SIAM J. COMPUT.
Vol. 27, No. 2, pp. 491–505, April 1998

COMPUTING MANY FACES IN ARRANGEMENTS OF LINES AND SEGMENTS[*]

PANKAJ K. AGARWAL[†], JIŘÍ MATOUŠEK[‡], AND 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.

Key words. arrangements, random sampling, duality

AMS subject classifications. 68Q20, 68R05, 68U05

PII. S009753979426616X

1. Introduction.

Given a finite set S of lines in the plane, the *arrangement* of S, denoted by $\mathcal{A}(S)$, is the cell complex induced by S. The 0-faces (or *vertices*) of $\mathcal{A}(S)$ are the intersection points of S, the 1-faces (or *edges*) are maximal portions of lines of S that do not contain any vertex, and the 2-faces (called *cells*) are the connected components of $\mathbb{R}^2 - \bigcup S$. 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 an arrangement of lines, 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 range of computational geometry problems can be formulated in terms of computing such arrangements or their parts [11, 14].

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

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

where the maximum is taken over all sets of n lines and over all sets of m points in

[*]Received by the editors April 18, 1994; accepted for publication (in revised form) March 11, 1996. 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 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 7141 (project ALCOM II: *Algorithms and Complexity*).

http://www.siam.org/journals/sicomp/27-2/26616.html

[†]Department of Computer Science, Box 90129, Duke University, Durham, NC 27708-0129 (pankaj@euclid.cs.duke.edu).

[‡]Department of Applied Mathematics, Charles University, Malostranské nám. 25, 118 00 Praha 1, Czech Republic (matousek@kam.mff.cuni.cz).

[§]Department of Computer Science, Utrecht University, P. O. Box 80.089, 3508 TB Utrecht, the Netherlands. Current address: Department of Computer Science, Hong Kong University of Science & Technology, Clear Water Bay, Kowloon, Hong Kong.

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] and the lower bound follows from a result of Szemerédi and Trotter [21]; previous results and related work can be found in Canham [4] and Edelsbrunner and Welzl [13].

In this paper we study the problem of computing $\mathcal{A}(S,P)$, that is, for each cell $C \in \mathcal{A}(S,P)$, we want to return the vertices of C in, say, clockwise order. We will refer to the cells of $\mathcal{A}(S,P)$ as the *marked* cells of $\mathcal{A}(S)$. Edelsbrunner, Guibas, and Sharir [12] presented a randomized algorithm, based on the random-sampling technique [16], for computing $\mathcal{A}(S,P)$, 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 variety of geometric problems, which add the input objects one by one in random order and maintain the desired structure; see e.g., [6, 10, 18, 19]. In our case, we can add the lines of S one by one in random order and maintain the marked cells in the arrangement of lines added so far. However, the expected running time of this approach is $\Omega(n\sqrt{m} + m\log n)$ 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 $\mathcal{A}(S,P)$. Notice that for $m \le \sqrt{n}$, the expected running time of the algorithm is $O(n\log n)$, which matches the known lower bound for computing a single cell in line arrangements. We then apply the randomized geometric divide-and-conquer technique in a standard way, obtaining an algorithm with expected time

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

We also study a similar but more difficult problem of computing the marked cells in an arrangement of n line 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 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 over all 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, Guibas, and Sharir [12], and a slightly faster deterministic algorithm is presented by Agarwal [1]. See [20] for results on computing a single cell in arrangements of segments.

Following the same strategy as for the case of lines, we first develop a randomized algorithm with $O((m^2 \log m + n \log m + n\,\alpha(n)) \log n)$ expected running time. Note that the above upper bound for $\eta(n,m)$ is not known to be tight, and a bound such as $\eta(n, \sqrt{n}) = O(n\,\alpha(n))$ (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 into 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

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

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

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

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 the 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 in the collection $\mathcal{A}(S,P)$, let $C^\parallel$ denote the collection of trapezoids in the vertical decomposition[1] of C, and let $\mathcal{A}^\parallel(S,P) = \bigcup_{C \in \mathcal{A}(S,P)} C^\parallel$ denote the set of trapezoids in the vertical decomposition of $\mathcal{A}(S,P)$. Abusing the notation slightly, we will use $\mathcal{A}^\parallel(S,P)$ to denote the corresponding planar subdivision as well. For any subset $X \subseteq S$ and any trapezoid $\Delta \in \mathcal{A}^\parallel(X,P)$, 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,

