Pseudo-line Arrangements: Duality, Algorithms, and Applications *

Pankaj K. Agarwal† Micha Sharir‡

Abstract

A collection L of n x-monotone unbounded Jordan curves in the plane is called a family of *pseudo-lines* if every pair of curves intersect in at most one point, and the two curves cross each other there. Let P be a set of m points in $\mathbb{R}^2$. We define a *duality* transform that maps L to a set L^* of points in $\mathbb{R}^2$ and P to a set P^* of pseudo-lines in $\mathbb{R}^2$, so that the incidence and the "above-below" relationships between the points and pseudo-lines are preserved. We present an efficient algorithm for computing the dual arrangement $\mathcal{A}(P^*)$ under an appropriate model of computation. We also propose a dynamic data structure for reporting, in $O(m^\varepsilon + k)$ time, all k points of P that lie below a query arc, which is either a circular arc or a portion of the graph of a polynomial of fixed degree. This result is needed for computing the dual arrangement for certain classes of pseudo-lines arising in our applications, but is also interesting in its own right. We present a few applications of our dual arrangement algorithm, such as computing incidences between points and pseudo-lines and computing a subset of faces in a pseudo-line arrangement.

Next, we present an efficient algorithm for cutting a set of circles into arcs so that every pair of arcs intersect in at most one point, i.e., the resulting arcs constitute a collection of *pseudo-segments*. By combining this algorithm with our algorithm for computing the dual arrangement of pseudo-lines, we obtain efficient algorithms for a number of problems involving arrangements of circles or circular arcs, such as detecting, counting, or reporting incidences between points and circles.

1 Introduction

The *arrangement* of a finite collection Γ of geometric curves or surfaces in $\mathbb{R}^d$, denoted as $\mathcal{A}(\Gamma)$, is the decomposition of the space into relatively open connected cells of dimensions $0, \ldots, d$ induced by Γ, where each cell is a maximal connected set of points lying in the intersection of a fixed subset of Γ and avoiding all other elements of Γ. Besides being interesting in their own right, due to the rich geometric, combinatorial, algebraic, and topological structures that they possess, arrangements also lie at the heart of numerous geometric problems arising in a wide range of applications, including robotics, computer graphics, molecular modeling, and computer vision. Study of arrangements of lines and hyperplanes has a long, rich history. A summary of early work on arrangements can be found in [20, 21]. Although hyperplane arrangements already possess a rich structure, many applications (e.g., motion-planning in robotics and molecular modeling) call for a systematic study of arrangements of arcs in the plane and of surface patches in higher dimensions. There has been much work in this area in the last two decades; see [8] for a review of recent results.

A collection L of n unbounded Jordan curves is called a family of *pseudo-lines* if every pair of curves intersects in at most one point, and the two curves cross each other there. Arrangements of pseudo-lines were probably first studied by Levi [24]; see [16, 19, 21] for the known results on pseudo-line arrangements. The work by Goodman and Pollack on allowable sequences [16] shows that any arrangement of pseudo-lines can be transformed into an arrangement of x-monotone pseudo-lines that is isomorphic to the original one. Such a transformation, however, is not efficient for the algorithms that we seek to develop, and we will thus confine our analysis to x-monotone pseudo-lines. For such pseudo-lines, the above/below relationship, which will be used a lot in our analysis, is naturally defined.

Many of the combinatorial results related to arrangements of lines (e.g., complexity of a single face, complexity of many faces, complexity of a level, etc.)

*Work on this paper has been supported by a grant from the U.S.-Israeli Binational Science Foundation. Work by Pankaj Agarwal was also supported by Army Research Office MURI grant DAAH04-96-1-0013, by a Sloan fellowship, and by NSF grants EIA-98-70724, EIA-99-72879, ITR-333-1050, CCR-97-32787, and CCR-00-86013. Work by Micha Sharir was also supported by NSF Grants CCR-97-32101 and CCR-00-98246, by a grant from the Israel Science Fund (for a Center of Excellence in Geometric Computing), and by the Hermann Minkowski–MINERVA Center for Geometry at Tel Aviv University.

†Department of Computer Science, Duke University, Durham, NC 27708-0129, USA. E-mail: pankaj@cs.duke.edu

‡School of Computer Science, Tel Aviv University, Tel Aviv 69978, Israel; and Courant Institute of Mathematical Sciences, New York University, New York, NY 10012, USA. E-mail: sharir@cs.tau.ac.il

hold for arrangements of pseudo-lines as well.

It has been shown that various families of arcs (e.g., circular, parabolic, etc.) can be converted into a family of *pseudo-segments* (subarcs, each pair of which intersect at most once), by cutting the arcs into a relatively small number of pieces. Chan [11] has shown that a collection of N pseudo-segments can be cut further into $O(N \log N)$ subarcs, each of which can be extended into an unbounded x-monotone curve, so that these curves constitute a family of pseudo-lines. One can then use the close relationship between line and pseudo-line arrangements to solve a variety of problems involving arrangements of arcs; see [2, 7, 9, 11, 30].

In this paper, we focus on algorithmic problems involving arrangements of pseudo-lines in the plane, problems that are much less studied than the corresponding combinatorial problems. Of course, one has to assume a reasonable representation of the given pseudo-lines, in order to develop efficient algorithms for their manipulation, so we assume, for example, that the given pseudo-lines are algebraic (or semi-algebraic) curves of fixed maximum degree, and that our model of computation allows us to perform, in constant time, exact computations involving any constant number of such curves. However, even with these assumptions, several algorithms for line arrangements do not extend routinely to pseudo-line arrangements. A stumbling block in many of these algorithms, when we try to extend them to the case of pseudo-lines, is that they use some kind of a *duality* transform that maps lines to points and points to lines. Typically, one uses the duality that maps a line $\ell : y = ax + b$ to a point $\ell^* = (a, b)$ and a point $p = (\alpha, \beta)$ to the line $p^* : y = -\alpha x + \beta$ [13]. Note that ℓ lies above (resp., below, on) p if and only if ℓ^* lies above (resp., below, on) p^*.

Burr *et al.* [10] had raised the question whether a similar dual transform exists for pseudo-lines. Goodman [15], based on his work with Pollack on allowable sequences [17, 18], defined a dual transform for (not necessarily x-monotone) pseudo-lines in the projective plane, that preserves the incidence relationship. That is, given a set L of n pseudo-lines and a set P of m points in $\mathbb{R}^2$, the transform yields a set L^* of points and a set P^* of pseudo-lines, so that a point p of P lies on a pseudo-line $\ell \in L$ if and only if the dual point ℓ^* lies on the dual pseudo-line p^*. Goodman's construction has several disadvantages from an algorithmic point of view. First, his construction is defined in the projective plane, and, consequently, it does not (and cannot, without considerable modifications) handle the above-below relationship. A more significant problem, from the algorithmic point of view, is that his construction requires that for each pair of the given points there exists an input pseudo-line passing through this pair. Although the existence of such a pseudo-line follows from the classical result of Levi [24], computing such a pseudo-line seems to be a highly nontrivial task.

