Q_i \in S, Q_i \neq P$, *and exactly one inward convex chain corresponding to the intra-bisector* $b_{\mathrm{f}}(p,P)$. *(Convexity is characterized as seen from the interior of* $hreg(p)$. *See Figure 4 where the intra-bisector chain of* $hreg(p)$ *is shown thickened).*

Proof. By Property 1, the farthest bisector of a point p and a shape Q, $p \notin CH(Q)$, must be a convex unbounded chain with convexity facing away from p. The property is derived by the fact that $hreg(p) = (\cap_{Q \in S} H(p,Q)) \cap freg(p)$ for any $p \in P$, where $H(p,Q)$ denotes the *half-plane* associated with p as partitioned by $b_{\mathrm{f}}(p,Q)$. □

Let's now define the *Hausdorff hull*, or *H-hull* for short, of S, denoted as *HH(S)*.

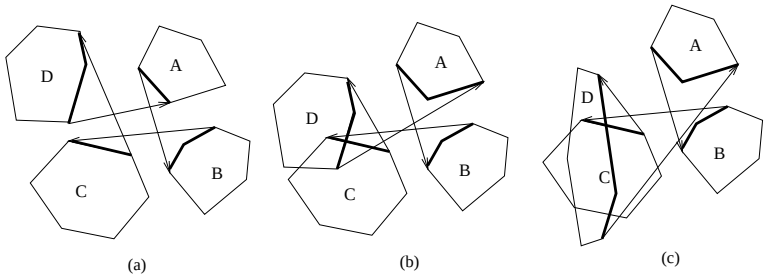

Fig. 5. The Hausdorff hull.

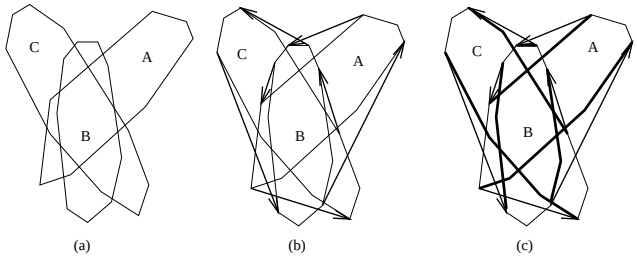

Fig. 6. The *Hausdorff hull* of pairwise crossing shapes.

The *H-hull* is related to the *H-Vor(S)* as the ordinary convex hull is related to the ordinary Voronoi diagram. Examples are depicted in Figure 5 and Figure 6.

Definition 8. A shape $P \in S$, in particular a vertex $p \in P$, is said to be on the H-hull of S, if and only if p admits a supporting line ℓ such that $CH(P)$ lies totally on one side of ℓ and none of the rest of shapes in S lie totally on the same side, except possibly from a shape having its boundary common to ℓ and its interior lying on the same side of ℓ as $CH(P)$. A segment $\overline{pq}$ joining two H-hull vertices $p \in P, q \in Q$ such that $\overline{pq}$ is a supporting segment of P, Q is called an H-hull *supporting segment*. An H-hull edge is either a convex hull edge joining H-hull vertices of one shape or an H-hull supporting segment joining H-hull vertices of different shapes.

By Definition 8, a supporting segment $\overline{pq}$, $p \in P$ and $q \in Q$, is an edge of the *H-hull* of S if and only if $CH(P)$ and $CH(Q)$ lie totally on one side of the line $\ell_{\overline{pq}}$ passing through $\overline{pq}$ and no other shape in S lies totally on the same side of $\ell_{\overline{pq}}$ (except possibly from a shape having its boundary common to $\ell_{\overline{pq}}$). Furthermore, a convex hull edge $\overline{pr} \in CH(P)$, is an H-hull edge if and only if $CH(P)$ is the only shape in S lying entirely on one side of the underlying line $\ell_{\overline{pr}}$. Thus, the boundary of the H-hull consists of a sequence of supporting segments, interleaved with a subsequence of convex chains. The order of traversal, say clockwise, of the H-hull edges satisfies the property that each edge defines a supporting line of a shape P on the H-hull: a right supporting line for H-hull supporting segments and a left supporting

line for convex hull chains. Figure 5 illustrates the H-hulls of (a) disjoint, (b) non-crossing, and (c) crossing shapes respectively. Supporting segments are illustrated in arrows according to a clockwise traversal. Figure 6 depicts the H-hull of pairwise crossing shapes. For clarity Figure 6(a) depicts only the three shapes, Figure 6(b) illustrates the supporting segments of the *H-hull*, and Figure 6(c) illustrates the whole *H-hull* in bold.

Property 4. *Region hreg*(p) *is unbounded if and only if vertex* p *lies on the H-hull of* S.

Proof. Suppose $p \in P$ is a vertex of *HH(S)*. Then by Definition 8 there exists a supporting line ℓ to $CH(P)$ passing through p that satisfies the condition stated. Consider the ray perpendicular to ℓ that is emanating from p into the half-plane where $CH(P)$ lies. Then for point x on the ray with $d(x,p)$ being sufficiently large, the circle $\mathcal{K}_x$ centered at x of radius $d(x,p)$ contains only shape P in its interior. That is, point x belongs in $hreg(p) \subset hreg(P)$. Thus, $hreg(p)$, and therefore $hreg(P)$, is unbounded.

Conversely suppose $hreg(p)$ is unbounded. Consider one of the unbounded edges of the boundary of $hreg(p)$. It may be an inter- or an intra-bisector b. If b is an intra-bisector then it is the perpendicular bisector of an edge $s = \overline{pp'}$ on $CH(P)$. If b is an inter-bisector, portion of $b_{\rm f}(P,Q)$, then b is the perpendicular bisector of a supporting segment $s = \overline{pq}$ between $CH(P)$ and $CH(Q)$. For any point x on b, the circle $\mathcal{K}_x$ of radius $d(x,p)$ entirely contains P. In case b is an inter-bisector, $\mathcal{K}_x$ also contains Q. However $\mathcal{K}_x$ can not contain any other shape, otherwise b would not be a Voronoi edge of the *H-Vor(S)*. Let ℓ be the line passing through the supporting segment s. Clearly any shape on the same side of ℓ as P would eventually be contained in $\mathcal{K}_x$ for some $x \in b$. Thus, ℓ must be a supporting line satisfying the condition of the H-hull definition that is, $p \in HH(S)$. □

Corollary 2. *All unbounded bisectors of H-Vor(S) (both inter- and intra-bisectors) are cyclically ordered in the same way as the H-hull edges are ordered on the boundary of HH(S).*

Definition 9. The tree structure of *f-Vor(P)* is called the *intra-bisector tree* of P and it is denoted as $T(P)$. The root of $T(P)$ is assumed to be the center of the minimum enclosing circle of P.

Every point y of the intra-bisector tree $T(P)$ corresponds to the center of a P-circle and it is weighted by $d_{\rm f}(y,P)$. The point of minimum weight along $T(P)$ is the center of the minimum enclosing circle of P and it is regarded as the root of $T(P)$. Note that root of $T(P)$ may be a point of degree 2. Let $\overline{y_r y_k} \in T(P)$ be the intra-bisector segment of chord $\overline{p_i p_j}$, where y_r is the parent of y_k in $T(P)$. Let y be an arbitrary point on $\overline{y_r y_k}$. Point y partitions $T(P)$ in two parts. Let $T(y)$ denote the part containing the descendents of y in the rooted $T(P)$ i.e., the subtree of

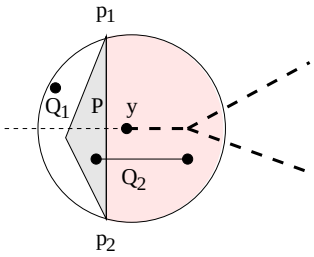

Fig. 7. $Q_1 \in \mathcal{K}_y^r \cup CH(P)$ and $Q_2 \in \mathcal{K}_y^f \cup CH(P)$.

$T(P)$ rooted at y that contains segment $\overline{yy_k}$, and let $T_c(y)$ denote the complement of $T(y)$. $T(y)$ is referred to as the *subtree of* $T(P)$ *rooted at* y. In Figure 7, $T(y)$, for the intra-bisector point $y \in b(p_1, p_2)$, is shown in dashed bold lines. Let $\mathcal{K}_y$ be the P-circle centered at y. Chord $\overline{p_ip_j}$ partitions $\mathcal{K}_y$ in two parts $\mathcal{K}_y^f$ and $\mathcal{K}_y^r$, where $\mathcal{K}_y^r$ is the *rear* part, enclosing the portion of $CH(P)$ inducing $T(y)$, and $\mathcal{K}_y^f$ is the *forward* part, enclosing the portion of $CH(P)$ inducing $T_c(y)$. Figure 7 depicts $\mathcal{K}_y^f \cup CH(P)$ shaded.

Definition 10. A shape $Q \in S$ is called *limiting* with respect to chord $\overline{p_ip_j} \in P$, if Q is enclosed within a P-circle $\mathcal{K}$ passing through p_i, p_j, and Q is non-crossing with $\overline{p_ip_j}$. (Note that Q may be limiting with respect to $\overline{p_ip_j}$, but still be crossing with P due to some other $\overline{p_kp_l}$). Q is called *forward limiting* if $Q \in \mathcal{K}_y^f \cup CH(P)$ or *rear limiting* if $Q \in \mathcal{K}_y^r \cup CH(P)$, where y is a point on the intra-bisector segment of $\overline{p_ip_j}$.

Note that any shape enclosed in a P-circle through $\overline{p_ip_j}$ must be forward limiting, rear limiting or crossing with $\overline{p_ip_j}$. In Figure 7, shape Q_1 is rear limiting and shape Q_2 is forward limiting with respect to y and chord $\overline{p_1p_2}$.

Lemma 2. *Let $Q \in S$ be a limiting shape with respect to chord $\overline{p_ip_j} \in P$ and let $y \in T(P)$ be the center of a P-circle enclosing Q. If Q is forward (resp. rear) limiting then the entire $T(y)$ (resp. $T_c(y)$) is closer to Q than to P.*

Proof. Let $y_k \in T(y)$. P-circle $\mathcal{K}_{y_k}$ must pass through $p_i, p_j \in CH(P)$ such that $p_i, p_j \in \mathcal{K}_y^r$. But then $\mathcal{K}_y^f \subset \mathcal{K}_{y_k}$. Similarly, it is easy to see that $\mathcal{K}_y^r \subset \mathcal{K}_{y_l}$, for any $y_l \in T_c(y)$. By definition, $CH(P) \subset \mathcal{K}_x$ for any $x \in T(P)$. Thus, the lemma follows by the definition of forward and rear limiting shapes. □

Lemma 2 implies a sufficient condition for a Voronoi region to be empty which is given in Property 5. In the case of non-crossing shapes it is easy to see that the condition is also necessary. For the subclass of disjoint convex shapes the condition had also been identified in Ref. [1]. Property 6 can be easily derived by the proof of Lemma 2 and it will be used throughout the time complexity analysis of our algorithm.