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

where c is a small constant; for instance, $c = 2$. Well-known results concerning the so-called ε-nets (Haussler and Welzl [16]) imply that, for every $\Delta \in \mathcal{A}^\parallel(R)$, $w(\Delta) \le a(n/r)\log r$ with high probability, where a is a suitable constant. This inequality can be used to derive a weaker 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 S be a set of n lines and P a set of m points in the plane. If $R \subseteq S$ is a random subset of size r, where each subset of size r is chosen with equal probability, then for any constant $c \ge 1$,*

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

[1]The vertical decomposition $C^\parallel$ of a cell C in an arrangement of segments (or of lines) is obtained by drawing a vertical line segment from each vertex of C in both directions (within C) until it hits another edge of C. If the segment does not intersect any edge of C, then it is extended to infinity.

(ii) *Let S be a set of n segments and P a set of m points in the plane. If $R \subseteq S$ is a 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 the approach by Chazelle and Friedman [7], which is somewhat different from the Clarkson–Shor method. Recently, de Berg, Dobrindt, and Schwarzkopf [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 $\mathrm{CT}(R)$; in Proposition 2.1 the objects are lines or segments, the regions are trapezoids, and $\mathrm{CT}(R) = \mathcal{A}^{\|}(R,P)$. 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 four 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 $\mathrm{CT}(R)$. 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 $S, \mathrm{CT}(R), D(\Delta)$, and $K(\Delta)$ be such that for any subset $R \subseteq S$, $\mathrm{CT}(R)$ satisfies the following axioms:

(i) For any $\Delta \in \mathrm{CT}(R)$, $D(\Delta) \subseteq R$ and $R \cap K(\Delta) = \emptyset$, and
(ii) If $\Delta \in \mathrm{CT}(R)$ and R' is a subset of R with $D(\Delta) \subseteq R'$, then $\Delta \in \mathrm{CT}(R')$.

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

For any natural number t, let

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

We establish the following.

LEMMA 2.2. *Given a set S of n 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* $\mathrm{CT}(R)$*,* $D(\Delta)$*, and* $K(\Delta)$ *satisfy axioms* (i) *and* (ii) *above, we have*

$$\mathrm{E}\left[|\mathrm{CT}_t(R)|\right] = O(2^{-t}) \cdot \mathrm{E}\left[|\mathrm{CT}(R')|\right], \tag{2.2}$$

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

[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 applications.

Roughly speaking, Lemma 2.2 says that the expected number of "large" trapezoids in $\mathrm{CT}(R)$, that is, 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 $\Delta \in \mathrm{CT}(R)$.

This assumption implies that the presence of Δ in $\mathrm{CT}(R)$ 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 $\mathrm{CT}(R)$ is the vertical decomposition of *all* cells in $\mathcal{A}(R)$.

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

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

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

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

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

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 $\Pr[\Delta \in \mathrm{CT}(R)]$ using the following definition of conditional probability:

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

and analogously,

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

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

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

To see this, consider the following random experiment. Select a set R' by including all the elements of $D(\Delta)$ into it, and adding $r' - |D(\Delta)|$ elements chosen randomly among the elements of $S \setminus (D(\Delta) \cup K(\Delta))$. By definition of the conditional probability, the probability that Δ appears in $\mathrm{CT}(R')$ is precisely $\Pr[\Delta \in \mathrm{CT}(R') \mid A'_\Delta]$. Now take this R' and add $r - r'$ randomly chosen elements of $S \setminus (R' \cup K(\Delta))$ to it, obtaining a set R. The distribution of this R is clearly the same as if we took the elements of $D(\Delta)$ and added $r - |D(\Delta)|$ random elements of $S \setminus (D(\Delta) \cup K(\Delta))$, and hence the probability of $\Delta \in \mathrm{CT}(R)$ is $\Pr[\Delta \in \mathrm{CT}(R) \mid A_\Delta]$. On the other hand, since R was created by adding extra elements to R', whenever Δ is present in $\mathrm{CT}(R)$ it must be in $\mathrm{CT}(R')$ as well, and thus (2.6) holds.

