Point Location in Zones of k-Flats in Arrangements*

Mark de Berg[†] Marc van Kreveld[†] Otfried Schwarzkopf[†] Jack Snoeyink[‡]

Abstract

Let $\mathcal{A}(H)$ be the arrangement of a set H of n hyperplanes in d-space. A k-flat is defined to be a k-dimensional affine subspace of d-space. The *zone* of a k-flat f with respect to H is the set of all faces in $\mathcal{A}(H)$ that intersect f. In this paper we study some problems on zones of k-flats. Our most important result is a data structure for point location in the zone of a k-flat. This structure uses $O(n^{\lfloor d/2 \rfloor + \varepsilon} + n^{k+\varepsilon})$ preprocessing time and space and has a query time of $O(\log^2 n)$. We also show how to test efficiently whether two flats are visible from each other with respect to a set of hyperplanes. Then point location in m faces in arrangements is studied. Our data structure for this problem has size $O(n^{\lfloor d/2 \rfloor + \varepsilon} \cdot m^{\lceil d/2 \rceil / d})$ and the query time is $O(\log^2 n)$.

Keywords. Arrangements, zones, k-flats, implicit point location, multidimensional parametric search.

1 Introduction

The subdivision of d-space into connected pieces—usually called faces—of various dimension, induced by a set H of hyperplanes, is called the *arrangement* $\mathcal{A}(H)$ of H. This concept was introduced to computational geometry by Edelsbrunner, O'Rourke and Seidel [11, 10], and they showed how to construct an arrangement optimally. The optimal construction time relies on the so-called *zone theorem*, a combinatorial bound on the maximum complexity of the zone of a hyperplane [12]. The *zone* of a hyperplane h is the subarrangement of all faces in $\mathcal{A}(H)$ that are intersected by h. (We consider the faces of the arrangement to be relatively open, that is they are topologically open within their affine hull. This means that the arrangement forms a partitioning of d-space into faces of various dimensions.)

See Figure ?? for an example in 2-space. The zone of the line h consists of all fat segments and vertices, together with the shaded faces. Zones are important in several contexts, as the efficiency of some algorithms depends on the complexity of zones. As mentioned, the bound on the complexity of the zone of a hyperplane guarantees optimal construction time of arrangements in d-space [11, 12]. A bound on the complexity of the vertical decomposition of the zone of a plane in 3-space is used to improve range searching in some cases [6]. As a third application, observe that the zone of a hyperplane h defines exactly the region that is visible from h, where the other hyperplanes are the obstacles. Therefore, zones are suitable for solving some visibility problems. Recently, upper bounds on the complexity of *zones of algebraic hypersurfaces* have also been shown [1]. These results have the following important application to lines in 3-space: To distinguish between the $O(n^4)$ isotopy classes of lines induced by n lines, one can consider the Plücker coordinates of lines on the Plücker hypersurface in 5-space. Since the total complexity of

*This research was supported by the ESPRIT Basic Research Action No. 3075 (project ALCOM). Research of the first author was also supported by the Netherlands' Organization for Scientific Research (N.W.O.).

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

[‡]Department of Computer Science, University of British Columbia, 201–2336 Main Mall, Vancouver, BC V6T 1Z4 Canada.

the zone of the Plücker hypersurface is $O(n^4 \log n)$, there is an $O(n^{4+\varepsilon})$ size data structure that solves the problem of distinguishing between isotopy classes of lines [4, 25].

The notion 'zone of a hyperplane' can be generalized to 'zone of a k-flat', where a k-flat is defined to be the intersection of $d-k$ hyperplanes with linearly independent normal vectors $(0 \leq k \leq d-1)$ [10]. Yet more generally, we define the zone of an arbitrary subset of d-space.

Definition 1 *The* zone *of a subset F of d-space with respect to H, denoted by $zone(H, F)$, is the subarrangement of all faces of $\mathcal{A}(H)$ that intersect F.*

Thus, for a point p not on any hyperplane of H, the $zone(H, p)$ is the convex polytope formed by the intersection of the open halfspaces that contain p and are bounded by the hyperplanes of H. In general, the zone of a point p is the face of $\mathcal{A}(H)$ that contains p. For a hyperplane h, the $zone(H, h)$ is the usual zone of a hyperplane [10, 11].

We concentrate on computational aspects of zones of k-flats rather than the combinatorial side. We obtain an efficient algorithm for point location in the zone of a k-flat. The data structure has preprocessing time and space $O(n^{\lfloor d/2 \rfloor+\varepsilon} + n^{k+\varepsilon})$ (for any $\varepsilon > 0$), where n is the size of the set H of hyperplanes. Notice that the first term of the size of the structure is close to the complexity of *one* single cell in the arrangement, and the second term is close to the *number* of faces in the k-flat itself. For almost all k this is considerably less than the complexity of the zone itself! With this structure it is possible to determine in $O(\log^2 n)$ time if a query point lies in the zone and, if so, in which face of the zone it lies. The problems of point location in a full arrangement and in a convex polytope have been studied before [7, 8]. The preprocessing time and space of these structures is $O(n^{d+\varepsilon})$ and $O(n^{\lfloor d/2 \rfloor+\varepsilon})$, respectively. The query time is $O(\log n)$. (Both these structures can be improved, removing the ε-term in the exponent [5, 19].)

