Point Location in Zones of k-Flats in Arrangements*

(extended abstract)

Mark de Berg Marc van Kreveld Jack Snoeyink†

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 [6] (see also [5]). They showed how to construct an arrangement optimally, and proved the so-called *zone theorem*, a combinatorial bound on the maximal complexity of the zone of a hyperplane. The *zone* of a hyperplane h consists of all faces of cells in $\mathcal{A}(H)$ that are supported by h. See Figure 1 for an example in 2-space. The zone of the line h consists of all bold-face segments and vertices, together with the shaded cells. Zones are important in several contexts, as the efficiency of some algorithms depends on the size of zones. In [6], the bound on the complexity of the zone of a hyperplane guarantees optimal construction time of arrangements in d-space. In [2], 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. 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.

We generalize the notion 'zone of a hyperplane' 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 ($0 \leq k \leq d-1$), see [5]. The *zone of a k-flat f* with respect to H, denoted by $zone(H, f)$, is the subarrangement of $\mathcal{A}(H)$ formed by the closure of all cells of $\mathcal{A}(H)$ that are intersected by f. Thus for a point p, the $zone(H, p)$ is the convex polytope formed by the intersection of the closed halfspaces that contain p and are bounded by the hyperplanes of H. For a hyperplane h, the $zone(H, h)$ is the zone of a hyperplane as defined in [5, 6].

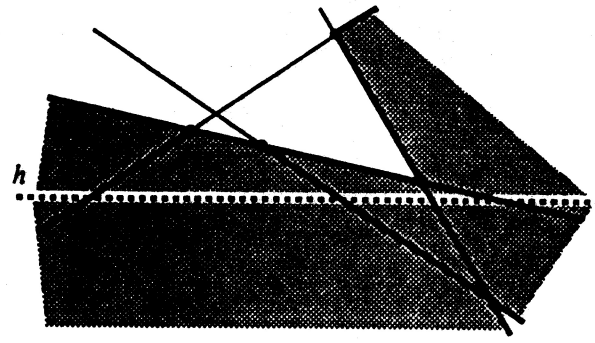

Figure 1: The zone of a line h in the plane.

The $zone(H, P)$ for a k-polytope P in a k-flat is defined in an analogous way.

We concentrate on algorithmic 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 $O(n^{\lfloor d/2 \rfloor + \epsilon} + n^{k+\epsilon})$ (for any $\epsilon > 0$), where n is the size of the set H of hyperplanes. Notice that the first term of the preprocessing is close to the size of one single cell in the arrangement, and the second term is close to the number of cells in the k-flat itself. For most k this is considerably less than the size of the zone itself!

*This research was supported by the ESPRIT Basic Research Action No. 3075 (project ALCOM). Research of the first author was also supported by the Dutch Organization for Scientific Research (N.W.O.). Authors address: Department of Computer Science, Utrecht University, P.O. Box 80.089, 3508 TB Utrecht, the Netherlands.

†On leave from the Department of Computer Science of the University of British Columbia.

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 cell of the zone it lies. Point location in a full arrangement or in a convex polytope has been studied before in [3, 4]. Preprocessing of the structures take $O(n^{d+\epsilon})$ and $O(n^{\lfloor d/2 \rfloor + \epsilon})$ time and space, respectively. The query time is $O(\log n)$.

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_1p_2 does not properly intersect any hyperplane in H. We obtain an efficient algorithm for this problem using two techniques of reducing the dimension of the problem, together with linear programming and the structure for point location in the zone. The precise bounds of the algorithm are given in Corollary 1.

In this extended abstract we omit almost all proofs, and also a number of details of the algorithms. These can be found in the full version [1].

2 Point location in the zone

We start this section with two easy properties on zones of k-flats. Then we give structure for point location in the zone.

Lemma 1 *A point q lies in the zone of a k-flat f with respect to H if and only if there is a point p in f such that the line segment pq does not properly intersect any hyperplane of H.*

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

Lemma 2 *For the maximum complexity of the zone of a k-flat f with respect to a set H of n hyperplanes in d-space, $z_d^k(n)$, we have: $z_d^k(n) = \Omega(n^{\lfloor (d+k)/2 \rfloor})$ and $z_d^k(n) = O(\min(n^{\lfloor d/2 \rfloor + k}, n^{d-1}))$.*

For a set H of n hyperplanes and a k-flat f in d-space, the problem of point location in the zone 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 cell of the zone q lies. If q lies on the boundary of one or more cells in the zone, then one of these cells should be found.