We define a different dual transform, which may be regarded as an extension of Goodman's construction, and which overcomes the technical problems mentioned above. Suppose we have a data structure for storing the m points of P, which can report, in $O(f(m)+k)$ time, all k points of P that lie below a query pseudo-line ℓ, which can determine, in $O(f(m))$ time, whether any point of P lies below a query pseudo-line ℓ, and which can be updated in $O(f(m))$ time after inserting or deleting a point into/from P. Using such a data structure as a "black box," we present a sweep-line algorithm for constructing the dual arrangement $\mathcal{A}(P^*)$ that runs in time $O((m^2 + n) f(m) \log m)$. We note that if $f(m)$ is small, say polylogarithmic in m or of the form $O(m^\varepsilon)$, for any $\varepsilon > 0$, then this bound is nearly optimal. It is a bound of this kind that was missing so far in the algorithmic applications alluded to above.

Next, we describe a data structure for preprocessing a set P of m points in the plane so that all k points of P lying below the graph of a query fixed-degree polynomial can be reported in $O(m^\varepsilon + k)$ time.[1] It can also determine, in $O(m^\varepsilon)$ time, whether any point of P lies below a query curve. A point can be inserted or deleted into/from P in $O(\log^2 m)$ time. Although our approach is closely based on Matoušek's algorithm [25] for reporting points that lie below a query line, a number of technical difficulties have to be overcome to extend this algorithm to the case of algebraic curves. A similar data structure also works for circular arcs.

Using our arrangement algorithm, we show that all incidences between a set P of m points and a set L of n pseudo-lines can be reported in time $O(m^{2/3-\varepsilon} n^{2/3+2\varepsilon} + n^{1+\varepsilon} + m^{1+\varepsilon})$, provided the pseudo-lines in L are extensions of bounded-degree polynomial arcs or of circular arcs (or for any other family of arcs for which a data structure with the above properties can be constructed). More precisely, we assume that all our arcs have the same x-projection, and that they are extended to pseudo-lines in some simple manner, e.g., by horizontal rays. We also describe an algorithm, with the same running time, for computing the faces of $\mathcal{A}(L)$ that have a nonempty intersection with a set P of m "marking points," none of which lie on any arc of L. Our algorithm works also for a set of congruent circles, thereby improving on the best-known algorithm, which required $O(n\sqrt{m} \log n)$ randomized expected time [27].

[1] We follow the convention that an upper bound of the form $O(g(n, \varepsilon))$ means that for each $\varepsilon > 0$ there is a constant c_ε such that the actual bound is $c_\varepsilon g(n, \varepsilon)$.

Let L be a family of n *pseudo-circles* in the plane, which is a collection of closed Jordan curves, each pair of which intersect at most twice. Recently, there has been considerable work on the problem of splitting the curves in such a family L into arcs (pseudo-segments), so that each pair of arcs intersect in at most one point. This work started with the paper of Tamaki and Tokuyama [30], and has continued with recent papers of Aronov and Sharir [9], Chan [11], and Agarwal et al. [7]. Since the resulting set of arcs is a collection of pseudo-segments, one can obtain bounds on the complexity of various substructures in arrangements of pseudo-circles by applying the known results for pseudo-segment arrangements. This approach has recently been used to obtain, among other results, nontrivial upper bounds on the complexity of a level in an arrangement of pseudo-circles [7, 11, 30], on the number of incidences between points and circles or parabolas [7, 9], and on the complexity of many faces in an arrangement of circles or parabolas [2, 7]. However, none of the preceding results were algorithmic.

In this paper we present an $O(n^{3/2+\varepsilon})$-time algorithm for splitting a set of n circles into $O(n^{3/2+\varepsilon})$ pseudo-segment arcs. The recent algorithms of Solan [29] and of Har-Peled [22] can be used or adapted for this task, but the running time of the resulting solutions would be close to $O(n^{7/4})$. Our algorithm follows the general approach of these algorithms, but it uses additional tools and a more refined analysis to obtain the bound stated above.

Combining this algorithm with our new algorithms for handling arrangements of pseudo-lines, we obtain algorithms that detect, count, or report all incidences between m points and n circles, in time that is close to the best upper bounds known for the number of such incidences (as provided in [7, 9]).

Finally, our duality result has recently found another application, in [28], where another duality, between graphs drawn in the plane and sets of vertices in pseudo-line arrangements, is obtained.

2 Duality for Points and Pseudo-lines

Let L be a set of n pseudo-lines and P a set of m points in the plane. Let W be a vertical strip that contains all points of P and all vertices of $\mathcal{A}(L)$. Let λ and ρ be the left and right boundary lines of W. We clip the pseudo-lines of L to within W, and thus assume that L is a set of x-monotone arcs whose left and right endpoints lie on λ and ρ, respectively; see Figure 1 (a). An x-monotone Jordan arc that crosses W completely splits W into two regions. We will refer to each of these regions as a *pseudo-halfplane.*

We now present a duality transform that maps L to a set L^* of n points and P to a set P^* of m pseudo-lines so that the incidences and the above-below relationships between the points and pseudo-lines are preserved. We first describe the duality in a manner that, albeit being constructive, is not concerned with real algorithmic efficiency. We then show how to implement the construction in an efficient manner. For simplicity, we assume that no point of P lies on any pseudo-line of L. The construction and the proof can easily be extended to handle this case. Sort the pseudo-lines of L in increasing order of their intercepts with λ. Map each pseudo-line $\ell \in L$ to the point $\ell^*(i_\ell, 0)$, where i_ℓ is the rank of the intercept $\ell \cap \lambda$ along λ. In other words, the dual points all lie on the x-axis, and appear there in the same order as the y-order of the intercepts of the corresponding curves with λ. Note that, since we are dealing with (x-monotone unbounded) pseudo-lines, the y-coordinates of the dual points, as well as the exact spacings between their x-coordinates, are not important. One can always move any dual point up or down (arbitrarily) or left or right (without passing over another dual point), and deform the dual pseudo-lines accordingly, so that the incidences, the above-below relationships, and the pseudo-line property, are all preserved. See Figure 1 for an illustration.

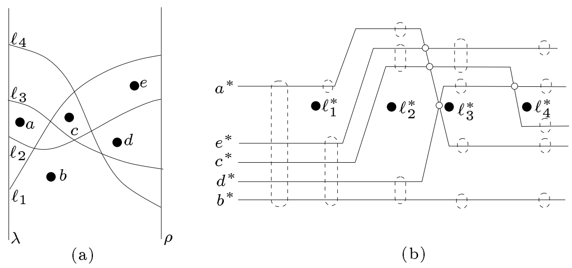

Figure 1: The duality transform: (a) The primal setting. (b) The dual representation; the dashed ovals show the bundles maintained by the sweep-line algorithm for constructing $\mathcal{A}(P^*)$.

Each point $p \in P$ is mapped to an x-monotone curve p^* that is constructed to obey the following (necessary) rule: For each pseudo-line $\ell \in L$, if p lies above (resp., below, on) ℓ, draw p^* to pass above (resp., below, through) the point ℓ^*. This rule does not fully specify the curves p^*, but, with some care, as we will see next, this rule yields a drawing of these curves as a collection of pseudo-lines.

We next show how to sort the dual curves p^*, for $p \in P$, at $x = -\infty$. Let us first assume that no pair of points of P lie in the same face of $\mathcal{A}(L)$. In the present course of analysis, we have no way to distinguish between any two points that lie in the same face, and we simply regard such a pair as identical.

initially	$(abcde)$					
after ℓ_1^*	$(bcde)$	$\bullet$	a			
after ℓ_2^*	(bd)	$\bullet$	(ce)	a		
after ℓ_3^*	b	a	$\bullet$	d	(ce)	
after ℓ_4^*	b	a	c	$\bullet$	d	e

Table 1: The evolution of the y-structure during the sweep. The symbol $\bullet$ denotes the location in the y-structure of the point ℓ_j^* just being swept.