Therefore,

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

(Note that $r' = \lfloor r/t \rfloor \ge d$, 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 $\delta = |D(\Delta)| \le d$, $w = w(\Delta)$, and for two nonnegative integers $a \le x$, let $x^{\underline{a}} = x(x-1)\cdots(x-a+1)$. Then

$$\frac{\Pr[A_\Delta]}{\Pr[A'_\Delta]} = \frac{\binom{n-w-\delta}{r-\delta}}{\binom{n}{r}} \cdot \frac{\binom{n}{r'}}{\binom{n-w-\delta}{r'-\delta}} \le \frac{r^{\underline{d}}}{r'^{\underline{d}}} \cdot \frac{(n-w-r')^{\underline{r-r'}}}{(n-r')^{\underline{r-r'}}}\,.$$

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

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

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

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

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

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

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 $R \subseteq S$ 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., $CT(R) = \mathcal{A}^{\Vert}(R, P)$. Obviously, $|CT(R)| \le \kappa(r, m)$. Now

$$\begin{aligned} &\mathrm{E}\left[\sum_{\Delta \in \mathcal{A}^{\Vert}(R,P)} w(\Delta)^c\right] \\ &= \mathrm{E}\left[\sum_{t \ge 1}\left(t\frac{n}{r}\right)^c (|CT_t(R)| - |CT_{t-1}(R)|)\right] \\ &\le \left(\frac{n}{r}\right)^c \sum_{t \ge 0}(t+1)^c \cdot \mathrm{E}\,[|CT_t(R)|] \\ &= \left(\frac{n}{r}\right)^c \sum_{t \ge 0} t^c O(2^{-t}) \cdot \kappa\left(\frac{r}{t}, m\right) \\ &\le \kappa(r,m)\left(\frac{n}{r}\right)^c \sum_{t \ge 0} O(t^c \cdot 2^{-t}) \\ &= \kappa(r,m) \cdot O\left((n/r)^c\right)\,. \end{aligned}$$ □

3. Computing cells in line arrangements. Let S 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 S are sorted by their slopes. In this section we describe a randomized algorithm for computing $\mathcal{A}(S, P)$. In fact, it computes the vertical decomposition $\mathcal{A}^{\parallel}(S, P)$ of $\mathcal{A}(S, P)$. Each face of $\mathcal{A}^{\parallel}(S, P)$ is a trapezoid, bounded by at most two vertical segments and portions of at most two edges of a cell of $\mathcal{A}^{\parallel}(S, P)$. We begin by presenting a very simple randomized algorithm for computing $\mathcal{A}^{\parallel}(S, P)$ with $O((m^2 + n) \log n)$ expected time, which we will use as a subroutine in the main algorithm. This algorithm is optimal for $m \le \sqrt{n}$. If $n \le n_0$, 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 $R \subseteq S$ of $r = \lfloor n/t \rfloor$ lines.
2. Partition P into $q = \lceil \sqrt{t}\, \rceil$ subsets $P_1, \dots, P_q$, each of size at most $k = \lfloor m/\sqrt{t} \rfloor$, where
$$\begin{aligned} P_i &= \{p_{(i-1)k+1}, \dots, p_{ik}\} \quad \text{for } i < q, \\ P_q &= \{p_{(q-1)k+1}, \dots, p_m\}. \end{aligned}$$
3. For each $i \le q$, compute $\mathcal{A}^{\parallel}(R, P_i)$ 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.) Since P is sorted in the x-direction, it is easy to detect multiple copies of a cell. In this way, we obtain $\mathcal{A}^{\parallel}(R, P)$.
4. For each line $\ell \in S \setminus R$, compute the cells of $\mathcal{A}(R, P)$ that ℓ intersects.
5. For each trapezoid Δ of $\mathcal{A}^{\parallel}(R, P)$, compute the set $S_\Delta \subseteq S \setminus R$ of lines that intersect the interior of Δ.
6. For each trapezoid $\Delta \in \mathcal{A}^{\parallel}(R, P)$, compute the arrangement of lines of S_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 $\mathcal{A}^{\parallel}(S, P)$ 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 $S \setminus R$. 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 $S \setminus R$ 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 four points.

