On the Complexity of Shattering Using Arrangements

Robert Freimer*
Joseph S. B. Mitchell**
Christine D. Piatko***

TR 91-1197
April 1991

Department of Computer Science
Cornell University
Ithaca, NY 14853-7501

An extended abstract of this paper has appeared in the *Proceedings of the Second Canadian Conference on Computational Geometry,* August 1990, pp. 218-222.
*Dept. of Computer Science. Supported in part by an AT&T Ph.D. Scholarship, and by NSF grant ECSE-8857642.
**School of Operations Research and Industrial Engineering. Supported in part by NSF grant ECSE-8857642, and by a grant from Hughes Research Laboratories.
***Dept. of Computer Science. Supported in part by an NSF Graduate Fellowship, and by NSF grant ECSE-8857642.

On the Complexity of Shattering Using Arrangements

Robert Freimer* Joseph S. B. Mitchell† Christine D. Piatko‡
Cornell University, Ithaca, NY 14853-7501
April 26, 1991

Abstract

A subdivision $\mathcal{S}$ of $\Re^d$ is said to shatter a set of objects if each object is contained within the closure of its own cell of $\mathcal{S}$. In this paper, we examine the problem of shattering a set of bounded polyhedral objects in $\Re^d$ by a subdivision formed by an arrangement of hyperplanes. We show for $\Re^d$, $d \geq 2$ that finding a minimum-cardinality set of hyperplanes whose arrangement shatters a set of points is *NP*-Complete. We then give algorithms to find a linear-size set of shattering hyperplanes for a set of n bounded polyhedral objects in $\Re^d$, if one exists. For $d = 2$, we provide two algorithms with worst-case time complexities $O(E + N \log N + n^2 \log n)$ and $O(E + N \log N + C \log n + nC^{.695})$, where E is the size of the visibility graph of the objects, N is the total number of vertices, and C is the number of candidate lines considered (at worst $\min\{E, n^2\}$). Our final algorithm has worst-case time complexity $O(N^{d+1})$ for $d \geq 3$.

1 Introduction

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

Related Work The version of the shattering problem in the plane using arrangements of parallel lines is the well-known "disjoint projections" or "shadow direction" problem: Find a direction in the plane such that the set of objects projects disjointly onto a line in

An extended abstract of this paper has appeared in the *Proceedings of the Second Canadian Conference on Computational Geometry*, August 1990, pp. 218–222.

*Department of Computer Science. Supported in part by an AT&T Ph.D. Scholarship, and by NSF grant ECSE-8857642.

†School of Operations Research and Industrial Engineering. Supported in part by NSF grant 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 grant ECSE-8857642.

this direction. This problem has been solved in time $O(n^2 \log n)$ (see [EOW83]), but it is open if it can be solved in time $O(n^2)$. The time-bound can be rewritten more precisely as $O(E \log n)$, where E is the size of the visibility graph of the objects.

Shattering problems are closely connected to questions of separability (see [Meg88]). For example, consider the following problem: Given two sets, can they be separated by some hyperplane or other curve? If each set is connected, then this is a shattering problem for two sets. Otherwise, each set may have many disjoint connected components. Depending on the complexity of the sets, a separator may consist of multiple disjoint curves. However, any separator for the two sets can be extended with additional curves to a subdivision that shatters all their connected components, which illustrates that separation is strongly linked to the shattering problem.

Various stabbing problems are also related to shattering. For example, consider one such problem: Given a set of objects, does there exist a stabbing set such that every object is intersected by at least one member of the stabbing set? Usually, the stabbing set consists of lines (see [HM88,MT82]), but circles or other curves can also be used. We can reformulate the shattering problem in terms of stabbing as follows: Does there exist a set of lines that stabs every edge of the visibility graph of the objects while avoiding the objects? The lines stab all the edges if and only if they shatter the objects. When the objects are points in the plane, we show in Section 2 that minimum-cardinality stabbing reduces to minimum-cardinality shattering.

Motivation Consider the problem of manufacturing widgets from sheets of plywood. Each sheet is cut into many widgets, possibly of differing type, according to a layout plan. A fast circular saw performs a set of preliminary cuts across each stack of boards, following straight-line paths. Each resulting piece either contains the material for one widget or is waste. The widgets are finished in parallel at other stations using slower jigsaws. For a given layout of widgets, one would like to minimize the number of cuts performed, because the stack must be prepositioned before each cut. The resulting optimization question is the problem of finding a minimum-cardinality shattering using an arrangement of lines for a set of objects in $\Re^2$.