We solve these problems using sampling (see e.g. [2, 3, 4, 7, 8] for other results on sampling). The algorithm returns a witness if the query point lies in the zone. The witness will be such that it uniquely determines a cell of the zone that contains the 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 k-flat in d-space. Then $\overline{H}$ denotes the set $\{\overline{h} \mid \overline{h} = h \cap f$ where $h \in H\}$, and $\mathcal{A}(\overline{H})$ is the arrangement in f formed by $\overline{H}$.
Construct a sample $\overline{R}$ of size r, where $\overline{R} \subset \overline{H}$, and r is a sufficiently large constant. We triangulate the arrangement $\mathcal{A}(\overline{R})$, for instance with a bottom-vertex triangulation. The triangulated arrangement $\mathcal{A}(\overline{R})$ consists of $O(r^k)$ simplices, and $\overline{R}$ can be constructed deterministically such that each simplex is intersected by $O(n/r)$ $(k-1)$-flats of $\overline{H}$, and, thus, hyperplanes of H [8]. Each simplex s partitions H into two subsets H_s and H'_s: the subset H_s contains the hyperplanes of H that properly intersect s, and H'_s contains the remaining hyperplanes. Notice that $R \subseteq H'_s$. Let c_s be that cell of $\mathcal{A}(H'_s)$ that contains the interior of the simplex s. In Figure 2, the upper part of the cell 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 shaded triangle s properly.

Lemma 3

(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 lies in c_s.*

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

Lemma 3 gives the recursive property on which the structure for point location in the zone of f

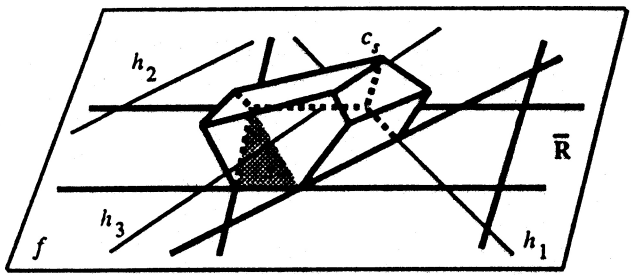

Figure 2: Situation for a 2-flat in 3-space: the polytope c_s for the shaded triangle.

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 3 (ii): the subtree rooted at γ_s is defined for $zone(H_s, f)$ instead of for $zone(H_s, f \cap c_s)$. This is remedied by filling in witnesses at certain cells 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 for each cell, take one point p in its interior. Locate p in the arrangement of the leaf where the search ends, and store p as a witness with the cell of this arrangement that contains p. (It may take too much preprocessing time to search with p for all children for which $p \in c_s$.)

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 p 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 a cell in the associated arrangement. If there is a witness stored at the cell, then q lies in the zone and we return this witness. Otherwise, q does not lie in the zone. 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 $\epsilon > 0$, a set H of n hyperplanes and a k-flat f (or polytope P in a k-flat) in d-space can be preprocessed in $O(n^{\lfloor d/2 \rfloor + \epsilon} + n^{k+\epsilon})$ 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 cell determined by the intersection of n halfspaces in d-space can be performed in $O(\log n)$ time, after $O(n^{\lfloor d/2 \rfloor + \epsilon})$ preprocessing time and space (for any $\epsilon > 0$), see [3]. Therefore, the initial preprocessing $S(n)$ of our structure satisfies the following recurrence:

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

This solves to $S(n) = O(n^{\lfloor d/2 \rfloor + \epsilon} + n^{k+\epsilon})$ preprocessing for any $\epsilon > 0$, if r is a large enough constant. Additionally, $O(n^k)$ 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:

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

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

3 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 1 *Flats f_1 and f_2 are* visible *(for each other) with respect to H if and only if there are points $q_1 \in f_1$ and $q_2 \in f_2$ such that the segment q_1q_2 does not properly intersect any hyperplane of 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 space of dimension k, which is the translation of a linear subspace spanned by k linearly independent vectors. 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 formed by taking the linear subspace spanned by the vectors defining flats f_1 and f_2, together with a vector from a point in f_1 to a point in f_2. This set of vectors is translated to contain a point in f_1. If k_1 and k_2 are small compared to the dimension d, then one can solve the visibility problem in the join of f_1 and f_2. On the other hand, if the summed dimension of the two flats is greater than $d-1$, then they must contain a parallel vector. It is possible to eliminate this vector in linear time, thus creating an instance of the visibility problem for a (k_1-1)-flat, a (k_2-1)-flat and a set of hyperplanes in $(d-1)$-space. In the full version we prove:

Lemma 4 *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 fewer dimensions.*

We present three methods for solving the visibility problem. These methods should be used after reducing the dimension with the above lemma, if possible. We first show how to use linear programming to determine visibility in $O(n^{k_1+1})$ time using $O(n^{k_1})$ space. When $k_1 = k_2$ our point location structure gives a better solution with $O(n^{k_1+\epsilon})$ space and time. When $k_1 = k_2 - 1$, we obtain a solution — by one more use of sampling — that is somewhat more efficient.

