SERIE B — INFORMATIK

On vertical ray shooting in arrangements

Jiří Matoušek*[†]

B 92–06
February 1992

Abstract

We consider the following problem: Given a collection H of n hyperplanes in E^d, preprocess it so that given a query point x, a hyperplane of H lying immediately above x can be detected quickly. We give a relatively simple solution with $O(n^d/\log^{d-1} n)$ space and deterministic preprocessing time and $O(\log n)$ query time. This gives a slightly more efficient and considerably simplified alternative of a previous solution due to Chazelle and Friedman.

*Department of Applied Mathematics, Charles University, Malostranské nám. 25, 118 00 Praha 1, Czechoslovakia and Institut für Informatik, Freie Universität Berlin, Arnimallee 2-6, W-1000 Berlin 33, Germany.
[†]Supported by Humboldt Research Fellowship

1 Introduction

Let H be a collection of n hyperplanes in E^d, where the dimension d is fixed (and we imagine it is a small number). A problem frequently encountered in computational geometry is the *point location* in the arrangement of H: construct a data structure so that, given a query point $x \in \mathrm{E}^d$, the face of the arrangement of H containing x can be determined quickly. Clarkson [Cla87] gave a solution requiring $O(n^{d+\delta})$ space[1] and expected preprocessing time and $O(\log n)$ query time. Chazelle and Friedman [CF92] described a solution achieving $O(n^d)$ space and $O(\log n)$ query time. Then Chazelle [Cha91] found another data structure with the same space/query time performance, but improving the preprocessing time to $O(n^d)$ and using a new and much simpler and cleaner method.

The data structure of Chazelle and Friedman [CF92] also enables to solve the *vertical ray shooting* problem, i.e. to find a hyperplane of H lying immediately above the query point (in the x_d-coordinate direction) within the same query time. Schwarzkopf [Sch92] simplified their solution of this problem significantly. In this note we show how to modify Chazelle's new solution of the point location problem also for the vertical ray shooting. In our modification of Chazelle's method, we use a variant of zone theorem due to Aronov et al. [AMS91]. We also give a somewhat different exposition of Chazelle's method.

For the above point location problem, one usually implicitly assumes that a full representation of the arrangement is stored in the data structure; then, of course, the $O(n^d)$ bound for space is optimal. However, there are various special versions and modifications of the problem, where no such trivial lower bound argument is applicable. The vertical ray shooting is one of such examples, and, indeed, the space required by Chazelle's and Friedman's data structure [CF92] can be reduced to $O(n^d/(\log n)^{\lceil d/2\rceil-\delta})$ if we only want to answer vertical ray shooting queries (in Schwarzkopf's simplified version, the space is $O(n^d/\log^{d-\delta}))$. In our solution, the space requirement is $O(n^d/(\log n)^{d-1})$, and a a further small reduction is possible with a much more complicated data structure, which we do not describe here. No lower bound for this problem is known, but we conjecture that $O(n^d/\log^d n)$ space is asymptotically optimal for the $O(\log n)$ query time.

One often considers a more general ray shooting problem, with rays of arbitrary directions (i.e. the query is specified as a semiline, and we are interested in its first intersection with a hyperplane of H). We do not see any way of extending our solution of the vertical ray shooting problem to the general ray shooting. General methods are known for turning a point location algorithm into a ray shooting algorithm (see [AM91], also implicitly in [dBHO$^+$92]); when applied to Chazelle's point location algorithm, these yield $O(n^d)$ space with $O(\log^2 n)$ query time. It would be interesting to get an algorithm for ray shooting in arrangements with $O(\log n)$ query time and $O(n^d)$ or smaller space. Finally let us remark that one can achieve a continuous tradeoff between storage and query time for ray shooting, with storage varying between $O(n)$ and $O(n^d)$, see [AM91].

Throughout the paper, we will assume that the considered collection H of hyperplanes is in general position (their arrangement is simple), and that no face of the arrangement is vertical. This assumption can be removed either by appealing to a general perturbation argument (*simulation of simplicity*, see [Ede87]), or by a more careful (and more complicated) direct analysis.

[1]Throughout this paper, δ denotes an arbitrarily small positive constant. The multiplicative constants in the asymptotic bounds may depend on δ.

