Computing Many Faces in Arrangements of Lines and Segments

P.K. Agarwal, J. Matoušek, and O. Schwarzkopf

UU-CS-1994-39
September 1994

Utrecht University

Department of Computer Science

Padualaan 14, P.O. Box 80.089,
3508 TB Utrecht, The Netherlands,
Tel. : ... + 31 - 30 - 531454

Computing Many Faces in Arrangements of Lines and Segments

P.K. Agarwal, J. Matoušek, and O. Schwarzkopf

Technical Report UU-CS-1994-39
September 1994

Department of Computer Science
Utrecht University
P.O.Box 80.089
3508 TB Utrecht
The Netherlands

ISSN: 0924-3275

Computing Many Faces in Arrangements of Lines and Segments *

Pankaj K. Agarwal† Jiří Matoušek‡ Otfried Schwarzkopf§

Abstract

We present randomized algorithms for computing many faces in an arrangement of lines or of segments in the plane, which are considerably simpler and slightly faster than the previously known ones. The main new idea is a simple randomized $O(n \log n)$ expected time algorithm for computing $\sqrt{n}$ cells in an arrangement of n lines.

1 Introduction

Given a finite set of lines, L, in the plane, the *arrangement* of L, denoted as $\mathcal{A}(L)$, is the cell complex induced by L. The 0-faces (or *vertices*) of $\mathcal{A}(L)$ are the intersection points of L, the 1-face (or *edges*) are maximal portions of lines of L that do not contain any vertex, and the 2-faces (called *cells*) are the connected components of $\mathbb{R}^2 - \bigcup L$. For a finite set S of segments we define the arrangement, $\mathcal{A}(S)$, in an analogous manner. Notice that while the cells are convex in a line arrangement, they need not even be simply connected in an arrangement of segments.

Line and segment arrangements have been extensively studied in computational geometry (as well as in some other areas), as a wide variety of computational geometry problems can be formulated in terms of computing such arrangements or their parts [11, 14].

Given a set L of n lines and a set P of m points in the plane, we define $\mathcal{A}(L, P)$ to be the collection of all cells of $\mathcal{A}(L)$ containing at least one point of P. The *combinatorial complexity* of a cell C, denoted by $|C|$, in $\mathcal{A}(L)$ is the number of edges of C. Let $\kappa(L, P) = \sum_{C \in \mathcal{A}(L,P)} |C|$ denote the total combinatorial complexity of all cells in $\mathcal{A}(L, P)$, and let

$$\kappa(n, m) = \max \kappa(L, P),$$

*A part of this work was done while the first and third authors were visiting Charles University and while the first author was visiting Utrecht University. The first author has been supported by National Science Foundation Grant CCR–93–01259 and an NYI award. The second author has been supported by Charles University grant No. 351 and Czech Republic Grant GAČR 201/93/2167. The third author has been supported by the Netherlands' Organization for Scientific Research (NWO) and partially supported by ESPRIT Basic Research Action No. 7141 (project ALCOM II: *Algorithms and Complexity*)

†Department of Computer Science, Box 90129, Duke University, Durham, NC 27708-0129, USA.

‡Department of Applied Mathematics, Charles University, Malostranské nám. 25, 118 00 Praha 1, Czech Republic.

§Department of Computer Science, Utrecht University, P.O. Box 80.089, 3508 TB Utrecht, the Netherlands.

where the maximum is taken over all sets of n lines and over all sets of m points in the plane. It is known that

$$\kappa(n,m) = \Theta(n^{2/3}m^{2/3} + n + m)\,.$$

The upper bound was proven by Clarkson et al. [9]; previous results and related work can be found in Canham [4], Edelsbrunner and Welzl [13], Szemerédi and Trotter [20].

In this paper we study the problem of computing $\mathcal{A}(L,P)$, that is, for each cell $C \in \mathcal{A}(L,P)$, we want return the vertices of C in, say, clockwise order. We will refer to the cells of $\mathcal{A}(L,P)$ as the marked cells of $\mathcal{A}(L)$. Edelsbrunner et al. [12] presented a randomized algorithm, based on the random sampling technique [16], for computing $\mathcal{A}(L,P)$, whose expected running time was $O(m^{2/3-\varepsilon}n^{2/3+2\varepsilon}\log n + n\log n\log m)$, for any fixed $\varepsilon > 0$. A deterministic algorithm with running time $O(m^{2/3}n^{2/3}\log^{O(1)} n + n\log^3 n + m\log n)$ was given by Agarwal [1]. These algorithms thus are nearly worst-case optimal, but both of them are rather involved.

Recently randomized incremental algorithms have been developed for a wide variety of geometric problems, which add the input objects one by one in a random order and maintain the desired structure; see e.g. [6, 10, 18, 19]. In our case, we can add the lines of L one by one in a random order and maintain the marked cells in the arrangement of lines added so far. However, this approach seems to yield expected running time of $\Omega(n\sqrt{m} + m\log n)$ in the worst case. We, therefore, do not quite follow the randomized incremental paradigm.

We begin by presenting an expected $O(m^2 + n\log n)$ time randomized algorithm for computing $\mathcal{A}(L,P)$. Notice that for $m \leq \sqrt{n\log n}$, this algorithm is optimal. We then apply the random sampling technique in a standard way, obtaining an expected $O(m^{2/3}n^{2/3}\log^{2/3}\frac{n}{\sqrt{m}} + (m+n)\log n)$ time algorithm.

We also study a similar but more difficult problem of computing the marked cells in an arrangement of n segments. Let S be a set of n segments in the plane. We use an analogous notation $\mathcal{A}(S,P)$ to denote the set of the cells in $\mathcal{A}(S)$ containing at least one point of P, and $\eta(n,m)$ to denote the maximum combinatorial complexity of $\mathcal{A}(S,P)$ over all sets S of n segments and sets P of m points in the plane. Aronov et al. [2] proved that

$$\eta(n,m) = O\left(m^{2/3}n^{2/3} + n\log m + n\,\alpha(n)\right)\,.$$

A randomized algorithm with expected running time

$$O(m^{2/3-\varepsilon}n^{2/3+2\varepsilon}\log n + n\alpha(n)\log^2 n\log m)$$

is described by Edelsbrunner et al. [12], and a slightly faster deterministic algorithm is presented by Agarwal [1].