Another potential application for shattering problems arises in the field of Constructive Solid Geometry (CSG), when 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 [SV90]). For example, consider an object X in the plane described by a boundary representation of line segments and circular arcs. The CSG representation for X is created by a Boolean combination of closed half-planes, closed discs, and complements of open discs. The half-planes are defined by extensions of line segments from the boundary of X. Similarly, discs and complements of discs are defined by circles containing circular arcs from the boundary. For some objects, additional half-planes defined by lines through endpoints of circular arcs are required to represent the object. These lines must shatter a set of pairs, where each pair consists of one point inside the object and another outside, with both points having the same Boolean combination of boundary half-planes, discs and complements of discs. The minimum CSG representation can be computed by finding a minimum-cardinality set of shattering lines for the pairs. Notice that the pairs do not need to be separated from each other in this shattering.

Overview of Paper We concentrate on the problem of shattering a set of n disjoint bounded polyhedral objects in $\Re^d$ using an arrangement of hyperplanes. We begin by restricting our attention in Section 2 to shattering sets of points. We describe various properties of such shatterings, and then show that the problem of finding a minimum-cardinality set of shattering hyperplanes is *NP*-complete for $d \geq 2$. A restricted version of this problem using only hyperplanes perpendicular to two of the coordinate axes is also *NP*-Complete. In Section 5, we present a simple greedy heuristic that always finds a shattering whose size is at worst $(1 + \log n)$ times the optimal for a set of n points.

In Section 3, we consider the case of polygonal objects in the plane, noting that the problem remains *NP*-Complete. Unlike point sets, some sets of polygonal objects are not shatterable by an arrangement of lines. Our first algorithm determines in time $O(E + N \log N + n^2 \log n)$ if a set of n polygonal objects with N vertices can be shattered by lines, and if so, outputs a linear-size 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 $O(E + N \log N + C \log n + nC^{.695})$, where C is the number of candidate lines considered (at worst $\min\{E, n^2\}$). The second algorithm is superior for small values of E, and generalizes to yield an algorithm with $O(N^{d+1})$ running time for $\Re^d$, which is described in Section 4.

2 Point Shattering

The simplest version of our problem is shattering a set of n points in $\Re^d$. We begin with the following facts:

Lemma 1: Any set of n points in $\Re^d$ can be shattered by $n - 1$ parallel hyperplanes. For rational points, the hyperplanes can be determined in $O(dn)$ time; in general, $O(dn \log n)$ time suffices. Any set of n points requires at least $\Omega(\sqrt[d]{n})$ shattering hyperplanes, and this bound is tight.

Proof: We start with an algorithm to determine $n-1$ parallel hyperplanes that shatter a set of n points.

$\boldsymbol{d = 1}$: We are given n distinct rational points $x_1, \ldots, x_n$, each expressed as a fraction ν_i/δ_i, where ν_i, δ_i are integers, and $\delta_i > 0$. Let $\delta_{max} = \max_i \delta_i$. The minimum distance between points is bounded below by $\Delta x_{min} = (\delta_{max})^{-2}$. For each x_i except the largest, define a shattering point $x_i + \frac{1}{2}\Delta x_{min}$. This procedure runs in $O(n)$ time. For real-valued points, denominators cannot be used to determine Δx_{min}. Instead, find Δx_{min} by sorting the points, which increases the time complexity to $O(n \log n)$.

$\boldsymbol{d \geq 2}$: Given n points $(x_1^1, x_1^2, \ldots, x_1^d), \ldots, (x_n^1, x_n^2, \ldots, x_n^d)$, we determine a $(d-1)$-dimensional hyperplane with normal $\vec{n}$, such that all the points project uniquely onto the hyperplane, using the orthogonal projection. We then apply this algorithm in $d-1$ dimensions to shatter the projected points. The resulting hyperplanes in $\Re^{d-1}$ are swept along the line containing $\vec{n}$ to produce $n-1$ parallel hyperplanes that shatter the original points. We determine $\vec{n}$ by finding a slope greater than any finite slope of a line defined by two of $(x_1^1, x_1^2), \ldots, (x_n^1, x_n^2)$. The maximum difference in x^2-coordinates is $\Delta x_{max}^2 = \max_i x_i^2 - \min_i x_i^2$. Let Δx_{min}^1 be the minimum difference between distinct x^1-coordinates, which is determined using one of the methods previously described. We use