We also investigate the following problem: Given a k_1-flat f_1, a k_2-flat f_2, and a set H of n hyperplanes in d-space, determine whether f_1 and f_2 can see each other with respect to H. In other words, determine whether there are points $p_1 \in f_1$ and $p_2 \in f_2$, such that the segment $p_1 p_2$ does not cross any hyperplane in H. We obtain efficient algorithms for this problem by reducing the dimension of the problem, together with multidimensional parametric search and the structure for point location in the zone. The precise bounds of the algorithm are given in Corollary 1.

Both our point location and visibility algorithms are based on sampling. Lately this technique has been used to solve a variety of problems [4, 6, 7, 8, 13, 27]. The rough idea of random sampling is as follows. Let H be a set of n hyperplanes in d-space. Choose a random subset R of H of size r. Consider the arrangement $\mathcal{A}(R)$, and triangulate it (that is, partition its faces into simplices). The triangulated arrangement has complexity $O(r^d)$, and with some constant positive probability, any simplex is intersected by only $O(n \log r / r)$ hyperplanes of H. This property can be used to define subproblems of the original problem recursively. For a more detailed treatment of random sampling and its applications we refer to the recent book by Mulmuley [23]. The main feature of our algorithms lies in the fact that we do not triangulate $\mathcal{A}(R)$ in d-space, but in the k-flat. This is the main reason for the efficiency of the algorithms.

Then we consider the problem of point location in m faces in arrangements. Our solution is similar to that of point location in the zone, but it requires somewhat more preprocessing time and space. The structure has size $O(n^{\lfloor d/2 \rfloor+\varepsilon} \cdot m^{\lceil d/2 \rceil / d})$, which is slightly smaller than the complexity of m faces for certain values of n, m and d. For instance, when d is odd and $m = n$, then the complexity of the faces can be $\Omega(n^{\frac{d+1}{2}})$ and the size of our structure is $O(n^{\frac{d+1/d}{2}+\varepsilon})$. The query time is $O(\log^2 n)$.

2 Basic properties

In this section we give an alternative definition of the zone of a k-flat, which will be used in the next section. We also state an upper bound and a lower bound on the maximum size of the zone.

We say that two points p and q are *visible* with respect to a hyperplane h if h intersects the (closed) line segment pq in a point. We call p and q visible with respect to a a set H of hyperplanes if they are visible with respect to every $h \in H$.

Lemma 1 *Two points p and q lie in the same face of a hyperplane arrangement $\mathcal{A}(H)$ if and only if they are visible with respect to H.*

The easy proof is left to the reader. The lemma immediately implies the following characterization of the zone of a set F.

Lemma 2 *A point q lies in the zone of a set F with respect to H if and only if there is a point p in F such that p and q are visible with respect to H.*

Such a point p is called a *witness* for q. The lemma shows that zones are useful for visibility problems, where the obstacles are hyperplanes.

We simply state some known bounds on the complexity of the zone of a k-flat.

Lemma 3 ([15], see also [26]) *Let $z_d^k(n)$ be the maximum complexity of the zone of a k-flat f with respect to a set H of n hyperplanes in d-space. We have: $z_d^k(n) = \Theta(n^{\lfloor (d+k)/2 \rfloor})$ if $d+k$ is even, and $z_d^k(n) = \Omega(n^{\lfloor (d+k)/2 \rfloor})$, $z_d^k(n) = O(n^{\lfloor (d+k)/2 \rfloor} \log n)$ if $d+k$ is odd.*

To actually construct the zone of a k-flat, one could construct the full arrangement $\mathcal{A}(H)$ and then remove the faces that are not in the zone. There are alternative ways to compute the zone of a k-flat that are more efficient [2]. In the next section, however, we show how to perform point location in the zone without actually constructing it.

3 Point location in the zone

For a set H of n hyperplanes and a set F that lies in a k-flat f in d-space, the problem of point location in the zone of F is defined as follows. Preprocess H and F, such that for any given query point q, one can determine efficiently whether q lies in $zone(H, F)$ and, if so, in which face of the zone q lies.

We solve this problem using sampling. The algorithm returns a witness if the query point lies in the zone. This witness is such that it uniquely determines the face of the zone which contains this query point. Rather than constructing the whole zone, we construct a tree that uses considerably less preprocessing time and space. Let H be a set of hyperplanes and let f be a fixed k-flat in d-space. Then $\overline{H}$ denotes the set $\{\overline{h} \mid \overline{h} = h \cap f \text{ where } h \in H\}$, and $\mathcal{A}(\overline{H})$ is the arrangement in f formed by $\overline{H}$.

Pick a sample $\overline{R}$ of size r, where $\overline{R} \subset \overline{H}$, and r is a sufficiently large constant. We triangulate the arrangement $\mathcal{A}(\overline{R})$ into a simplicial complex, for instance with a bottom-vertex triangulation. The triangulated arrangement $\mathcal{A}(\overline{R})$ consists of $O(r^k)$ relatively open simplices of dimensions 0 to k.