2 Vertical decompositions

Let τ a *simplicial prism* in E^d, that is a vertical prism whose bases are $(d-1)$-dimensional simplices. Let $B(\tau)$ denote the set of the (at most $d+2$) hyperplanes bounding τ. Let H be a collection of hyperplanes, and let c be a cell of the arrangement of $H \cup B(\tau)$ contained in τ. For each nonvertical facet f of c, we consider the "vertical wall" of f inside c, i.e. the set

$$w(f) = c \cap \{(x_1, \ldots, x_d) \in \mathrm{E}^d;\ (x_1, \ldots, x_{d-1}, t) \in f \text{ for some t}\}\,.$$

Let $V_0(c)$ denote the set of all nonempty vertical prisms of the form $w(f) \cap w(f')$, f, f' nonvertical facets of c. It is easily seen that these prisms form a subdivision of c, and that each base of such a prism is contained in a single facet of c. For each prism of $V_0(c)$, let us choose a triangulation of its base into simplices, using a number of simplices proportional to the combinatorial complexity of the base. This is possible e.g., using so-called canonical (or bottom-vertex) triangulation of the base, see e.g., [Cla88]. This defines a subdivision of the prism into simplicial prisms. The collection of all simplicial prisms thus arising for all prisms of $V_0(c)$ will be denoted by $V(c)$ and called the *vertical decomposition*[2] of c.

The following lemma seems to belong to the folklore:

Lemma 2.1 *For every c, $|V(c)| = O(v(c)^2)$, where $v(c)$ denotes the number of vertices of the cell c.*

Proof: By our general position assumptions, c is "almost" a simple polytope, meaning that the number of hyperplanes incident to each vertex is bounded by a constant. The complexity of each prism $w(f) \cap w(f') \in V_0(c)$ is at most proportional to $v(f)v(f')$, thus the total complexity of $V_0(c)$ is at most proportional to $\left(\sum_f v(f)\right)^2$, where the sum is over all facets f of c. But since each vertex of c is incident to a bounded number of hyperplanes, it is $\sum_f v(f) = O(v(c))$. Since the number of simplicial prisms in $V(c)$ is proportional to the total complexity of the prisms of $V_0(c)$, the claim follows. □

For τ, H as above, let us call the union of the vertical decompositions $V(c)$ for all cells c of the arrangement of $H \cup B(\tau)$ contained in τ the *vertical decomposition of τ according to H*.

Let us say that a collection H of hyperplanes is *sparse* for a simplicial prism τ if no vertex of the arrangement of H is contained in the interior of τ. The combinatorial result needed for our subsequent algorithm is the following:

Lemma 2.2 *Let τ be a simplicial prism in E^d, let H be a collection of n hyperplanes which is sparse for τ. Then the number of simplicial prisms in the vertical decomposition of τ according to H is $O(n^{d-1}\log^{d-2} n)$.*

Proof: Let $\mathcal{C}$ be the collection of cells of the arrangement of $H \cup B(\tau)$ contained in τ. By Lemma 2.1, it suffices to bound the sum $\sum_{c\in\mathcal{C}} v(c)^2$. Since each vertex of a cell $c \in \mathcal{C}$ belongs to a hyperplane of $B(\tau)$, this sum is upper bounded by

$$\sum_{b\in B(\tau)} \sum_{c\in \mathrm{zone}(b, H\cup B(\tau))} v(c)v(c\cap b)$$

[2]Let us remark that sometimes one defines a vertical decomposition otherwise, in such a way that also the bases of the prisms are (recursively) vertically decomposed. For this kind of decomposition, no satisfactory bounds for its complexity are known.

where zone(h, H) denotes the collection of all cells incident with a hyperplane h in the arrangement of H. By so-called Extended zone theorem [AMS91], the inner sum is bounded by $O(n^{d-1}\log^{d-2} n)$ for each b, and since there are at most $d+2 = O(1)$ hyperplanes in $B(\tau)$, the claim follows. $\square$

3 The data structure

Let K be a constant, n_0 a parameter (both to be determined later). Let τ be a simplicial prism, let H be a collection of n hyperplanes, each intersecting the interior of τ. Let N be the number of vertices of the arrangement of H in the interior of τ. We say that

- τ is *poor* if $N \leq 2n^d/K$, and
- τ is *rich* if $N \geq n^d/K$.