Property 5. *H-Vor(P) = ∅ if there exist two limiting shapes Q, R, enclosed in the same P-circle $\mathcal{K}_y$, $y \in b(p_i, p_j)$, $\overline{p_i p_j} \in P$, such that $Q \in \mathcal{K}_y^f \cup CH(P)$ and $R \in \mathcal{K}_y^r \cup CH(P)$. In case of a non-crossing S the condition is also necessary (except from the trivial case where CH(P) entirely contains another shape).*

Property 6. *Any rear limiting shape with respect to a chord $\overline{p_i p_j} \in P$ must be enclosed in the minimum enclosing circle of P.*

We now derive an improved upper bound on the structural complexity of *H-Vor(S)*. Recall that the case of interest for our application is the case of non-crossing polygons and the case of polygons with only a small number of crossings.

Theorem 1. *The Hausdorff Voronoi diagram of an arbitrary set of polygons S has size $O(n+m)$, where n is the number of vertices on the convex hulls of shapes in S and $m = \Sigma_{(P,Q)} m(P,Q)$, where $m(P,Q)$ is the number of crossing mixed vertices[c] on $b_{\mathrm{f}}(P,Q)$ for any pair of crossing shapes (P,Q). In the worst case, m is $O(n^2)$. In the case of a non-crossing S, there is at most one connected Voronoi region for each polygon $P \in S$, and H-Vor(S) has size $O(n)$.*

Proof. By Euler's theorem for planar graphs and since any vertex of *H-Vor(S)* has degree at least 3, the size of *H-Vor(S)* must be proportional to the number of its faces. By Property 3, each face $hreg(p), p \in P$, is bounded by exactly one intra-bisector chain whose endpoints are mixed Voronoi vertices or extend to infinity. Hence, the size of *H-Vor(S)* is proportional to the total number of mixed Voronoi vertices in *H-Vor(S)*, including those at infinity. A mixed Voronoi vertex y is the center of a P-circle, $\mathcal{K}_y$, passing through the endpoints of chord $\overline{p_i p_j} \in P$ and a vertex $q \in Q$, $Q \neq P$, such that both P and Q are enclosed in $\mathcal{K}_y$. Vertex y can be of two types: *crossing*, if Q is crossing $\overline{p_i p_j}$, or *non-crossing*, otherwise. In the latter case Q must be limiting with respect to $\overline{p_i p_j}$. By Lemma 2, if Q is forward (resp. rear) limiting then the entire $T(y)$ (resp. $T_c(y)$) gets eliminated from *H-Vor(S)*. Thus, the number of non-crossing mixed Voronoi vertices on $T(P)$ is upper-bounded by $|T(P)|$. Hence, the total number of non-crossing mixed Voronoi vertices is $O(n)$. Since any crossing mixed Voronoi vertex in *H-Vor(S)* corresponds to a crossing vertex of some inter-bisector $b_{\mathrm{f}}(P,Q)$, the bound is derived. By Lemma 1 any mixed vertex is part of *f-Vor*$(P \cup Q)$ and thus the total number of mixed Voronoi vertices cannot exceed $O(n^2)$.

For a non-crossing S, there are no crossing mixed Voronoi vertices and thus the size of *H-Vor(S)* is $O(n)$. The connectivity of Voronoi regions follows from Properties 2 and 3. □

The bound of Theorem 1 has been refined to a tight bound in a recent companion paper.[13] In Ref. [13] it is shown that $m = \Theta(m_r)$, where m_r is the number of crossing

[c] A mixed vertex v induced by points $p_i, p_j \in P$ and $q_r \in Q$ is called *crossing* if Q is crossing with $\overline{p_i p_j}$.

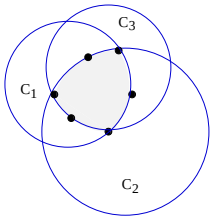

Fig. 8. Proof of Lemma 3.

mixed Voronoi vertices that are *rear*, where a mixed Voronoi vertex v induced by $p_i, p_j \in P$ and $q_r \in Q$ is called *rear* (resp. *forward*) if $q_r \in \mathcal{K}_v^r$ (resp. $q_r \in \mathcal{K}_v^f$). It is also shown that every rear mixed Voronoi vertex on $T(P)$ corresponds to a unique pair of supporting segments between $CH(P)$ and $CH(Q)$, called *crucial*, entirely enclosed in the minimum enclosing circle of P. In Ref. [13] an example is given showing that the size of *H-Vor(S)* is $\Omega(n + m_r)$, that is, $\Omega(n + m)$ as $m = \Theta(m_r)$.

Hausdorff Voronoi regions are disconnected in general and thus the methodology of abstract Voronoi diagrams is not applicable for an arbitrary set S. If S is non-crossing, however, the conditions of abstract Voronoi diagrams (as given in Ref. [9]) are satisfied (see Property 1, Corollary 1, Theorem 1, and Property 3) and thus generic techniques for the construction of abstract Voronoi diagrams can be applied. The property required by abstract Voronoi diagrams that any two bisecting curves can intersect at most a constant number of times is shown in the following Lemma. The direct algorithm given in the following section is more efficient however.

Lemma 3. *The farthest bisectors of three pairwise non-crossing polygons P, Q, R, $b_{\mathrm{f}}(P, Q)$ and $b_{\mathrm{f}}(P, R)$, may intersect at most twice.*

Proof. Suppose to the contrary that there exist three pairwise non-crossing shapes, P, Q, R, such that $b_{\mathrm{f}}(P, Q)$ and $b_{\mathrm{f}}(P, R)$ intersect three times. Let $I_i, i = 1, 2, 3$, denote the intersection points. Then I_i corresponds to the center of a circle C_i defined by three points, one on each shape, such that C_i entirely encloses all three shapes. That is, P, Q and R must be totally enclosed within the common intersection of the circles, $A = \cap_i C_i$. A is shown shaded in Figure 8. Furthermore, each circular arc bounding area A must contain at least one point from each shape. But then shapes P, Q, R cannot be non-crossing; a contradiction. □

4. A Divide and Conquer Algorithm

Given a vertical dividing line $\mathcal{L}$, let S_l and S_r be the sets of shapes in S to the left and to the right respectively of $\mathcal{L}$, where a shape P is said to be to the *left* (resp. *right*) of $\mathcal{L}$ if the leftmost x-coordinate of P is to the left (resp. right) of $\mathcal{L}$. Note that unlike shapes in S_r, shapes in S_l may intersect $\mathcal{L}$. Let's assume that *H-Vor*(S_l) and *H-Vor*(S_r) have been computed. We shall compute *H-Vor*(S) by merging *H-Vor*(S_l) and *H-Vor*(S_r). Let $\sigma(S_l, S_r)$ denote the *merge curve* between *H-Vor*(S_l)

and *H-Vor*(S_r), that is, $\sigma(S_l, S_r)$ is the collection of bisectors $b(p_l, p_r)$ in *H-Vor*(S) such that $p_l \in S_l$ and $p_r \in S_r$. The merge curve has the following property.

Lemma 4. *The merge curve $\sigma(S_l, S_r)$ is a collection of edge disjoint unbounded chains and cycles. The cycles can only enclose regions of shapes in S_l that intersect the dividing line and regions of shapes in S_r that are crossing with shapes in S_l (if any). $\sigma(S_l, S_r)$ must consist of at least one unbounded chain unless HH(S)=HH(S_r).*

Proof. The first statement is derived by the the fact that *H-Vor*(S) is a two-colorable map, similarly to the ordinary Voronoi diagram case.[18] Suppose now that shapes in S_r are non-crossing with shapes in S_l. Let τ be a cycle in $\sigma(S_l, S_r)$ and let S_τ denote the subset of S consisting of shapes whose regions border τ or are enclosed by τ. Suppose to the contrary that τ encloses regions of S_r. Consider *H-Vor*(S_τ). Since any region enclosed by τ is bounded, all regions of shapes in $S_\tau \cap S_r$ must be bounded. Let Q_r be the rightmost shape in $S_\tau \cap S_r$ (The x-coordinate of a shape is implied by its leftmost point). Q_r is the only shape in S_τ entirely lying to the right of the vertical line through the leftmost x-coordinate of Q_r. Thus, Q_r must be on the *H-hull* of S_τ. That is, *hreg*(Q_r) must be unbounded, a contradiction. Thus, τ cannot enclose any regions of shapes in S_r that are non-crossing with shapes in S_l. It can be shown similarly that τ can not enclose the region of any shape P in S_l unless P intersects the dividing vertical line $\mathcal{L}$ between S_l and S_r.

By Corollary 2, any unbounded inter-bisector of $\sigma(S_l, S_r)$ corresponds to a supporting segment $\overline{p_i q_j}$ of the *H-hull* of S such that $p_i \in S_l$ and $q_j \in S_r$. Thus, if $HH(S) \neq HH(S_l)$ and $HH(S) \neq HH(S_r)$, $\sigma(S_l, S_r)$ must consist of at least one unbounded portion. Otherwise, $\sigma(S_l, S_r)$ can have no unbounded portion and thus $\sigma(S_l, S_r)$ must consist solely of cycles. But since the entire S_r is to the right of the dividing line $\mathcal{L}$, *HH(S)* must contain at least one vertex in S_r that is, $HH(S) \neq HH(S_l)$. The same need not hold for S_l if no shape in S lies entirely to the left of $\mathcal{L}$. Hence, $\sigma(S_l, S_r)$ contains no unbounded chain if and only if $HH(S) = HH(S_r)$. □

Figure 9 and Figure 10 indicate that $\sigma(S_l, S_r)$ may indeed contain cycles. In particular, Figure 9 shows *H-Vor(S)*, $S = \{P_1, P_2, Q_1, Q_2\}$ with *hreg*(P_2) depicted shaded. Dividing S as $S_l = \{P_1, P_2\}$ and $S_r = \{Q_1, Q_2\}$ shows that $\sigma(S_l, S_r)$ can have cycles enclosing regions of shapes in S_l that intersect $\mathcal{L}$. In Figure 10 the shaded regions depict *hreg*(Q). Dividing S as $S_l = \{P_1, P_2\}$ and $S_r = \{Q\}$ shows that, in case of crossing shapes, a cycle in $\sigma(S_l, S_r)$ can enclose regions of shapes in S_r. Identifying the cycles of $\sigma(S_l, S_r)$ poses the main computational difficulty of the divide and conquer algorithm.

To trace the merge curve $\sigma(S_l, S_r)$ we need to identify a starting point on every connected component of $\sigma(S_l, S_r)$. Then each component can be traced in linear time as in the ordinary Voronoi diagram case. The tracing algorithm for the Hausdorff Voronoi diagram however is different as the properties that induce

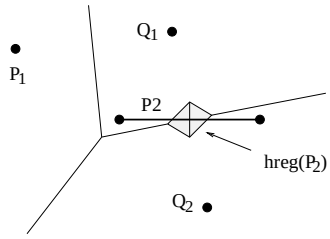

Fig. 9. *H-Vor(S)*, $S = \{P_1, P_2, Q_1, Q_2\}$.