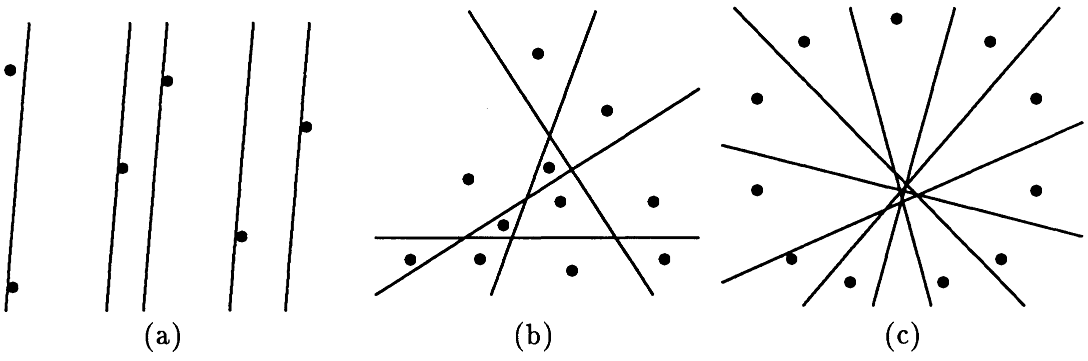


Figure 1: (a) Parallel shattering lines. (b) Four lines can shatter at most eleven points. (c) Eleven convex points require six shattering lines.

$m = 2\Delta x^2_{max}/\Delta x^1_{min}$ as the slope. Thus, $\vec{n} = (\frac{1}{m+1}, \frac{m}{m+1}, 0, \ldots, 0)$. Determining $\vec{n}$ requires $O(n \log n)$ time ($O(n)$ time if the points are rational). The overall time complexity for the algorithm is $O(dn \log n)$ ($O(dn)$ for rational points). Figure 1a provides an example for $\Re^2$.

The bottleneck in the above algorithm is in finding a lower bound on the closest distance between two points. This step is closely related to the problem of ϵ-closeness, which has an $\Omega(n \log n)$ lower bound (see [PS85]). This suggests that the time required for shattering real-valued points is $\Omega(n \log n)$, although we know of no better lower bound than $\Omega(n)$.

An arrangement of r hyperplanes in $\Re^d$ can have at most $\sum_{i=0}^{d} \binom{n}{i}$ cells ([Ede87], page 10). This fact immediately implies a lower bound on the number of hyperplanes necessary to shatter n objects. By solving for r, we determine that $\Omega(\sqrt[d]{n})$ hyperplanes are necessary in any shattering. This bound is tight, since an object can be placed in every cell of an arrangement of hyperplanes (see Figure 1b). The exact bounds for $d = 1, 2, 3$ are respectively $n-1$, $\left\lceil \frac{1}{2}(\sqrt{8n-7}-1) \right\rceil$, and $\left\lceil \frac{-5}{3R} + R \right\rceil$ where $R = \sqrt[3]{3n-3+\sqrt{\frac{125}{27}+(3-3n)^2}}$. ■

For n collinear points, $n-1$ hyperplanes is an optimal size shattering. If the points are in general position (no subset of $d+1$ points lies on a common hyperplane), $\Omega(n)$ shattering hyperplanes may still be needed. For example in $\Re^2$, n points in convex position require $\left\lceil \frac{n}{2} \right\rceil$ shattering lines (see Figure 1c). Similar examples exist in higher dimensions (see [Fre91]). However, other sets of points can be shattered by as few as $\Omega(\sqrt[d]{n})$ hyperplanes. We are therefore motivated to study the problem of finding a minimum cardinality set of hyperplanes that shatter a set of objects. Unfortunately, this problem turns out to be *NP*-Complete. However, there is an approximation algorithm that finds a shattering whose size is at most $(1 + \log n)$ times the optimal (see Section 5).