Following the same strategy as for the case of lines, we first develop a randomized algorithm with $O((m^2 + n\log m + n\,\alpha(n))\log n)$ expected running time. Let us remark that the above upper bound for $\eta(n,m)$ is not known to be tight, and a bound like $\eta(n,\sqrt{n}) = O(n\,\alpha(n))$ (which is conjectured to be the complexity of $\sqrt{n}$ cells) will immediately improve the expected running time of our algorithm to $O(n\log n\,\alpha(n))$. Plugging this algorithm to

the standard random sampling technique, as in the case of lines, we obtain a randomized algorithm for computing $\mathcal{A}(S,P)$ whose expected running time is

$$O(m^{2/3}n^{2/3}\log^{4/3}\frac{n}{\sqrt{m}}\,\alpha^{1/3}(\frac{n}{\sqrt{m}}) + (m + n\log m + n\,\alpha(n))\log n)\,.$$

If the segments of S have only $k = o(n^2)$ intersection points, the expected running time of the algorithm is

$$O(m^{2/3}k^{1/3}\log^{4/3}\frac{k}{m}\alpha^{1/3}(\frac{k}{m}) + (m + n\log m + n\,\alpha(n))\log n).$$

For the analysis of the expected running time of our algorithms we will use a generalization of a lemma due to Chazelle and Friedman [7]. (An alternative analysis could probably be obtained using a method similar to that of Chazelle et al. [6], but we hope that our approach is somewhat more intuitive).

2 A generalization of Chazelle-Friedman lemma

Let S be a set of lines or segments, and P a set of points in the plane. For a cell C of the collection $\mathcal{A}(S,P)$, let $C^{\|}$ denote the collection of trapezoids in the vertical decomposition of C,[1] and let $\mathcal{A}^{\|}(S,P) = \bigcup_{C\in\mathcal{A}(S,P)} C^{\|}$ denote the set of trapezoids in the vertical decomposition of $\mathcal{A}(S,P)$. Abusing the notation slightly, we will use $\mathcal{A}^{\|}(S,P)$ to denote the corresponding planar subdivision as well.

Let R be a subset of S. For a trapezoid $\Delta \in \mathcal{A}^{\|}(R,P)$, let $w(\Delta)$ denote the number of elements of S intersecting the interior of Δ.

Let $n = |S|$, and let R be a random subset of S of size r. For the analysis of our algorithms, we are interested in estimating the expectation, over all random choices of R,

$$\mathrm{E}\Big[\sum_{\Delta\in\mathcal{A}^{\|}(R,P)} w(\Delta)^c\Big]\,, \tag{2.1}$$

where c is a small constant like $c = 2$. Well-known results concerning the so-called ε-nets (Haussler and Welzl [16]) imply that, for every $\Delta \in \mathcal{A}^{\|}(R)$, $w(\Delta) \le C(n/r)\log r$ with high probability, where C is a suitable constant. From this one can derive a bound for (2.1). We are, however, interested in the following, slightly stronger bound (better by a factor of $\log^c r$):

Proposition 2.1 *(i) Let L be a set of n lines and P a set of m points in the plane. If $R \subseteq L$ is random subset of size r, where each subset of size r is chosen with equal probability, then for any constant $c \ge 1$.*

$$\mathrm{E}\,[\sum_{\Delta\in\mathcal{A}^{\|}(R,P)} w(\Delta)^c] = \kappa(r,m)\cdot O((n/r)^c)\,.$$

[1] The vertical decomposition $C^{\|}$ of a cell C in an arrangements of segments (or of lines) is obtained by drawing a vertical line from each vertex of C in both directions (within C) until it hits another edge of C.

(ii) Let S be a set of n segments and P a set of m points in the plane. If $R \subseteq S$ is random subset of size r, where each subset of size r is chosen with equal probability, then for any constant $c \geq 1$.

$$\mathrm{E}\left[\sum_{\Delta \in \mathcal{A}^{\|}(R,P)} w(\Delta)^c\right] = \eta(r,m) \cdot O((n/r)^c)\,.$$

These bounds essentially say that the cth moment of the quantities $w(\Delta)$ behaves as if $w(\Delta)$ were $O(n/r)$. If we sum $w(\Delta)$ over all cells in $\mathcal{A}(R)$—the case where every cell of $\mathcal{A}(R)$ contains a point of P—then Proposition 2.1 follows from a result of Clarkson and Shor [10]. In our situation, where the sum is taken over only some of the cells, the Clarkson-Shor framework does not apply directly anymore (the main distinction between these two situations will be outlined below). We give a proof based on a generalization of an approach due to Chazelle and Friedman [7], which is somewhat different from the Clarkson-Shor method. Recently, de Berg et al. [3] gave an alternative proof of Proposition 2.1.

We derive a key lemma in a somewhat abstract framework; see also [6, 7, 10] for various approaches to axiomatize similar situations.

Let S be a set of objects. For a subset $R \subseteq S$, we define a collection of 'regions' called $CT(R)$; in Proposition 2.1 the objects are segments, the regions are trapezoids, and $CT(R) = \mathcal{A}^{\|}(R,P)$. Let $T = T(S) = \bigcup_{R \subseteq S} CT(R)$ denote the set of regions defined by all possible subsets of S. We associate two subsets $D(\Delta), K(\Delta) \subseteq S$ with each region $\Delta \in T$.

$D(\Delta)$, called the *defining set*, is a subset of S defining the region Δ in a suitable geometric sense.[2] We assume that for every $\Delta \in T$, $|D(\Delta)| \leq d$ for a (small) constant d. In Proposition 2.1, each trapezoid Δ is defined by at most 4 segments (or lines) of S, which constitute the set $D(\Delta)$; details can be found in Chazelle et al. [6].

$K(\Delta)$, called the *killing set*, is a set of objects of S, such that including any object of $K(\Delta)$ into R prevents Δ from appearing in $CT(R)$. In many applications $K(\Delta)$ is the set of objects intersecting the cell Δ; this is also the case in Proposition 2.1. Set $w(\Delta) = |K(\Delta)|$.

Let $S, CT(R), D(\Delta), K(\Delta)$ be such that for any subset $R \subseteq S$, $CT(R)$ satisfies the following axioms:

(i) For any $\Delta \in CT(R)$, $D(\Delta) \subseteq R$ and $R \cap K(\Delta) = \emptyset$, and

(ii) If $\Delta \in CT(R)$ and R' is a subset of R with $D(\Delta) \subseteq R'$, then $\Delta \in CT(R')$.

It is easily checked that these axioms hold in the situations of Proposition 2.1.

For any natural number t, let us denote

$$CT_t(R) = \{\Delta \in CT(R) \mid w(\Delta) \geq tn/r\}\,.$$

We establish the following:

[2] We need not make this precise here, as this is only an intuitive meaning of $D(\Delta)$. The analysis depends only on the axioms involving $D(\Delta)$ given below, and these will be satisfied in our specific examples.

Lemma 2.2 *Given a set S of b objects, let R be a random sample of size $r \leq n$ drawn from S, and let t be a parameter, $1 \leq t \leq r/d$, where $d = \max|D(\Delta)|$. Assuming that $CT(R)$, $D(\Delta)$ and $K(\Delta)$ satisfy Axioms (i) and (ii) above, we have*

$$\mathrm{E}\,|CT_t(R)| = O(2^{-t}) \cdot \mathrm{E}\,|CT(R')|\,, \tag{2.2}$$

where $R' \subseteq S$ denotes a random sample of size $r' = \lfloor r/t \rfloor$.