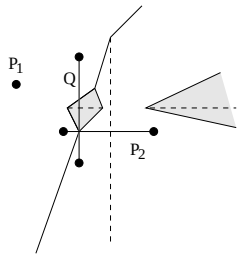

Fig. 10. *H-Vor(S)*, $S = \{P_1, P_2, Q\}$.

the linear-time tracing in the ordinary Voronoi diagram case (see e.g. Ref. [18]) are no longer valid. Furthermore, the same Voronoi region may participate in the tracing of several components of $\sigma(S_l, S_r)$. To maintain the linear time complexity of the tracing phase we exploit Properties 2 and 3 of the previous section. In the following, let $P \in S$ be a shape whose region may be enclosed in a cycle. To avoid differentiating between S_l and S_r, let $S_P = S_l$ and $S_Q = S_r$ (resp. $S_P = S_r$ and $S_Q = S_l$) if $P \in S_l$ (resp. $P \in S_r$). To distinguish between the Voronoi region(s) of a shape P in *H-Vor*(S_P) and *H-Vor*(S) we use a prime. That is, *hreg*(p), $p \in P$, denotes the region of p in *H-Vor*(S) and *hreg*$'(p)$, denotes the region of p in *H-Vor*(S_P). The following property shows that a linear-time tracing of the components $\sigma(S_l, S_r)$ is possible.

Lemma 5. *For any point $x \in$ hreg$'(p)$, $\sigma(S_l, S_r)$ may intersect segment $\overline{px} \cap$ hreg$'(p)$ at most once.*

Proof. Let y be the intersection point $\overline{px} \cap$ *freg*(p). By Property 2, segment $\overline{yx}$ lies entirely in *hreg*$'(p)$ ($\overline{yx} = \overline{px} \cap$ *hreg*$'(p)$). Suppose to the contrary that $\sigma(S_l, S_r)$ intersects $\overline{yx}$ twice at points r_1 and r_2 such that r_1 is closer to P than r_2 and there are no other intersection points on $\overline{r_1 r_2}$. Then either both $\overline{yr_1}$ and $\overline{r_2 x}$ or only $\overline{r_1 r_2}$ must remain in *hreg*(p). But neither option is possible as *hreg*(p) must satisfy Property 2. □

Definition 11. The *visibility-based decomposition* of *hreg*$(p), p \in P$ is a decom-

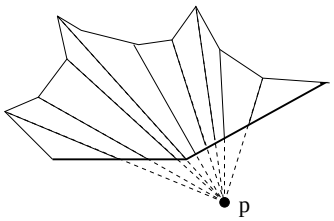

Fig. 11. The visibility-based decomposition of $hreg(p)$.

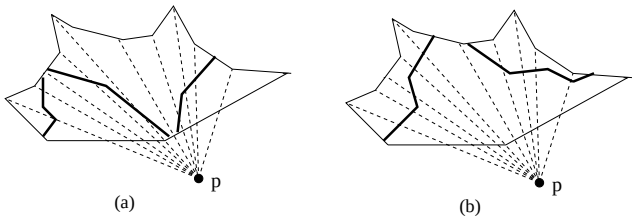

Fig. 12. Examples of $\sigma(S_l, S_r)$ in $hreg(p)$.

position of $hreg(p)$ into *quasi-triangles* obtained by drawing segments $\overline{pv}$ for every vertex v on the boundary of $hreg(p)$ (see Figure 11).

The tracing of any component ρ of $\sigma(S_l, S_r)$ can be performed using the visibility-based decomposition. Recall that during tracing we will always follow a bisector $b_f(p, q)$ in regions $hreg'(p)$ and $hreg'(q)$, $p \in S_P, q \in S_Q$. While in region $hreg'(p)$ and $hreg'(q)$, ρ moves from quasi-triangle to quasi-triangle until it hits the boundary of either $hreg'(p)$ or $hreg'(q)$. Every time ρ hits the boundary of a region, a vertex (mixed or inter-vertex) of ρ is determined and the tracing continues at the neighboring region. By Lemma 5, no backtracking of ρ to an already visited quasi-triangle is possible. Also by Lemma 5, the quasi-triangles visited by different components of $\sigma(S_l, S_r)$ (except from those containing endpoints of different components) must be disjoint. Thus, the tracing of all components of $\sigma(S_l, S_r)$ can be done in linear time even in case of regions intersected by multiple components of $\sigma(S_l, S_r)$. Figure 12 illustrates examples of $\sigma(S_l, S_r)$, shown in bold lines, in a connected component of $hreg'(p)$, in which the turning points occur at the boundary of $hreg'(q)$ for some q . Note that after tracing $\sigma(S_l, S_r)$, $hreg'(p)$ may be decomposed into multiple disjoint areas (see e.g. Figure 12(a)).

The remaining problem for computing $\sigma(S_l, S_r)$ is to determine a starting point on every component. A starting point on the unbounded portions of $\sigma(S_l, S_r)$ can be determined by the H-hull as shown by Corollary 2. As it will be shown in Section 4.1, the H-hull of S can be derived by merging $HH(S_l)$ and $HH(S_r)$ in linear time. Thus, starting points on the unbounded portions of $\sigma(S_l, S_r)$ can be computed in linear time. In Section 4.2 we show how to identify a starting point on every cycle of $\sigma(S_l, S_r)$. Identifying starting points for cycles is the main computational

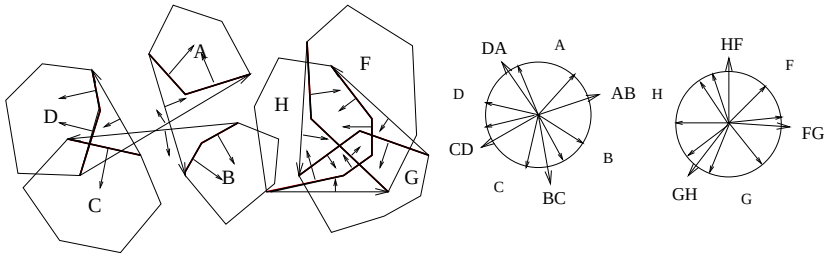

Fig. 13. The Gaussian maps of $HH(\{A, B, C, D\})$ and $HH(\{F, H, G\})$.

bottleneck of our algorithm.

4.1. *Merging two H-hulls*

The H-hull of S is represented by an ordered (e.g. clockwise) list of its vertices. An alternative representation can be obtained by the *Gaussian Map*[3] of S, for short *GMap(S)*, onto the unit circle. In the Gaussian Map every H-hull edge e_i is mapped to a point $\nu(e_i)$ on the circumference of a unit circle $\mathcal{K}_o$ as obtained by the outward pointing unit normal. That is, point $\nu(e_i)$ represents a normal vector pointing in the half-plane bordered by the supporting line ℓ_{e_i} through e_i away from the H-hull i.e., towards the interior of the shape(s) where the endpoints of e_i belong. We will use the same notation to denote both the vector and the corresponding point in the GMap. The arc of $\mathcal{K}_o$ from $\nu(e_i)$ to $\nu(e_{i+1})$ for any two consecutive H-hull edges e_i and e_{i+1} is denoted by $\alpha(m_i)$, where m_i is the common H-hull vertex between e_i and e_{i+1}. Figure 13 illustrates two H-hulls and their respective Gaussian maps. In the Gaussian maps of Figure 13 the supporting segments are represented by longer arrows and are marked by the names of the two shapes they support. In the following, unless explicitly noted otherwise, we assume a clockwise ordering of both the H-hull and the GMap of S.

Lemma 6. *The normal vectors in GMap(S) appear in the same cyclic order (e.g. clockwise) as the respective cyclic traversal of the boundary of HH(S).*

Proof. Let's assume without loss of generality that $HH(S)$ follows a clockwise ordering. Consider the GMap points $\nu(e_i)$ and $\nu(e_{i+1})$ of any two consecutive edges $e_i = \overline{m_{i-1}m_i}$ and $e_{i+1} = \overline{m_i m_{i+1}}$ of the $HH(S)$. We shall show that $\nu(e_i)$ and $\nu(e_{i+1})$ are also consecutive on $\mathcal{K}_o$. Let $P_i \in S$ be the shape containing vertex m_i and let l_{e_i}, $l_{e_{i+1}}$ denote the supporting lines through e_i and e_{i+1} respectively. By definition, $l_{e_{i+1}}$ can be reached by rotating l_{e_i} clockwise around m_i. Thus, any point along the arc of $\mathcal{K}_o$ between $\nu(e_i)$ and $\nu(e_{i+1})$ corresponds to the normal vector of a supporting line l_{m_i}, through m_i, with slope in-between the slopes of l_{e_i} and $l_{e_{i+1}}$. Suppose to the contrary that there is an H-hull edge e_j, $e_j \neq e_i, e_{i+1}$, parallel to l_{m_i} such that $\nu(e_j)$ is in between $\nu(e_i)$ and $\nu(e_{i+1})$. Then e_j must be supporting to at least one different shape P_j. However, if e_j were in the same side of l_{m_i} as P_i,

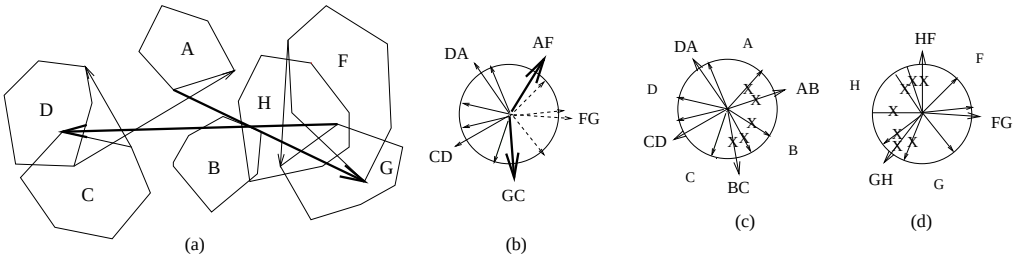

Fig. 14. Merging $HH(S_l)$ and $HH(S_r)$.

m_i would not be an H-hull vertex. Similarly, if e_j were at the opposite side of l_{m_i}, e_j could not be an H-hull edge. Thus, e_j must be on l_{m_i} i.e., it must be incident to m_i, a contradiction. □

Corollary 3. *The cyclic ordering of the arcs around $\mathcal{K}_o$ coincides with the cyclic ordering of the vertices of $HH(S)$.*

By Lemma 6 and Corollary 3 it is clear that merging two H-hulls is equivalent to merging their Gaussian maps. Because of the cyclic ordering, merging the Gaussian maps of two H-hulls becomes similar to merging two ordinary convex hulls. An edge or vertex of $HH(S_l)$ and $HH(S_r)$ is called *valid* if it also belongs in *HH(S)*, otherwise it is called *invalid*. To merge $GMap(S_l)$ and $GMap(S_r)$ we simply need to identify their valid portions, merge them in cyclic order, and add vectors of the new supporting segments between $HH(S_l)$ or $HH(S_r)$. Figure 14 illustrates the merging process of the two H-hulls appearing in Figure 13. In particular, Figure 14(a) illustrates $HH(S), S = S_l \cup S_r, S_l = \{A, B, C, D\}, S_r = \{G, F, H\}$, Figure 14(b) illustrates $GMap(S)$, and Figures 14(c) and 14(d) illustrate $GMap(S_l)$ and $GMap(S_r)$ respectively. In Figures 14(c) and 14(d) the invalid vectors are shown crossed out. The new vectors corresponding to the new supporting segments between $HH(S_l)$ and $HH(S_r)$ are indicated by thicker arrows.