First the linear programming solution. In d-space, a point $p = (x_1, x_2, \ldots, x_d)$ is contained in a hyperplane h if $a_1x_1 + \cdots + a_dx_d = b_h$, where $(a_1, \ldots, a_d)^T$ is the normal vector of h. By definition, a given k-flat is the intersection of $d-k$ hyperplanes with linearly independent normal vectors. Thus, a point p in a k-flat satisfies the matrix equation $M \cdot p = b$, where the rows of M are the normal vectors of the hyperplanes defining the k-flat, and b is the column vector with the corresponding b_h's. We proceed as follows. Construct the arrangement $\mathcal{A}(\overline{H})$ in the k_1-flat f_1. For a given cell, take a candidate witness point q_1 from its interior. For each hyperplane $h \in H$, choose the sign of the hyperplane equation so that the halfspace $\{p \mid (a_1, \ldots, a_d) \cdot p \geq b_h\}$ contains q_1; expressed as an $n \times d$ matrix inequality, $M_H \cdot q_1 \geq b_H$.

We now wish to find a witness q_2 in the k_2-flat f_2 that lies in the intersection of these halfspaces. Thus, we wish to find a feasible solution to $M_H \cdot p \geq b_H$ and the matrix equation defining f_2, which is $M_{f_2} \cdot p = b_{f_2}$. This can be tested in $O(n)$ time by solving a linear program [9]. Hence, testing visibility of a k_1-flat and a k_2-flat can be done by solving $O(n^{k_1})$ linear programs.

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 $O(n^{k_1})$ space whether f_1 and f_2 see each other with respect to H.*

When both flats have the same dimension, our query structure gives a better solution than the linear programming approach. We choose a set of $O(n^k)$ candidate witnesses in the one flat (one in each cell of $\mathcal{A}(\overline{H})$), and query with them in the point location structure for the zone in the other flat.

Theorem 3 *Let H be a set of n hyperplanes, and f_1 and f_2 two k-flats in d-space. For any $\epsilon > 0$, one can decide in $O(n^{k+\epsilon})$ time whether the two k-flats can see each other with respect to H.*

In the full version[1] we also give an algorithm that is better than the previous ones when $k_1 = k_2 - 1$.

It makes use of sampling and our point location structure for the zone. We just state the result here:

Theorem 4 *Let H be a set of n hyperplanes, f_1 a k_1-flat and f_2 a k_2-flat in d-space, where $1 \leq k_1 \leq k_2$. For any $\epsilon > 0$, one can decide whether f_1 and f_2 can see each other with respect to H in time $O(n^{k_1/2+k_2/2+\epsilon})$.*

To summarize the results of this section, we state:

Corollary 1 *For any fixed $\epsilon > 0$, 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*

$$\begin{array}{ll} O(n^{k_1+\epsilon}) & \text{if } k_1 = k_2 \\ O(n^{k_1+1/2+\epsilon}) & \text{if } k_1 = k_2 - 1 \\ O(n^{k_1+1}) & \text{always.} \end{array}$$

4 Conclusions and open problems

In this paper we introduced the notion of a zone of a k-flat in an arrangement of hyperplanes in d-space and studied two algorithmic aspects. Firstly, 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. Secondly, an efficient algorithm was given to determine whether two flats are visible for each other with respect to a set of hyperplanes. All algorithms also work for portions of k-flats, such as k-polytopes.

Some open algorithmic problems that remain are the optimal construction of the zone itself, and the improvement of our algorithms. One can also think of a generalization to arrangements of hyperspheres or curves.

The main open combinatorial problem is to find sharp upper and lower bounds on the complexity of the generalized notion of the zone.

Acknowledgements

The authors thank Otfried Schwarzkopf for helpful discussions.

References

[1] de Berg, M., M. van Kreveld, and J. Snoeyink, *Point Location in Zones of k-Flats in Arrangements*, Techn. Rep., Dept. of Comp. Science, Utrecht University, 1991.

[2] Chazelle, B., M. Sharir, and E. Welzl, Quasi-Optimal Upper Bounds for Simplex Range Searching and New Zone Theorems, *Proc. 6th ACM Symp. on Comp. Geom.* (1990), pp. 23-33.

[3] Clarkson, K.L., New Applications of Random Sampling in Computational Geometry, *Discr. & Comp. Geom.* **2** (1987), pp. 195-222.

[4] Clarkson, K.L., and P.W. Shor, Applications of Random Sampling in Computational Geometry II, *Discr. & Comp. Geom.* **4** (1989), pp. 387-422.

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

[6] Edelsbrunner, H., J. O'Rourke, and R. Seidel, Constructing Arrangements of Lines and Hyperplanes with Applications, *SIAM J. Comput.* **15** (1986), pp. 341-363.

[7] Haussler, D., and E. Welzl, ϵ-Nets and Simplex Range Queries, *Discr. & Comp. Geom.* **2** (1987), pp. 127-151.

[8] Matoušek, J., Approximations and Optimal Geometric Divide-and-Conquer, *manuscript*, 1990.

[9] Meggido, N., Linear Programming in Linear Time when the Dimension is Fixed, *J. ACM* **31** (1984), pp. 114-127.