Roughly speaking, Lemma 2.2 says that the expected number of "large" trapezoids in $CT(R)$, that is, the trapezoids for which the value of $w(\Delta)$ exceeds the "right" value n/r more than t times, decreases exponentially with t.

Chazelle and Friedman [7] proved a result analogous to Lemma 2.2 under the following stronger axiom replacing (ii):

(ii′) If $D(\Delta) \subseteq R$ and $K(\Delta) \cap R = \emptyset$, then $\Delta \in CT(R)$.

This assumption implies that the presence of Δ in $CT(R)$ depends only on $D(\Delta)$ and $K(\Delta)$, thus it is determined purely "locally." Notice that (ii′) may fail in the situation of Proposition 2.1. However, (ii′) holds in the special case, when $CT(R)$ is the vertical decomposition of *all* cells in $\mathcal{A}(R)$.

Proof of Lemma 2.2: Let $T_t = \bigcup_{R \subseteq S} CT_t(R)$. We have

$$\mathrm{E}\,|CT_t(R)| = \sum_{\Delta \in T_t} \Pr[\Delta \in CT(R)]\,, \tag{2.3}$$

$$\begin{aligned} \mathrm{E}\,|CT(R')| &= \sum_{\Delta \in T} \Pr[\Delta \in CT(R')] \\ &\geq \sum_{\Delta \in T_t} \Pr[\Delta \in CT(R')]\,. \end{aligned} \tag{2.4}$$

We will prove that, for each $\Delta \in T_t$,

$$\Pr[\Delta \in CT(R)] = O(2^{-t}) \cdot \Pr[\Delta \in CT(R')]\,, \tag{2.5}$$

which in conjunction with (2.3) and (2.4) implies (2.2).

Let A_Δ denote the event $D(\Delta) \subseteq R$ and $K(\Delta) \cap R = \emptyset$, and let A'_Δ denote the event $D(\Delta) \subseteq R'$ and $K(\Delta) \cap R' = \emptyset$.

We rewrite $\Pr[\Delta \in CT(R)]$ using the definition of conditional probability:

$$\Pr[\Delta \in CT(R)] = \Pr[A_\Delta] \cdot \Pr[\Delta \in CT(R) \mid A_\Delta]$$

and analogously

$$\Pr[\Delta \in CT(R')] = \Pr[A'_\Delta] \cdot \Pr[\Delta \in CT(R') \mid A'_\Delta]\,.$$

We observe that, by Axiom (ii), we have

$$\Pr[\Delta \in CT(R) \mid A_\Delta] \leq \Pr[\Delta \in CT(R') \mid A'_\Delta]\,. \tag{2.6}$$

Indeed, $\Pr[\Delta \in CT(R') \mid A'_\Delta]$ is the probability that Δ appears in $CT(R')$, where R' is created as follows: Include all elements of $D(\Delta)$, and then choose the remaining $r' - |D(\Delta)|$ elements randomly among the elements of $S \setminus (D(\Delta) \cup K(\Delta))$. We may continue this experiment by choosing R to be R' plus a random subset of $r - r'$ elements of $S \setminus (R' \cup K(\Delta))$. Clearly for such subsets R' and R, $\Pr[\Delta \in CT(R)] \leq \Pr[\Delta \in CT(R')]$. Moreover, the subset R selected by this experiment contains $D(\Delta)$ plus $r - |D(\Delta)|$ random elements of $S \setminus (D(\Delta) \cup K(\Delta))$, so $\Pr[\Delta \in CT(R)]$ is the same as the left hand side of (2.6).

Therefore

$$\frac{\Pr[\Delta \in CT(R)]}{\Pr[\Delta \in CT(R')]} \leq \frac{\Pr[A_\Delta]}{\Pr[A'_\Delta]}.$$

(Note that $r' = \lfloor r/t \rfloor \geq d$, and hence both denominators are nonzero.)

It remains to estimate the latter ratio, which can be done in the same way as by Chazelle and Friedman. Let $d = |D(\Delta)|$, $w = w(\Delta)$, and for two non-negative integers $a \leq x$, let $x^{\underline{a}} = x(x-1)\cdots(x-a+1)$. Then

$$\begin{aligned} \frac{\Pr[A_\Delta]}{\Pr[A'_\Delta]} &= \frac{\binom{n-w-d}{r-d}}{\binom{n}{r}} \cdot \frac{\binom{n}{r'}}{\binom{n-w-d}{r'-d}} \\ &= \frac{r^{\underline{d}}}{r'^{\underline{d}}} \cdot \frac{(n-w-r')^{\underline{r}}}{(n-r')^{\underline{r}}}. \end{aligned}$$

By our assumption $r' \geq d$, so we obtain

$$\frac{r-i}{r'-i} \leq dt \quad \text{for } i = 0, 1, \ldots, d-1.$$

Thus, the first factor in the above expression is $O(t^d)$. To bound the second factor, we observe that, for $i = r', r'+1, \ldots, r-1$,

$$\frac{n-w-i}{n-i} = 1 - \frac{w}{n-i} \leq 1 - \frac{w}{n} \leq \exp(-w/n).$$

Since $w \geq tn/r$, we have $w/n \geq t/r$, and therefore

$$\begin{aligned} \frac{\Pr[A_\Delta]}{\Pr[A'_\Delta]} &\leq O(t^d) \exp\left(\frac{-t(r-r')}{r}\right) \\ &= O(t^d)\exp(-(t-1)) = O(2^{-t}), \end{aligned}$$

as desired. □

We now prove Proposition 2.1.

Proof of Proposition 2.1: We will only prove the first part, the second part is identical. For any subset $R \subseteq L$ of size r, let $CT(R)$ denote the set of trapezoids in the vertical decomposition of the marked cells of $\mathcal{A}(R)$, i.e., $CT(R) = \mathcal{A}^{\shortparallel}(R, P)$. Obviously, $|CT(R)| \leq \kappa(r, m)$. Now

$$\mathrm{E}\Big[\sum_{\Delta \in \mathcal{A}^{\shortparallel}(R,P)} w(\Delta)^c\Big]$$

$$
\begin{aligned}
&= \mathrm{E}\left[\sum_{t\geq 1}\left(t\frac{n}{r}\right)^c\left(|CT_t(R)|-|CT_{t-1}(R)|\right)\right]\\
&\leq \left(\frac{n}{r}\right)^c\sum_{t\geq 0}(t+1)^c\cdot \mathrm{E}\,|CT_t(R)|\\
&= \left(\frac{n}{r}\right)^c\sum_{t\geq 0}t^cO(2^{-t})\cdot\kappa(r/t,m)\\
&\leq \kappa(r,m)\left(\frac{n}{r}\right)^c\sum_{t\geq 0}O(t^c\cdot 2^{-t})\\
&= \kappa(r,m)\cdot O\left((n/r)^c\right).
\end{aligned}
$$