We now present the details of the merging process. For an H-hull edge $e \in HH(S_l)$ (resp. $HH(S_r)$) let the *neighboring* edges of e in $HH(S_r)$ (resp. $HH(S_l)$) be $f_j, f_{j+1} \in HH(S_r)$ (resp. $HH(S_l)$) such that $\nu(e)$ is located between $\nu(f_j)$ and $\nu(f_{j+1})$ in a clockwise traversal of $GMap(S_r)$ (resp. $GMap(S_l)$). The common vertex m_j of (f_j, f_{j+1}) is said to be the vertex *neighboring* e in $HH(S_r)$ (resp. $HH(S_l)$). Note that $\nu(e)$ falls on arc $\alpha(m_j)$.

Lemma 7. *Edge $e \in HH(S_l)$ (resp. $HH(S_r)$) remains valid iff the vertex q neighboring e in $HH(S_r)$ (resp. $HH(S_l)$) lies on the opposite side of the supporting line l_e through e as vector $\nu(e)$ i.e., q lies on the side of l_e towards the interior of the H-hull.*

Proof. Let $e \in HH(S_l)$ and let f_j, f_{j+1} be the neighboring edges of e in $HH(S_r)$. Let q be the common endpoint of f_j and f_{j+1} and let $q \in Q$. Let e have an

endpoint on shape P and let l_e denote the line through e. Since $\nu(e)$ defines a point on $\alpha(q)$, the line l_q parallel to l_e passing through q must be supporting $CH(Q)$ and must leave all shapes in $S_r \setminus Q$ entirely or partially at opposite side of l_q as Q. Thus, if q lies on opposite sides of l_e as P then all shapes in S_r (including Q) must lie partially or entirely on opposite sides of l_e as P. Thus, e must remain a valid edge. If q lies on the same side of l_e as P then Q must also lie entirely on the same side. Thus, at least one endpoint of e can not remain on the H-hull of $S_r \cup S_l$ and thus, e must be invalid. □

Lemma 8. *Let e_i, e_{i+1} be two consecutive H-hull edges in $HH(S_l)$ (resp. $HH(S_r)$). If at least one of e_i, e_{i+1} remains valid then their common vertex m_i also remains valid. If both e_i and e_{i+1} are invalid, then m_i remains valid iff there is an invalid edge $f_j \in HH(S_r)$ (resp. $HH(S_l)$) such that $\nu(f_j)$ is between $\nu(e_i)$ and $\nu(e_{i+1})$.*

Proof. The first statement is trivially true as both endpoints of an H-hull edge must be H-hull vertices. Suppose that e_i and e_{i+1} are invalid. Let $m_i \in P$, $P \in S_l$. The following notation is used. Let l_e denote the line through any H-hull edge e. Let $H(e)$ (resp. $H(l)$) denote the half plane on the same side of l_e (resp. line l) as the shape supported by l_e (resp. l).

Suppose that there is a vector $\nu(f_j) \in GMap(S_r)$, such that m_i is the neighboring vertex of f_j in $GMap(S_l)$ i.e., $\nu(f_j)$ is located between $\nu(e_i)$ and $\nu(e_{i+1})$, and suppose that f_j has been determined invalid. Since f_j is invalid, $P \in H(f_j)$. Since $f_j \in HH(S_r)$, no shape in S_r can lie in $H(f_j)$. Since $\nu(f_j)$ is between $\nu(e_i)$ and $\nu(e_{i+1})$ the line through m_i parallel to l_{f_j}, denoted as l_{m_i}, must fall between l_{e_i} and $l_{e_{i+1}}$ and must be supporting to P. Thus, no other shape in S_l can lie at the same side of l_{m_i} as P. Since no shape in S_r can be in $H(f_j)$, no shape in S_r can lie at the same side of l_{m_i} as P. Thus, m_i must be a valid vertex in the H-hull of S.

Conversely suppose that m_i is valid (see Figure 15). Then there is a line l_{m_i} that is supporting P at m_i, l_{m_i} lies between l_{e_i} and $l_{e_{i+1}}$, and l_{m_i} leaves no shape in S other than P on the same side as P. Figure 15 shows several combinations of e_i, e_{i+1} as supporting or convex hull edges. In Figure 15(a), e_i is a supporting segment and e_{i+1} is convex hull edge. In Figures 15(b) and 15(c), both e_i and e_{i+1} are convex hull edges and supporting segments respectively. Since e_i and e_{i+1} are both invalid there must be two shapes $Q_1, Q_2 \in S_r, Q_1 \neq Q_2$, such that $Q_1 \in H(e_i), Q_2 \in H(e_{i+1})$, and the supporting segment between Q_1 and Q_2 is an H-hull edge $f_j \in HH(S_r)$. But Q_1 and Q_2 must lie at least partially in the complement of $H(l_{m_i})$, since P is the only shape in S lying entirely in $H(l_{m_i})$. Thus, the endpoints of f_j must be in $H(e_i) \setminus H(l_{m_i})$ and $H(e_{i+1}) \setminus H(l_{m_i})$ respectively. This area is shown shaded in Figure 15. Hence, P must lie entirely on the same side of l_{f_j}, as Q_1, Q_2, and the slope of f_j must be between the slopes of e_i and e_{i+1}. Thus, f_j is invalid and $\nu(f_j)$ is between $\nu(e_i)$ and $\nu(e_{i+1})$. □

Lemma 7 and Lemma 8 indicate an algorithm to determine the valid vertices

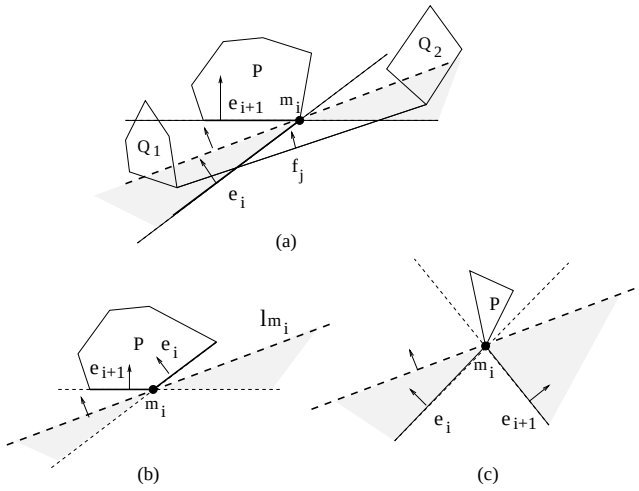

Fig. 15. Proof of Lemma 8.

of $HH(S_l)$ and $HH(S_r)$. Those are exactly the vertices of the H-hull of S since any such vertex must necessarily be a valid vertex in $HH(S_l)$ or $HH(S_r)$. To complete the construction, for any pair of consecutive valid H-hull vertices (p_i, q_j) such that $p_i \in S_l$ (resp. $p_i \in S_r$) and $q_j \in S_r$ (resp. $q_j \in S_l$), segment $\overline{p_i q_j}$ must be added to *HH(S)* and $\nu(\overline{p_i q_j})$ must be added to $GMap(S)$. In detail, let $GMap_v(S_l)$ (resp. $GMap_v(S_r)$) denote the valid portion of $GMap(S_l)$ (resp. $GMap(S_r)$) where valid vectors remain unchanged and invalid vectors represent their neighboring vertex in S_r (resp. S_l) that is, the vertex in S_r that made them invalid. Consider $GMap_v(S_l) \cup GMap_v(S_r)$ as a cyclic collection of valid and invalid vectors around the origin o. As the following lemma shows, $GMap_v(S_l) \cup GMap_v(S_r)$ indicates the ordering of the vertices of *HH(S)*. Note that any valid vector in $GMap_v(S_l)$ (resp. $GMap_v(S_r)$) corresponds to an ordered pair of valid vertices in S_l (resp. S_r) and any invalid vector corresponds to the neighboring valid vertex in S_r (resp. S_l) that made it invalid (see Lemma 9). $GMap(S)$ can now trivially be derived from $GMap_v(S_l) \cup GMap_v(S_r)$ by adding $\nu(\overline{p_i q_j})$ for any pair of consecutive vertices (p_i, q_j) such that $p_i \in S_l$ (resp. $p_i \in S_r$) and $q_j \in S_r$ (resp. $q_j \in S_l$) and deleting all invalid vectors.

Lemma 9. *For any pair of consecutive vertices (p_i, q_j) indicated by the cyclic ordering of $GMap_v(S_l) \cup GMap_v(S_r)$, segment $\overline{p_i q_j}$ must be an edge on the H-hull of S.*

Proof. Consider an invalid vector $\nu(e_i) \in GMap(S_l)$ and let $q_j \in Q$ be the neighboring vertex of e_i in $GMap(S_r)$. By Lemma 8, q_j is a valid vertex and the line l_{q_j} through q_j of slope normal to $\nu(e_i)$ must be supporting to Q, with Q being the only shape at the same side of l_{q_j} as $\nu(e_i)$. Thus, $\alpha(q_j) \in HH(S)$ must contain the endpoint of $\nu(e_i)$. Thus, $GMap_v(S_l) \cup GMap_v(S_r)$ indicates the vertices of *HH(S)*

in their correct cyclic order. Since no additional vertex that is not represented in $GMap_v(S_l) \cup GMap_v(S_r)$ can be part of *HH(S)*, the segment joining any two consecutive vertices must be an H-hull edge. □

Lemma 7, Lemma 8 and Lemma 9 clearly indicate a linear time algorithm to derive $HH(S)$ from $HH(S_l)$ and $HH(S_r)$. We thus conclude with the following theorem.

Theorem 2. *Merging $HH(S_l)$ and $HH(S_r)$ into HH(S) can be computed in linear time.*

4.2. *Tracing the cycles of $\sigma(S_l, S_r)$*

Suppose that the unbounded components of $\sigma(S_l, S_r)$ have been traced. Our goal is to identify a starting point on every cycle of $\sigma(S_l, S_r)$. Recall (Lemma 4) that cycles can only enclose regions of shapes in S_l that intersect the dividing line $\mathcal{L}$ and regions of shapes in S_r that are crossing with shapes in S_l (if any). Let this set of shapes be denoted as $S_{\mathcal{L}}$.

