On the Complexity of Shattering Using Arrangements

(extended abstract)

Robert Freimer[*] Joseph S. B. Mitchell[†] Christine D. Piatko[‡]
Cornell University, Ithaca, NY 14853

June 27, 1990

1 Introduction

Let $\mathcal{O} = \{X_1, \ldots, X_n\}$ be a set of objects in $\Re^d$. We say that a subdivision $\mathcal{S}$ of $\Re^d$ *shatters* $\mathcal{O}$ if the following conditions hold:

1. Each object X_i is contained within the closure of some d-dimensional cell of $\mathcal{S}$; and
2. The closure of any d-dimensional cell of $\mathcal{S}$ contains no more than one object of $\mathcal{O}$.

The conditions assure that $\mathcal{S}$ separates each X_i from all the other objects. The problem of finding a subdivision that shatters the set $\mathcal{O}$ is called the *shattering problem*. We say that a set of hyperplanes shatters $\mathcal{O}$ if the subdivision induced by their arrangement shatters $\mathcal{O}$.

In this paper, we concentrate on the two-dimensional version of the shattering problem. In particular, we are interested in the problem of shattering a set of disjoint polygonal objects with an arrangement of lines in the plane.

Shattering problems are closely related to stabbing problems (e.g., [HM88,MT82]) and other questions of separability (see [Meg88]). A potential application for shattering problems arises in the field of Constructive Solid Geometry (CSG), where one is interested in converting a boundary representation (straight edges and circular arcs) for a planar solid into a minimum CSG (Boolean formula) representation (see [SV89]). This application requires the minimum-cardinality shattering of a set of pairs by lines chosen from the set of lines defined by endpoints of circular arcs.

We show that the problem of finding a minimum-cardinality set of shattering lines of a set of n objects is NP-complete, even if the objects are all points. We then give an $O(E \log N + n^2 \log n)$ time algorithm to determine if a set of polygonal objects with N vertices can be shattered by lines, and, if so, to output a "small" witness set of shattering lines. Here, E is the size of the visibility graph of the scene. We give a second algorithm with running time $O(E \log N + nE^{.695})$, which is superior for small values of E, and generalizes to yield a slightly super-cubic algorithm for shatterability of polyhedra by planes in three dimensions.

2 Point Shattering

A special case of our problem is that of shattering a set of n points in the plane. First, we have the following fact:

Lemma 1: Any set of n points in the plane can be shattered by $n-1$ parallel lines. For rational points, the lines can be determined in linear time. In general, $O(n \log n)$ time suffices. Any set of n points requires at least $\lceil \frac{1}{2}(\sqrt{8n-7}-1) \rceil$ shattering lines, and this bound is tight.

For n collinear points, $n-1$ lines is an optimal size shattering. Even if the points are in general position, $O(n)$ shattering lines may be needed, since n convex points require $\lceil \frac{n}{2} \rceil$ shattering lines. However, other sets of points can be shattered by as few as $O(\sqrt{n})$ lines. Given this variety, it would be desirable to have an efficient algorithm to find a minimum size shattering for a set of points. Unfortunately, this turns out to be *NP*-Complete.

[*]Department of Computer Science. Supported in part by an AT&T Ph.D. Scholarship.

[†]School of Operations Research and Industrial Engineering. Supported in part by NSF grants IRI-8710858 and ECSE-8857642, and by a grant from Hughes Research Laboratories.

[‡]Department of Computer Science. Supported in part by an NSF Graduate Fellowship, and by NSF grants IRI-8710858 and ECSE-8857642.

We show that finding a minimum-cardinality shattering of a set of points is *NP*-Complete by a reduction from the Point Covering problem, which is *NP*-Complete by Megiddo and Tamir [MT82]. Formally, the problems we are considering are:

Problem Point Covering (PC): Given a set of rational points $(x_1, y_1), \ldots, (x_n, y_n)$ and an integer k_{pc}, does there exist a set of k_{pc} straight lines such that each point (x_i, y_i) lies on at least one line?

Problem Point Shattering (PS): Given a set of rational points $(x_1, y_1), \ldots, (x_n, y_n)$ and an integer k_{ps}, does there exist a set of k_{ps} straight lines that shatter the n points (i.e., that form an arrangement such that no cell contains more than one point (x_i, y_i))?