□

3 Computing cells in line arrangements

Let L be a set of n lines and P a set of m points in the plane. We assume that the points of P are sorted in nondecreasing order of their x-coordinates, and that the lines of L are sorted by their slopes. In this section we describe a randomized algorithm for computing $\mathcal{A}(L,P)$. In fact, it computes the vertical decomposition $\mathcal{A}^{\|}(L,P)$ of $\mathcal{A}(L,P)$. Each face of $\mathcal{A}^{\|}(L,P)$ is a trapezoid, bounded by at most two vertical segments and portions of at most two edges of a cell of $\mathcal{A}^{\|}(L,P)$. We begin by presenting a very simple randomized algorithm for computing $\mathcal{A}^{\|}(L,P)$ with $O(m^2+n\log n)$ expected time, which we will use as a subroutine in the main algorithm. Notice that this algorithm is optimal for $m\leq\sqrt{n\log n}$. If $n\leq n_0$, where n_0 is an appropriate constant, the algorithm computes the vertical decomposition of the entire arrangement using any standard algorithm. Otherwise, it proceeds as follows.

1. Let t be a sufficiently large constant. Choose a random subset $R\subseteq L$ of $r=\lfloor n/t\rfloor$ lines.

2. Partition P into $q=\lceil\sqrt{t}\,\rceil$ subsets $P_1,\ldots,P_q$, each of size at most $k=\lfloor m/\sqrt{t}\rfloor$, where $P_i=\{p_{(i-1)k+1},\cdots,p_{ik}\}$, for $i<q$, and $P_q=\{p_{(q-1)k+1},\cdots,p_m\}$.

3. For each $i\leq q$, compute $\mathcal{A}^{\|}(R,P_i)$ recursively. If a cell C of $\mathcal{A}(R)$ is computed more than once, retain only one copy of C. (Note that multiple copies of a cell C are computed if C contains the points of more than one P_i's.) Since P is sorted in the x-direction, it is easy to detect multiple copies of a cell. In this way, obtain $\mathcal{A}^{\|}(R,P)$.

4. For each line $\ell\in L\setminus R$, compute the cells of $\mathcal{A}(R,P)$ that ℓ intersects.

5. For each trapezoid Δ of $\mathcal{A}^{\|}(R,P)$, compute the set $L_\Delta\subseteq L\setminus R$ of lines that intersect the interior of Δ.

6. For each trapezoid $\Delta\in\mathcal{A}^{\|}(R,P)$, compute the arrangement of lines of L_Δ, clip it within Δ, and compute the vertical decomposition of the clipped arrangement.

For each cell $C \in \mathcal{A}(R,P)$, perform a graph search on trapezoids of these vertical decompositions to merge appropriate trapezoids and to discard superfluous ones, thus forming the portion of $\mathcal{A}^{\|}(L,P)$ within the cell C.

Steps 1–3 are trivial, so we only describe Steps 4–6 in more detail.

Step 4. We want to compute the cells of $\mathcal{A}(R,P)$ intersected by each line in $L \setminus R$. The situation can be viewed as follows: we have a collection $\mathcal{C}$ of disjoint convex polygons (the cells of $\mathcal{A}(R,P)$), and a set $L \setminus R$ of lines. The collection $\mathcal{C}$ has at most m polygons with a total of $O(n+m^2)$ edges[3]. For each polygon $C \in \mathcal{C}$, consider C^*, the set of points that are dual to the lines intersecting C. C^* is a polygonal region, bounded by an infinite convex chain from above and by an infinite concave chain from below. Each vertex of C^* is dual to the line supporting an edge of C. For a pair of polygons $C_1, C_2 \in \mathcal{C}$, an intersection point of the edges of C_1^*, C_2^* is dual to a common tangent of C_1 and C_2. Since C_1, C_2 are disjoint, the boundaries of C_1^*, C_2^* intersect in at most 4 points.

Let us consider the arrangement $\mathcal{A}(\mathcal{C}^*)$ of the polygonal chains bounding the regions C^*, for all $C \in \mathcal{C}$. It has $O(n+m^2)$ complexity, and can be computed in expected time $O(m^2 + n\log n)$, for instance by Mulmuley's randomized incremental algorithm [18, 6]. This algorithm actually computes the vertical decomposition $\mathcal{A}^{\|}(\mathcal{C}^*)$ of the arrangement, together with a point location data structure with $O(\log n)$ expected query time. We use this data structure to locate the points ℓ^* dual to all lines $\ell \in L \setminus R$. From this we can determine, for every ℓ, the regions of $\mathcal{C}^*$ containing ℓ^*, or in other words, the polygons of $\mathcal{C}$ intersecting ℓ. Indeed, after having located all points of the form ℓ^*, we traverse the adjacency graph of the trapezoids in $\mathcal{A}^{\|}(\mathcal{C}^*)$. At each trapezoid $\tau \in \mathcal{A}^{\|}(\mathcal{C}^*)$ we compute $\mathcal{C}^*(\tau)$, the set of regions that contain the trapezoid $\tau \in \mathcal{A}^{\|}(\mathcal{C}^*)$, and output the pairs (ℓ, C) for $\ell^* \in \tau$ and $C^* \in \mathcal{C}^*(\tau)$. Suppose we arrive at τ from τ', then $\mathcal{C}^*(\tau)$ and $\mathcal{C}^*(\tau')$ differ by at most one region (the region whose boundary separates τ from τ'), and thus $\mathcal{C}^*(\tau)$ can be obtained from $\mathcal{C}^*(\tau')$ in $O(1)$ time.

The total time spent in this step is $O(m^2 + n\log n)$ plus the number of polygon/line incidences. The expected number of these incidences is bounded by $O(\kappa(r,m)\cdot(n/r)) = O(n+m^2)$, using Proposition 2.1 with $r = n/t$ and $c = 1$.

Step 5. Let C be a cell in $\mathcal{A}(R,P)$, and let $L_C \subseteq L \setminus R$ be the set of lines intersecting the interior of C. For each line $\ell \in L_C$, we compute the trapezoids of $C^{\|}$ intersected by ℓ, as follows. Since the lines in L are sorted by their slopes, by being careful in Step 4, we can ensure that the lines of L_C are also sorted by their slopes. For each line $\ell \in L_C$ we compute the two vertices v_1, v_2 of C that support the lines parallel to ℓ (see Figure 1). This can be done, over all lines of L_C, in $O(|L_C|)$ time by merging the slopes of L_C with the slopes of the edges of C; we leave out the easy details for the reader. Next, we traverse ∂C in clockwise as well as counter-clockwise order in a lock-step fashion, starting from both v_1 and v_2 simultaneously (so we perform 4 traversals in a lock-step fashion, as depicted in Figure 1), until we reach an intersection point σ of ℓ and C. Since ℓ intersects C, we will eventually find such an intersection point. Finally, by tracing ℓ through $C^{\|}$,

[3]The latter estimate follows from the bound for $\kappa(n,m)$ mentioned in Section 1, in fact it is the weaker bound proved by Canham [4].