Let $H\text{-}Vor_\sigma(S)$ denote the intermediate diagram derived by merging the components of $\sigma(S_l, S_r)$ computed so far with the relevant portions of $H\text{-}Vor(S_l)$ and $H\text{-}Vor(S_r)$, and let $H\text{-}Vor^*_\sigma(S)$ denote the complement diagram derived by merging the components of $\sigma(S_l, S_r)$ computed so far with the remaining portions of $H\text{-}Vor(S_l)$ and $H\text{-}Vor(S_r)$. That is, $H\text{-}Vor^*_\sigma(S) = H\text{-}Vor(S_l) \cup H\text{-}Vor(S_r) - H\text{-}Vor_\sigma(S)$ plus the components of $\sigma(S_l, S_r)$ computed so far. When all components of $\sigma(S_l, S_r)$ are computed, $H\text{-}Vor_\sigma(S) = H\text{-}Vor(S)$, and $H\text{-}Vor^*_\sigma(S) = H\text{-}Vor^*(S)$, where $H\text{-}Vor^*(S)$ denotes the final diagram to be discarded i.e., $H\text{-}Vor^*(S) = H\text{-}Vor(S_l) \cup H\text{-}Vor(S_r) \cup \sigma(S_l, S_r) - H\text{-}Vor(S)$. When a new cycle τ of $\sigma(S_l, S_r)$ is determined, the regions of $H\text{-}Vor^*_\sigma(S)$ enclosed in τ become part of $H\text{-}Vor_\sigma(S)$ and the respective regions of $H\text{-}Vor_\sigma(S)$ become part of $H\text{-}Vor^*_\sigma(S)$. $H\text{-}Vor_\sigma(S)$ and $H\text{-}Vor^*_\sigma(S)$ are well defined initially if $\sigma(S_l, S_r)$ consists of at least one unbounded merge curve. Otherwise, $H\text{-}Vor_\sigma(S)$ and $H\text{-}Vor^*_\sigma(S)$ are initialized to $H\text{-}Vor(S_r)$ and $H\text{-}Vor(S_l)$ respectively as indicated by Lemma 4. Every time a component of $\sigma(S_l, S_r)$ is identified, both $H\text{-}Vor_\sigma(S)$ and $H\text{-}Vor^*_\sigma(S)$ get updated. At the end, when no more cycles can be determined, $H\text{-}Vor^*(S)$ can be safely discarded.

Consider an intra-bisector segment $\overline{y_i y_j} \in H\text{-}Vor^*_\sigma(S)$ induced by $p_i, p_j \in P$, $P \in S_{\mathcal{L}}$, such that y_i is an ancestor of y_j in $T(P)$ (i.e., $d_{\mathrm{f}}(y_i, P) < d_{\mathrm{f}}(y_j, P)$). Let S_P denote the portion of S containing P and S_Q denote the complement of S_P, that is, let $S_P = S_l$, $S_Q = S_r$ (resp. $S_P = S_r$, $S_Q = S_l$) if $P \in S_l$ (resp. $P \in S_r$). Let $q_i \in Q_i$ and $q_j \in Q_j$ be the owners of y_i and y_j respectively in $H\text{-}Vor(S_Q)$ i.e., $y_i \in hreg'(q_i)$ and $y_j \in hreg'(q_j)$, where Q_i and Q_j may or may not be equal. (Recall that a prime is used to distinguish the Voronoi region(s) of a shape P in $H\text{-}Vor(S_P)$ from the Voronoi region(s) of P in $H\text{-}Vor(S)$). Comparing $d(y_k, p_k)$ and $d(y_k, q_k)$ for $k = i, j$, we can easily determine whether y_k is closer to P or Q_k, that is, whether $y_k \in hreg(P)$ or $y_k \in hreg(Q_k)$. If y_k is equidistant from P and Q_k and

y_k is not a point of an already computed component of $\sigma(S_l, S_r)$, then y_k must be the starting point of a new cycle. By default, if y_k is a point of an already computed component of $\sigma(S_l, S_r)$, y_k is assumed to be closer to P than its (equidistant) owner in S_Q. The following Lemma gives conditions under which a cycle may intersect $\overline{y_iy_j}$. It is important to note that $\overline{y_iy_j}$ is an intra-bisector segment of *H-Vor*$^*_\sigma(S)$ and not one of *H-Vor*$_\sigma(S)$.

Lemma 10. *Let $\overline{y_iy_j}$ be an intra-bisector segment in H-Vor$^*_\sigma(S)$ induced by $p_i, p_j \in P$, $P \in S_\mathcal{L}$, such that $d_f(y_i, P) < d_f(y_j, P)$.*

(1) If $y_i \in hreg(P)$ and $y_j \in hreg(Q_j)$ (resp. $y_j \in hreg(P)$ and $y_i \in hreg(Q_i)$) then there exists a cycle τ intersecting $\overline{y_iy_j}$ enclosing y_i (resp. y_j).

(2) If both $y_i, y_j \in hreg(P)$ and no shapes in S_Q are crossing with chord $\overline{p_ip_j}$, then there can be no cycle intersecting $\overline{y_iy_j}$; the entire $\overline{y_iy_j} \in$ H-Vor(S) and it must be enclosed in a merge cycle τ. Note that no conclusion regarding cycles intersecting $\overline{y_iy_j}$ can be made if $\overline{p_ip_j}$ is crossing with shapes in S_Q.

(3) Suppose $y_i \in hreg(Q_i)$ and $y_j \in hreg(Q_j)$. If $Q_i = Q_j$, or if Q_i is forward limiting, or if Q_j is rear limiting, or if $d_f(y_i, Q_j) < d_f(y_i, P)$, or if $d_f(y_j, Q_i) < d_f(y_j, P)$, then there can be no cycle intersecting $\overline{y_iy_j}$; the entire $\overline{y_iy_j}$ must remain in H-Vor$^*(S)$. Furthermore, if Q_i is forward limiting, $T(y_i) \notin$ H-Vor(S), and if Q_j is rear limiting, $T_c(y_j) \notin$ H-Vor(S).

(4) If $y_i \in hreg(Q_i)$ and $y_j \in hreg(Q_j)$ but the conditions of item 3 do not hold, then there may be a cycle τ intersecting $\overline{y_iy_j}$. This cycle must intersect $\overline{y_iy_j}$ twice (Figure 16(b)) or it must be part of a pair of nested cycles as shown in Figure 16(a),(c). In the non-crossing case, there is either exactly one cycle intersecting $\overline{y_iy_j}$ twice or $hreg(P) = \emptyset$.

Proof. The first item of this lemma is trivially true. When the conditions of item 2 are satisfied the entire $\overline{y_iy_j}$ must be in $hreg(P)$ as both $y_i, y_j \in hreg(P)$ and, by Lemma 2, no two components of $\sigma(S_l, S_r)$ may intersect $b(p_i, p_j)$ when no shapes in S_Q are crossing with $\overline{p_ip_j}$. Since $\overline{y_iy_j} \in$ *H-Vor*$^*_\sigma(S)$, the entire segment must be enclosed in a cycle of $\sigma(S_l, S_r)$. Let's now assume that the conditions of item 3 are satisfied. By Lemma 2, if Q_i is forward limiting for y_i then the entire $T(y_i)$ must be closer to Q_i than to P and thus, no cycle can intersect the entire $T(y_i)$. Since $\overline{y_iy_j} \in$ *H-Vor*$^*_\sigma(S)$, no portion of $T(y_i)$ can be part of *H-Vor*(S). Similarly if Q_j is rear limiting for y_j, no portion of $T_c(y_j)$ can be part of *H-Vor*(S). If $y_i \in hreg(Q_i)$, $y_j \in hreg(Q_j)$, and $d_f(y_j, Q_i) < d_f(y_j, P)$ (resp. $d_f(y_i, Q_j) < d_f(y_j, P)$), then $Q_i \in \mathcal{K}_{y_i}$ and $Q_i \in \mathcal{K}_{y_j}$ (resp. $Q_j \in \mathcal{K}_{y_j}$ and $Q_j \in \mathcal{K}_{y_i}$). But then, by the proof of Lemma 2, Q_i (resp. Q_j) must be enclosed in $\mathcal{K}_y$ for every $y \in \overline{y_iy_j}$, and thus, the entire $\overline{y_iy_j}$ must be closer to Q_i (resp. Q_j) than P, that is, $\overline{y_iy_j} \in$ *H-Vor*$^*(S)$. Similarly for $Q_i = Q_j$. Hence in all cases of item 3, no cycle can intersect or enclose $\overline{y_iy_j}$ and thus, $\overline{y_iy_j}$ must remain entirely in *H-Vor*$^*(S)$.

In case of item 4, there may be one or more cycles intersecting $\overline{y_iy_j}$. Let τ be such a cycle (if any). Since $y_i \in hreg(Q_i)$ and $y_j \in hreg(Q_j)$, either τ must intersect $\overline{y_iy_j}$ twice, as shown in Figure 16(b), or τ must be part of a nested pair of cycles

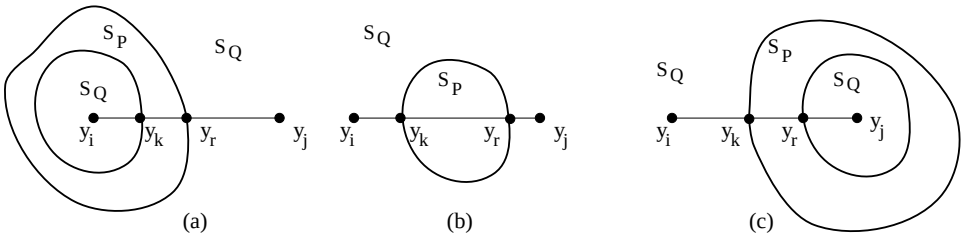

Fig. 16. A type 2 intra-bisector segment $\overline{y_i y_j} \in T(P)$ can be intersected by a single cycle twice or by a pair of nested cycles.

(τ, τ'), both intersecting $\overline{y_i y_j}$ once as shown in Figures 16(a),(c), such that the one is enclosed within the other. Note that in the general crossing case it is conceivable for a cycle of $\sigma(S_l, S_r)$ to be enclosed in another cycle. In the non-crossing case, Q_i and Q_j must be rear and forward limiting respectively, and thus we must have a P-circle $\mathcal{K}_y$, $y \in \overline{y_i y_j}$, which is either empty or it contains two limiting shapes $Q_r, Q_t \in S_Q$, such that one is rear limiting and the other is forward limiting. In the former case τ must intersect $\overline{y_i y_j}$ twice, and in the latter $hreg(P) = \emptyset$ (Property 5). □

By Lemma 10 we have two types of intra-bisector segments in $H\text{-}Vor^*_\sigma(S)$ that may contain the starting point of a cycle or be entirely enclosed in a cycle: those with at least one endpoint in $H\text{-}Vor(S)$ (see items 1,2 of Lemma 10), referred to as *type 1*, and those with no endpoint in $H\text{-}Vor(S)$ satisfying the conditions of item 4 of Lemma 10, referred to as *type 2*. A cycle intersecting or enclosing an intra-bisector segment of type 1 is referred to as a *cycle of type 1*, and a cycle intersecting an intra-bisector segment of type 2 is referred to as a *cycle of type 2*. Figure 16 depicts possible cycles of type 2. In the following we are giving algorithms to identify cycles of type 1 and type 2 respectively.

Algorithm 1 – Identify a cycle of type 1.