With positive probability, each simplex intersects $O(n \log r/r)$ $(k-1)$-flats of $\overline{H}$, and, thus, hyperplanes of H. If this is not the case, we pick a new sample.[1] Each simplex s defines two subsets H_s and H'_s of H. The subset H_s contains the hyperplanes of H that intersect s without containing it, and H'_s contains the remaining hyperplanes (the ones that fully contain s or avoid it completely). Notice that $R \subseteq H'_s$ for every simplex s. Let c_s be that face of $\mathcal{A}(H'_s)$ that contains s.

An illustration of these definitions is given in Figure ??, where the upper part of the boundary of the face c_s in 3-space is shown. The planes h_1 and h_2 are in H'_s, and they contribute to c_s. The plane h_3 is in H_s, because it intersects the interior of the shaded triangle s.

[1]Randomization can be removed using techniques of Matoušek [16]; therefore, we do not elaborate on picking and testing samples.

Lemma 4 *Let F be a subset of the k-flat f. Then we have the following properties.*

(i) If a point q lies in $zone(H, F)$ then there is a simplex s in the triangulated arrangement $\mathcal{A}(\overline{R})$ such that $q \in c_s$.

(ii) For all simplices s in the triangulated arrangement $\mathcal{A}(\overline{R})$ and any point $q \in c_s$, we have: $q \in zone(H, F)$ if and only if $q \in zone(H_s, F \cap c_s)$.

Proof: *(i)* Suppose that q lies in $zone(H, F)$. By Lemma 2 there is a witness $p \in F$ such that p and q are visible with respect to H. Let s be the simplex in the triangulated arrangement $\mathcal{A}(\overline{R})$ that contains p. Since p and q are visible with respect to $H'_s \subseteq H$, q lies in the face c_s of $\mathcal{A}(H'_s)$.
(ii) Let s be a simplex in the triangulated arrangement $\mathcal{A}(\overline{R})$, such that $q \in c_s$.
$\Rightarrow$: Suppose that $q \in zone(H, F)$. Then there is a witness $p \in F$ such that p and q are visible with respect to H. As above, this implies that p and q lie in the same face of $\mathcal{A}(H'_s)$, which is c_s. Thus $p \in F \cap c_s$ and $q \in zone(H_s, F \cap c_s)$.
$\Leftarrow$: Suppose that q lies in $zone(H_s, F \cap c_s)$. Then there is a witness $p \in F \cap c_s$, such that p and q are visible with respect to H_s. Because $p, q \in c_s$ and all hyperplanes in H'_s do not intersect c_s, it follows that p and q are visible with respect to $H = H_s \cup H'_s$. Thus, p witnesses that $q \in zone(H, F)$. □

Lemma 4 gives the recursive property on which the structure for point location in the zone of a set F lying in a k-flat f is based. The structure for $zone(H, F)$ is a tree $\mathcal{T}$ of degree $O(r^k)$. Let δ be the root of $\mathcal{T}$. The root δ has one child γ_s for every simplex s in the triangulated arrangement $\mathcal{A}(\overline{R})$. With γ_s we store the simplex s and an associated structure for deciding whether a point lies in c_s. The child γ_s is the root of a recursively defined tree for $zone(H_s, F)$. If the number of hyperplanes in H is smaller than some constant, we do not take a sample, but we store the full arrangement $\mathcal{A}(H)$.

This recursive definition does not completely correspond to the recursive property of Lemma 4 (ii): the subtree rooted at γ_s is defined for $zone(H_s, F)$ instead of for $zone(H_s, F \cap c_s)$. To remedy this, we will fill in witnesses at some faces in the arrangements stored in the leaves in such a way that any query point finds a witness if and only if it lies in the zone. To this end, impose an arbitrary order on the children of each node. A query with a point q should continue in the *first* child γ_s for which the query point lies in the polytope c_s. To finish the preprocessing, consider the arrangement $\mathcal{A}(\overline{H})$, and take one point p in each face that intersects F. Locate p in the arrangement of the leaf where the search ends, and store p as a witness with the face of this arrangement that contains p. It takes too much preprocessing time to search with p for all children for which $p \in c_s$. But we should make sure that any query point q that lies in the same face of $\mathcal{A}(H)$ as point p does, follows the same path down the tree.

A query with a point q is performed as follows. Start at the root δ. Find the first child γ_s (with respect to the chosen order on children) for which q lies in c_s; this is determined by searching in the associated structure of each child of δ. Continue the search recursively at this child. If q does not lie in c_s for any child, then q does not lie in the zone. If the current node is a leaf, then we locate q in the associated arrangement. If there is a witness stored at the face, then q lies in the zone and we return this witness. Otherwise, q does not lie in the zone. Note that it is not important what ordering is chosen at any node during the preprocessing. The filling in of witnesses is correct, because a query point that lies in the zone will follow the same path down $\mathcal{T}$ as its witness. A query point that does not lie in the zone will, at some point, take a different path down $\mathcal{T}$ than any witness.