We define the following relation on P^*: For two points $p, q \in P$ we say that $p^* \prec q^*$ if the pseudo-line $\ell \in L$ with the lowest λ-intercept that separates p and q is such that p lies below ℓ and q lies above ℓ. We denote this relationship by $p < \ell < q$.

LEMMA 2.1. *The relationship $\prec$ is a total order on P^*.*

Proof. (This simplified proof was suggested by Pavel Valtr.) For each $p \in P$, let σ_p be the sequence of pseudo-lines of L that lie below p, sorted in increasing λ-intercept order. Order the sequences σ_p lexicographically, but with the twist that after removing a common identical prefix, a nonempty sequence precedes an empty one. The relationship $\prec$ is then identical with this lexicographical order, as is easily checked. $\square$

We now show how to draw the curves of P^* so that they form an arrangement of pseudo-lines. As noted, for the time being, we are not concerned about the efficiency of the procedure given below; we only want to show that the pseudo-line property can be enforced. First we sort the curves by $\prec$ and draw them at $x = -\infty$ in this increasing order. In general, we draw the curves from left to right as horizontal, parallel curves until we are about to sweep past some dual point ℓ^*. We compute the sets $A(\ell^*)$, $B(\ell^*)$, consisting of those points that lie above (resp., below) ℓ. This allows us to find all *inversions* enforced by ℓ^*, namely, all pairs (p, q), such that p^* passed above q^* before ℓ^*, but at ℓ^* we have that p^* passes below ℓ^* whereas q^* passes above that point. We then draw the curves past ℓ^* so that exactly those inverted pairs cross each other. To achieve this, we take all the curves in $A(\ell^*)$ (that have to pass above ℓ^*), and bend them simultaneously, keeping them parallel to each other, so that they do not intersect among themselves. We apply a symmetric deformation to the curves in $B(\ell^*)$. In this way, it is clear that intersections arise exactly between the inverted pairs. After sweeping past ℓ^*, we bend all curves back to horizontal and continue like this to the right. If ℓ^* contains any point of P, then the corresponding pseudo-lines pass through ℓ^*. See Figure 4. We invite the reader to verify that Figure 1(b) is a (somewhat deformed but topologically equivalent) realization of this drawing procedure, applied to the configuration in Figure 1(a). We prove that this procedure does indeed produce an arrangement of pseudo-lines. We first need the following lemma.

LEMMA 2.2. *There do not exist three pseudo-lines $\ell_1, \ell_2, \ell_3 \in L$ and two points $p, q \in P$ such that (i) ℓ_1^* lies to the left of ℓ_2^*, which lies to the left of ℓ_3^*, (ii) the curve p^* passes above ℓ_1^* and ℓ_3^*, and below ℓ_2^*, and (iii) the curve q^* passes below ℓ_1^* and ℓ_3^*, and above ℓ_2^*.*

Proof. Refer to Figure 2. Suppose to the contrary that there exists such a configuration. Interpreting it in the primal plane, we have that point p lies above ℓ_1 and below ℓ_2, and point q lies below ℓ_1 and above ℓ_2. Hence these points lie in different wedges of the double wedge formed between ℓ_1 and ℓ_2. Since ℓ_1 has smaller λ-intercept than that of ℓ_2, it is easily seen that p has to lie to the left of q. Repeating the same argument for the pseudo-lines ℓ_2 and ℓ_3, we conclude this time that q lies to the left of p, a contradiction that establishes the lemma. $\square$

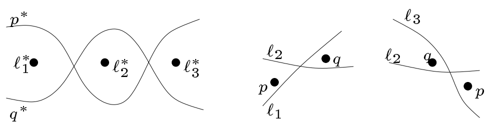

Figure 2: Illustration to the proof of Lemma 2.2.

Using the above lemma, we can prove the following.

LEMMA 2.3. *There do not exist two pseudo-lines $\ell_1, \ell_2 \in L$ and two points $p, q \in P$ such that (i) ℓ_1^* lies to the left of ℓ_2^*, (ii) the curve p^* passes below q^* at $x = -\infty$, (iii) the curve p^* passes above ℓ_1^* and below ℓ_2^*, and (iv) the curve q^* passes below ℓ_1^* and above ℓ_2^*.*

Omitting further details, we obtain the main result of this section:

THEOREM 2.1. *For a finite set P of points and a finite set L of pseudo-lines in the plane, the above transformation maps L into a set of points and P into a set of pseudo-lines, so that the incidence and the above-below relationships between P and L are preserved.*

3 Constructing the Dual Arrangement

Let L^* denote the set of points dual to the pseudo-lines of L, and let P^* denote the set of pseudo-lines dual to the points of P, as defined in the preceding section. We describe an efficient algorithm for computing the arrangement $\mathcal{A}(P^*)$. That is, we compute an *incidence* *graph* of $\mathcal{A}(P^*)$ in which there is a node for every face — vertex, edge, and two-dimensional facet — of $\mathcal{A}(P^*)$, and two nodes associated with the faces ϕ_1 and ϕ_2 are connected by an arc if $\phi_1 \subseteq \partial\phi_2$ and $\dim(\phi_1) + 1 = \dim(\phi_2)$. Moreover, the output also records, for each $\ell^* \in L^*$, the vertex, edge, or 2-face of $\mathcal{A}(P^*)$ that contains ℓ^*.

We construct $\mathcal{A}(P^*)$ by sweeping a vertical line from left to right that stops at every point of L^*. The difficulty in performing the sweep is that we do not know how to compare the y-ordering of two dual pseudo-lines at a given vertical line. For example, suppose we want to compare two dual pseudo-lines p^*, q^* at $x = -\infty$. By definition, we need to find, in the primal plane, all the pseudo-lines ℓ that separate p and q, and determine the order of p and q using the line with the smallest λ-intercept. Computing this set of separating pseudo-lines is nontrivial and time consuming, and we cannot afford to do it explicitly. We therefore sweep the line without maintaining the total ordering of pseudo-lines in P^*, which is only progressively revealed as the sweep proceeds. More precisely, let $\ell_1^*, \ell_2^*, \ldots, \ell_n^*$ be the sequence of points in L^* sorted by their x-coordinates. The algorithm maintains the invariant that it has computed the following structure after processing ℓ_i^*:

(I.1) A partition $\Pi_i = \langle P_1, \ldots, P_{u_i}\rangle$ of P into subsets, referred to as as *bundles*. Two points of P lie in the same bundle of Π_i if and only if they lie in the same face of $\mathcal{A}(L_i)$, where $L_i = \{\ell_1, \ldots, \ell_i\}$. For any $p \in P_j$ and $q \in P_{j+1}$, the pseudo-line p^* lies below q^* immediately to the right of ℓ_i^*. That is, the bundles are sorted by the y-ordering along the sweep line, but the vertical order of the pseudo-lines within each bundle is yet undetermined. See Figure 1 (b) and Table 1.

(I.2) Regard all dual pseudo-lines in each bundle P_j as a single "thick" pseudo-line γ_j (say, choose a representative dual pseudo-line from each bundle), and let $\Gamma_i = \{\gamma_j \mid 1 \le j \le u_i\}$. The algorithm has computed the portion of $\mathcal{A}(\Gamma_i)$ up to the vertical line passing through ℓ_i^*.