Let $\overline{y_i y_j} \in H\text{-}Vor^*_\sigma(S)$ be an intra-bisector segment of type 1, induced by $p_i, p_j \in P$, such that $y_i \in hreg(P)$. Locate y_i in $H\text{-}Vor(S_Q)$ and traverse segment $\overline{y_i y_j}$ through the visibility based decomposition of $H\text{-}Vor(S_Q)$ until a point y is reached that is equidistant from p_i and S_Q or $y = y_j$. In the former case point y is the starting point of a cycle intersecting $\overline{y_i y_j}$. In the latter case, $\overline{y_i y_j}$ becomes known to be entirely in $H\text{-}Vor(S)$, and thus it must be entirely enclosed in a cycle. In the latter case, lets assume that a point x on the boundary of the region of p_i in $H\text{-}Vor^*_\sigma(S)$ is known to be in $H\text{-}Vor^*(S)$. Let y be point $\overline{y_i y_j} \cap \overline{p_i x}$. It is easy to see that a cycle must intersect $\overline{yx}$. Traverse segment $\overline{yx}$ through the visibility-based decomposition of $H\text{-}Vor(S_Q)$, starting at y, until a point equidistant from p_i and S_Q (i.e., the starting point of a cycle), is encountered.

End

Algorithm 2 – Identify a cycle (or a pair of cycles) of type 2.

Let $\overline{y_iy_j} \in H\text{-}Vor_\sigma(S)$ be an intra-bisector segment of type 2. That is, $\overline{y_iy_j}$ is induced by $p_i, p_j \in P$, $y_i \in hreg(Q_i)$, $y_j \in hreg(Q_j)$, where $Q_i, Q_j \in S_Q$, Q_i is rear limiting or crossing with $\overline{p_ip_j}$, and Q_j is forward limiting or crossing with $\overline{p_ip_j}$. Furthermore, $Q_i \neq Q_j$, $d_f(y_i, Q_j) > d_f(y_j, P)$, and $d_f(y_j, Q_i) > d_f(y_j, P)$. Segment $\overline{y_iy_j}$ is assumed to be oriented from y_i to y_j, where y_i is the ancestor of y_j in $T(P)$ (i.e., $d_f(y_i, P) < d_f(y_j, P)$). Let y_k be the first intersection of $\overline{y_iy_j}$ with a cycle τ of $\sigma(S_l, S_r)$ (if any). Determine y_k as follows:

- Let $y'_k \in \overline{y_iy_j}$ be a point equidistant from P and Q_i (i.e., $y'_k = b_f(P, Q_i) \cap \overline{y_iy_j}$). Since $d_f(y_j, Q_i) > d_f(y_j, P)$ and $d_f(y_i, Q_i) < d_f(y_i, P)$, such a point must exist and no cycle can intersect $\overline{y_iy'_k}$. To determine y'_k we simply traverse $\overline{y_iy_j}$ against $f\text{-}Vor(Q_i)$. To bound the time complexity, the traversal starts at y_j and not at y_i (see Lemma 12 and Ref. [13]). Segment $\overline{y_iy'_k}$ is now known to be in $H\text{-}Vor^*(S)$.
- Locate y'_k in $H\text{-}Vor(S_Q)$ and let $q_k \in Q_k$ be the owner of y'_k. If $Q_k = Q_i$ then clearly $y_k = y'_k$. Otherwise proceed as follows:
  - If $\overline{y'_ky_j}$ satisfies the conditions of item 3 of Lemma 10 then there can be no cycle intersecting $\overline{y'_ky_j}$. The entire $\overline{y_iy_j}$ must be in $H\text{-}Vor^*(S)$.
  - Else the conditions of item 4 of Lemma 10 must be satisfied. Repeat the algorithm for segment $\overline{y'_ky_j}$.

If y_k exists, let y_r denote the next intersection of $\overline{y_iy_j}$ with a cycle (see Figure 16). By Lemma 10, y_r is either the second intersection of τ with $\overline{y_iy_j}$ (see Figure 16(b)) or y_r is the intersection of $\overline{y_iy_j}$ with a cycle τ' that forms a nested pair with τ (see Figure 16(a)(c)). Point y_r can be easily identified either as the second point of intersection of τ with $\overline{y_iy_j}$ in the case of Figure 16(b), or by applying Algorithm 1 on the type 1 segment $\overline{y_ky_j}$ in the case of Figures 16(a),(c).

End

Let's now discuss the entire algorithm to identify the cycles of $\sigma(S_l, S_r)$. For clarity, subdivisions $H\text{-}Vor_\sigma(S)$ and $H\text{-}Vor^*_\sigma(S)$ are assumed to be explicitly maintained throughout the algorithm and be augmented with their visibility-based decomposition. Alternatively, the computation of $H\text{-}Vor_\sigma(S)$ could be delayed for the end after all components of $\sigma(S_l, S_r)$ were computed. In that case, $H\text{-}Vor_\sigma(S)$ and $H\text{-}Vor^*_\sigma(S)$ could be maintained implicitly by augmenting $H\text{-}Vor(S_l)$ and $H\text{-}Vor(S_r)$ with the mixed Voronoi vertices of the components of $\sigma(S_l, S_r)$ computed so far. For clarity, in the following we explicitly maintain $H\text{-}Vor_\sigma(S)$ and $H\text{-}Vor^*_\sigma(S)$ as there is no loss in time complexity. At the end, $H\text{-}Vor_\sigma(S)$ is readily available and $H\text{-}Vor^*_\sigma(S)$ can be immediately discarded.

To identify cycles we simply traverse subdivision $H\text{-}Vor^*_\sigma(S)$ and make use of Lemma 10 to characterize intra-bisector segments as type 1, type 2, or mark them as permanently residing in $H\text{-}Vor^*(S)$. In particular, $H\text{-}Vor^*_\sigma(S)$ consists of two types of edges: those that are known to be in $H\text{-}Vor^*(S)$ and need not be consid-

ered again, and those that have not been determined yet i.e., those that may be enclosed in a new cycle of $\sigma(S_l, S_r)$. Any edge that is known to be in $H\text{-}Vor^*(S)$ gets *marked* as such. Otherwise it remains *unmarked*. Initially, all components of $\sigma(S_l, S_r)$ computed so far as well as all bisectors involving shapes of $S - S_{\mathcal{L}}$ can safely be *marked* as being in $H\text{-}Vor^*(S)$. The traversal of $H\text{-}Vor^*_\sigma(S)$ can start at any region adjacent to a marked edge or vertex. If $\sigma(S_l, S_r)$ consists of at least one unbounded component, there must be a number of marked edges in the beginning of the traversal. Otherwise, as indicated by Lemma 4, $H\text{-}Vor(S_l)$ is used to initialize $H\text{-}Vor^*_\sigma(S)$ and all its unbounded portions can be marked. The traversal proceeds by visiting regions neighboring an already marked edge. When a region is visited, the intra-bisector segments of that region get traversed, and get characterized as type 1, or type 2, or as being part of $H\text{-}Vor^*(S)$, by using Lemma 10 and performing point location of their endpoints in the opposite diagram. Algorithm 1 or Algorithm 2 is then applied to identify cycles. During this process, new edges of $H\text{-}Vor^*_\sigma(S)$ get marked as part of $H\text{-}Vor^*(S)$ and the traversal continues until there are no more edges in $H\text{-}Vor^*_\sigma(S)$ left unmarked. Every time a new cycle is identified, the cycle gets traversed as shown in Section 4 and both subdivisions $H\text{-}Vor^*_\sigma(S)$ and $H\text{-}Vor_\sigma(S)$ get updated by exchanging regions enclosed in that cycle. When all edges in $H\text{-}Vor^*_\sigma(S)$ get marked the search for cycles ends and $H\text{-}Vor^*(S)$ can be discarded. Note that once the intra-bisectors of a region have been marked as being in $H\text{-}Vor^*(S)$ the entire region can also be marked as such. Note also that it is enough to search $H\text{-}Vor^*_\sigma(S)$ for cycles as, by Lemma 2, any new cycle of $\sigma(S_l, S_r)$ must enclose at least one intra-bisector segment of $H\text{-}Vor^*_\sigma(S)$. In detail the algorithm is as follows:

Algorithm 3 – Determine the cycles of $\sigma(S_l, S_r)$.

Let $hreg'(p_i)$ be an (unmarked) region of $p_i \in P \in S_{\mathcal{L}}$ in $H\text{-}Vor^*_\sigma(S)$, neighboring a marked bisector segment $\overline{v_iv_j}$. Let $\overline{y_iy_j}$ be the intra-bisector segment of $hreg'(p_i)$ intersected by $\overline{p_iv_i}$ and let $\overline{y_iy_j}$ be a portion of $b(p_i, p_j)$. Segment $\overline{y_iy_j}$ is readily available from the visibility-based decomposition of $hreg'(p_i)$. Note that y_i or y_j may be identical to v_i or v_j respectively. Do the following:

(1) Locate y_i and y_j in $H\text{-}Vor(S_Q)$, if y_i, y_j have not already been located or marked. Let $q_i \in Q_i$ and $q_j \in Q_j$ be the owners of y_i and y_j respectively in $H\text{-}Vor(S_Q)$. If $d(y_i, q_i) < d(y_i, p_i)$ (resp. $d(y_j, q_j) < d(y_j, p_i)$), mark y_i (resp. y_j) to be in $H\text{-}Vor^*(S)$.
(2) If the conditions of item 3 of Lemma 10 are met, mark the entire $\overline{y_iy_j}$ as being part $H\text{-}Vor^*(S)$. Mark also all inter-bisectors of $hreg'(p_i)$ and $hreg'(p_j)$ forming quasi-triangles with $\overline{y_iy_j}$.
(3) If $\overline{y_iy_j}$ is of type 1, apply Algorithm 1 to determine the starting point of a cycle τ intersecting or entirely enclosing $\overline{y_iy_j}$. Pass vertex v_i or v_j to Algorithm 1 as a vertex on the boundary of $hreg'(p_i)$ that has already been marked. Go to step 5 to trace τ and update the diagrams.
(4) If $\overline{y_iy_j}$ is of type 2, apply Algorithm 2 to determine the first intersection

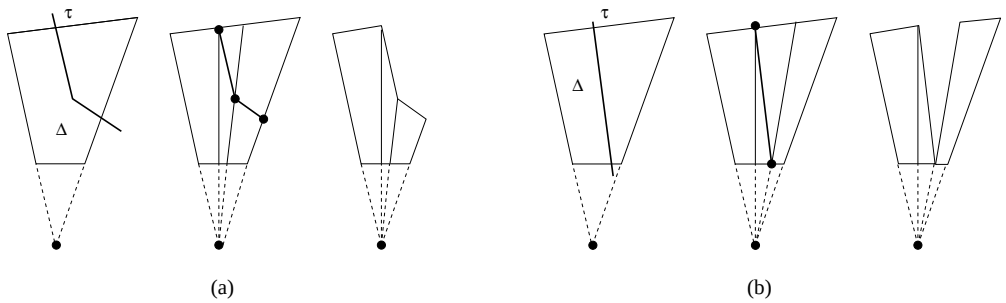

Fig. 17. Two examples of splitting of a quasi-triangle Δ intersected by τ into three new quasi-triangles bounding τ.

point y_k of a cycle τ on $\overline{y_i y_j}$. If no cycle exists mark the entire $\overline{y_i y_j}$ as being part of $H\text{-}Vor^*(S)$. Otherwise mark $\overline{y_i y_k}$ (assuming that y_i is an ancestor of y_j in $T(P)$). Mark also all inter-bisectors of $hreg'(p_i)$ and $hreg'(p_j)$ forming quasi-triangles with $\overline{y_i y_k}$.