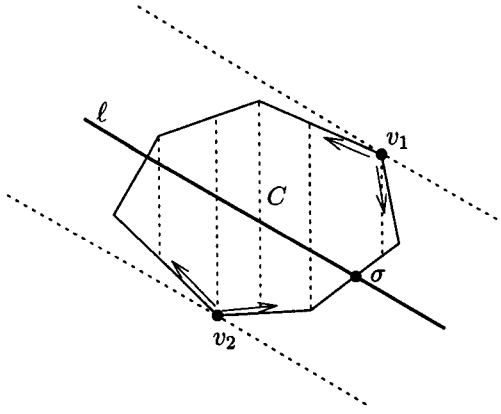

Figure 1: Finding σ.

starting from σ, we compute all k trapezoids of $C^{\shortparallel}$ that ℓ intersects. The time spent in finding σ and tracing ℓ is easily seen to be $O(k)$. Summing over all cells $C \in \mathcal{A}(R,P)$ and over all lines of L_C, the total time spent is $O(\sum_{\Delta \in \mathcal{A}^{\shortparallel}(R,P)} w(\Delta))$, whose expected value, by Proposition 2.1 (i), is $O(m^2+n)$.

Step 6. Let Δ be a trapezoid of $\mathcal{A}^{\shortparallel}(R,P)$. After having computed L_Δ, we compute the arrangement $\mathcal{A}(L_\Delta)$ using, say, a randomized incremental algorithm. We clip $\mathcal{A}(L_\Delta)$ within Δ, and compute the vertical decomposition of the clipped arrangement. For each point $p \in P \cap \Delta$, we also compute the trapezoid of this vertical decomposition containing p. The time spent in this step is easily seen to be $O(w(\Delta)^2 + |P \cap \Delta| \log w(\Delta))$ per trapezoid $\Delta \in \mathcal{A}^{\shortparallel}(R,P)$.

For a cell $C \in \mathcal{A}(R,P)$, let $\mathbf{\Delta}_C$ be the set of the resulting trapezoids that lie in C. We now define a graph $\mathcal{G}_C$ on the trapezoids of $\mathbf{\Delta}_C$. The vertices of $\mathcal{G}_C$ are the trapezoids of $\mathbf{\Delta}_C$, and two trapezoids are connected by an edge if they share a vertical edge. By performing a depth first search on $\mathcal{G}_C$, we can extract all connected components of $\mathcal{G}_C$ whose trapezoids contain any point of P. That is, we pick a point $p \in P \cap C$. Let $\tau_p \in \mathbf{\Delta}_C$ be the trapezoid containing p. We perform a depth first search in $\mathcal{G}_C$ starting from τ_p until we find the entire connected component of $\mathcal{G}_C$ containing τ_p. Let $\mathbf{\Delta}_C(p)$ be the set of trapezoids in this component; then the union of these trapezoids is exactly the cell of $\mathcal{A}(L,\{p\})$. The vertices of the cell, sorted in the clockwise order, can be easily obtained by merging the trapezoids of $\mathbf{\Delta}_C(p)$ in an obvious manner.

If there is a point $q \in P \cap C$ that does not lie in $\mathbf{\Delta}_C(p)$, we repeat the same procedure with q. We continue this process until we have extracted all components of $\mathcal{G}_C$ that contain any point of $P \cap C$. This gives $\mathcal{A}(L, P \cap C)$.

Repeating this step for all cells of $\mathcal{A}(R,P)$, we obtain all cells of $\mathcal{A}(L,P)$. Finally, we compute the vertical decomposition of all the cells. The total running time for Step 6 is

$$O(m \log n) + \sum_{\Delta \in \mathcal{A}(R,P)} O(w(\Delta)^2),$$

and its expected value is

$$O(m \log n + \kappa(r, m)(n/r)^2) = O(m^2 + n) .$$

Putting all the pieces together, the total expected running time of Steps 4–6 is $O(m^2 + n \log n)$. Let $T(n, m)$ denote the maximum expected time of the entire algorithm, then we obtain the following recurrence.

$$T(n, m) \leq \begin{cases} c_1 & \text{if } n \leq n_0, \\ \sum_{i=1}^{q} T(\lfloor n/t \rfloor, m_i) + C(m^2 + n \log n) & \text{if } n > n_0, \end{cases}$$

where $m_i \leq m/\sqrt{t}$ for $i \leq q = \lceil \sqrt{t} \rceil$, $\sum_{i=1}^{q} m_i = m$, and C is an appropriate constant. The solution of this recurrence is

$$T(n, m) = O(m^2 + n \log n) .$$

If $m > \sqrt{n \log n}$, we can divide the points of P into groups of size $\sqrt{n \log n}$, and solve the subproblems separately. This standard batching technique yields a more convenient bound for the expected running time, namely $O(m\sqrt{n \log n} + n \log n)$. Hence, we can conclude

Lemma 3.1 *Given a set L of n lines and a set P of $m \leq n^2$ points in the plane, the cells of $\mathcal{A}(L)$ containing the points of P can be computed by a randomized algorithm in expected time $O(m\sqrt{n \log n} + n \log n)$.*

We now present another randomized algorithm whose running time is significantly better for larger values of m. Although the basic idea is the same as in [1], the algorithm presented here is simpler because we allow randomization.

We choose a random subset $R \subseteq L$ of size r, where

$$r = \left\lceil \frac{m^{2/3}}{n^{1/3} \log^{1/3}(n/\sqrt{m})} \right\rceil .$$

Using a randomized incremental algorithm, we construct $\mathcal{A}^{\|}(R)$ plus a point-location data structure for $\mathcal{A}^{\|}(R)$ in expected time $O(r^2)$ [6]. For each trapezoid $\Delta \in \mathcal{A}^{\|}(R)$, let $L_\Delta \subseteq L \setminus R$ be the set of lines that intersect the interior of Δ and $P_\Delta \subseteq P$ the set of points that are contained in Δ. L_Δ can be computed in time $O(nr)$ by tracing each line through $\mathcal{A}^{\|}(R)$ and P_Δ can be computed in expected time $O(m \log n)$ by locating each point of P in $\mathcal{A}^{\|}(R)$. Set $n_\Delta = |L_\Delta|$ and $m_\Delta = |P_\Delta|$. For the sake of convenience, we assume that $\mathcal{A}(L_\Delta), \mathcal{A}(L_\Delta, P_\Delta)$ are clipped within Δ, and $\mathcal{A}^{\|}(L_\Delta), \mathcal{A}^{\|}(L_\Delta, P_\Delta)$ are their vertical decompositions. Let Z_Δ denote the set of cells in $\mathcal{A}(L_\Delta)$ that intersect the vertical edges of Δ. It is well-known that the number of edges in the faces in Z_Δ is $O(n_\Delta)$ [9].

