DERANDOMIZING AN OUTPUT-SENSITIVE CONVEX HULL ALGORITHM IN THREE DIMENSIONS

Bernard Chazelle
Jiri Matousek

CS-TR-361-92

February 1992

Derandomizing an Output-Sensitive Convex Hull Algorithm in Three Dimensions*

BERNARD CHAZELLE
Department of Computer Science
Princeton University
Princeton, NJ 08544, USA

JIŘÍ MATOUŠEK
Department of Applied Mathematics
Charles University
Malostranské nám. 25, 118 00 Praha 1, Czechoslovakia

February 5, 1992

Abstract

We consider the computation of the convex hull of a given n-point set in three-dimensional Euclidean space in an output-sensitive manner. Clarkson and Shor proposed an optimal randomized algorithm for this problem, with an expected running time $O(n \log h)$, where h denotes the number of points on the surface of the convex hull. In this note we point out that the algorithm can be made deterministic by using recently developed techniques, thus obtaining an optimal deterministic algorithm.

Keywords: Computational complexity, computational geometry, randomized algorithm, convex hull

The computation of the convex hull of a given n-point set in an Euclidean space of a small fixed dimension d has been intensively studied in computational geometry. Most of the work has concentrated on the worst-case scenario; it is known that for certain inputs, the computation may require $\Omega(n \log n)$ time for $d = 2, 3$ and $\Omega(n^{\lfloor d/2 \rfloor})$ for $d \geq 4$. These worst-case lower bounds can be matched: optimal algorithms were given by Graham [6] for $d = 2$, by Hong and Preparata [13] for $d = 3$, by Seidel [15] for every even d, by Clarkson and Shor [2] for all d (with randomization) and finally by Chazelle [1] who constructed an optimal deterministic algorithm for a general fixed dimension. For many point sets, however, the actual complexity of the convex hull is much smaller than the worst-case bound and it is interesting to investigate algorithms which can take advantage of this. Kirkpatrick and Seidel [8] gave a convex hull algorithm in dimension 2 which is optimal in this setting, attaining a running time $O(n \log h)$, where h denotes the number of vertices of the convex hull. Seidel [14] gave an output-sensitive algorithm for a general dimension d with running time $O(n^2 + h \log n)$. The quadratic overhead can be reduced to $O(n^{2-\frac{2}{\lfloor d/2 \rfloor + 1} + \varepsilon})$ for any fixed $\varepsilon > 0$ using the results of [11].

In this paper we consider the case of dimension 3. For this case, Edelsbrunner and Shi [5] gave a fairly complicated $O(n \log^2 h)$ output-sensitive algorithm. Clarkson and Shor [2] discovered an optimal $O(n \log h)$ randomized output-sensitive algorithm. In this note we observe that general methods for an efficient derandomization of geometric algorithms, developed in [9], [10], can be applied to Clarkson and Shor's algorithm, yielding the following result:

*Work by Bernard Chazelle has been supported by NSF Grant CCR-90-02352 and the Geometry Center.

Theorem 1 *Given a set P of n points in E^3, its convex hull can be computed deterministically in time $O(n \log h)$, where h is the number of points of P on the surface of the convex hull.*

Let us remark that for point sets in highly degenerate positions, the algorithm guaranteed by the previous theorem doesn't have the best kind of output-sensitivity one might desire: in a "truly" output-sensitive algorithm, h should stand for the number of vertices of the convex hull. We decided to use the above weaker formulation (similar to Clarkson and Shor's), since it allows us to make a general position assumption and brings a considerable technical simplification of the presentation. With some more technicalities, one can get also an output-sensitivity in the stronger sense.

In the sequel, we first introduce some tools, and then we give a reasonably self-contained proof of Theorem 1. Actually, there are now several ways to proceed, and we tried to choose one using the simplest machinery. For a comparison with the randomized algorithm we refer to [2], and for more background and examples of use of the derandomization methods we refer, e.g., to papers [9], [1], [10].