Theorem 1 *For any $\varepsilon > 0$, a set H of n hyperplanes and a set F of constant complexity lying in a k-flat f in d-space can be preprocessed in $O(n^{\lfloor d/2 \rfloor + \varepsilon} + n^{k+\varepsilon})$ time and space, such that point location queries in the zone can be performed in $O(\log^2 n)$ time.*

Proof: Testing whether a query point lies in a convex face determined by the intersection of n halfspaces in d-space can be performed in $O(\log n)$ time, after $O(n^{\lfloor d/2 \rfloor + \varepsilon})$ preprocessing time and space (for any $\varepsilon > 0$) [7]. Therefore, the initial preprocessing $S(n)$ of our structure satisfies the following recurrence:

$$S(n) = O(r^k) \cdot S(n \log r/r) + O(r^k) \cdot O(n^{\lfloor d/2 \rfloor + \varepsilon})$$

This solves to $S(n) = O(n^{\lfloor d/2 \rfloor + \varepsilon} + n^{k+\varepsilon})$ preprocessing for any $\varepsilon > 0$, if r is a large enough constant. Additionally, at most $O(n^k)$ queries, each taking $O(\log^2 n)$ time (see below) are required to fill in the witnesses in $\mathcal{T}$.

The query time $Q(n)$ satisfies the following recurrence:

$$Q(n) = Q(n \log r/r) + O(r^k) \cdot O(\log n)$$

which solves to $Q(n) = O(\log^2 n)$ time. $\square$

It should be mentioned that this structure includes as special cases the point location in an arrangement (for $k = d$) and the point location in the intersection of halfspaces (for $k = 0$), for which the best known solutions need space about $O(n^d)$ and $O(n^{\lfloor d/2 \rfloor})$, respectively.

4 Visibility among k-flats

Since the zone of a k-flat f defines the region from which f is visible (without looking through a hyperplane), it is natural to use zones to solve visibility problems in arrangements of hyperplanes. In particular, we consider the problem: given a k_1-flat f_1 and a k_2-flat f_2, with $k_1 \leq k_2$, can f_1 and f_2 see each other?

Definition 2 *Flats f_1 and f_2 are (mutually)* visible *with respect to H if and only if there are points $q_1 \in f_1$ and $q_2 \in f_2$ such that q_1 and q_2 are visible with respect to H. The points q_1 and q_2 are called* witnesses.

Before we consider solutions that use our query structure, we note some useful facts about affine spaces. Recall that a k-flat is an affine subspace of dimension k, the affine hull of $k + 1$ affinely independent points. The *join* of flats f_1 and f_2 is the affine space of smallest dimension that contains both flats. The join of a k_1-flat and a k_2-flat has dimension at most $k_1 + k_2 + 1$; it is the affine hull of the $k_1 + k_2 + 2$ points spanning f_1 and f_2. Since any segment q_1q_2 connecting a point $q_1 \in f_1$ with a point $q_2 \in f_2$ lies in the join of f_1 and f_2, it is clearly sufficient to work in this join space. This property can also be used when the visibility problem for subsets of the k-flats is considered.

In the following, however, we will consider the visibility problem only for entire flats. In that case it turns out that one can always reduce the dimension of the problem to the case where $d = k_1 + k_2 + 1$.

Lemma 5 *Given an instance of the visibility problem with a k_1-flat and a k_2-flat in d-space. If $k_1 + k_2 + 1 \neq d$ then one can find in linear time an equivalent instance of the visibility problem with smaller dimension.*

Proof: If $d > k_1 + k_2 + 1$, then we can reduce the problem to the join of f_1 and f_2.

If $d < k_1 + k_2 + 1$, then the join of the k_1-flat f_1 and the k_2-flat f_2 has dimension strictly less than $k_1 + k_2 + 1$; either the flats intersect or the points defining them are not affinely independent. We can test intersection in constant time.

If f_1 and f_2 do not intersect, we find a direction v that is contained in both flats (by Gaussian elimination) in constant time. Let's call v the vertical direction. We can partition the set of hyperplanes into the vertical hyperplanes H_v, those containing a line parallel to v, and the non-vertical hyperplanes $H_{non-v} = H \setminus H_v$. Notice that every non-vertical hyperplane intersects every vertical line.

In linear time, we can project the flats f_1 and f_2 and the vertical hyperplanes H_v in the direction v to form an instance of the visibility problem with a $(k_1 - 1)$-flat f_1', a $(k_2 - 1)$-flat f_2' and a set H_v' of hyperplanes in $(d-1)$-space. We can show that this reduced problem is equivalent to the original: If f_1' and f_2' can see each other, then there are witness points q_1' and q_2' for the reduced problem. The vertical lines that project to q_1' and q_2' are contained in f_1 and f_2; any two points on these lines can see each other with respect to H_v. Furthermore, above some height *top*, both lines are above all hyperplanes of H_{non-v}. Choose two points, q_1 and q_2, above *top* as witnesses that these lines see each other with respect to H_{non-v} (see Figure ??). Thus the flats f_1 and f_2 see each other with respect to H. We can determine this height *top* in linear time. On the other hand, if the flats f_1' and f_2' cannot see each other, then f_1 and f_2 cannot see each other with respect to $H_v \subseteq H$. ◻

We present two methods for solving the visibility problem. These methods should be used after reducing the dimension with the above lemma, if possible. In the following, we always assume $k_1 \leq k_2$, to avoid cluttering up the notation. As a warm-up, we first show how to use linear programming to determine visibility in $O(n^{k_1+1})$ time using $O(n)$ space. We then show how multidimensional parametric search can be employed to make our point location structure give a solution with $O(n^{k_1+\varepsilon})$ space and time for $k_2 \leq 2k_1+1$. For $k_2 > 2k_1+1$, we still obtain a slightly faster solution than with linear programming.