At the end, after processing ℓ_n^*, each bundle in Π_n consists of a single dual pseudo-line. (Two points that remain in the same bundle at the end of the algorithm must lie in the same face of $\mathcal{A}(L)$, and, for our purpose, can be considered identical.) Therefore Π_n gives the ordering of P^* at $x = +\infty$ and $\mathcal{A}(\Gamma_n) = \mathcal{A}(P^*)$.

In the ith step, while processing ℓ_i^*, the algorithm constructs Π_i and $\mathcal{A}(\Gamma_i)$ from Π_{i-1} and $\mathcal{A}(\Gamma_{i-1})$, respectively, as follows.

Computing Π_i**.** For each bundle $P_j \in \Pi_{i-1}$, split P_j into two subsets P_j^- and P_j^+, where P_j^- (resp., P_j^+) is the set of points in P_j that lie below (resp., above) ℓ_i. Let $\Pi_i^- = \langle P_j^- \mid P_j \in \Pi_{i-1}, P_j^- \ne \emptyset\rangle$ and $\Pi_i^+ = \langle P_j^+ \mid P_j \in \Pi_{i-1}, P_j^+ \ne \emptyset\rangle$. Set $\Pi_i = \Pi_i^- \circ \Pi_i^+$, where $\circ$ denotes concatenation.

Computing $\mathcal{A}(\Gamma_i)$**.** For each $P_j \in \Pi_{i-1}$, if both P_j^- and P_j^+ are nonempty, then $p \prec q$ for any $(p, q) \in P_j^- \times P_j^+$, so we can refine the ordering of pseudo-lines in P^* at $x = -\infty$ (this is not done explicitly — it will be a byproduct of the other steps described next). We split the corresponding thick pseudo-line γ_j into two pseudo-lines γ_j^- and γ_j^+ and refine $\mathcal{A}(\Gamma_{i-1})$, with γ_j^- lying below γ_j^+; γ_j^+ lies above ℓ_i^*, and γ_j^- lies below it. Roughly speaking, every edge of $\mathcal{A}(\Gamma_i)$ that lies on γ_j is now replaced by a thin "pseudo-rectangle," as shown in Figure 3. We omit the details from this abstract.

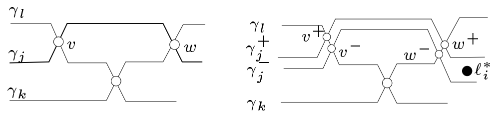

Figure 3: Splitting a thick pseudo-line; every edge lying on γ_j becomes a pseudo-rectangular face.

Next, if we have two nonempty bundles P_j^- and P_k^+ such that $k < j$, then γ_j^- lies above γ_k^+ just to the left of ℓ_i^* but below γ_k^+ at ℓ_i^*, so they induce a vertex of $\mathcal{A}(\Gamma_i)$ to the left of ℓ_i^*. We create this new vertex and update the incidence graph. Note that in general many pairs (P_j^-, P_k^+) may create such a crossing before ℓ_i^*, as shown in Figure 4. We update $\mathcal{A}(\Gamma_i)$ accordingly.

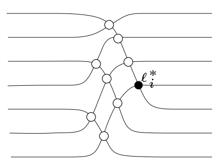

Figure 4: Several pairs of bundles cross before ℓ_i^*.

With some extra care, the algorithm can also handle pseudo-lines that pass through points of P. Details are omitted in this abstract.

Lemma 3.1. *The above two steps maintain the invariant (I.1) and (I.2).*

Once we have computed Π_i^- and Π_i^+ and determined the bundles that have been split into two nonempty bundles, the rest of the computation can be carried out in time proportional to the change in the size of the incidence graph of the arrangement, whose accumulated cost is only $O(m^2)$. It thus suffices to describe how to compute Π_i^- and Π_i^+ efficiently. We maintain a weight-balanced binary tree $\mathcal{T}$ whose jth leftmost leaf stores the bundle P_j [26]. For each node $v \in \mathcal{T}$, let $S_v \subseteq P$ denote the set of points stored at the leaves of the subtree rooted at v. At each node $v \in \mathcal{T}$, we maintain a data structure $\mathcal{D}_v = \mathcal{D}(S_v)$ that supports the following operations on S_v:

EMPTY$_v(\gamma)$: Is one of the pseudo-halfplanes determined by γ empty (of points of S_v)? If so, which one?

INSERT$_v(p)$: Insert a point p into S_v.

DELETE$_v(p)$: Delete a point p from S_v.

SPLIT$_v(\gamma)$: Let S_v^+, S_v^- be the subset of points of S_v that lie above and below γ, respectively. Split $\mathcal{D}(S_v)$ into $\mathcal{D}(S_v^+)$ and $\mathcal{D}(S_v^-)$.

We will describe in the next section a data structure that supports these operations efficiently. For now, assume that each of these operations can be performed in $O(f(m))$ (amortized) time. Then we compute Π_i^- and Π_i^+, as follows.

While processing ℓ_i, we visit $\mathcal{T}$ in a top-down manner. Suppose we are at a node $v \in \mathcal{T}$. We execute EMPTY$_v(\ell_i)$ on $\mathcal{D}(S_v)$. If it returns "yes," then we mark v by '+' (resp., by '−') if S_v lies entirely above (resp., below) ℓ_i. If the procedure returns "no" and v is a leaf, then we perform SPLIT$_v(\ell_i)$, create two children v^- and v^+ of v, mark v^- (resp., v^+) by '−' (resp., by '+'), store S_v^- (resp., S_v^+) at v^- (resp., at v^+), and associate $\mathcal{D}(S_v^-)$ (resp., $\mathcal{D}(S_v^+)$) with v^- (resp., with v^+). Otherwise (if the points of S_v lie on both sides of ℓ_i and v is not a leaf), we recursively visit the two children of v. Let V^- (resp. V^+) denote the nodes of $\mathcal{T}$ marked '−' (resp., '+'). Let $A = \langle u_{i_1}, \ldots, u_{i_a} \rangle$ and $B = \langle w_{j_1}, \ldots, w_{j_b} \rangle$ be the sequence of (new) leaves of $\mathcal{T}$, sorted from left to right, in the subtrees rooted at nodes in V^- and V^+, respectively. Note that we have $\Pi_i^- = \langle S_{u_{i_1}}, \ldots, S_{u_{i_a}} \rangle$ and $\Pi_i^+ = \langle S_{w_{j_1}}, \ldots, S_{w_{j_b}} \rangle$. We finish the step by re-arranging the leaves, the interior nodes, and the secondary structures of $\mathcal{T}$, so that all leaves of A appear before those of B, i.e., the sequence of leaves after re-ordering is $(u_{i_1}, \ldots, u_{i_a}, w_{j_1}, \ldots, w_{j_b})$. Suppose $u_{i_t}, \ldots, u_{i_a}$ appear to the right of w_{j_1}. Then we delete these leaves from $\mathcal{T}$, and we also delete the points stored at these leaves from the secondary data structures stored at the ancestors of these nodes. We then re-insert these leaves, and the corresponding points, before w_{j_1} in the correct order. We also update the data strucures stored at the ancestors of these leaves.