(5) Trace cycle τ (or the nested pair of cycles τ and τ' in case of Algorithm 2) as described in Section 4. For every quasi-triangle Δ intersected by τ mark the portion of the inter-bisector segment remaining at the opposite side of τ as being in $H\text{-}Vor^*(S)$.

(6) Update $H\text{-}Vor_\sigma(S)$ and $H\text{-}Vor^*_\sigma(S)$ by adding the cycle τ (or the nested pair of cycles (τ, τ')), and exchanging the regions enclosed by τ (or pair (τ, τ')) between $H\text{-}Vor_\sigma(S)$ and $H\text{-}Vor^*_\sigma(S)$. Note that only regions in both diagrams intersected by τ (or pair (τ, τ')) need to be visited to perform the update. The actual update can be done as follows using standard operations of planar subdivision data-structures (see e.g. Refs. [4, 7]):
For any quasi-triangle Δ intersected by τ, split Δ in finer quasi-triangles to include all vertices of τ so that there is at least one quasi-triangle in each subdivision for every segment of τ. For every pair of quasi-triangles, one in $H\text{-}Vor^*_\sigma(S)$ and one in $H\text{-}Vor_\sigma(S)$, intersected by a segment of τ split them along τ and produce four new faces: two quasi-triangles that join $H\text{-}Vor_\sigma(S)$ and the two remaining portions that join $H\text{-}Vor^*_\sigma(S)$. The two remaining portions that join $H\text{-}Vor^*_\sigma(S)$ can directly be marked as being in $H\text{-}Vor^*(S)$ as they contain no intra-bisectors. Update the adjacencies between the four produced faces to complete the update. Clean-up by merging into one any two neighboring quasi-triangles that after the update border the same inter-bisector segment. Figure 17 illustrates the splitting and the updating of a quasi-triangle by two examples of τ.

(7) While there are still unmarked edges in $H\text{-}Vor^*_\sigma(S)$ proceed to a region neighboring an already marked edge and repeat the process.

End

In the following we bound the time complexity of the above algorithm.

Lemma 11. *The total time spent to compute starting points for cycles of $\sigma(S_l, S_r)$ of type 1, throughout the entire algorithm, is $O(n \log n + m)$, plus an additional $O((n \log n) + m) \log n)$-time for point location.*

Proof. Let $\overline{y_i y_j} \in H\text{-}Vor^*_\sigma(S)$ be an intra-bisector segment of type 1 considered by Algorithm 1. To determine the starting point of a cycle, segment $\overline{y_i y_j}$ is traversed through the visibility based decomposition of $H\text{-}Vor(S_Q)$. Let v be a vertex of $hreg'(q_i) \in H\text{-}Vor(S_Q)$ considered during the traversal of $\overline{y_i y_j}$. We claim that unless a cycle is determined when v is considered, v can never be considered again during the search for some other cycle of type 1 within this or any other merge step of the divide and conquer algorithm. This can be shown as follows: When v is considered, either a starting point on $\overline{y_i y_j}$ of a cycle is detected, or the point $y = \overline{q_i v} \cap \overline{y_i y_j}$ is determined to be in $hreg(P)$. Thus, $v \in H\text{-}Vor^*(S)$ and v must be enclosed in the cycle τ that will be determined by Algorithm 1. Suppose v is considered again by Algorithm 1 while considering a different intra-bisector segment $\overline{y'_i y'_j} \in T(P')$ during the search for another cycle τ' of type 1. Then v must also be enclosed in τ' and point $y' = \overline{q_i v} \cap \overline{y'_i y'_j}$ must be in $hreg(P')$. As a result, cycles τ and τ' must be enclosed within one another and the inner one must intersect segment $\overline{yy'}$. But then a portion of segment $\overline{q_i v}$ between y and y' must remain in $hreg(q_i)$ which is impossible by Property 2.

Thus, the number of vertices considered during the search for cycles of type 1 is upper bounded by the total number of vertices appearing in *H-Vor(S)* throughout the algorithm, plus the total number of cycles of type 1 discovered throughout the algorithm. But by Theorem 1, the number of Voronoi vertices in $H\text{-}Vor(S)$ during one merge step of the divide and conquer algorithm, excluding crossing mixed vertices, is $O(n)$, and thus $O(n \log n)$ throughout the divide and conquer algorithm. Since any Voronoi vertex can be generated only once, the total number of crossing mixed Voronoi vertices that get generated throughout the algorithm can not exceed m. Thus the total number of vertices ever appearing in $H\text{-}Vor(S)$ throughout the algorithm is $O(n \log n + m)$. Similarly the total number of cycles generated throughout the algorithm can not exceed $O(n \log n + m)$.

For every intra-bisector segment of type 1 considered, point location of its endpoints is performed in $H\text{-}Vor(S_Q)$ to determine its type. The number of point locations performed for identifying cycles of type 1 is upper bounded by the number of intra- and mixed Voronoi vertices in $H\text{-}Vor^*_\sigma(S)$. Following the same argument as above the total number of vertices ever appearing in $H\text{-}Vor^*_\sigma(S)$ throughout the algorithm is $O(n \log n + m)$. Since point location can be performed in $O(\log n)$ time the bound is derived. □

Definition 12. A point $q \in CH(Q)$, $Q \in S$ is called *interacting* with $P \in S$ if q is enclosed in the minimum enclosing circle of P, and either Q is entirely enclosed in the minimum enclosing circle of P or Q is crossing with P. Shape Q is also called

interacting with P. The number of points interacting with P is denoted by $M(P)$. The number of shapes entirely enclosed within the minimum enclosing circle of P is denoted by $K(P)$. Let $M = \Sigma_{P \in S} M(P)$ and $K = \Sigma_{P \in S} K(P)$.

Lemma 12. *The total time spent to compute starting points for cycles of $\sigma(S_l, S_r)$ of type 2, throughout the entire algorithm, is $O(M + m + n \log n)$ plus an additional $O((K + (n \log n) + m) \log n)$ time for point location, where M and K are as defined in Def. 12.*

Proof. Similarly to Lemma 11, a total $O(n \log n + m)$ time is spent throughout the divide and conquer algorithm to simply traverse $H\text{-}Vor^*_\sigma(S)$ searching for cycles. In addition, to determine starting points of cycles of type 2, Algorithm 2 generates additional mixed vertices that need not be part of $H\text{-}Vor^*_\sigma(S)$ nor $H\text{-}Vor_\sigma(S)$. Let $\overline{y_i y_j}$ be an intra-bisector segment of type 2 under consideration by Algorithm 2 such that $\overline{y_i y_j} \in b(p_i, p_j), p_i, p_j \in P$, and y_i is an ancestor of y_j in $T(P)$. Let y'_k be a mixed vertex generated by Algorithm 2 on $\overline{y_i y_j}$ induced by $Q_i \in S_Q$. Since P and Q_i are at opposite sides of the dividing line, pair (P, Q_i) can be considered with respect to only one dividing line. Furthermore, P and Q_i can have at most one equidistant point along the same intra-bisector segment $\overline{y_i y_j}$. Thus, y'_k can be generated once throughout the algorithm. Hence, the number of additional crossing mixed vertices ever generated by Algorithm 2 cannot exceed $O(m)$. Suppose now that y'_k is non-crossing. Then Q_i must be rear limiting with respect to $y_i \in \overline{y_i y_j}$. But then by Property 6, Q_i must be enclosed in the minimum enclosing circle of P. Furthermore, by Lemma 2 (see also Ref. [13]), Q_i can induce at most one rear non-crossing mixed vertex on $T(P)$. Thus, the number of additional non-crossing vertices generated on $T(P)$ by Algorithm 2 throughout the divide and conquer construction can not exceed $K(P)$. Hence, the total number of additional mixed vertices generated by Algorithm 2 throughout all merge steps is $O(m + K)$.

Let's now derive the total time needed to generate those additional mixed vertices. As shown above, Q_i is either crossing $\overline{p_i p_j}$ or Q_i must be entirely contained in the minimum enclosing circle of P. Thus, Q_i must be *interacting* with P. Vertex y'_k can be clearly determined in $O(|CH(Q_i)|)$ time by walking on $\overline{y_i y_j}$ and considering intersections with $f\text{-}Vor(Q_i)$, until y'_k is determined. If y'_k is non-crossing then, as shown above, $CH(Q_i)$ is enclosed in the minimum enclosing circle of P and Q_i is considered only once with respect to $T(P)$. But if y'_k is crossing, Q_i may be considered a number of times during the search for different cycles of type 2 intersecting $T(P)$. However, we claim that any $q_r \in Q_i$ visited while determining y'_k can not be considered again, and q_r must be enclosed in the minimum enclosing circle of P. This was basically shown in Lemma 9 of Ref. [13] and it is repeated here for completeness. Assuming that this claim is true, the total time to compute any mixed vertex induced by Q_i on $T(P)$ during the search for a cycle of type 2 is $O(|CH(Q_i) \cap \mathcal{K}_0(P)|)$, where $\mathcal{K}_0(P)$ denotes the minimum enclosing circle of P. As a result, and since pair (P, Q_i) can be considered with respect to a single dividing line, the total time to generate the additional mixed vertices of Algorithm

2 on $T(P)$ is $O(M(P))$. Thus, the total time for the generation of additional mixed vertices spent by Algorithm 2 is $O(M)$.

We now prove the claim that any $q_r \in Q_i$ visited while determining y'_k can only be considered once, furthermore, $q_r \in \mathcal{K}_0(P)$ (see also Lemma 9 of Ref. [13]). Recall that in Algorithm 2 the traversal of $\overline{y_i y_j}$ over *f-Vor*(Q_i) starts at y_j and that the entire segment $\overline{y'_k y_j}$ is closer to P than Q_i. Thus, the owner q_r of any $y \in \overline{y'_k y_j}$ in *f-Vor*(Q_i) must have the property that $q_r \notin \mathcal{K}_y$ (with the exception of y'_k) and $q_r \in \mathcal{K}_{y_i}$. Since $\mathcal{K}^f_{y_i} \subset \mathcal{K}_y$, for any $y \in \overline{y'_k y_j}$ (see the proof of Lemma 2), we conclude that $q_r \in \mathcal{K}^r_{y_i}$ while $q_r \notin \mathcal{K}_{y_j}$. Thus, $q_r \notin \mathcal{K}^r_y$ for any $y \in T(y_j)$, as $\mathcal{K}^r_{y_j} \subset \mathcal{K}^r_y$ (see Lemma 1 of Ref. [13]). In addition, for any $y \in T_c(y_i)$ that is not an ancestor of y_i, $\mathcal{K}^r_{y_i} \subset \mathcal{K}^f_y$ (see Lemma 1 of Ref. [13]), and thus $q_r \in \mathcal{K}^f_y$, i.e., $q_r \notin \mathcal{K}^r_y$. Hence, q_r can not be considered again with respect to any other intra-bisector segment of $T(P)$. Furthermore, since $q_r \in \mathcal{K}^r_{y_i}$, by Property 6, q_r must be enclosed in the minimum enclosing circle of P. The claim is now shown.