Theorem 2: The Point Shattering problem is *NP*-Complete.

Proof (sketch): Given an instance of PC with n (even) points, we construct an instance of PS with $10n+40$ points. The points are carefully constructed so that a shattering by $k_{ps} = k_{pc} + n + 11$ lines exists if and only if a covering by k_{pc} lines exists. We build the instance of PS by replacing each of the points of *PC* by a pair of points. The pairs are chosen so that for any set of pairs, there is a shattering line for the set if and only if there is a covering line for the corresponding set of points. If the pairs did not have to be separated from each other in a shattering, then the reduction would be finished. To control how the center pairs are separated from each other, we construct $8n + 40$ points to the left and the right of the center pairs that induce a grid consisting of $n + 11$ horizontal and vertical lines (Figure 1 illustrates a typical grid). The grid separates the center pairs from each other, without shattering individual pairs or leaving any new points that must be shattered. This completes the description of the instance of PS. If a shattering of size k_{ps} exists for the instance of PS, then we show that the k_{pc} non-grid lines in the shattering correspond to the covering lines for the instance of PC. ■

Even if we restrict the shattering lines to be horizontal or vertical, the problem remains *NP*-Complete, by a reduction from 3-SAT (similar to Hassin and Megiddo's *NP*-Completeness proof for the problem of stabbing unit horizontal segments with horizontal and vertical lines [HM88]).

3 Shattering Polygonal Objects

In this section, we give two algorithms that determine if a set of polygonal objects is shatterable by an arrangement of lines. If the objects are shatterable, then we require that our algorithms produce a "small" witness: a collection of a "small" number of lines whose arrangement shatters the given objects. The notion of "small" is made precise in the following lemma.

Lemma 3: Let $\mathcal{O}$ be a set of n objects in $\Re^d$. If $\mathcal{O}$ is shatterable by an arrangement of hyperplanes, then it is shatterable by an arrangement of $n-1$ or fewer hyperplanes.

As shown in Figure 2a, the question of shatterability is nontrivial even for a collection of three line segment objects. Figure 2b shows that the naive approach of attempting to find lines that cut off one object at a time can fail to find a shattering when one exists.

Assume that we are given a set $\mathcal{O}$ of n simple polygons, with a total of N vertices. By a simple plane-sweep algorithm, we can detect in time $O(N \log N)$ if there are any overlapping polygons. Since overlapping objects cannot be shattered, we assume from now on that the polygons are disjoint.

We describe two algorithms which decide if a set of polygonal objects in the plane can be shattered by an arrangement of lines. The first algorithm runs in time $O(E \log N + n^2 \log n)$, where E is the number of edges in the visibility graph of $\mathcal{O}$. The second algorithm runs in time $O(E \log N + nE^{.695})$, which is slower than the first algorithm when E is large (e.g., E may be quadratic in N). The second algorithm has the advantage that it generalizes to three dimensions.

Both algorithms begin by computing the visibility graph (VG) of $\mathcal{O}$ in time $O(E + N \log N)$, using the algorithm of [GM87]. As a by-product of building VG, we can, within the same time bounds, identify the set C of all candidate shattering lines. The candidate lines are those that contain VG edges (not on the convex hull of $\mathcal{O}$) such that no object is intersected by the line. We can further restrict attention to those lines that are "pinned" such that no counterclockwise rotation is possible without intersecting an object. If the set $\mathcal{O}$

is shatterable, then it is shattered by the arrangement of $O(E)$ candidate lines C. Our goal is to determine if $\mathcal{O}$ is indeed shatterable, and, if so, find a subset of no more than $n-1$ candidate lines that shatter $\mathcal{O}$.

Our algorithms incrementally build a set of "productive" candidate lines that will be used in our attempt to shatter $\mathcal{O}$. At any given stage, the objects have been partially shattered by the productive lines. Let a *cluster* be the set of objects that belong to a cell that currently has two or more objects. Initially, there is only one cluster (all of $\mathcal{O}$).