(Note that τ can be both rich and poor by our definition; this is to allow for approximate estimates of the quantity N in the algorithm.)

The following lemma summarizes the part of results of Chazelle [Cha91] we will use essentially as a black box in our development.

Lemma 3.1 *Let τ be a simplicial prism and H collection of n hyperplanes, each intersecting the interior of τ, $n \geq n_0$. Let r be a prescribed constant, $K = K(r)$ a large enough constant. There is a deterministic algorithm with $O(n)$ running time, which (correctly) claims τ poor or rich (if τ is both poor and rich, it is free to choose either outcome), and it computes a set $S \subseteq H$ with $|S| \leq Cr\log r$ for an absolute constant C, and such that the interior of each simplicial prism in the vertical decomposition of τ according to S is intersected by no more than n/r hyperplanes of H. Moreover, if τ was declared poor, then S is sparse for τ.* $\square$

Let us remark that Chazelle's results are proved for simplices instead of simplicial prisms, but the reader familiar with [Cha91] may check that this makes no real difference in the proof. Also, Chazelle gives an upper bound on the number of vertices of the arrangement of S inside τ, expressed in terms of K, r, n; out requirement of no such vertices follows from his bound for large enough K.

We are ready to describe the data structure for the vertical ray shooting in the arrangement of H. It will be a rooted tree $\mathcal{T}$, whose each node v stores a simplicial prism τ_v. The bases of each τ_v are contained in certain hyperplanes of H, and such hyperplanes are stored together with τ_v. If v is a leaf of $\mathcal{T}$, it also stores the list of hyperplanes of H intersecting the interior of the simplicial prism τ_v.

We describe a recursive algorithm for building the tree $\mathcal{T}$. The algorithm accepts a simplicial prism τ and the collection H_τ of the hyperplanes of H intersecting the interior of τ. The algorithm is first called with the whole space E^d standing for τ (and thus $H_\tau = H$). With a current τ and H_τ, it proceeds as follows: It creates a node v and stores τ as τ_v in it. Let us denote $|H_{\tau_v}|$ by n_v. If $n_v < n_0$, v becomes a leaf. Otherwise the algorithm finds $S \subseteq H_{\tau_v}$ as in Lemma 3.1 (with H_{τ_v} standing for H in that Lemma) and computes the vertical decomposition $\mathcal{D}_v$ of τ_v according to S. For each simplicial prism τ' in $\mathcal{D}_v$, it computes the collection $H_{\tau'}$, recursively calls itself on $\tau', H_{\tau'}$ and attaches the resulting tree as one of the subtrees of the node v. This finishes the description of the algorithm.

Lemma 3.1 guarantees that for a child w of a node v,

$$n_w \leq n_v / r\,, \tag{1}$$

and so for (say) $r \geq 2$ the tree $\mathcal{T}$ has depth $O(\log n)$. The number of children of each node is bounded by a constant. Hence, given a query point x, we can find, in $O(\log n)$ time, a leaf node v such that $x \in \tau_v$. Then the hyperplane of H lying immediately above x must be either the one defining the top base of τ_v, or among the hyperplanes intersecting the interior of τ_v. Thus it can be detected in $O(n_0)$ additional time. If we choose $n_0 = \log n$, our data structure can answer vertical ray shooting queries in $O(\log n)$ time (here n stands for the cardinality of the original collection of hyperplanes).

It remains to bound the space and preprocessing time required by our data structure. The time spent for the preprocessing in a node v is, by Lemma 3.1, proportional to n_v (not counting the recursive calls of the algorithm for building the subtrees of v). The storage needed for v is clearly also $O(n_v)$. Since each node v has $O(1)$ sons and since $n_w \leq n_v$ for every son w of v, it suffices to bound the sum $\Sigma(\mathcal{T})$ of n_v over all inner nodes v of the tree.