For every mixed vertex generated by Algorithm 2, point location in *H-Vor*(S_Q) is performed, thus, $O((m + K)\log n)$ time is attributed to point location of the additional vertices. In addition, point location is performed for the endpoints of $\overline{y_i y_j}$ in order for algorithm 2 to be invoked. Following the same reasoning as in Lemma 11 this requires an additional $O((n \log n + m)\log n)$ time. □

The following lemma illustrates how to determine whether a shape Q is limiting or crossing with respect to a a chord $\overline{p_i p_j} \in P$.

Lemma 13. *Let $P, Q \in S$ such that Q is entirely enclosed in a P-circle through $p_i, p_j \in P$ and $q_i \in Q$. Whether Q is limiting or crossing with $\overline{p_i p_j}$ can be determined in $O(\log n)$ time.*

Proof. We first need to determine whether there is a vertex $q_j \in Q$, at opposite side of $\overline{p_i p_j}$ as q_i, i.e., if $\overline{p_i p_j}$ crosses Q. This can be done in $O(\log |Q|)$ time. If there is no intersection, Q is limiting. If so, we need to check whether $q_i \notin CH(P)$ or whether there exists a q_j at the opposite side of $\overline{p_i p_j}$ as q_i such that $q_j \notin CH(P)$. This can be done as follows: Shoot from p_i and p_j the supporting rays to $CH(Q)$. This can be done by binary search on $CH(Q)$ in logarithmic time. Let C_p be the chain of $CH(P)$ between p_i and p_j at opposite side of q_i. Let C_q be the chain of $CH(Q)$, including the supporting segments, between p_i and p_j at opposite side of q_i. In Figure 18, C_p is shown in bold and the supporting segments from p_i, p_j to $CH(Q)$ are shown dashed. We need to determine whether C_p and C_q, two convex chains with the same endpoints, intersect. This can be done in logarithmic time using binary search. The same is then done for the other side of $\overline{p_i p_j}$ to check if $q_i \notin CH(P)$. □

Theorem 3. *The Hausdorff Voronoi diagram of S can be computed by divide and conquer in time $O(M + n\log^2 n + (m + K)\log n)$, where n is the number of points on the convex hulls of input shapes, $O(n+m)$ is the worst case bound on the size of*

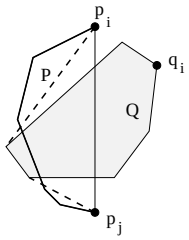

Fig. 18. The proof of Lemma 13.

H-Vor(S) given in Theorem 1 and Ref. [13], $M = \Sigma_{P \in S} M(P)$, $K = \Sigma_{P \in S} K(P)$, where $M(P)$ is the number of vertices in the minimum enclosing circle of P interacting with P (see Def. 12), and $K(P)$ is the number of shapes enclosed in the minimum enclosing circle of P. An $O(|S| \log n)$ preprocessing to identify the convex hulls of input shapes is assumed.

Proof. The time complexity in the merge step of the divide and conquer scheme is dominated by the time to compute starting points for the cycles of $\sigma(S_l, S_r)$. Recall that starting points on the unbounded portions of $\sigma(S_l, S_r)$ can be computed in $O(n)$-time by the use of the *H-hull* as shown in Theorem 2. Recall also that the actual tracing of the components of $\sigma(S_l, S_r)$ can be done in $O(n + m)$ time as a consequence of Lemma 5. Furthermore, by Lemma 5, any vertex of *H-Vor*$_\sigma(S)$ and *H-Vor*$^*_\sigma(S)$ visited during the tracing of a component of $\sigma(S_l, S_r)$ will not be visited again during the tracing of another component of $\sigma(S_l, S_r)$. Thus the total time spent on the tracing of merge curves throughout the divide and conquer algorithm cannot exceed the total number of vertices ever appearing in *H-Vor*$^*_\sigma(S)$ or *H-Vor*$_\sigma(S)$. But this number was shown in Lemma 11 to be $O(n \log n + m)$. Similarly, the update of *H-Vor*$_\sigma(S)$ and *H-Vor*$^*_\sigma(S)$ can be done in total $O(n \log n + m)$ time throughout the algorithm as the actual splitting of a quasi-triangle by a segment requires only constant time. Checking whether a shape is crossing with respect to chord can be done in $O(\log n)$ time as shown in Lemma 13. Thus, the total time complexity of the algorithm is derived by Lemma 11 and Lemma 12. The correctness of the algorithm follows from Lemma 10, the proof of Lemma 11, and the proof of Lemma 12.

In the case of non-crossing shapes, $S_{\mathcal{L}}$ consists only of shapes in S_l intersecting the dividing line $\mathcal{L}$ and $S_{\mathcal{L}}$ is easy to identify. Thus, point location needs only be performed for shapes in $S_{\mathcal{L}}$. Since there are no crossing shapes the time complexity simplifies to $O(M+(n+N+K) \log n)$, where N denotes the total number of vertices of the convex hulls in $S_{\mathcal{L}}$. Asymptotically N remains $O(n \log n)$. □

5. Conclusion

In this paper we have formulated the Hausdorff Voronoi diagram of a set S of polygonal objects in the plane and investigated its structural and combinatorial

properties. Our study of this type of diagram was motivated by the *critical area* computation problem for *via-blocks* in VLSI designs. We have presented a divide and conquer algorithm for the construction of this diagram improving upon previous results. In particular, we have shown that the size of the Hausdorff Voronoi diagram is $O(n+m)$, where n is the number of points on the convex hulls of the polygons in S, and m is the total number of crossing mixed vertices on the inter-bisectors between pairs of crossing shapes. The term m is $O(n^2)$ in the worst case, and reflects the number of *crossings* among shapes in S. This bound has since been refined and shown to be tight in a companion paper.[13] For a non-crossing set S, Hausdorff Voronoi regions remain connected and the size of the diagram has been shown to be $O(n)$. The time complexity of our algorithm is $O(M + n\log^2 n + (m+K)\log n)$, where $M = \Sigma_{P\in S} M(P)$, $K = \Sigma_{P\in S} K(P)$, $M(P)$ is the number of vertices in the minimum enclosing circle of P *interacting* with P (see Def. 12), and $K(P)$ is the number of shapes enclosed in the minimum enclosing circle of P. We have also introduced the *Hausdorff-hull*, a structure that relates to the Hausdorff Voronoi diagram in the same way as an ordinary convex hull relates to the ordinary Voronoi diagram. Whether the terms M, K and $n\log^2 n$ can be eliminated from the time complexity remains an open problem even in the case of non-crossing polygonal objects.

References

1. M. Abellanas, G. Hernandez, R. Klein,V. Neumann-Lara, and J. Urrutia, "A combinatorial property of convex sets", *Discrete Computational Geometry* 17, 1997, 307-318.
2. M. Abellanas, F. Hurtado, C. Icking, R. Klein, E. Langetepe, L. Ma, B. Palop, V. Sacristán, "The farthest color Voronoi diagram and related problems", *Abstracts 17th European Workshop Comput. Geom. CG 2001*, Freie Universität Berlin, Berlin 2001, 113–116.
3. L.L. Chen, S.Y. Chou, and T.C. Woo, "Parting directions for mould and die design", *Computer-Aided Design*, Vol. 25, No. 12, 1993, 762-768.
4. M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf, "Computational Geometry, Algorithms and Applications" Springer-Verlag Berlin Heidelberg 1997.
5. H. Edelsbrunner, L.J. Guibas, and M. Sharir, "The upper envelope of piecewise linear functions: algorithms and applications", *Discrete Computational Geometry* 4, 1989, 311-336.
6. D. P. Huttenlocher, K. Kedem, and M. Sharir, "The upper envelope of Voronoi surfaces and its applications", *Discrete Computat. Geometry*, 9, 1993, 267-291.
7. L. Kettner, "Using generic programming for designing a data structure for polyhedral surfaces", *Computational Geometry - Theory and Applications* 13, 1999, 65-90.
8. R. Klein, "Concrete and Abstract Voronoi Diagrams", vol. 400, *Lecture Notes in Computer Science*, Springer-Verlag, 1989.
9. R. Klein, K. Mehlhorn, S. Meiser, "Randomized incremental construction of abstract Voronoi diagrams", *Computational Geometry: Theory and Applications*, 3, 1993, 157-184
10. W. Maly, "Computer Aided Design for VLSI circuit manufacturability," *Proc. IEEE*,

vol.78, no.2, Feb. 1990, 356-392.
11. B. R. Mandava, "Critical Area for yield models", IBM Technical Report TR22.2436, East Fishkill, NY, 12 Jan 1982.
12. C.H. Ouyang, W.A. Pleskacz, W. Maly, "Extraction of Critical Area for opens in large VLSI circuits", *IEEE Trans. on Computer-Aided Design*, vol. 18, no 2, Feb. 1999, 151-162.
13. E. Papadopoulou, "The Hausdorff Voronoi diagram of point clusters in the plane", *Algorithmica*, 40, 2004, 63-82.
14. E. Papadopoulou, "Critical Area computation for missing material defects in VLSI circuits", *IEEE Transactions on Computer-Aided Design*, vol. 20, no.5, May 2001, 583-597.
15. E. Papadopoulou and D.T. Lee, "The min-max Voronoi diagram of polygonal objects and applications in VLSI manufacturing", *Proc. 13th International Symposium on Algorithms and Computation*, Nov. 2002, Vancouver, Canada, *LNCS 2518*, 511-522.
16. E. Papadopoulou and D.T. Lee, "Critical Area computation via Voronoi diagrams", *IEEE Trans. on Computer-Aided Design*, vol. 18, no.4, April 1999, 463-474.
17. E. Papadopoulou and D.T. Lee, "The L_∞ Voronoi diagram of segments and VLSI applications", *International Journal of Computational Geometry and Applications*, Vol. 11, No. 5, 2001, 503-528.
18. Preparata, F. P. and M. I. Shamos, *Computational Geometry: an Introduction*, Springer-Verlag, New York, NY 1985.
19. I. A. Wagner and I. Koren, "An interactive VLSI CAD tool for yield estimation," *IEEE Trans. on Semiconductor Manufacturing* Vol. 8, No.2, Feb. 1995, 130-138.
20. H. Walker and S.W. Director, " VLASIC: A yield simulator for integrated circuits," *IEEE Trans. on Computer-Aided Design*, CAD-5,4, Oct. 1986, 541-556.