We show that finding a minimum-cardinality shattering of a set of points in $\Re^2$ 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 $(x_1, y_1), \ldots, (x_n, y_n)$ in $\Re^2$ and an integer k_{pc}, does there exist a set of k_{pc} straight lines such that each point lies on at least one line?

Problem Point Shattering (PS): Given a set of rational points $(x_1, y_1), \ldots, (x_n, y_n)$ in $\Re^2$ 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)?

Theorem 2: The Point Shattering Problem in $\Re^2$ is *NP*-Complete.

Proof: PS is in *NP* since it is easy to check in polynomial time if a set of lines shatters n points. We show that PS is *NP*-Complete by the following reduction from PC.

Given an instance of PC with n 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 complete. 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 2 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.

Formally, we start with an instance of PC: n rational points $(x_1, y_1), \ldots, (x_n, y_n)$ and the integer k_{pc}. Without loss of generality, assume each y-coordinate is unique, and the points are sorted by increasing y-coordinate. If the y-coordinates are not all unique, then compute a new coordinate system and use it implicitly to prevent the introduction of irrational coordinates. Also, assume n is even and at least four; this can be assured by adding points above (x_n, y_n). We build an instance of PS with $10n + 40$ points and $k_{ps} = k_{pc} + n + 11$.

For each point (x_i, y_i), construct a pair of points (x_i, y_i), $(x_i + \epsilon_c, y_i)$, with $\epsilon_c > 0$. We choose ϵ_c so that for any set of pairs, there is a line that shatters the set if and only if the corresponding set of points are collinear. Since ϵ_c is positive, any set of pairs corresponding to collinear points can be shattered by shifting the points' covering line slightly. We assure that only pairs corresponding to collinear points can be shattered by a line, by choosing ϵ_c small enough. For each line ℓ defined by two points (x_i, y_i), (x_j, y_j), $i < j$, determine an ϵ as the minimum of the following (see Figure 3 for an illustration): the distance to the closest point not on ℓ, the length of the line segment perpendicular to ℓ at (x_i, y_i) ending at the line rotated counterclockwise about (x_j, y_j) that first hits a point, and the length of the line segment perpendicular to ℓ at (x_j, y_j) ending at the clockwise rotated line about (x_i, y_i) that first hits a point. Choose ϵ_c to be a fraction of the smallest ϵ (let the fraction be $\frac{1}{M}$ for some large integer M). We use this ϵ_c to construct all the *center pairs.*

The purpose of the grid, which is forced to be part of any shattering of size k_{ps}, is to separate the center pairs from each other and leave only the center pairs unshattered. By constructing additional points to the left and the right of the center pairs, we induce the grid consisting of horizontal and vertical lines. (To simplify this proof, we call horizontal[vertical] any line homeomorphic to a horizontal[vertical] line.)

The main portion of the grid is composed of horizontal lines which separate neighboring center pairs. The remainder of the grid is composed of two horizontal lines above, two