First the linear programming solution.

Theorem 2 *Let H be a set of n hyperplanes in d-space, let f_1 be a k_1-flat and let f_2 be a k_2-flat. One can decide in $O(n^{k_1+1})$ time and $O(n)$ space whether f_1 and f_2 see each other with respect to H.*

Proof: The hyperplanes H partition the k_1-flat f_1 into $O(n^{k_1})$ faces. We pick a witness point in every such face, and then have to test whether this candidate witness point q_1 lies in the zone of f_2.

To do that, we orient the hyperplanes $h \in H$ such that the intersection of the resulting halfspaces defines the face of $\mathcal{A}(H)$ containing q_1. We then intersect every such halfspace with f_2 to obtain a set of k_2-dimensional halfspaces, and it remains to test whether their intersection is empty. That is equivalent to solving a linear program in n inequalities and k_2 variables, and can be done in $O(n)$ time using the deterministic technique of Megiddo [21, 22] or the randomized algorithm by Seidel [27].

The $O(n^{k_1})$ witness points can be determined by computing the k_1-dimensional arrangement formed by H in f_1, using $O(n^{k_1})$ storage. A different approach that uses only linear space is to look at every k_1-tuple of hyperplanes in H, and to use a symbolic perturbation scheme to deal with degeneracies in the arrangement. ◻

To improve upon the time bound of this simple solution, we use a variation of Megiddo's parametric search [20] called multidimensional parametric search [9, 18, 24]. We follow the description of Matoušek and Schwarzkopf [18]. Let S be a set of halfspaces, and let A be an algorithm that determines whether a query point q lies in the polytope P (not necessarily full-dimensional), which is the intersection of the halfspaces of S. We also assume that the only kind of operation A applies to q is to test whether q is contained in some halfspace (not necessarily from S), and, if $q \notin P$, that A returns a witness halfspace $h^+ \in S$ such that $q \notin h^+$. We can say that q is *obscured* by the hyperplane h since it keeps q from seeing the relative interior of P.

We sketch the proof of the following theorem.

Theorem 3 *([18]) Given an algorithm A as above, which runs in $q(n) = O(\log^{O(1)} n)$ time, then A can be used to test whether a k-flat f intersects the polytope P in time $O(\log^{O(1)} n)$, or to find a set of $k+1$ hyperplanes, such that any point $q \in f$ is obscured by at least one of the hyperplanes.*

Proof: Notice first that Helly's theorem proves the existence of an obscuring $(k+1)$-tuple of hyperplanes if f does not intersect P [10, 14].

We prove the Theorem 3 by induction on k. For $k = 0$, the problem is solved by algorithm A, which, by assumption, runs in polylogarithmic time.

So assume that we have a solution for $(k-1)$-flats, and we are now given a k-flat f. We employ parametric search and run A *generically*, that is, with an undetermined query point q^* (supposed to lie in f). The algorithm A goes through $O(\log^{O(1)} n)$ steps, and in some of these the generic point q^* is tested against a halfspace h^+. We answer questions about the position of q^* as follows: Given a question "is $q \in h^+$?", consider the $(k-1)$-flat $f \cap h$ and query with it (possible by the induction hypothesis). Either the query will find a point $p \in f \cap h$ in P, which solves the problem for f as well, or it will return an obscuring k-tuple T. Let R denote the region of points *not* obscured by T. Since R is the intersection of k halfspaces, it is convex. Furthermore, R does not intersect $f \cap h$ and hence, $f \cap R$ can lie on one side of h only. So only one of the halfspaces $f \cap h^+$ or $f \cap h^-$ remains as a potential location for q^*.

When algorithm A finally stops, it has compared q^* with $O(\log^{O(1)} n)$ hyperplanes, so we know that q^* lies in the intersection of $O(\log^{O(1)} n)$ halfspaces. There are two possibilities. Either the intersection of these halfspaces is non-empty, implying that the answer (of algorithm A) for all points in its interior must be the same. If it is YES, we have found a point q in $f \cap P$; otherwise, we take the witness halfspace returned by algorithm A and the at most $O(\log^{O(1)} n)$ obscuring k-tuples returned by the tests done before, and observe that these at most $k \cdot O(\log^{O(1)} n) + 1$ halfspaces obscure f, and thus contain an obscuring $(k+1)$-tuple for f as well.

If, however, the intersection of the $O(\log^{O(1)} n)$ halfspaces is empty, this implies that the tests done have been answered inconsistently, which is only possible if no point q^* exists. Thus, it follows that f does not intersect P. Again, we observe that the $O(\log^{O(1)} n)$ obscuring k-tuples computed previously contain an obscuring $(k+1)$-tuple for f, which can be found in polylogarithmic time (by testing all $(k+1)$-tuples, for instance).