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

[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].

$O((m^2+n)\log n)$, using a randomized incremental algorithm [6, 18]. This algorithm actually computes the vertical decomposition $\mathcal{A}^{\parallel}(\mathcal{C}^*)$ 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 $\ell \in S \setminus R$. 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 $\mathcal{A}^{\parallel}(\mathcal{C}^*)$. At each trapezoid $\tau \in \mathcal{A}^{\parallel}(\mathcal{C}^*)$ we compute $\mathcal{C}^*(\tau)$, the set of regions that contain the trapezoid $\tau \in \mathcal{A}^{\parallel}(\mathcal{C}^*)$, 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 $O(\kappa(r,m)\cdot (n/r)) = O(m^2+n)$, 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 $S_C \subseteq S \setminus R$ be the set of lines intersecting the interior of C. For each line $\ell \in S_C$, we compute the trapezoids of $C^{\parallel}$ intersected by ℓ, as follows. We compute, in $O(\log n)$ time, the intersection points of ∂C and ℓ and also the trapezoids of $C^{\parallel}$ containing these intersection points. Next, by tracing ℓ through $C^{\parallel}$, we compute all b trapezoids of $C^{\parallel}$ that ℓ intersects. The time spent in finding the intersection points and tracing ℓ is $O(\log n + b)$. By repeating this procedure for all cells C and all lines $\ell \in L \setminus R$, we obtain S_Δ, for every $\Delta \in \mathcal{A}^{\parallel}(R,P)$. The running time of this step is $O(\sum_{\Delta \in \mathcal{A}^{\parallel}(R,P)} w(\Delta)\log n)$. The expected value of $O(\sum_{\Delta \in \mathcal{A}^{\parallel}(R,P)} w(\Delta))$, by Proposition 2.1 as before, is $O(m^2+n)$. Hence, the expected time spent in step 5 is $O((m^2+n)\log n)$.

Step 6. Let Δ be a trapezoid of $\mathcal{A}^{\parallel}(R,P)$. After having computed S_Δ, we compute the arrangement $\mathcal{A}(S_\Delta)$ using, say, a randomized incremental algorithm. We clip $\mathcal{A}(S_\Delta)$ 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 $\Delta \in \mathcal{A}^{\parallel}(R,P)$.

For a cell $C \in \mathcal{A}(R,P)$, let $\mathbf{\Delta}_C$ be the set of 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 $\mathcal{A}(S, \{p\})$. The vertices of the cell, sorted in clockwise order, can be 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 $\mathcal{A}(S, P \cap C)$.

Repeating this step for all cells of $\mathcal{A}(R,P)$, we obtain all cells of $\mathcal{A}(S,P)$. 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.

$$T(n,m) \le \begin{cases} c_1 & \text{if } n \le n_0, \\ \displaystyle\sum_{i=1}^{q} T\left(\lfloor n/t \rfloor, m_i\right) + c_2(m^2+n)\log n & \text{if } n > n_0, \end{cases}$$

where $m_i \le m/\sqrt{t}$ for $i \le q = \lceil \sqrt{t}\, \rceil$, $\sum_{i=1}^{q} m_i = m$, and c_1, c_2 are appropriate constants. The solution of this recurrence is