The algorithm is best described in a dual setting, where we consider a collection of n halfspaces in E^3, and we want to compute their intersection. In this setting, the parameter h stands for the number of facets of the intersection, and this is in turn proportional to the number of its vertices.

We will assume that the bounding planes of the halfspaces are in general position. This assumption can be removed either by *simulation of simplicity* (see [4]), or by a careful analysis of possible degenerate cases.

It is known that in linear time, one can decide whether the intersection of n halfspaces has a nonempty interior, and if yes, also find one its interior point o (see [3], [12]). Hence, our problem becomes the following: we have a collection H of n planes in general position, and a point o lying in none of the planes, and we seek the intersection of the halfspaces determined by the planes of H and the point o. This intersection will be denoted by $\mathcal{P}(H)$. Further, let $\Delta(H)$ denote a triangulation of $\mathcal{P}(H)$, constructed as follows: first every face of $\mathcal{P}(H)$ is triangulated from its vertex with the lexicographically smallest coordinate vector (so-called *bottom-vertex triangulation*), and then each triangle of these triangulations is lifted into a simplex with a vertex at the point o. It is well-known that the resulting decomposition of $\mathcal{P}(H)$ is a simplicial complex, and it has $O(|H|)$ simplices.

For a simplex s, let H_s denote the collection of planes of H intersecting it. We will use the following lemma, whose first part is by now a standard random sampling result:

Lemma 2

(i) [2] *Let R be a random r-tuple of planes drawn from H. Then, for suitable constants C, C', the following two conditions hold with a fixed positive probability:*

(a) *For every simplex $s \in \Delta(R)$, $|H_s| \leq C(|H|/r)l$.*

(b)
$$\sum_{s \in \Delta(R)} |H_s| \leq C'|H|.$$

(ii) *Given H and r, a sample R as in (i) can be computed deterministically, in time polynomial in $|H|$.*

Part (ii) of this lemma is proved by a standard application of the method of conditional probabilities of Raghavan and Spencer; see, e.g., [9] for an example of use of this method in a similar context. □

Now we need the concept of ε-approximations. In our setting, we say that a subset $A \subseteq H$ is an *ε-approximation* for H (with respect to simplices; $0 < \varepsilon < 1$ is a real number), if for every simplex s it is

$$\left| \frac{|H_s|}{|H|} - \frac{|A_s|}{|A|} \right| < \varepsilon.$$

We need the following particular case of results about computing ε-approximations:

Lemma 3 [10] *There exists a constant $\alpha > 0$, such that given H and r, $r < n^\alpha$, a $(1/r)$-approximation for H of cardinality $O(r^3)$ can be computed deterministically in time $O(n \log r)$.* □

Putting the previous two results together, we get

Corollary 4 *There exists a constant $\alpha > 0$, such that given H and r, $r < n^\alpha$, a sample R with the properties (a), (b) from Lemma 2 can be computed in time $O(n \log r)$.*

Proof: We compute R in two steps. First, in time $O(n \log r)$, we find a $(1/r)$-approximation A for H, of cardinality $O(r^3)$. Then we apply Lemma 2(ii), computing a sample R which has the required properties relative to A. If r is small enough, the running time of this second step will also be dominated by $O(n \log r)$. And using the definition of an ε-approximation, it is easy to see that a sample R which has the required properties relative to A will also be good with respect to H, only with somewhat worse constants of proportionality. □

Before we start with the algorithm, we need two more auxiliary results. The first one replaces a randomized incremental construction in [2] by a deterministic counterpart.