Let μ_i and ν_i be, respectively, the number of leaves of $\mathcal{T}$ that are split, and the number of vertices of $\mathcal{A}(\Gamma_i)$ that are created in the ith step. Then $|V^-| + |V^+| = O((\mu_i + 1)\log m)$. Hence, the total time spent in traversing $\mathcal{T}$ and splitting the leaves is $O(\mu_i f(m) \log m)$. Since $\sum_{i=1}^{n} \mu_i \leq m - 1$, the total time spent in these steps during the entire sweep is $O((n + m) f(m) \log m)$. The total time spent in re-arranging the tree is $O((\sum_{j=t}^{a} |S_{i_j}|) f(m) \log m)$ because a point is deleted from the secondary structures of only $O(\log m)$ nodes. However, $\sum_{j=t}^{a} |S_{i_j}| \leq 2\nu_i$, so the total time spent in re-arranging $\mathcal{T}$ and updating its secondary structures is $O(\nu_i f(m) \log m)$. Since $\sum_i \nu_i = O(m^2)$, we conclude:

THEOREM 3.1. *Let L be a set of n pseudo-lines and P a set of m points in the plane. Suppose we have a data structure that supports each of the four operations described above in $O(f(m))$ amortized time. Then we can construct $\mathcal{A}(P^*)$ (in the sense prescribed in the beginning of this section), in $O((m^2 + n) f(m) \log m)$ time.*

We will show in the next section that if L is a set of circular arcs or bounded-degree polynomial arcs (with a common x-projection), then $f(m) = O(m^\varepsilon)$, so we obtain the following.

COROLLARY 3.1. *Let L be a set of n circular arcs or bounded-degree polynomial arcs with a common x-projection, each pair of which intersect at most once, and let P be a set of m points in the plane. Then we can construct $\mathcal{A}(P^*)$ (with respect to an extension of the arcs of L into pseudo-lines) in $O((m^2 + n) m^\varepsilon)$ time. Moreover, for each point $p \in P$, the above algorithm can also return, within the same asymptotic time bound, the set of arcs in L that contain p. If there is no arc passing through p, then the algorithm can return the arcs that lie immediately above and below p.*

4 Pseudo-Halfplane Range Reporting

Let W be a vertical strip, and let Γ be a collection of x-monotone arcs whose endpoints lie on the left and right boundaries of W. Each arc $\gamma \in \Gamma$ splits W into two (closed) regions. As above, we call each of these regions a *pseudo-halfplane* bounded by γ. Let S be a set of m points lying inside the strip W. We wish to preprocess S into a data structure that supports the four operations described in the previous section – EMPTY, INSERT, DELETE, and SPLIT, with respect to arcs $\gamma \in \Gamma$. In addition, we want the data structure to support the following REPORT (g, k) operation: Let g be one of the pseudo-halfplanes bounded by an arc $\gamma \in \Gamma$, and let k be an integer. REPORT (g, k) reports $\min\{|S \cap g|, k\}$ points of $S \cap g$. Note that EMPTY (γ) can be answered by performing the queries REPORT $(\gamma^+, 1)$

and REPORT $(\gamma^-, 1)$, where γ^-, γ^+ are the two pseudo-halfplanes bounded by γ. The following lemma is easy to prove.

LEMMA 4.1. *If a data structure supports the operations* INSERT, DELETE *in $O(f(m))$ amortized time and* REPORT *(g, k) in $O(f(m) + k)$ time, then* SPLIT *can be performed in $O(f(m) \log m)$ amortized time.*

Proof. Let $\gamma \in \Gamma$ be the query arc, and let γ^+, γ^- be the two pseudo-halfplanes bounded by γ. By invoking REPORT $(\gamma^-, 2^i)$ and REPORT $(\gamma^+, 2^i)$ repeatedly and alternately, with $i = 1, 2, 3, \ldots$, we can determine which of γ^-, γ^+ contains fewer points, up to a factor of 2. Suppose $|S \cap \gamma^-| = \mu \le 2|S \cap \gamma^+|$. Then the above procedure reports all points of $S \cap \gamma^-$ in $O(f(m) \log m + \mu)$ time. We delete the points of $S \cap \gamma^-$ from the data structure and reconstruct a new data structure on $S \cap \gamma^-$. A standard analysis shows that each point is deleted at most $O(\log m)$ times. The amortized cost of each split operation is thus $O(f(m) \log m)$. □

Hence, it suffices to describe a data structure that supports the INSERT, DELETE, and REPORT operations efficiently. We present such a data structure for two special cases: (i) Γ is a set of circular arcs, and (ii) Γ is a set of (portions of the) graphs of polynomials of bounded degree.

4.1 Querying with circular arcs

Let Γ be the set of circular arcs whose endpoints lie on the left and right boundaries of W. We construct a weight-balanced binary tree $\mathcal{T}$ on the y-coordinates of the points in S [26]. For a node $v \in \mathcal{T}$, let $S_v \subseteq S$ be the set of points whose y-coordinates are stored at the leaves of the subtree rooted at v, and put $m_v = |S_v|$. We map each point $p = (x_p, y_p) \in S_v$ to the point $\bar{p} = (x_p, y_p, x_p^2 + y_p^2)$ in $\mathbb{R}^3$. Let $\bar{S}_v = \{\bar{p} \mid p \in S_v\}$. We preprocess $\bar{S}_v$ into a dynamic data structure, proposed by Agarwal and Matoušek [6], for reporting, in time $O(m_v^\varepsilon + k)$, all k points of $\bar{S}_v$ that lie in a query halfspace h in $\mathbb{R}^3$. This data structure can easily be modified, so that queries of the following form can also be answered efficiently: given a parameter μ, report $\min\{|\bar{S}_v \cap h|, \mu\}$ points of S_v lying in the query halfspace. A query takes $O(m_v^\varepsilon + \mu)$ time, and a point can be inserted into or deleted in $O(\log^2 m)$ time.

Let g be the region lying below an arc $\gamma \in \Gamma$. Suppose γ lies in the upper semicircle of the circle C_γ; let a denote the y-coordinate of the center of C_γ, and let D_γ denote the disk bounded by C_γ. A REPORT (g, k) query is answered as follows. We first identify $O(\log m)$ nodes $v_1, \ldots v_s$ of $\mathcal{T}$ so that $\bigcup_i S_{v_i}$ is the set of points in S whose y-coordinates are at most a. Since each $S_{v_i} \subseteq g$, by visiting the v_i's one by one, we can report, in $O(\mu)$ time, $\mu = \min\{|\bigcup_i S_{v_i}|, k\}$ points of S whose y-coordinates are at most a. If we have reported fewer than k points, then we identify $O(\log m)$ nodes $w_1, \ldots, w_r$ of $\mathcal{T}$ so that $\bigcup_i S_{w_i}$ is the set of points whose y-coordinates are at least a. A point $p \in S_{w_i}$ lies in the halfplane g if and only if $p \in D_\gamma$. We map C_γ to a plane $\bar{C}_\gamma$ in $\mathbb{R}^3$, using the standard lifting transform, so that $p \in D_\gamma$ if and only if $\bar{p}$ lies below the plane $\bar{C}_\gamma$. We visit the w_i's one by one, and at each node w_i we do the following: Suppose we have reported μ points so far. We then report $\min\{k - \mu, |S_{w_i} \cap C_\gamma|\}$ points of $S_{w_i} \cap C_\gamma$ using the secondary structure stored at w_i. If we have reported a total of k points, we stop. Otherwise, we update the value of μ and visit w_{i+1}. The total time spent in this procedure is $O(m^\varepsilon + k)$.

Using the standard partial-rebuilding technique [26], a point can be inserted or deleted into/from the overall structure in $O(\log^3 m)$ time. Hence, we obtain the following:

THEOREM 4.1. *Let Γ and S be as above. Then each of the operations* EMPTY, INSERT, DELETE, *and* SPLIT *can be performed in $O(m^\varepsilon)$ amortized time.*

Remark. The above data structure can be extended to the case in which the endpoints of the query circular arc do not lie on the boundary of W. Details are omitted.