For the first algorithm, we begin by sorting the slopes of the VG edges (in time $O(E \log N)$). Let θ be the current "vertical" direction. Initially, $\theta = \pi/2$. Our goal is to maintain the *visibility profile* (VP(θ)) of the convex hulls of the clusters with respect to the vertical direction θ, as we vary θ through a range of 2π. We also store the extreme left and right points of each cluster, with respect to the current direction θ. Initially, the VP is trivial, since there is only one cluster $\mathcal{O}$.

As we rotate the vertical direction θ, critical changes occur only at directions corresponding to the VG edges (which are encountered in slope order). At each such critical direction, we may have a combinatorial change in the current VP or we may discover a new productive candidate line.

If we encounter a VG edge e that corresponds to a candidate line, we consult the current VP to detect if the candidate line stabs at least one cluster (in query time $O(\log n)$). If it does, then we add this candidate line to our set of productive lines, and we update the VP, breaking all clusters stabbed by this line. This requires that we identify all VG edges within each stabbed cluster that cross the line; this can be done in time proportional to the number of VG edges crossed, using the methods of [MW90] to update a VG when adding a new obstacle (we think of the candidate line as the new "obstacle" in our case). We can determine, in linear time, the set of stabbed clusters by checking if the candidate line stabs the segment joining the extreme left and right point of each cluster. We then use a plane sweep to update the VP in $O(n \log n)$ time. (Since there can be at most $n-1$ productive candidate lines, the overall cost of these sweeps will be $O(n^2 \log n)$.) If the candidate line stabs no cluster, then it is not productive in refining the current partial shattering.

Also, for each VG edge e encountered, we must check to see if the VP changes (combinatorially) as we sweep through this direction. A change occurs when e corresponds to an edge incident to an extreme left or extreme right point of a cluster (in which case the extreme point changes accordingly), or when e corresponds to an edge between two vertices that are currently extreme (left or right) on the clusters (in which case a simple constant-size update is done on the VP). Since only constant-size changes occur at each such critical direction, the total cost of all updates is only $O(E \log N)$.

If the set of clusters becomes empty during our sweep in θ, then we stop and conclude that $\mathcal{O}$ is shatterable by the current set of productive lines (which is no greater than $n-1$). Otherwise, we finish the sweep in θ and have one or more clusters of unshattered objects, and we conclude that $\mathcal{O}$ is not shatterable. The total complexity of this algorithm is $O(E \log N + n^2 \log n)$.

Theorem 4: Given a set of n polygonal objects $\mathcal{O}$ in the plane with a total of N vertices, one can detect if $\mathcal{O}$ can be shattered by a set of lines and, if so, output such a set of size at most $n-1$ in time $O(E \log N + n^2 \log n)$, where E is the size of the visibility graph induced by $\mathcal{O}$. The space required by the algorithm is $O(E)$.

Our second algorithm differs in the method by which it selects the subset of candidate lines. We compute the visibility graph and the candidate lines as before. Consider the set C^* of points that are dual to the candidate lines C. Preprocess C^* to be able to handle triangle range queries of the following type: Given a query triangle (in dual space), either return a point within the triangle or answer that the triangle is empty. By [Ede87,EW86], this preprocessing can be done in time $O(E \log E) = O(E \log N)$, yielding a data structure of size $O(E)$ that answers each query in time $O(E^{.695})$.

Let e be a VG edge on the convex hull of $\mathcal{O}$ (thus, e does not generate a candidate line). We now call procedure $SPLIT(VG, e)$, which returns a set of candidate lines that shatter $\mathcal{O}$.

$SPLIT(G, e)$:

1. **Find a candidate line ℓ that stabs e: The dual of e is a double wedge. By performing up to two triangle queries, we find a point of C^* within the double wedge. This step takes time $O(E^{.695})$.**
2. **Remove all edges of G stabbed by ℓ: This can be done in time proportional to the number of edges stabbed, using the results of [MW90] on maintaining visibility graphs under insertions of new obstacles.**

G is now split into two pieces G_1 and G_2.

3. For each G_i with at least two of the original objects, determine an edge e_i on its convex hull (this can be done in constant time), and let $L_i = SPLIT(G_i, e_i)$. Otherwise, let $L_i = \emptyset$. Return $\{\ell\} \cup L_1 \cup L_2$.