Lemma 5 *Given H and $\Delta(R)$ as in Lemma 2, one can compute the collection H_s for every $s \in \Delta(R)$, in total time $O(n \log r)$.*

Proof: We will trace the incidences with the simplices of $\Delta(R)$ for every plane of H separately. Once we know one simplex s intersected by a plane h, we can trace all the remaining incidences of h in time proportional to their number, by a graph search algorithm (here one uses the special properties of the triangulation $\Delta(R)$). If suffices to show how to find a starting simplex; apparently it suffices to find a vertex of $\mathcal{P}(R)$ separated by h from o. A dual version of this problem is to find a facet of a polyhedron separating a query point from infinity; this in turn can be transformed into a two-dimensional point location problem, for which optimal solutions are known (e.g., [7]). Hence, with $O(r)$ preprocessing, we can find a starting simplex (or decide that h misses $\Delta(R)$ completely) in $O(\log r)$ time. Thus the total time for computing all the H_s's is at most $O(n \log r) + \sum_{s \in \Delta(R)} |H_s| = O(n \log r)$. □

Lemma 6 [2] *Let s be a simplex and H_s the collection of planes intersecting it. One can decide whether s contains a vertex of $\mathcal{P}(H)$ by a deterministic algorithm in time $O(|H_s| \log h)$, where h denotes the number of vertices of $\mathcal{P}(H)$.*

For completeness, we sketch the proof here: We consider each of the planes ρ bounding the simplex s, and we compute $\rho \cap \mathcal{P}(H_s)$. Each such problem is just a dual of a computation of

the convex hull of a planar point set, and we use the output-sensitive algorithm of Kirkpatrick and Seidel [8] for this computation. Hence in time $O(|H_s| \log h)$ we can find the portion of $\mathcal{P}(H)$ within every face of s. With this knowledge we can use a straightforward consistency check to detect whether s contains a vertex of $\mathcal{P}(H)$. □

Now we can describe the output-sensitive convex hull algorithm. One of the problems we must handle is that we do not know the value of h in advance. To circumvent this problem, we establish the following:

Lemma 7 *Let H be a collection of n planes in E^3, and $b \leq n$ a parameter. There exists a constant B and a deterministic algorithm, such that if $\mathcal{P}(H)$ has at most b vertices, then the algorithm computes $\mathcal{P}(H)$ in time at most $Bn \log b$; otherwise it always finishes within the above time bound and it may either compute $\mathcal{P}(H)$ or report a failure.*

Once this lemma is proved, we complete the proof of Theorem 1 as follows: We define a sequence $b_1, b_2, \ldots$ by setting b_1 to a large enough constant, and $b_{i+1} = b_i^2$. We then run the algorithm from the Lemma first with $b = b_1$, then if a failure is reported we repeat it with $b = b_2$, etc. Obviously, we gain the desired answer no later than for $b_i \geq h$, so the total running time is at most

$$\sum_{i; b_i < h} Bn \log b_i \leq Bn \sum_{i=1}^{\log\log h} 2^i = O(n \log h).$$

In the proof of Lemma 7, we may concentrate on the case when b is relatively small compared to n, namely $b < n^\beta$ for some fixed $\beta > 0$ (otherwise an $O(n \log n)$ convex hull algorithm will do). In such case, we use the following algorithm:

We set $H_0 = H$. We assume inductively that in the i-th step of the algorithm ($i = 0, 1, \ldots$), we have a collection H_i of cardinality $n_i \leq n/2^i$, which contains all planes of H contributing a facet of $\mathcal{P}(H)$. If $n_i \log n \leq n \log b$, we use an $O(n \log n)$ convex hull algorithm to compute $\mathcal{P}(H_i) = \mathcal{P}(H)$, in time $O(n_i \log n) = O(n \log b)$, and this will be the last step of the algorithm. Otherwise our goal will be to discard at least half of the planes of H_i. Note that in this case n_i must be quite large compared to b, in particular we may assume $b \leq n_i^\beta$ for a prescribed constant $\beta > 0$. This allows us to apply the approximation tools for the deterministic computation.