4.2 Querying with polynomial arcs

Next let Γ be the set of all arcs that are intersections with a fixed strip W of graphs of polynomials of degree at most d. We describe a dynamic data structure that reports all points of S lying above an arc in Γ. A similar data structure can be constructed for reporting points that lie below an arc.

We call an arc $\gamma \in \Gamma$ *k-shallow* if at most k points of S lie above γ. We call a simply connected cell with at most four edges a *pseudo-trapezoid* if its top and bottom edges are portions of arcs in Γ and its left and right edges are vertical segments. An *elementary partition* of S is a family $\Xi = \{(S_1, \triangle_1), \ldots, (S_u, \triangle_u)\}$, where $S_1, \ldots, S_u$ form a partition of S, $\triangle_i$ is a pseudo-trapezoid, and $S_i \subseteq \triangle_i$. The following lemma, whose proof is omitted, is obtained by extending the results of Matoušek [25] and of Agarwal and Matoušek [5].

LEMMA 4.2. *Let S and Γ be as defined above, and let r be a parameter. Then there exists an elementary partition $\Xi = \{(S_1, \triangle_1), \ldots, (S_u, \triangle_u)\}$ of S so that $m/r \le |S_i| \le 2m/r$, for each i, and any (m/r)-shallow arc of Γ crosses $O(\log m)$ pseudo trapezoids of Ξ. If r is a constant, then Ξ can be computed in $O(m)$ time.*

As in [25], using the above lemma, we can construct, in $O(m \log m)$ time, a partition tree of size $O(m)$ for answering REPORT (g, k) queries in time $O(m^\varepsilon + k)$. A point can be inserted or deleted in $O(\log^2 m)$ amortized time. Hence, we conclude the following.

THEOREM 4.2. *Let Γ and S be as above. Then each of the operations* EMPTY, INSERT, DELETE, *and* SPLIT *can be performed in $O(m^\varepsilon)$ amortized time.*

5 Incidences in Pseudo-line Arrangements

Let P be a set of m points and L a set of n pseudo-lines that are extensions of circular or polynomial arcs, and let $\mathcal{I}(P, L)$ denote the set of pairs $(p, \ell) \in P \times L$ such that p lies on ℓ. We wish to report $\mathcal{I}(P, L)$, compute $|\mathcal{I}(P, L)|$, or just determine whether $\mathcal{I}(P, L)$ is nonempty. For simplicity, we focus on the first subproblem. Corollary 3.1 implies that $\mathcal{I}(P, L)$ can be computed in $O((m^2 + n)m^\varepsilon)$ time. By partitioning P into $\lceil m/\sqrt{n} \rceil$ subsets $P_1, \ldots, P_s$, each of size at most $\sqrt{n}$, and computing $\mathcal{I}(P_i, L)$ for each subset separately, $\mathcal{I}(P, L)$ can be computed in $O(mn^{1/2+\varepsilon} + n^{1+\varepsilon})$ time, which is near optimal for $m \le \sqrt{n}$. We now describe an algorithm that is efficient for all values of m and n. For a parameter $r \le n$, a $(1/r)$-*cutting* of L is a decomposition of $\mathbb{R}^2$ into pseudo-trapezoids with disjoint interiors so that each pseudo-trapezoid crosses at most n/r pseudo-lines of L. Chazelle's algorithm [12] for computing a $(1/r)$-cutting of hyperplanes can be modified to compute a $(1/r)$-cutting of pseudo-lines of size $O(r^2)$ in $O(nr)$ time, under an appropriate model of computation.

We choose a parameter $r < n$ and construct a $(1/r)$-cutting Ξ of L of size $O(r^2)$. For a cell $\tau \in \Xi$, let $L_\tau \subseteq L$ be the set of pseudo-lines that intersect the interior of τ. We can compute the incidences between L and those points of P that lie at the vertices of Ξ in $O(nr)$ time. For a cell $\tau \in \Xi$, let $P_\tau \subseteq P$ be the set of points that either lie in the interior of τ or that lie on an edge of τ. Set $n_\tau = |L_\tau|$ and $m_\tau = |P_\tau|$. Then $\sum_\tau m_\tau \le 2m$ and $n_\tau \le n/r$. At most one pseudo-line ℓ_e of L can contain an edge e of Ξ. If there is such a pseudo-line, we report all incidences between e and the points that lie on e, over all edges e, in a total time of $O(r^2 + m)$. Finally, we compute $\mathcal{I}(P_\tau, L_\tau)$ in time $O(m_\tau n_\tau^{1/2+\varepsilon} + n_\tau^{1+\varepsilon})$ using the algorithm outlined above. Choosing the value of r appropriately, we obtain the following.

THEOREM 5.1. *The incidences between m points and n pseudo-lines that are extendions of circular or polynomial arcs of bounded degree can be detected, counted, or reported in time $O(m^{2/3-\varepsilon} n^{2/3+2\varepsilon} + m^{1+\varepsilon} + n^{1+\varepsilon})$.*

6 Many Faces in Pseudo-line Arrangements

For a set L of pseudo-lines as above and for a set P of points in the plane, none lying on any pseudo-line of L, let $\mathcal{F}(P, L)$ be the set of faces in $\mathcal{A}(L)$ that contain at least one point of P. We compute $\mathcal{F}(P, L)$ by following an approach similar to the one for computing $\mathcal{I}(P, L)$. We first describe an $O((m^2 + n)n^\varepsilon)$ algorithm for computing $\mathcal{F}(P, L)$: We compute the arrangement $\mathcal{A}(P^*)$ of pseudo-lines dual to P, and then compute its vertical decomposition $\mathcal{A}^{||}(P^*)$. For each face $\varphi \in \mathcal{A}^{||}(P^*)$, we compute the subset $L_\varphi \subseteq L$ of pseudo-lines whose dual points lie inside φ. This step takes $O((m^2 + n)n^\varepsilon)$ time. Next, we compute an Eulerian tour Π of the planar graph dual to $\mathcal{A}^{||}(P^*)$ so that each face of $\mathcal{A}^{||}(P^*)$ is visited $O(1)$ times; see [3]. Each node of Π corresponds to a face of $\mathcal{A}^{||}(P^*)$. If an edge e of Π crosses two adjacent faces of $\mathcal{A}(P^*)$, we set $\pi(e)$ to be the point of P whose dual pseudo-line separates these two faces. Otherwise, i.e., e connects two faces of $\mathcal{A}^{||}(P^*)$ separated by a vertical line, we set $\pi(e) = \emptyset$. Next, we construct a minimum-height binary tree T on Π. Each leaf of T is associated with a node of Π, and thus with a face of $\mathcal{A}^{||}(P^*)$, and each node v of T is associated with a subpath Π_v of Π. For each node v of T, we set $L_v = \bigcup_\varphi L_\varphi$, where the union is taken over all faces of $\mathcal{A}^{||}(P^*)$ associated with the nodes in Π_v. Similaly, we define $P_v \subseteq P$ to be the set of points associated with the edges in Π_v. Set $m_v = |P_v|$ and $n_v = |L_v|$. At any level of T, $\sum_v m_v = O(m^2)$ and $\sum_v n_v = O(n)$. By construction, any point in $P \setminus P_v$ lies either above all pseudo-lines in L_v or below all of them. We therefore add two points, one at $y = +\infty$ and another $y = -\infty$, to each P_v, and compute $\mathcal{F}(P_v, L_v)$ at each node v of T, in a bottom-up manner. Let κ_v be the complexity of $\mathcal{F}(P_v, L_v)$. For each leaf $w \in T$, we compute the lower and upper envelopes of L_w in $O(n_w \log n_w)$ time. For each internal node $v \in T$, with children w and z, we compute $\mathcal{F}(P_v, L_v)$ from $\mathcal{F}(P_w, L_w)$ and $\mathcal{F}(P_z, L_z)$ in $O((\kappa_v + \kappa_w + \kappa_z + m_v + n_v)\log n)$ time, using the "red-blue-merge" algorithm proposed by Edelsbrunner *et al.* [14]. It can be shown that the total time spent in computing $\mathcal{F}(P_v, L_v)$ at all nodes of T is $O((m^2 + n)\log^2 n)$. Next, we use $(1/r)$-cuttings to obtain an algorithm for computing $\mathcal{F}(P, L)$, which is efficient for all ranges of m and n, as in [1], and in the general spirit of the preceding section. Omitting further details, we conclude the following.