Our time analysis is quite crude: We only observe that we have done $q(n)$ tests taking polylogarithmic time (by the induction hypothesis), and that we have some polylogarithmic overhead in the end. ◻

We will now apply this to the visibility problem for the flats f_1 and f_2. Assume that $1 \leq k_1 \leq k_2$. As in the linear programming solution, we compute the arrangement of $\overline{H}$ in f_1, and consider each of its $O(n^{k_1})$ faces in turn. For each of these faces we (implicitly) consider the face P of $\mathcal{A}(H)$ containing it, and determine whether the flat f_2 intersects it. By Theorem 3 this can be done in polylogarithmic time once we have an algorithm A which tests whether a query point lies in P. But we can use our point location structure to do that: since P is a face of the zone of f_1, and the structure returns a witness if the query point q lies in the zone, we can decide whether q lies in P. Hence, we can decide whether the flat f_2 intersects P.

To determine whether the flats f_1 and f_2 can see each other, we compute the point location structure for the flat f_1 once in the beginning. Then we go through the list of all $O(n^{k_1})$ witness points of flat f_1. For every such witness point p we want to determine whether it can see flat f_2. To do so, we trace p through our search structure, marking the path in the tree we take. Since any query point lying in the same face as p must follow exactly the same path, we have a sequence of $O(\log n)$ polytope containment queries to test whether a query point q lies in the same face as p. This gives us our algorithm A, with query time $q(n) = O(\log^2 n)$. The only operation applied to q is to test it against a hyperplane, and it is also easy to make sure that when q does not lie in the face of p, we can return a witness hyperplane (Clarkson's algorithm for polytope containment does that, and we just have to keep track of it). We now apply Theorem 3 to test in polylogarithmic time whether f_2 intersects p's face.

We need one more observation. The preprocessing of the point location structure can be reduced, since it is known that all query points will lie in the flat f_2. Therefore, we only need an associated structure for point location in a k_2-dimensional polytope. With this improvement, the preprocessing time is $O(n^{k_1+\varepsilon} + n^{\lfloor k_2/2 \rfloor+\varepsilon})$. Hence, we obtain:

Theorem 4 *Given a k_1-flat and a k_2-flat (where $1 \leq k_1 \leq k_2$) and a set of n hyperplanes, it can be tested in time $O(n^{k_1+\varepsilon} + n^{\lfloor k_2/2 \rfloor+\varepsilon})$ whether the flats can see each other with respect to the hyperplanes. Within the same time bounds we can report all distinct visibilities.*

For $k_2 \leq 2k_1 + 1$, this is $O(n^{k_1+\varepsilon})$. For $k_2 > 2k_1 + 1$, we can still do slightly better than the linear programming solution. The idea is to replace Clarkson's polytope containment structure by Matoušek's structure [17], using storage $M = \Theta(n^{k_1+1-\frac{k_1+1}{\lfloor k_2/2 \rfloor+1}})$, and query time $O(n^{1+\varepsilon}/M^{1/\lfloor k_2/2 \rfloor}) = O(n^{1-\frac{k_1+1}{\lfloor k_2/2 \rfloor+1}+\varepsilon})$. We then need a refined version of Theorem 3 (see [18] for details) to conclude that we can again employ multidimensional parametric search. The refinement consists primarily of the observation that Matoušek's structure is a partition tree of low depth and can be evaluated in a parallel fashion, which makes the requirement that $q(n)$ be polylogarithmic in Theorem 3 unnecessary. To summarize the results of this section, we state:

Corollary 1 *For any fixed $\varepsilon > 0$, one can check if a k_1-flat and a k_2-flat can see each other with respect to a set of n hyperplanes in d dimensions in time*

$$\begin{array}{ll} O(n^{k_1+\varepsilon}) & \text{if } k_2 \leq 2k_1 + 1, \\ O(n^{k_1+1-\frac{k_1+1}{\lfloor k_2/2 \rfloor+1}+\varepsilon}) & \text{otherwise.} \end{array}$$

We notice that $\Omega(n^{k_1})$ is a lower bound for reporting a witness pair for every face of $\mathcal{A}(H)$ that is intersected by both f_1 and f_2, since we can choose two k_1-dimensional flats f_1, f_2 such that there are that many witness pairs. Finally, the results generalize when we consider subsets of constant complexity in flats rather than whole flats.

5 Point location in several faces

A variant of the data structure of Section 3 can be used for point location in several faces in arrangements of hyperplanes. Let H be a set of n hyperplanes, and let m faces of the arrangement $\mathcal{A}(H)$ be given by a set P of m points in these faces. We show how to store these m faces for efficient point location using $O(n^{\lfloor d/2 \rfloor+\varepsilon} \cdot m^{\lceil d/2 \rceil/d})$ storage.

Note that it is known that the maximum number of *facets* of m (d-dimensional) cells in an arrangement of n hyperplanes is $\Omega(mn)$ if $m = O(n^{d-2})$, and $\Omega(n^{d-1})$ if $m = \Omega(n^{d-2})$, and $\Omega(m^{2/3}n^{d/3})$ if $m = \Omega(n^{d-3/2})$ [10]. The total complexity can be higher. In 5-space, for instance, n cells can have $\Omega(n^3)$ vertices whereas they can have only $O(n^2)$ facets. Our structure has size $O(n^{13/5+\varepsilon})$ in this case.