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

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

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

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 $R \subseteq S$ of size r, where $r = \lceil m^{2/3}/n^{1/3} \rceil$. Using a randomized incremental algorithm, we construct $\mathcal{A}^{\Vert}(R)$ plus a point-location data structure for $\mathcal{A}^{\Vert}(R)$ in expected time $O(r^2)$ [6]. For each trapezoid $\Delta \in \mathcal{A}^{\Vert}(R)$, let $S_\Delta \subseteq S \setminus R$ 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. S_Δ can be computed in time $O(nr)$ by tracing each line through $\mathcal{A}^{\Vert}(R)$ and P_Δ can be computed in expected time $O(m\log n)$ by locating each point of P in $\mathcal{A}^{\Vert}(R)$. Set $n_\Delta = |S_\Delta|$ and $m_\Delta = |P_\Delta|$. For the sake of convenience, we assume that $\mathcal{A}(S_\Delta), \mathcal{A}(S_\Delta, P_\Delta)$ are clipped within Δ, and $\mathcal{A}^{\Vert}(S_\Delta), \mathcal{A}^{\Vert}(S_\Delta, P_\Delta)$ are their vertical decompositions. Let Z_Δ denote the set of cells in $\mathcal{A}(S_\Delta)$ 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 $\mathcal{A}(S_\Delta)$ containing p lies entirely in the interior of Δ, then $\mathcal{A}(S, \{p\}) = \mathcal{A}(S_\Delta, \{p\})$. Otherwise, $A(S,\{p\})$ may have edges that lie outside Δ, but each such edge lies on the boundary of faces in $Z_{\Delta'}$, $\Delta' \in A^{\Vert}(R)$. Hence, for each trapezoid Δ, it is sufficient to compute $\mathcal{A}(S_\Delta, P_\Delta)$ and Z_Δ. We compute $\mathcal{A}(S_\Delta, P_\Delta)$ in expected time $O((m_\Delta\sqrt{n_\Delta}+n_\Delta)\log n_\Delta)$ using Lemma 3.1. If we clip the lines of S_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 $O(n_\Delta \log n_\Delta)$ using a (simplified version of) the algorithm by Chazelle et al. [6]. Hence, the expected running time of the algorithm is

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

By a result of Clarkson and Shor [10, Theorem 3.6], we have

$$\mathrm{E}\left[\sum_{\Delta\in\mathcal{A}^{\|}(R)} n_\Delta \log n_\Delta\right] = O\left(nr\log\frac{n}{r}\right) \quad \text{and}$$

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

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

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

Substituting the value r in the above expression, we obtain the following theorem.

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

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

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 with a simpler algorithm, which is effective for computing few cells, and then plug in 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, the boundary of m cells may consist of $m+n$ polygonal chains. 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 the details of Steps 4 and 5.

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; 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 $|\mathcal{O}|\le m$ and $|\mathcal{I}|\le m+n$. Let μ be the total number of edges of all polygons in $\mathcal{O}\cup\mathcal{I}$; obviously,

$$\mu \le \eta(n/t,m) = O(m^2\log m + n\log m + n\alpha(n)); \tag{4.1}$$

the last inequality follows from a weaker result of Aronov et al. [2].

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 $\mathcal{A}^{\|}(R, P)$ intersected by σ.

Suppose that we have already computed Σ in Step 4. Then in Step 5 we compute S_Δ, for all $\Delta \in \mathcal{A}^{\|}(R, P)$, 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 subsegment of Σ that lies in the interior of the outer component of the boundary of a cell C. We trace σ through $C^{\|}$ to compute the trapezoids of $C^{\|}$ intersected by σ. In more detail, let a, b be the endpoints of σ, and let $\Delta(a)$ be the trapezoid of $C^{\|}$ 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 $C^{\|}$ 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 $C^{\|}$ intersected by σ and 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 $C^{\|}$ containing a_2, and we continue tracing σ through $C^{\|}$.

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

$$\sum_{\sigma \in \Sigma} O(k_\sigma \log n) = O\left(\sum_{\Delta \in \mathcal{A}^{\|}(R,P)} n_\Delta \log n \right),$$

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 $\mathcal{J}$ be the set of intervals corresponding to the x-projections of the polygonal chains in $\mathcal{O}$. We construct in time $O(m \log m)$ an interval tree T on $\mathcal{J}$; see [17] for details on interval trees. T is a minimum-height binary tree with at most $2m$ leaves, each of whose nodes v is associated with a vertical strip W_v and a point x_v; x_v is the median of the endpoints of $\mathcal{J}_v$ that lie in the interior of W_v. For the root u, W_u is the entire plane. If v, z are the children of u, then W_v and W_z are obtained by splitting W_u into two vertical strips by the vertical line passing through x_u. An interval $J \in \mathcal{J}$ is stored at the highest node v of T for which $x_v \in J$.