THEOREM 6.1. *Let L be a set of n pseudo-lines that are extensions of circular or polynomial arcs of bounded degree in the plane, and let P be a set of m points, none lying on any pseudo-line. One can compute $\mathcal{F}(P, L)$ in time $O(m^{2/3-\varepsilon} n^{2/3+2\varepsilon} + m^{1+\varepsilon} + n^{1+\varepsilon})$.*

Let C be a set of n congruent circles and P a set of points. We wish to compute $\mathcal{F}(P, C)$. We partition each circle in C into two semicircles by splitting it at its leftmost and rightmost points. Let U and L denote the sets of resulting upper and lower semicircles, respectively. Each pair of arcs within U (or L) intersects in at most one point. Although U and L do not conform to the framework described in the beginning of Section 2, we can nevertheless use Theorem 6.1 to compute $\mathcal{F}(P, L)$ and $\mathcal{F}(P, U)$ in time $O(m^{2/3-\varepsilon}n^{2/3+2\varepsilon} + m^{1+\varepsilon} + n^{1+\varepsilon})$. We can then compute $\mathcal{F}(P, C) = \mathcal{F}(P, U \cup L)$ from $\mathcal{F}(P, L)$ and $\mathcal{F}(P, U)$ in time $O(\kappa \log n)$ by using the red-blue-merge algorithm of [14], where κ is the total number of vertices in $\mathcal{F}(P, L), \mathcal{F}(P, U)$, and $\mathcal{F}(P, C)$. Hence we obtain the following.

THEOREM 6.2. *Let P be a set of m points and C a set of n congruent circles in the plane. We can compute $\mathcal{F}(P, C)$ in time $O(m^{2/3-\varepsilon}n^{2/3+2\varepsilon} + m^{1+\varepsilon} + n^{1+\varepsilon})$.*

7 Cutting Lenses

One of our main motivations for studying arrangements of pseudo-lines was the problem of computing incidences (and many faces) between points and circles. The recent analysis of Aronov and Sharir [9] shows that a collection of n circles can be cut into $O(n^{3/2+\varepsilon})$ arcs that are pseudo-segments, meaning that any pair of arcs intersect at most once. One can then apply known bounds for incidences between points and pseudo-segments, to obtain a bound that is roughly $O(m^{2/3}n^{2/3} + n^{3/2})$ on the number of incidences between m points and n circles. (This bound can then be further refined, for small values of m; see [9] for details.) Our goal is to make this combinatorial analysis constructive, so as to obtain a comparably-efficient algorithm for detecting, counting, or reporting these incidences. The first task that we face is to find, in time $O(n^{3/2+\varepsilon})$, a set of $O(n^{3/2+\varepsilon})$ points that cut the given circles into pseudo-segments. If two circles $\gamma, \gamma' \in C$ intersect, then the boundaries of the three bounded faces of $\mathcal{A}(\{\gamma, \gamma'\})$ are called *lenses*. Our goal is thus to cut the circles in C so that all lenses will be cut, i.e., a cut is made on at least one of the two edges of each lens.

The algorithm proceeds in two stages. In the first stage, we use standard range-searching techniques [4], to decompose the intersection graph of the circles in C into a union of complete bipartite subgraphs $\{A_i \times B_i\}_i$ so that the following condition holds (see also [9]).

$$\sum_i (|A_i| + |B_i|)^{3/2} = O(n^{3/2+\varepsilon}). \tag{7.1}$$

In the second stage, we cut circles in each bipartite subgraph independently. Let A be a set of "red" circles and B a set of "blue" circles, so that every red circle intersects every blue circle, and let $m = |A| + |B|$. We cut circles in A and B into circular arcs so that all *bichromatic* lenses, i.e., lenses formed by a red circle and a blue circle, are cut. We describe a recursive algorithm for making these cuts. At each step, we have a pseudo-trapezoid τ and two sets of circular arcs Γ and Γ' clipped to within τ. The arcs in Γ and Γ' lie on the circles in A and B, respectively. Initially, Γ (resp. Γ') is the set of upper and lower semicircles in A (resp. B), and τ is the entire plane. We omit the proof of the following lemma; see [23] for a similar result.

LEMMA 7.1. *If the endpoints of all arcs in Γ and Γ' lie on $\partial\tau$, then we can determine, in $O((|\Gamma| + |\Gamma'|)\log^3 m)$ time, whether Γ and Γ' induce at least one bichromatic lens that lies entirely in the interior of τ.*

If the endpoints of all arcs in $\Gamma \cup \Gamma'$ lie on $\partial\tau$ and Γ and Γ' do not form a bichromatic lens that is fully contained inside τ, then we stop. Otherwise (i.e., an endpoint lies inside τ, or there is a bichromatic lens lying inside τ), we choose a sufficiently large constant r, and compute a $(1/r)$-cutting Ξ of $\Gamma \cup \Gamma'$, of size $O(r^2)$ within τ. For every arc $\gamma \in \Gamma \cup \Gamma'$ and for every cell $\Delta \in \Xi$ that is crossed by γ, we cut γ at its intersection points with $\partial\Delta$. The total number cuts made is $O(mr) = O(m)$. After this step all lenses that lie in more than one cell of Ξ have been cut, so we recursively solve the problem within each cell Δ of Ξ, with the sets Γ_Δ and Γ'_Δ, which are the sets of arcs in Γ and Γ', respectively, clipped to within Δ, that intersect the interior of Δ.

It is clear that the algorithm cuts all bichromatic lenses inside τ. (Initially, τ is the whole plane.) In order to analyze the running time of the algorithm, we need the following observations. (See also [7] for a proof of a similar result.)