If we ever fail to stab an edge e (in Step 1), then we know that the set $\mathcal{O}$ cannot be shattered, so we return with failure. Otherwise, the algorithm produces a shattering, since the set of lines output is constructed in such a way that every visibility graph edge is cut. Each step of $SPLIT$ requires $O(E^{.695})$ time for the triangle queries. The effort of removing edges is charged off to the edges of the visibility graph, so it requires only $O(E)$ time overall. There can be at most $n-1$ calls to $SPLIT$ since each call yields at least one new productive candidate shattering line. Thus, the algorithm requires total time $O(E \log N + nE^{.695})$.

Although the running time of our second algorithm is sometimes inferior to our first, it has the advantage of generalizing to higher dimensions. In three dimensions, it allows us to obtain a slightly super-cubic time algorithm to detect shatterability of polyhedra by planes.

4 Conclusion

Several directions for extensions and future research are possible for our problem. We list here a few problems on which we are working:

1. Higher dimensions – How quickly can one determine if a given set of polyhedra in three dimensions can be shattered by a set of planes? We have a slightly super-cubic bound at the moment.
2. Approximation algorithms – Given that the problem of finding a minimum cardinality shattering is hard, what can be said about provably good approximation schemes to find a small size shattering?
3. Shattering with other subdivisions – Instead of using line arrangements to shatter a set of objects in the plane, we may ask if there exists a convex subdivision that shatters the objects. This question is answered by the methods of [ERS87] or [Wen90]. However, the complexity of the following problem remains open: Determine if there exists a convex subdivision that shatters a set $\mathcal{O}$ of objects in such a way that each bounded face of the subdivision contains exactly one object. Figure 2 gives an example of three line segments for which such a shattering does not exist.
4. CSG Application – What is the complexity of the shattering problem of [SV89]?

References

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

[EOW83] H. Edelsbrunner, M. H. Overmars, and D. Wood. Graphics in flatland: a case study. In *Advances in Computing Research*, chapter 2, pages 35–59. JAI Press, 1983.

[ERS87] H. Edelsbrunner, A. Robison, and X. Shen. Covering convex sets with non-overlapping polygons. Technical report, Dept. Computer Science, U. Illinois at Urbana-Champaign, 1987.

[EW86] H. Edelsbrunner and E. Welzl. Halfplanar range search in linear space and $O(n^{.695})$ query time. *Information Processing Letters*, 23:289–293, 1986.

[GM87] S. Ghosh and D. Mount. An output sensitive algorithm for computing visibility graphs. In *Proc. 28th FOCS*, pages 11–19, October 1987.

[HM88] R. Hassin and N. Megiddo. Approximation algorithms for hitting objects by straight lines. Technical report, Tel Aviv University, May 1988. To appear in *SIAM J. Discrete and Applied Math.*

[Meg88] N. Megiddo. On the complexity of polyhedral separability. *Discrete and Computational Geometry*, 3:325–337, 1988.

[MT82] N. Megiddo and A. Tamir. On the complexity of locating linear facilities in the plane. *Operations Research Letters*, 1(5):194–197, November 1982.

[MW90] J. S. B. Mitchell and E. Welzl. Dynamically maintaining a visibility graph under insertions of new obstacles. Manuscript, 1990.

[SV89] V. Shapiro and D. Vossler. Efficient CSG representations of planar solids. Technical report, Sibley School of M&AE, Cornell University, November 1989.

[Wen90] R. Wenger. Upper bounds on geometric permutations for convex sets. *Discrete and Computational Geometry*, 5(1):27–33, 1990.