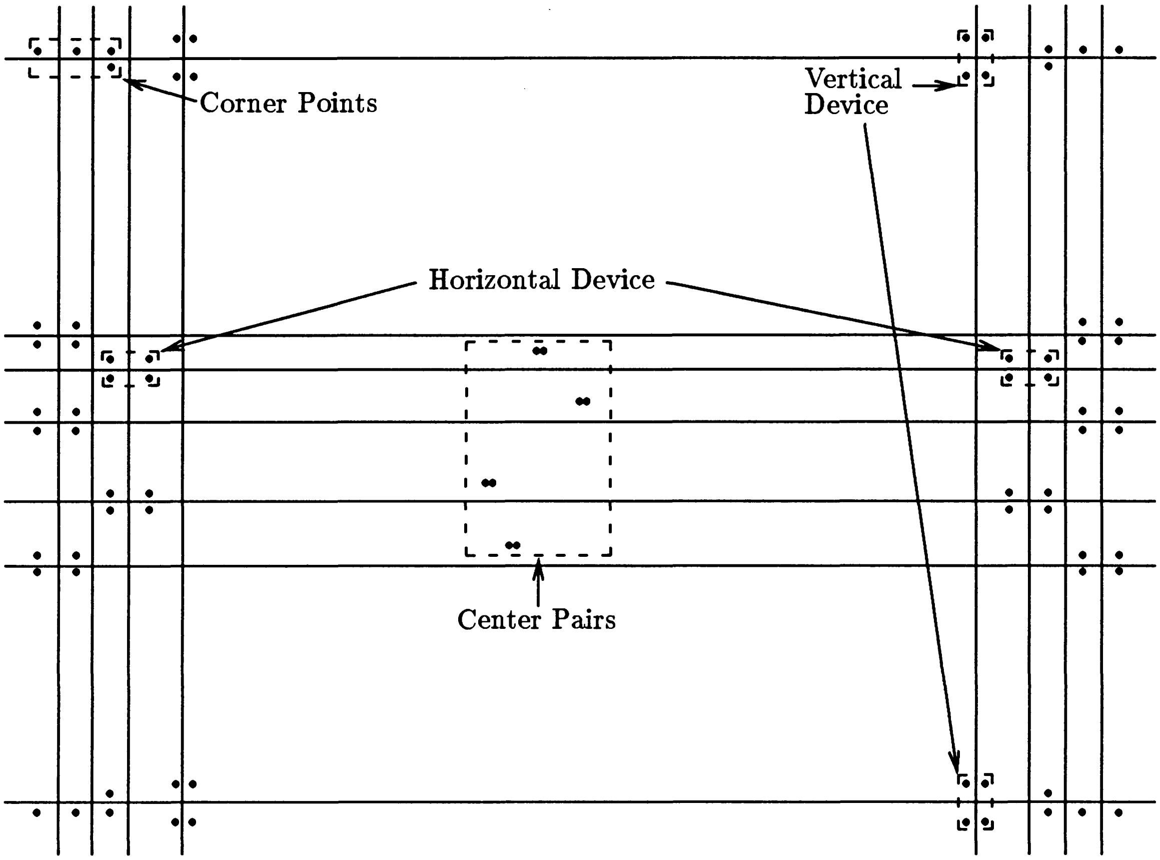


Figure 2: A typical instance of PS, along with the associated grid. Each center pair corresponds to a point in the related instance of PC.

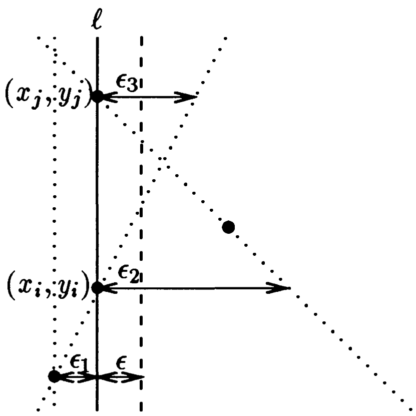


Figure 3: ϵ_1 is the distance to the closest point not on ℓ. $\epsilon_2[\epsilon_3]$ is the $\perp$-distance between ℓ at $(x_i, y_i)[(x_j, y_j)]$ and the line rotated counterclockwise[clockwise] about $(x_j, y_j)[(x_i, y_i)]$ that first hits a point not on ℓ. ϵ is chosen to be the smallest ϵ_i.

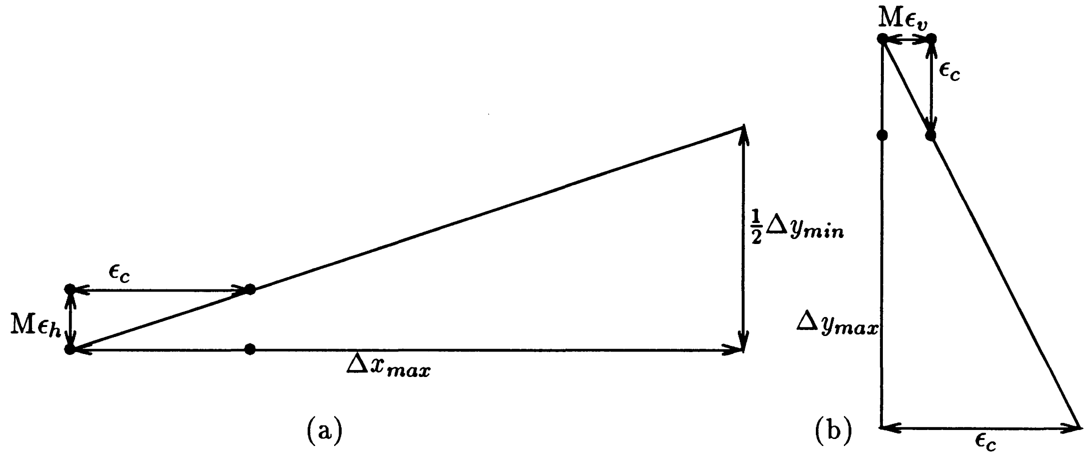


Figure 4: (a) The derivation of ϵ_h. (b) The derivation of ϵ_v. In each figure, no point lies within the large triangle, so ϵ_h and ϵ_v can be derived by equating ratios.