Let p be a point of P_Δ. If the cell $\mathcal{A}(L_\Delta)$ containing p lies entirely in the interior of Δ, then $\mathcal{A}(L, \{p\}) = \mathcal{A}(L_\Delta, \{p\})$. Otherwise, $A(L, \{p\})$ may have edges that lie outside

Δ, but each such edge lies on the boundary of faces in $Z_{\Delta'}$, $\Delta' \in A^{\|}(R)$. Hence, for each trapezoid Δ, it is sufficient to compute $\mathcal{A}(L_\Delta, P_\Delta)$ and Z_Δ. We compute $\mathcal{A}(L_\Delta, P_\Delta)$ in expected time $O(m_\Delta\sqrt{n_\Delta \log n_\Delta} + n_\Delta \log n_\Delta)$ using Lemma 3.1. If we clip the lines of L_Δ within Δ, then Z_Δ is the unbounded face in the arrangement of the clipped segments, and we can compute it in expected time $O(n \log n)$, using a (simplified version of) the algorithm by Chazelle et al. [6]. Hence, the expected running time of the algorithm is

$$\mathrm{E}\Big[\sum_{\Delta\in\mathcal{A}^{\|}(R)} O\left(m_\Delta\sqrt{n_\Delta \log n_\Delta} + n_\Delta \log n_\Delta\right)\Big] + O(nr) + O(m \log n)\,.$$

By a result of Clarkson and Shor [10] (or also by Proposition 2.1), we have

$$\begin{aligned}\mathrm{E}\Big[\sum_{\Delta\in\mathcal{A}^{\|}(R)} n_\Delta\Big] &= O(nr) \quad \text{and} \\ \mathrm{E}\Big[\sum_{\Delta\in\mathcal{A}^{\|}(R)} m_\Delta\sqrt{n_\Delta \log n_\Delta}\Big] &= O\left(m\sqrt{\frac{n}{r}\log\frac{n}{r}}\right).\end{aligned}$$

Thus, the expected running time of the algorithm is bounded by

$$O(m\sqrt{\frac{n}{r}\log(n/r)} + nr\log\frac{n}{r} + m\log n)\,.$$

Substituting the value r in the above expression, we obtain

Theorem 3.2 *Given a set L of n lines and a set P of m points, the faces of $\mathcal{A}(L)$ containing the points of P can be computed by a randomized algorithm in expected time*

$$O(m^{2/3}n^{2/3}\log^{2/3}\frac{n}{\sqrt{m}} + (m+n)\log n)\,.$$

4 Computing cells in segment arrangements

Next, we present an algorithm for computing marked cells in arrangements of segments. Let S be a set of n segments and P a set of m points in the plane. The goal is to compute $\mathcal{A}(S, P)$ and its vertical decomposition $\mathcal{A}^{\|}(S, P)$. Again, we begin by a simpler algorithm which is effective for few cells, and then plug the random sampling technique to handle larger values of m.

The outline of the first algorithm is the same as in the previous section, except that we must now interpret the operations in terms of segments. Since the cells of $\mathcal{A}^{\|}(R, P)$ are not necessarily simply connected, we may have to deal with $m+n$ polygons even though there are only m cells. Consequently, the computation of the sets of cells intersected by each segment of $S \setminus R$ in Step 4 and the computation of S_Δ for each trapezoid $\Delta \in \mathcal{A}^{\|}(R, P)$ in Step 5 now become considerably more complicated. Another difficulty in computing S_Δ is that we now have to detect intersections between simple polygons and segments rather

than between convex polygons and lines. In the remainder of this section we will describe how to compute the sets S_Δ.

The boundary ∂C of each cell $C \in \mathcal{A}(R,P)$ is composed of (at most) one *outer* component and a family of *inner* components such that C lies in the interior of the outer component and in the exterior of each inner component. Each component of ∂C can be regarded as a simple polygonal chain. Let $\mathcal{O}$ be the set of outer boundary components of the cells in $\mathcal{A}(R,P)$, and let $\mathcal{I}$ be the set of the inner boundary components of these cells. We have $|\mathcal{O}| \leq m$ and $|\mathcal{I}| \leq m+n$. Let μ be the total number of edges of all polygons in $\mathcal{O} \cup \mathcal{I}$; obviously, $\mu \leq \eta(n/t, m)$.

We first decompose each segment $g \in S \setminus R$ into maximal subsegments, so that each subsegment lies in the interior of some outer component O. We cut each segment at the intersection points of $\mathcal{O}$ and S, and discard the subsegments that lie in the exterior of $\mathcal{O}$. Let Σ be the set of resulting subsegments. Next, for each subsegment $\sigma \in \Sigma$, we compute the trapezoids of $\mathcal{A}^{\text{II}}(R,P)$ intersected by σ.

Suppose that we have already computed Σ in Step 4. Then in Step 5 we compute S_Δ, for all $\Delta \in \mathcal{A}^{\text{II}}(R,P)$, as follows. We preprocess each polygonal chain $I \in \mathcal{I}$, in linear time, for ray shooting queries, so that the first intersection point of a query ray and I can be computed in logarithmic time, see [5, 15]. The total time spent in preprocessing $\mathcal{I}$ is $O(\mu) = O(\eta(n,m))$.

Let σ be a segment of Σ that lies in the interior of the outer component $O \in \mathcal{O}$ of ∂C. Let a, b be the endpoints of σ, and let $\Delta(a)$ be the trapezoid of C^{II} containing a. If a is not an endpoint of a segment of $S \setminus R$, then a lies on the boundary of $\Delta(a)$. We check whether $b \in \Delta(a)$. If the answer is 'yes', then $\Delta(a)$ is the only trapezoid of C^{II} intersected by σ, and we stop. If $b \notin \Delta(a)$, we compute the other intersection point, a_1, of σ and $\Delta(a)$. If a_1 lies on a vertical edge of $\Delta(a)$, we also compute, in constant time, the next trapezoid $\Delta(a_1)$ of C^{II} intersected by σ. We then repeat the same step with a_1 and $\Delta(a_1)$. If a_1, on the other hand, lies on an edge of the cell C, then a_1 lies on the boundary of some inner component $I \in \mathcal{I}$ of C, and the portion of the segment σ immediately following a_1 lies outside C. Using the ray shooting data structure, we compute the next intersection point a_2 of the polygonal chain I and the segment $\overrightarrow{a_1 b}$. Once we know a_2, we can also compute the trapezoid of C^{II} containing a_2, and we continue tracing σ through C^{II}.