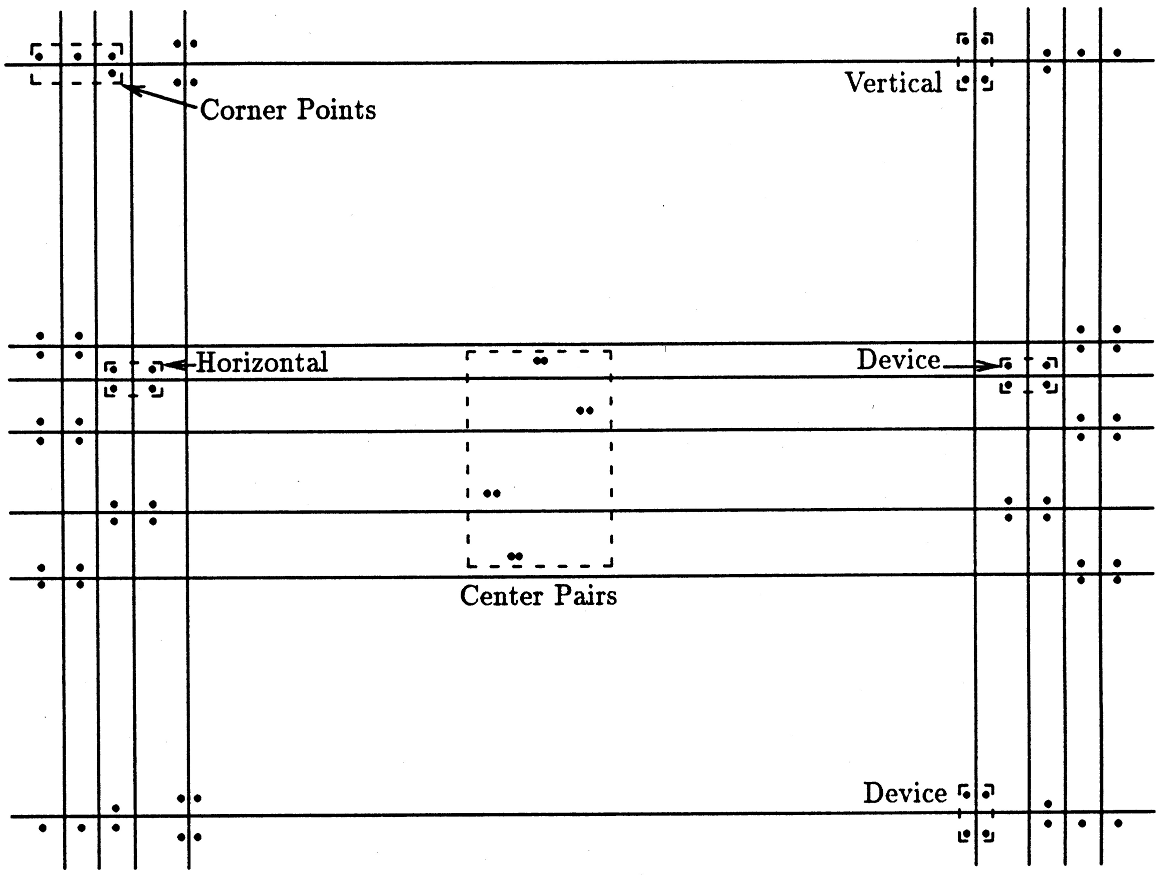

Figure 1: A typical instance of PS, along with the associated grid.

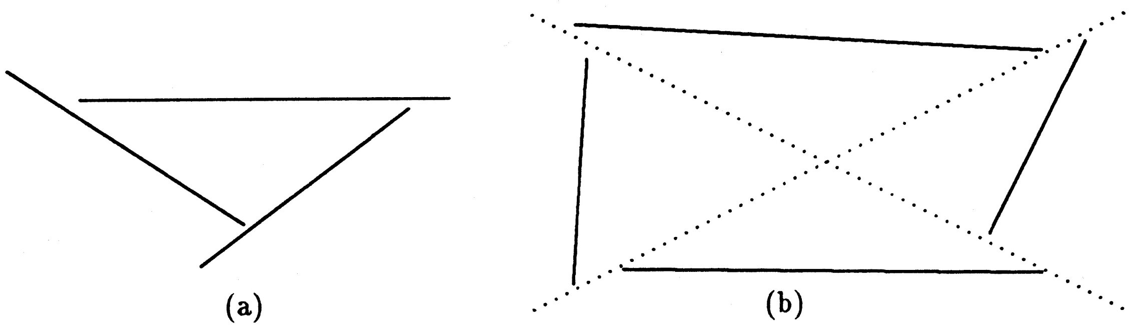

Figure 2: (a) Three line segments that cannot be shattered. (b) Four shatterable line segments that cannot be cut off one by one.