LEMMA 7.2. *Let λ be a lens in $\mathcal{A}(C)$. Then λ contains (in the closure of its interior) a lens λ' (possibly $\lambda' = \lambda$) such that any circle that crosses λ' intersects both of its arcs (either once or twice).*

We call a lens λ *elementary* if the lens that satisfies Lemma 7.2 is λ itself. Returning to the subproblem inside the trapezoid τ, if τ contains a lens in its interior, then it also contains an elementary lens. Let $m = |\Gamma| + |\Gamma'|$, let k be the number of endpoints of arcs in $\Gamma \cup \Gamma'$ that lie in the interior of τ plus the number of elementary bichromatic lenses that lie in the interior of τ, and let $T(m, k)$ denote the maximum running time of the above algorithm, for sets Γ, Γ' that have parameters m and k. Then the above observation gives the following recurrence for $T(m, k)$.

$$T(m, k) = \sum_{\Delta \in \Xi} T(m/r, k_\Delta) + O(m \log^3 m),$$

where k_Δ is the number of endpoints of $\Gamma \cup \Gamma'$ plus the number of elementary lenses that lie in the interior of the cell Δ of Ξ, so $\sum_\Delta k_\Delta \le k$. We also have $T(m, 0) = O(m \log^3 m)$. Hence, the same analysis as in [23, 29] implies that $T(m, k) = O(m^{1+\varepsilon}\sqrt{k})$. The following lemma is a re-statement of a recent result in [7].

LEMMA 7.3. *The number of elementary bichromatic lenses formed by* Γ *and* Γ' *is* $O(m)$.

Hence, $k = O(m)$, so the total time spent in cutting the bichromatic lenses formed by A and B is $O(m^{3/2+\varepsilon})$. Repeating this procedure to all bipartite graphs $A_i \times B_i$, and adding up the resulting complexity bounds using (7.1), we obtain the following:

THEOREM 7.1. *A collection of* n *circles can be cut into* $O(n^{3/2+\varepsilon})$ *pseudo-segments, in time* $O(n^{3/2+\varepsilon})$.

8 Circular Arrangements

Combining Theorem 7.1 with Theorem 5.1, we can conclude that the incidences between m points and n circles can be detected, counted or reported in time $O(m^{2/3-\varepsilon}n^{2/3+2\varepsilon} + m^{1+\varepsilon} + n^{3/2+\varepsilon})$. This bound is nearly worst-case optimal for m larger than roughly $n^{5/4}$. Aronov and Sharir [9] show how to improve such a bound for the number of incidences when m is smaller. The extra step that they use, constructing a dual partitioning for the set of circles, represented as points in $\mathbb{R}^3$, is in fact constructive. Putting it all together, and omitting any further details in this abstract, we obtain:

THEOREM 8.1. *The incidences between* m *points and* n *circles can be detected, counted or reported in time* $O(m^{2/3-\varepsilon}n^{2/3+2\varepsilon} + m^{6/11+3\varepsilon}n^{9/11-\varepsilon} + m^{1+\varepsilon} + n^{1+\varepsilon})$.

The following result on range searching can also be obtained by modifying our incidence algorithm.

THEOREM 8.2. *Given a set* C *of* n *circles and a set* P *of* m *points in the plane, we can count the number of points lying inside each circle in time* $O(m^{2/3-\varepsilon}n^{2/3+2\varepsilon} + m^{6/11+3\varepsilon}n^{9/11-\varepsilon} + m^{1+\varepsilon} + n^{1+\varepsilon})$.

References

[1] P. K. Agarwal, Partitioning arrangements of lines: II. Applications, *Discrete Comput. Geom.*, 5 (1990), 533–573.

[2] P. K. Agarwal, B. Aronov and M. Sharir, On the complexity of many faces in arrangements of pseudo-segments and of circles, to appear in *Proc. 42nd Annual Sympos. Foundations of Computer Science*, 2001.

[3] P. K. Agarwal, B. Aronov, M. Sharir, and S. Suri, Selecting distances in the plane, *Algorithmica*, 9 (1993), 495–514.

[4] P. K. Agarwal and J. Erickson, Geometric range searching and its relatives, in: *Advances in Discrete and Computational Geometry* (B. Chazelle, J. E. Goodman, and R. Pollack, eds.), *Contemporary Mathematics*, Vol. 223, American Mathematical Society, Providence, RI, 1999, pp. 1–56.

[5] P. K. Agarwal and J. Matoušek, On range searching with semialgebraic sets, *Discrete Comput. Geom.*, 11 (1994), 393–418.

[6] P. K. Agarwal and J. Matoušek, Dynamic half-space range reporting and its applications, *Algorithmica*, 13 (1995), 325–345.

[7] P. K. Agarwal, E. Nevo, J. Pach, R. Pinchasi, M. Sharir, and S. Smorodinsky, Lenses in arrangements of pseudo-circles and their applications, manuscript, 2001.

[8] P. K. Agarwal and M. Sharir, Arrangements and their applications, in: *Handbook of Computational Geometry* (J.-R. Sack and J. Urrutia, eds.), Elsevier Science Publishers B.V. North-Holland, Amsterdam, 2000, pp. 49–119.

[9] B. Aronov and M. Sharir, Cutting circles into pseudo-segments and improved bounds on incidences, *Discrete Comput. Geom.*, to appear.

[10] S. A. Burr, B. Grünbaum, and N. J. A. Sloane, The orchard problem, *Geom. Dedicata*, 2 (1974), 397–424.

[11] T.M. Chan, On levels in arrangements of curves, *Proc. 41st IEEE Symp. Found. Comput. Sci.*, 2000, 219–227.

[12] B. Chazelle, Cutting hyperplanes for divide-and-conquer, *Discrete Comput. Geom.*, 9 (1993), 145–158.

[13] H. Edelsbrunner, *Algorithms in Combinatorial Geometry*, Springer-Verlag, Heidelberg, 1987.

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

[15] J. E. Goodman, Proof of a conjecture of Burr, Grünbaum and Sloane, *Discrete Math.*, 32 (1980), 27–35.

[16] J. E. Goodman, Pseudoline arrangements, in *Handbook of Discrete and Computational Geometry* (J.E. Goodman and J. O'Rourke, eds.), CRC Press, Boca Raton, FL, 1987, pp. 83–109.

[17] J. E. Goodman and R. Pollack, On the combinatorial classification of nondegenerate configurations in the plane, *J. Combin. Theory Ser. A*, 29 (1980), 220–235.

[18] J. E. Goodman and R. Pollack, A theorem of ordered duality, *Geom. Dedicata*, 12 (1982), 63–74.

[19] J. E. Goodman and R. Pollack, Allowable sequences and order types in discrete and computational geometry, in: *New Trends in Discrete and Computational Geometry* (J. Pach, ed.), Springer-Verlag, 1993, pp. 103–134.

[20] B. Grünbaum, Arrangements of hyperplanes, *Congr. Numer.*, 3 (1971), 41–106.

[21] B. Grünbaum, *Arrangements and Spreads*, American Mathematical Society, Providence, RI, 1972.

[22] S. Har-Peled, Constructing cuttings in theory and practice, *SIAM J. Comput.* 29 (2000), 2016–2039.

[23] S. Har-Peled and M. Sharir, On-line point location in planar arrangements and its applications, *Discrete Comput. Geom.* 26 (2001), 19–40.

[24] F. Levi, Die Teilung der projektiven Ebene durch Gerade oder Pseudogerade, *Ber. Math.-Phys. Kl. sächs. Akad. Wiss. Leipzig*, 78 (1926), 256–267.

[25] J. Matoušek, Reporting points in halfspaces, *Comput. Geom. Theory Appl.*, 2 (1992), 169–186.

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

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

[28] M. Sharir and S. Smorodinsky, On generalized geometric graphs and pseudolines, manuscript, 2001.

[29] A. Solan, Cutting cycles of rods in space, *Proc. 14th Annual Sympos. Comput. Geom.*, 1998, 135–142.

[30] H. Tamaki and T. Tokuyama, How to cut pseudoparabolas into segments, *Discrete Comput. Geom.*, 19 (1998), 265–290.