For each trapezoid intersected by σ, we spend $O(\log n)$ time, so the total time spent in computing the k_σ trapezoids intersected by σ is $O(k_\sigma \log n)$. Summing over all segments of Σ, the total time spent is $\sum_{\sigma \in \Sigma} O(k_\sigma \log n) = O\left(\sum_{\Delta \in \mathcal{A}^{\text{II}}(R,P)} n_\Delta \log n\right)$, where $n_\Delta = |S_\Delta|$.

Next, we describe how to compute the set Σ. Notice that it is sufficient to compute all intersection points between the segments of $S \setminus R$ and the outer polygonal chains in $\mathcal{O}$.

Let J_O be an interval corresponding to the projection of the polygoal chain $O \in \mathcal{O}$ onto the x-axis, and let $\mathcal{J} = \{J_O \mid O \in \mathcal{O}\}$.

We construct an interval tree T on $\mathcal{J}$; see Mehlhorn [17] for details on interval trees. T is a minimum height binary tree with at most $2m$ leaves. Each node v of T is associated with an interval U_v, and a point x_v. Let $W_v = U_v \times [-\infty, +\infty]$ be a vertical strip, and let

h_v be the vertical line passing through x_v. For the root u, W_u is the entire plane and h_u is the vertical line passing the middle endpoint of the intervals of $\mathcal{J}$. Each interval $J \in \mathcal{J}$ is stored at the highest node v of T such that $x_v \in J$.

Let $\mathcal{J}_v$ be the set of intervals stored at v. We associate two subsets $\mathcal{O}_v, Z_v$ of $\mathcal{O}$ with v. Let $\mathcal{O}_v = \{O \mid J_O \in \mathcal{J}_v\}$ and $Z_v = \bigcup_w \mathcal{O}_w$, where the union is taken over all descendants of v, including v; set $m_v = |\mathcal{O}_v|$ and $z_v = |Z_v|$. Finally, let μ_v (resp. ζ_v) denote the total number of edges in $\mathcal{O}_v$ (resp. Z_v). Since each polygonal chain of $\mathcal{O}$ appears in exactly one $\mathcal{O}_v$, we have $\sum_{v \in T} \mu_v = \mu$ and $\sum_{v \in T} \zeta_v \leq 2\mu \log m$. Moreover, it can be shown that if v_1, v_2 are the children of v then $z_{v_1}, z_{v_2} \leq z_v/2$, which implies that $\sum_{v \in T} z_v^2 = O(m^2)$.

Since the polygonal chains in $\mathcal{O}_v$ are pairwise disjoint and all of them intersect a vertical line, we can regard $\mathcal{O}_v$ along with appropriate portions of the vertical line h_v as a simple polygon Π_v, and preprocess Π_v in $O(\mu_v)$ time for answering ray shooting queries. Using this data structure, one can report all k intersection points of a segment g and $\mathcal{O}_v$ in time $O((k+1)\log \mu_v)$.

Next, we take the convex hull of each polygonal chain in Z_v, and preprocess the resulting convex polygons into a data structure, as described in the previous section, so that all convex polygons intersected by a query line can be reported quickly. Since any two polygonal chains of $\mathcal{O}$ are disjoint, the boundaries of their convex hulls intersect in at most two points, and so they have at most 4 common tangents. Consequently, the line intersection searching structure has size $O(z_v^2 + \zeta_v)$. Moreover, it can be computed in time $O(z_v^2 + z_v \log \zeta_v + \zeta_v)$, using the algorithm of [19]. We also preprocess each $O \in \mathcal{O}$ in linear time for ray shooting queries as in [15]. It can be shown that the total preprocessing time is $O(m^2 + \sum_v (z_v \log \zeta_v + \zeta_v)) = O(m^2 + m \log m \log n + \mu \log m)$. We omit the details.

Let $g \in S \setminus R$ be a segment. All intersection points of g and $\mathcal{O}$ can be computed as follows. We search the tree T with g starting from the root. Let v be a node visited by the query procedure. If the endpoints of g do not lie in the vertical strip W_v, i.e., g completely crosses W_v, then g intersects $O \in Z_v$ if and only if the line supporting g intersects the convex hull of O. Thus, we first compute all polygonal chains of Z_v intersected by g, using the line intersection searching structure, and then, for each $O \in Z_v$ intersected by g, we compute the intersection points of g and O using the ray shooting data structure. If k_g^v is the number of intersection points between g and the polygonal chains of Z_v, then the total time in reporting these intersections is $O((k_g^v + 1)\log \zeta_v)$.

If one of the endpoints of g lies in W_v, we can compute all a_g^v intersection points between $\mathcal{O}_v$ and g in time $O((a_g^v + 1)\log \mu_v)$, using the ray shooting data structure for $\mathcal{O}_v$. Let v_1, v_2 be the children of the node v. If g intersects W_{v_1} (resp. W_{v_2}), we visit v_1 (resp. v_2). It is easily shown that the query procedure visits $O(\log m)$ nodes, and the query time is $O((\log m + k_g)\log n)$, where k_g is the total number of intersection points reported.

We repeat this procedure for all segments $g \in S \setminus R$. Since

$$\mu \leq \eta(n,m) = O(m^2 + n(\log m + \alpha(n))) \quad \text{and} \quad \sum_{g \in S \setminus R} k_g \leq \sum_{\Delta \in \mathcal{A}^{\parallel}(R,P)} n_\Delta ,$$

the total cost of computing the intersection points is

$$O((m^2+\mu)\log n)+\sum_{s\in S\setminus R} O((\log m+k_s)\log n)=$$

$$O\Big((m^2+n\log m+n\,\alpha(n))\log n+\sum_{\Delta\in\mathcal{A}^{\|}(R,P)} n_\Delta\log n\Big).$$

As in the previous section, the time spent in Step 6 (refining the cells of $\mathcal{A}^{\|}(R,P)$) is $O(\sum_\Delta n_\Delta^2)$. Using Proposition 2.1 (ii), we obtain that the total expected time spent in the merge step is $O\left((m^2+n\log m+n\,\alpha(n))\log n\right)$.