Let $\mathcal{O}_v \subseteq \mathcal{O}$ be the subset of polygonal chains whose projections are stored at v. Each chain belongs to exactly one $\mathcal{O}_v$. Let $Z_v = \bigcup_w \mathcal{O}_w$, where the union is taken over all descendants of v, including v; set $z_v = |Z_v|$. Finally, let ζ_v denote the total number of edges in Z_v. Since each polygonal chain of $\mathcal{O}$ appears in at most $O(\log m)$ Z_v's, we have $\sum_{v \in T} \zeta_v = O(\mu \log m)$, where μ is, as above, the total number of edges in $\mathcal{O}$. Moreover, by the construction of T, if v_1, v_2 are the children of v, then $z_{v_1}, z_{v_2} \leq z_v/2$, and therefore $\sum_{v \in T} z_v^2 = O(m^2)$.

The polygonal chains in $\mathcal{O}_v$ are pairwise disjoint and all of them intersect the vertical line passing through x_v, so we can regard $\mathcal{O}_v$ along with appropriate portions of the vertical line as a simple polygon and preprocess it in linear time for answering ray-shooting queries. The time spent in this step is $O(\mu)$, as each polygonal chain belongs to exactly one O_v. Using this data structure, all a intersection points of a segment g and $\mathcal{O}_v$ can be reported in time $O((a+1)\log n)$.

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 they have at most four common tangents. Consequently, the line intersection-searching structure has size $O(z_v^2+\zeta_v)$, and it can be computed in time $O(z_v^2 + z_v \log \zeta_v + \zeta_v)$, using the algorithm described in [19]. We also preprocess each $O \in \mathcal{O}$ in linear time for ray-shooting queries as in [15]. The total time spent in preprocessing $\mathcal{O}$ is therefore

$$
\begin{aligned}
&O\left(m \log m + \mu + \sum_v \left(z_v^2 + z_v \log \zeta_v + \zeta_v\right)\right) \\
(4.2) \qquad &= O(m^2 + m \log m \log n + \mu \log m) \\
&= O((m^2 \log m + n \log m + n\alpha(n)) \log m),
\end{aligned}
$$

where the last inequality follows from (4.1).

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. We need to compute the intersection points of g and the polygonal chains in Z_v. If the endpoints of g do not lie in the vertical strip W_v, that is, g completely crosses W_v, then g intersects a polygonal chain $O \in Z_v$ 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 the line supporting 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 a is the number of intersection points between g and the polygonal chains of Z_v, then the total time in reporting these intersections is $O((a+1)\log n)$. If, on the other hand, one of the endpoints of g lies in W_v, we can compute all b intersection points between $\mathcal{O}_v$ and g in time $O((b+1)\log n)$ 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 recursively visit v_1 (resp., v_2).

It is easily seen that the query procedure visits $O(\log m)$ nodes and that 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

$$\sum_{g \in S\setminus R} k_g \;\le \sum_{\Delta \in \mathcal{A}^{\parallel}(R,P)} n_\Delta\,,$$

the total cost of computing the intersection points between $S \setminus R$ and $\mathcal{O}$ is

$$\sum_{g\in S\setminus R} O((\log m + k_g)\log n) = O\left(n \log m \log n + \sum_{\Delta\in\mathcal{A}^{\parallel}(R,P)} n_\Delta \log n\right).$$

The expected value of $\sum_\Delta n_\Delta$ is $\eta(r,m)O(n/r) = O(\eta(r,m))$; therefore we obtain

$$(4.3) \qquad \sum_{g\in S\setminus R} O((\log m + k_g)\log n) = O(n \log m \log n + \eta(r,m)\log n).$$

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

$$\text{E}\left[\sum_\Delta n_\Delta^2\right] = \eta(r,m)O((n/r)^2) = O(\eta(r,m)). \tag{4.4}$$

Summing (4.2), (4.3), and (4.4), and using the fact that $m \le n^2$, the total expected time spent in the merge step is $O((m^2 \log m + n \log m + n\,\alpha(n)) \log n)$.

This gives the following recurrence for $T(n, m)$ (the maximum expected running time):

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

where $m_i \le m/\sqrt{t}$ for all $i \le \sqrt{t}$. The solution of this recurrence is

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

Hence, we can conclude that the expected running time of the overall algorithm for computing $\mathcal{A}(S, P)$ is

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

We can again use the same batching technique if m is large. That is, partition P into groups of size $\sqrt{n}$ and solve the subproblems separately. Omitting the details, we obtain the following lemma.