We proceed as follows: we set $r = 3Cb \log b$, and we compute a sample $R \subseteq H_i$ according to Corollary 4, in time $O(n_i \log b)$. Then we compute $\Delta(R)$, the collections of planes of H_i intersecting each simplex of $\Delta(R)$ (using Lemma 5) and we use Lemma 6 to detect the simplices of $\Delta(R)$ which contain at least one vertex of $\mathcal{P}(H_i)$.

If $h \leq b$, this computation takes time at most $O(r) \cdot O((n_i/r) \log b) = O(n_i \log b)$. If $h > b$, then the running time might be longer; we thus let this computation run for only $B' n_i \log b$ steps for a suitable constant B', and if it does not finish within this limit, we terminate it and we report a failure of the algorithm.

If this computation has finished successfully, we form a collection H_{i+1}. We include all the planes intersecting the simplices which do contain a vertex of $\mathcal{P}(H)$. In this way we have included all relevant planes. If $h \leq b$, then at most b simplices of $\Delta(R)$ contain a vertex, and by the property (a) from Lemma 2, we get $n_{i+1} = |H_{i+1}| \leq b \cdot C(n_i/r) \log r \leq n_i/2$. Hence if $h \leq b$ holds, we may certainly continue with the $(i+1)$-st step of the algorithm, having spent $O(n_i \log b)$ time on the i-th step. Otherwise it may happen that $n_{i+1} > n_i/2$, and in such a case we report a failure and finish.

The total running time of this algorithm is $O(n \log b) + \sum_i O(n_i \log b) = O(n \log b)$ as required, and this finishes the proof of Lemma 7. □

References

[1] B. Chazelle. An optimal convex hull algorithm and new results on cuttings. In *Proc. 32nd Ann. IEEE Symp. Theory Comput. Sci.*, Oct. 1991, 29–38.

[2] K. L. Clarkson and P. Shor. New applications of random sampling in computational geometry II. *Disc. & Comput. Geo.*, 4 (1989), 387–421.

[3] M. E. Dyer. *Linear time algorithms for two- and three-variable linear programs. SIAM J. Comput.*, 13 (1984), 31–45.

[4] H. Edelsbrunner. *Algorithms in combinatorial geometry.* Springer-Verlag, 1987.

[5] H. Edelsbrunner and W. Shi. *An $O(n \log^2 h)$ time algorithm for the three-dimensional convex hull problem. SIAM J. Comput.*, 20 (1991), 259–269.

[6] R. L. Graham. An efficient algorithm for determining the convex hull of a planar point set. *Inform. Proc. Lett.*, 1 (1972), 132–133.

[7] D. G. Kirkpatrick. Optimal search in planar subdivisions. *SIAM J. Comput.*, 12 (1983), 28–35.

[8] D. G. Kirkpatrick and R. Seidel. The ultimate planar convex hull algorithm? *SIAM J. Comput.*, 15 (1986), 287–299.

[9] J. Matoušek. Approximations and optimal geometric divide-and-conquer. In *Proc. 23rd. ACM Symp. Theory of Comput.*, 1991, 506–511.

[10] J. Matoušek. Efficient partition trees. In *Proc. 7th ACM Symp. Comput. Geo.*, 1991, 1–9.

[11] J. Matoušek. Linear optimization queries. Submitted for publication, 1991.

[12] N. Megiddo. Linear programming in linear time when the dimension is fixed. *J. ACM*, 31 (1984), 114–127.

[13] F. P. Preparata and S. J. Hong. Convex hulls of finite point sets in two and three dimensions. *Comm. of the ACM*, 20 (1977), 89–73.

[14] R.Seidel. Constructing higher-dimensional convex hulls at logarithmic cost per face. In *Proc. 18th ACM Symp. Theory of Comput.*, 1986, 404–413.

[15] R. Seidel. A convex hull algorithm optimal for point sets in even dimensions. Univ. British Columbia, Technical Report 81–14, 1981.