We thus obtain the following recurrence for $T(n,m)$ (the overall running time):

$$T(n,m)\le\begin{cases} c_1 & \text{if } n\le n_0,\\ \displaystyle\sum_{i=1}^{\sqrt{t}} T\left(\frac{n}{t},m_i\right)+O(m^2\log n+n(\log m+\alpha(n))\log n) & \text{if } n>n_0,\end{cases}$$

where $m_i\le m/\sqrt{t}$ for all $i\le\sqrt{t}$, and $\sum_i m_i=m$. The solution of this recurrence is $T(n,m)=O(m^2\log n+n(\log m+\alpha(n))\log n)$. Hence, we can conclude that the total running time of the first algorithm for computing $\mathcal{A}(S,P)$ is

$$O\left((m^2+n\log m+n\,\alpha(n))\log n\right).$$

We can again use the batching technique if m is large. Omitting the details, we obtain

Lemma 4.1 *Given a set S of n segments and a set P of m points, the faces of $\mathcal{A}(S)$ that contain at least one point of P can be computed by a randomized algorithm in expected time $O\left((m\sqrt{n\log m}+n(\log m+\alpha(n)))\log n\right)$.*

For larger values of m, we again use the random sampling technique as in the previous section. That is, we choose a random subset $R\subseteq S$ of size

$$r=\left\lceil\frac{m^{2/3}}{n^{1/3}}\cdot\frac{\log^{1/3}(n/\sqrt{m})}{\alpha^{2/3}(n/\sqrt{m})}\right\rceil,$$

and compute $\mathcal{A}^{\|}(R)$. For each $\Delta\in\mathcal{A}^{\|}(R)$, we compute $P_\Delta=P\cap\Delta$ and S_Δ, the set of segments that intersect Δ. We clip the segments within Δ. The total time spent in this step is $O(r^2+(m+nr)\log r)$. Let z be a point lying in the unbounded face of $\mathcal{A}(S)$. For each $\Delta\in\mathcal{A}^{\|}(R)$, we compute $\mathcal{A}^{\|}(S_\Delta,P_\Delta\cup\{z\})$, in time $O((m_\Delta\sqrt{n_\Delta\log m_\Delta}+n_\Delta(\log m_\Delta+\alpha(n_\Delta)))\log n_\Delta)$, using Lemma 4.1, and then glue them together. We omit the rather routine details from here. The overall expected running time of the algorithm is

$$\mathrm{E}\Big[\sum_{\Delta\in\mathcal{A}^{\|}(R)} O((n_\Delta(\alpha(n_\Delta)+\log m_\Delta)+m_\Delta\sqrt{n_\Delta\log m_\Delta}\Big]+O((m+nr)\log r).$$

Again, using the results by Clarkson-Shor [10] and substituting the value of r, we obtain

Theorem 4.2 *Given a set S of n segments and a set P of m points, the faces of $\mathcal{A}(S)$ that contain a point of P can be computed by a randomized algorithm in expected time $O(m^{2/3}n^{2/3}\log^{4/3}\frac{n}{\sqrt{m}}\,\alpha^{1/3}(\frac{n}{\sqrt{m}}) + (m + n\log m + n\,\alpha(n))\log n)$.*

Finally, let us remark that if $\mathcal{A}(S)$ has only $k = o(n^2)$ vertices, then using the fact that the expected number of trapezoids in $\mathcal{A}^{\|}(R)$ is $O(kr^2/n^2 + r)$, we can do a more careful analysis. Choosing $r = \left\lceil n\,(\frac{m}{k})^{2/3}\,\frac{\log^{1/3}(k/m)}{\alpha^{2/3}(k/m)}\right\rceil$, it can be shown that the expected running time of the algorithm is

$$O(m^{2/3}k^{1/3}\log^{4/3}\frac{k}{m}\alpha^{1/3}(\frac{k}{m}) + (m + n\log m + n\,\alpha(n))\log n)\,.$$

Acknowledgments. The authors thank Mark de Berg, Mark Overmars, and Micha Sharir for several useful discussions.

References

[1] P. Agarwal, Partitioning arrangements of lines: II. Applications, *Discrete and Computational Geometry* 5 (1990), 533–573.

[2] B. Aronov, H. Edelsbrunner, L. Guibas and M. Sharir, Improved bounds on the complexity of many faces in arrangements of segments, *Combinatorica*, 12 (1992), 261–274.

[3] M. de Berg, K. Dobrindt and O. Schwarzkopf, On lazy randomized incremental construction, to appear in *Proceedings 26th Annual ACM Symposium on Theory of Computing*, 1994.

[4] R. Canham, A theorem on arrangements of lines in the plane, *Israel J. Math.* 7 (1969), 393–397.

[5] B. Chazelle, H. Edelsbrunner, M. Grigni, L. Guibas, J. Hershberger, M. Sharir and J. Snoeyink, Ray shooting in polygons using geodesic triangulations, *Proc. 17th Int. Colloq. Automata, Languages and Programming*, 1991, pp. 661–673.

[6] B. Chazelle, H. Edelsbrunner, L. Guibas, M. Sharir and J. Snoeyink, Computing a face in an arrangement of line segments, *SIAM J. Computing* 22 (1993), 1286–1302.

[7] B. Chazelle and J. Friedman, A deterministic view of random sampling and its use in geometry, *Combinatorica* 10 (1990), 229–249.

[8] K. Clarkson, Computing a single face in an arrangement of segments, 1990, manuscript.

[9] K. Clarkson, H. Edelsbrunner, L. Guibas, M. Sharir and E. Welzl, Combinatorial complexity bounds for arrangements of curves and spheres, *Discrete and Computational Geometry* 5 (1990), 99–160.

[10] K. Clarkson and P. Shor, Applications of random sampling in computational geometry II, *Discrete and Computational Geometry* 4 (1989), 387–421.

[11] H. Edelsbrunner, *Algorithms in Combinatorial Geometry*, Springer-Verlag, Berlin, 1987.

[12] H. Edelsbrunner, L. Guibas and M. Sharir, The complexity of many faces in arrangements of lines and of segments, *Discrete and Computational Geometry* 5 (1990), 161–196.

[13] H. Edelsbrunner and E. Welzl, On the maximal number of edges of many faces in an arrangement, *Journal of Combinatorial Theory, Series A* 41 (1986), 159–166.

[14] L. Guibas and M. Sharir, Combinatorics and algorithms of arrangements, in *New Trends in Discrete and Computational Geometry* (J. Pach, ed.), Springer-Verlag, New York-Berlin-Heidelberg, 1993, 9–36.

[15] J. Hershberger and S. Suri, A pedestrian approach to ray shooting: Shoot a ray, take a walk, *Proc. 4th ACM-SIAM Symp. Discrete Algorithms*, 1993, pp. 54–63.

[16] D. Haussler and E. Welzl, ϵ-nets and simplex range queries, *Discrete Comput. Geom.* 2 (1987), 127–151.

[17] K. Mehlhorn, *Data Structures and Algorithms 3: Multi-dimensional Searching and Computational Geometry*, Springer-Verlag, Berlin, 1984.

[18] K. Mulmuley, A fast planar partition algorithm, I, *J. Symbolic Computation* 10 (1990), 253–280.

[19] K. Mulmuley, A fast planar partition algorithm, II, *J. Assoc. Comput. Mach.* 38 (1991), 74–103.

[20] E. Szemerédi and W. Trotter Jr., Extremal problems in discrete geometry, *Combinatorica* 3 (1983), 381–392.