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

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 $r = \lceil m^{2/3}/n^{1/3} \rceil$ 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 expected 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 expected time

$$O((m_\Delta \sqrt{n_\Delta} \log m_\Delta + n_\Delta \log m_\Delta + n_\Delta \alpha(n_\Delta)) \log n_\Delta),$$

using Lemma 4.1, and then glue these faces together. The overall expected running time of the algorithm is

$$\text{E}\left[\sum_{\Delta \in \mathcal{A}^{\|}(R)} O((m_\Delta \sqrt{n_\Delta} \log n_\Delta + n_\Delta \alpha(n_\Delta) + n_\Delta \log m_\Delta) \log n_\Delta)\right] + O((m + nr) \log r) . \tag{4.5}$$

Again, using the results by Clarkson–Shor [10], we have

$$\text{E}\left[\sum_{\Delta \in \mathcal{A}^{\|}(R)} m_\Delta \sqrt{n_\Delta} \log n_\Delta\right] = O\left(m\sqrt{\frac{n}{r}} \log \frac{n}{r}\right) \tag{4.6}$$

$$\sum_{\Delta \in \mathcal{A}^{\|}(R)} n_\Delta = \text{E}\left[|\mathcal{A}^{\|}(R)|\right] O\left(\frac{n}{r}\right) = O(nr). \tag{4.7}$$

Substituting (4.5), (4.6), and the value of r in (4.7), we obtain the following theorem.

THEOREM 4.2. *Given a set S of n segments and a set P of m points in the plane, the faces of $\mathcal{A}(S)$ that contain a point of P can be computed by a randomized algorithm in expected time*

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

Finally, we 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 obtain a better bound on the expected running time of the algorithm. In particular, choosing $r=\lceil n(m/k)^{2/3}\rceil$ and using (4.4), (4.5), it can be shown that the expected running time of the algorithm is

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

Acknowledgments. The authors thank Mark de Berg, Mark Overmars, and Micha Sharir for several useful discussions. The comments by an anonymous referee helped improve the presentation of the paper.

REFERENCES

[1] P. K. AGARWAL, *Partitioning arrangements of lines:* II. *Applications*, Discrete Comput. Geom., 5 (1990), pp. 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), pp. 261–274.

[3] M. DE BERG, K. DOBRINDT, AND O. SCHWARZKOPF, *On lazy randomized incremental construction*, Discrete Comput. Geom., 14 (1995), pp. 261–286.

[4] R. CANHAM, *A theorem on arrangements of lines in the plane*, Israel J. Math., 7 (1969), pp. 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*, Algorithmica, 12 (1994), pp. 54–68.

[6] B. CHAZELLE, H. EDELSBRUNNER, L. GUIBAS, M. SHARIR, AND J. SNOEYINK, *Computing a face in an arrangement of line segments*, SIAM J. Comput., 22 (1993), pp. 1286–1302.

[7] B. CHAZELLE AND J. FRIEDMAN, *A deterministic view of random sampling and its use in geometry*, Combinatorica, 10 (1990), pp. 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 Comput. Geom., 5 (1990), pp. 99–160.

[10] K. CLARKSON AND P. SHOR, *Applications of random sampling in computational geometry* II, Discrete Comput. Geom., 4 (1989), pp. 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 Comput. Geom., 5 (1990), pp. 161–196.

[13] H. EDELSBRUNNER AND E. WELZL, *On the maximal number of edges of many faces in an arrangement*, J. Combin. Theory Ser. A, 41 (1986), pp. 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, Heidelberg, 1993, pp. 9–36.

[15] J. HERSHBERGER AND S. SURI, *A pedestrian approach to ray shooting: Shoot a ray, take a walk*, J. Algorithms, 18 (1995), pp. 403–431.

[16] D. HAUSSLER AND E. WELZL, *ϵ-nets and simplex range queries*, Discrete Comput. Geom., 2 (1987), pp. 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 Comput., 10 (1990), pp. 253–280.

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

[20] M. Sharir and P. K. Agarwal, *Davenport-Schinzel Sequences and Their Geometric Applications*, Cambridge University Press, New York, 1995.

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