Take a random sample R of H of size r. Triangulate $\mathcal{A}(R)$, giving $O(r^d)$ simplices. For each simplex s of the triangulated arrangement, let $H_s \subseteq H$ be the set of hyperplanes that intersect s without containing it fully, and let $H'_s = H \setminus H_s$. Let c_s be the face of the arrangement $\mathcal{A}(H'_s)$ that contains s. Place an arbitrary but fixed order on the simplices. Locate the m points of P in the polytopes c_s, and assign each point to the first simplex s for which it lies in c_s. Let $s_1, \ldots, s_j$ be the simplices that are assigned at least one point of P, and let $P_{s_1}, \ldots, P_{s_j}$ be the subsets of P assigned to these simplices. We can prove similar to Lemma 4:

Lemma 6 *Let $p \in P$ be a point that is assigned to s_i. For any point q in d-space, we have that p and q lie in the same face of $\mathcal{A}(H)$ if and only if $q \in c_{s_i}$ and p and q lie in the same face of $\mathcal{A}(H_{s_i})$.*

We define a data structure $\mathcal{T}$ for point location in the faces of $\mathcal{A}(H)$ defined by P. Let δ be the root of $\mathcal{T}$. For each simplex s_i with $1 \leq i \leq j$, let γ_{s_i} be a child node of δ. Associate with γ_{s_i} a point location structure for the convex polytope c_{s_i} [7]. Let the subtree rooted at γ_{s_i} be a recursively defined tree for point location in the faces of $\mathcal{A}(H_{s_i})$ defined by P_{s_i}. When m, the number of faces at a node γ, drops below some constant chosen sufficiently large, then we let γ be a leaf, and associate m point location structures with it for the m convex polytopes that contain the m points. We omit further details and the query algorithm; they are the same as in Section 3.

Theorem 5 *For any $\varepsilon > 0$, a set H of n hyperplanes and a set P of m points can be preprocessed in $O(n^{\lfloor d/2 \rfloor + \varepsilon} \cdot m^{\lceil d/2 \rceil / d})$ time and space, such that point location queries in the faces of $\mathcal{A}(H)$ defined by P can be performed in $O(\log^2 n)$ time.*

Proof: Let m_{s_i} denote the size of the set P_{s_i}. Then $\sum_i m_i = m$, and the size and preprocessing of the structure described above satisfies:

$$\begin{aligned}
S(n,m) &= O(n^{\lfloor d/2 \rfloor + \varepsilon}) \text{ if } m = O(1), \\
S(n,m) &= O(m) \text{ if } n = O(1), \\
S(n,m) &= \sum_{i=1}^{j} S(n \log r/r,\, m_i) + j \cdot O(n^{\lfloor d/2 \rfloor + \varepsilon}) \text{ otherwise,}
\end{aligned}$$

where $j = \min(m,\, c \cdot r^d)$ for some constant c. For any $\varepsilon > 0$, we prove by induction that $S(n,m) \leq b \cdot n^{\lfloor d/2 \rfloor + \varepsilon} \cdot m^{\lceil d/2 \rceil / d}$ for some constant b.

$$\begin{aligned}
S(n,m) &\leq \sum_{i=1}^{j} b \cdot (n \log r/r)^{\lfloor d/2 \rfloor + \varepsilon} \cdot m_i^{\lceil d/2 \rceil / d} + j \cdot a \cdot n^{\lfloor d/2 \rfloor + \varepsilon} \\
&\leq b \cdot (n \log r/r)^{\lfloor d/2 \rfloor + \varepsilon} \cdot \sum_{i=1}^{j} m_i^{\lceil d/2 \rceil / d} + c \cdot r^d \cdot a \cdot n^{\lfloor d/2 \rfloor + \varepsilon} \\
&\leq b \cdot n^{\lfloor d/2 \rfloor + \varepsilon} \cdot j \cdot (m/j)^{\lceil d/2 \rceil / d} \cdot (\log r/r)^{\lfloor d/2 \rfloor + \varepsilon} + c \cdot r^d \cdot a \cdot n^{\lfloor d/2 \rfloor + \varepsilon} \\
&\leq b \cdot n^{\lfloor d/2 \rfloor + \varepsilon} \cdot m^{\lceil d/2 \rceil / d} \cdot j^{1 - \lceil d/2 \rceil / d} \cdot (\log r/r)^{\lfloor d/2 \rfloor + \varepsilon} + c \cdot r^d \cdot a \cdot n^{\lfloor d/2 \rfloor + \varepsilon}
\end{aligned}$$

We must now prove that $j^{1 - \lceil d/2 \rceil / d} \cdot (\log r/r)^{\lfloor d/2 \rfloor + \varepsilon}$ is strictly less than 1. If m is large enough with respect to r, then the induction hypothesis follows. We proceed as follows:

$$\begin{aligned}
j^{1 - \lceil d/2 \rceil / d} \cdot (\log r/r)^{\lfloor d/2 \rfloor + \varepsilon} &\leq (c \cdot r^d)^{1 - \lceil d/2 \rceil / d} \cdot (\log r/r)^{\lfloor d/2 \rfloor + \varepsilon} \\
&\leq c^{1 - \lceil d/2 \rceil / d} \cdot r^{-\varepsilon} \log^{\lfloor d/2 \rfloor + \varepsilon} r \\
&< 1
\end{aligned}$$

if r is a large enough constant. $\square$

6 Conclusions

In this paper we considered the zone of a k-flat in an arrangement of hyperplanes in d-space, and studied two algorithmic aspects. First, we presented a structure for $O(\log^2 n)$ time point location in the zone, which in many cases uses less space than the zone itself. Furthermore, an efficient algorithm was given to determine whether two flats are visible from each other with respect to a set of hyperplanes. Thirdly, we presented a data structure for point location in m faces. All our results can be made deterministic rather than randomized by using cuttings rather than random samples [16].