Let R_i denote the collection of the rich inner nodes v of $\mathcal{T}$ with $r^{i-1}n_0 \leq n_v < r^i n_0$ ($i = 1, 2, \ldots$). By (1), the simplicial prisms corresponding to nodes from the same R_i have disjoint interiors. Since the arrangement of H has fewer than n^d vertices and each simplicial prism stored in a node of R_i contains at least $n_0^d r^{(i-1)d}/K$ vertices, we obtain the estimate

$$|R_i| = O\left(\frac{n^d}{n_0^d\, r^{di}}\right)\,. \tag{2}$$

We will now assign poor nodes to rich nodes, as follows. For a rich node v, let $P_j(v)$ be the set of the poor inner nodes w in the subtree rooted at v, which are j levels below v in the tree and such that there is no other rich node on the path from v to w. We also set $P_0(v) = \{v\}$. Since the root of $\mathcal{T}$ is a rich node, each poor node belongs to exactly one $P_j(v)$.

Let $v \in R_i$. For $w \in P_j(v)$, we have

$$n_w \leq n_v / r^j \leq r^{i-j} n_0\,,$$

thus, in particular, $P_j(v) = \emptyset$ for $j > i$.

Let $f(n) = O(n^{d-1} \log^{d-2} n)$ be the bound from Lemma 2.2, and let $g(r) = f(Cr \log r)$, where C is as in the bound on the size of S in Lemma 3.1, so that $g(r)$ is an upper bound on the number of children of a poor node in the tree. We have $g(n) = o(n^d)$, so we may pick the value of r in the algorithm so large that $g(r) \leq r^d/2$. Since the node v has $O(1)$ sons, we get $|P_j(v)| = O(r^{jd}/2^j)$. Hence

$$\sum_{j=0}^{i} \sum_{w \in P_j(v)} n_w \leq \sum_{j=0}^{i} |P_j(v)| r^{i-j} n_0 = O\left(\sum_{j=0}^{i} \frac{r^{i+(d-1)j}}{2^j} n_0\right) = O\left(\frac{r^{di}}{2^i} n_0\right)\,.$$

Using (2), we obtain

$$\Sigma(\mathcal{T}) \leq \sum_i |R_i| O\left(\frac{r^{di}}{2^i} n_0\right) = O\left(\sum_{i=1}^{\infty} \frac{n^d}{n_0^{d-1} 2^i}\right) = O(n^d / n_0^{d-1}) = O(n^d / \log^{d-1} n)\,.$$

We have proved the following theorem:

Theorem 3.2 *The vertical ray shooting problem in an arrangement of n hyperplanes can be solved with $O(n^d/\log^d n)$ space and deterministic preprocessing time and $O(\log n)$ query time.*

References

[AM91] P. K. Agarwal and J. Matoušek. Ray shooting and parametric search. In *Proc. 23. ACM symposium on Theory of Computing*, 1992. To appear. Also Tech. Report CS-1991-22, Duke University, 1991.

[AMS91] B. Aronov, J. Matoušek, and M. Sharir. On the sum of squares of cell complexities in hyperplane arrangements. In *Proc. 7. ACM Symposium on Computational Geometry*, pages 307–313, 1991.

[CF92] B. Chazelle and J. Friedman. Point location among hyperplanes and vertical ray shooting. *Computational Geometry: Theory and Applications*, 1992. To appear.

[Cha91] B. Chazelle. Cutting hyperplanes for divide-and-conquer. Tech. report CS-TR-335-91, Princeton University, 1991. Preliminary version: *Proc. 32. IEEE Symposium on Foundations of Computer Science*, October 1991.

[Cla87] K. L. Clarkson. New applications of random sampling in computational geometry. *Discrete & Computational Geometry*, 2:195–222, 1987.

[Cla88] K. L. Clarkson. A randomized algorithm for closest-point queries. *SIAM Journal on Computing*, 17:830–847, 1988.

[dBHO$^+$92] M. de Berg, D. Halperin, M. Overmars, J. Snoeyink, and M. van Kreveld. Efficient ray shooting and hidden surface removal. *Algorithmica*, 1992. To appear. Extended abstract: Proc. 7. ACM Symposium on Computational Geometry, 1991.

[Ede87] H. Edelsbrunner. *Algorithms in Combinatorial Geometry.* Springer-Verlag, 1987.

[Sch92] O. Schwarzkopf. Lecture at Freie Universität Berlin, January 1992.