An interesting generalization of our result would be to consider the zone of surfaces or the zone in arrangements of hyperspheres or curves. Our algorithms extend to the zone of a surface in a hyperplane arrangement, if a random sample of the hyperplanes in the surface can be decomposed into a small number of faces of constant description. Such decompositions are studied by Chazelle et al. [3], who show that roughly r^{2k-3} faces are sufficient for a random sample of size r in a k-dimensional surface.

References

[1] B. Aronov, M. Pellegrini, and M. Sharir. On the zone of a surface in a hyperplane arrangement. *Discrete Comput. Geom.*, 9(2):177–186, 1993.

[2] M. de Berg, K. Dobrindt, and O. Schwarzkopf. On lazy randomized incremental construction. *Discrete Comput. Geom.*, to appear.

[3] B. Chazelle, H. Edelsbrunner, L. J. Guibas, and M. Sharir. A singly-exponential stratification scheme for real semi-algebraic varieties and its applications. In *Proc. 16th Int. Coll. on Automata, Languages and Programming* (1989), Lect. Notes in Comp. Science 372, pp. 179-193.

[4] B. Chazelle, H. Edelsbrunner, L. J. Guibas, and M. Sharir. Lines in space: combinatorics, algorithms, and applications. In *Proc. 21st Annu. ACM Sympos. Theory Comput.* (1989), pp. 382–393.

[5] B. Chazelle and J. Friedman. Point location among hyperplanes and unidirectional ray-shooting. *Comput. Geom. Theory Appl.*, to appear.

[6] B. Chazelle, M. Sharir, and E. Welzl. Quasi-optimal upper bounds for simplex range searching and new zone theorems. *Algorithmica*, 8:407–429, 1992.

[7] K. L. Clarkson. A randomized algorithm for closest-point queries. *SIAM J. Comput.*, 17:830–847, 1988.

[8] K. L. Clarkson and P. W. Shor. Applications of random sampling in computational geometry II. *Discrete Comput. Geom.*, 4:387-422, 1989.

[9] R. Cole, M. Sharir, and C. Yap. On k-hulls and related problems. *SIAM J. Comput.*, 16:61-67, 1987.

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

[11] H. Edelsbrunner, J. O'Rourke, and R. Seidel, Constructing arrangements of lines and hyperplanes with applications. *SIAM J. Comput.*, 15:341-363, 1986.

[12] H. Edelsbrunner, R. Seidel, and M. Sharir. On the zone theorem for hyperplane arrangements. *SIAM J. Comput.*, 22:418–429, 1993.

[13] D. Haussler and E. Welzl. ε-Nets and simplex range queries. *Discrete Comput. Geom.*, 2:127-151, 1987.

[14] E. Helly. Über Mengen konvexer Körper mit gemeinschaftlichen Punkten. *Jahresber. Deutsch. Math.-Verein.*, 32:175-176, 1923.

[15] M. E. Houle and T. Tokuyama. *On zones of flats in hyperplane arrangements.* Manuscript, 1991.

[16] J. Matoušek. Cutting hyperplane arrangements. *Discrete Comput. Geom.*, 6:385–406, 1991.

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

[18] J. Matoušek and O. Schwarzkopf. Linear optimization queries. In *Proc. 8th ACM Symp. on Comp. Geom.* (1992), pp. 16-25.

[19] J. Matoušek and O. Schwarzkopf. On ray shooting in convex polytopes. *Discrete Comput. Geom.*, 10(2):215–232, 1993.

[20] N. Megiddo. Applying parallel computation algorithms in the design of serial algorithms. *J. ACM*, 30:852-865, 1983.

[21] N. Megiddo. Linear-time algorithms for linear programming in $\mathbf{R}^3$ and related problems. *SIAM J. Comput.*, 12:759-776, 1983.

[22] N. Megiddo. Linear programming in linear time when the dimension is fixed. *J. ACM*, 31:114-127, 1984.

[23] K. Mulmuley. *Computational Geometry: An Introduction Through Randomized Algorithms.* Prentice Hall, New York, 1993.

[24] C. H. Norton, S. A. Plotkin, and E. Tardos. Using separation algorithms in fixed dimension. In *Proc. 1st ACM-SIAM Symp. on Discr. Algorithms* (1990), pp. 377-378.

[25] M. Pellegrini. Ray shooting and isotopy classes of lines in 3-space. In *Proc. 2nd Workshop on Algorithms and Data Structures* (1991), Lect. Notes in Comp. Science 519, Springer-Verlag, pp. 20-31.

[26] M. Pellegrini. On the zone of a co-dimension p surface in a hyperplane arrangement. In *Proc. 3rd Canad. Conf. Comput. Geom.* (1991), pp. 233–238.

[27] R. Seidel. Small-dimensional linear programming and convex hulls made easy. *Discrete Comput. Geom.*, 6:423–434, 1991.