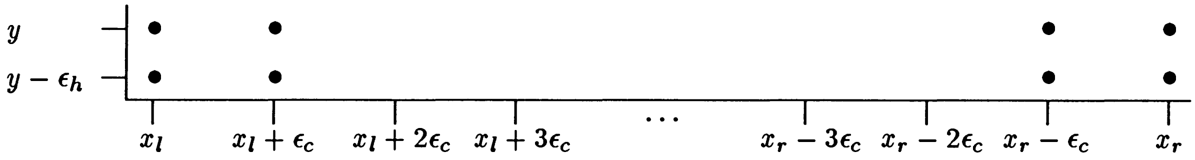
---

horizontal lines below, and four vertical lines on each side of the center pairs. The points inducing the grid are constructed as follows:

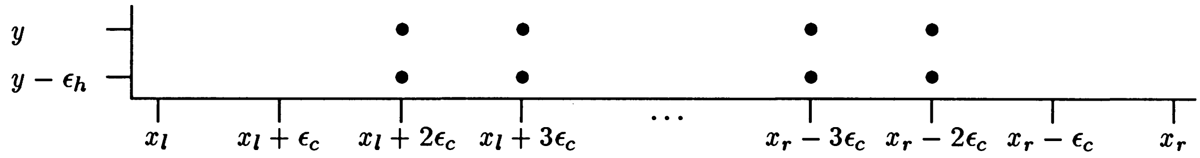
The top and bottom y-coordinates used are $y_t = y_n + 2(y_n - y_1)$ and $y_b = y_1 - 2(y_n - y_1)$ respectively. Let Δy_{min} be the minimum difference in y-coordinates, and $\Delta y_{max} = y_t - y_b$. The leftmost and rightmost x-coordinates used are x_l and x_r, which are finalized at the conclusion of the grid's definition. Let Δx_{max} be a number greater than $x_l - x_r$. (If the initial guess is too small, then Δx_{max} can be revised once x_l and x_r are known, and the grid recomputed. Since the grid changes only slightly, this procedure does not have to be repeated.)

Devices, constructed from four pairs, are used to induce most of the vertical and horizontal lines. The remaining lines in the grid are induced by the placement of the devices and additional corner points. The pair spacings used for horizontal and vertical devices are $\epsilon_h = \frac{1}{2M}\epsilon_c \Delta y_{min}/\Delta x_{max}$ and $\epsilon_v = \frac{1}{M}{\epsilon_c}^2/\Delta y_{max}$. We choose ϵ_h so that any line horizontally shattering two pairs of a horizontal device cannot shatter any other pair except the remaining pairs of the device (see Figure 4a). We choose ϵ_v similarly for vertical pairs (see Figure 4b).

For $i = 1, \ldots, n$ we build a horizontal device to force a horizontal line below the i^{th} pair in the shattering of size k_{ps}. The device is constructed from four vertical pairs forming a rectangle on each side of the center pairs. The rectangles have dimension ϵ_c by ϵ_h. The devices are alternated between two positions so that no extra lines are required to horizontally separate the devices. Their vertical alignment induces the three outermost vertical lines on each side. Let $y = y_i - \frac{1}{2}(\Delta y_{min} - \epsilon_h)$. If i is odd, then the device is:

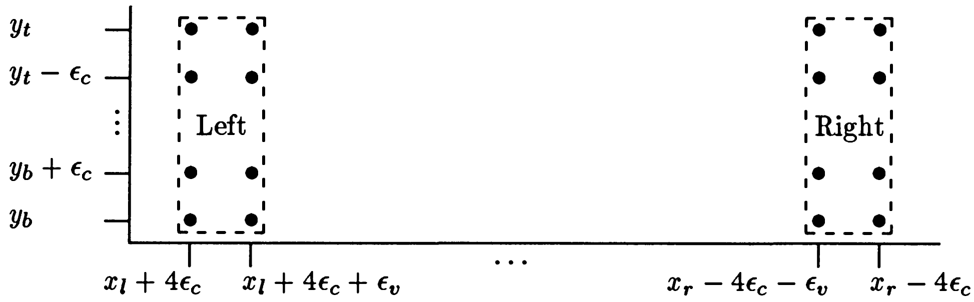


Otherwise, the device is:

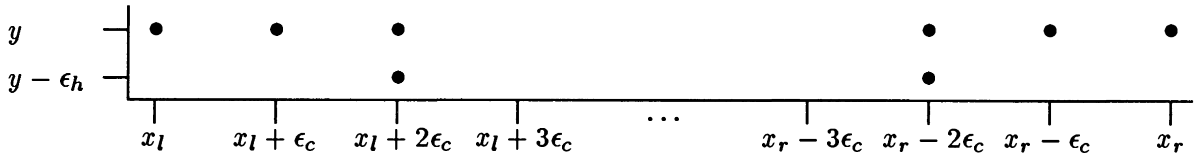


A horizontal device is built for $y = y_n + \frac{1}{2}(\Delta y_{min} + \epsilon_h)$ using the odd position, to force a horizontal line above all the center pairs.

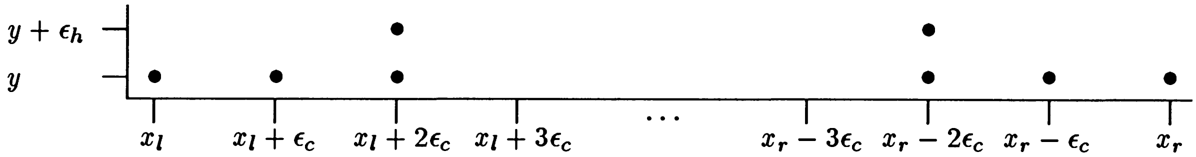
To separate the horizontal devices from the center pairs, vertical devices are constructed on both sides of the center pairs. Each vertical device is formed from two rectangles placed above and below the outermost horizontal lines previously defined. The rectangles have dimension ϵ_v by ϵ_c. The left and right devices are:



The definition of the grid is completed by four points in each corner. Letting $y = y_t - \frac{1}{2}(\epsilon_c - \epsilon_h)$, the upper points are:



Letting $y = y_b + \frac{1}{2}(\epsilon_c + \epsilon_h)$, the lower points are:



The corner points along with the vertical devices act like two horizontal devices to induce the two extreme horizontal lines. On each side of the center pairs, the two stacks of horizontal device rectangles and corner points induce three lines that vertically shatter the stacks. The stacks are placed so that a line vertically shattering the stacks cannot shatter other pairs. Without loss of generality, we assume that the lines used to shatter the stacks in any shattering of size k_{ps} are vertical, since if two or more nearly-vertical lines are used, they can be replaced by an equal number of vertical lines, and the resulting set of lines is still a shattering of size k_{ps}.

We choose x_l to be small enough so that all the left-side grid points are in a single cell of the arrangement formed the lines $y = y_t$ and $y = y_b$, and the internal tangents between every two center pairs. In addition, x_l is small enough so that all the center pairs are in a single cell of the arrangement formed by lines defined by two points chosen from different left-side devices.

Similarly, x_r is chosen to be large enough so all the right-side grid points are in a single cell of each of the arrangements defined above. In addition, x_r is chosen so that all the center pairs and the left-side grid points are in a single cell of the arrangement formed by lines defined by two points chosen from different right-side devices.

This completes the definition of the grid points. We still have to show that a shattering of size k_{ps} of the constructed points contains the grid and that the other lines in the shattering correspond to the lines of the related minimum covering. Since only a linear number of points are used and the arrangements in the definitions of x_l and x_r have quadratic size, we will clearly have a polynomial time reduction of PC to PS.

It is easy to check that any shattering of just the grid points requires at least $n + 11$ lines; $n + 11$ lines are also sufficient, since the $n + 3$ horizontal and eight vertical lines previously described are a shattering. The grid previously described is the only shattering, up to homeomorphism and replacement of nearly-vertical lines, by $n+11$ lines. To see this, suppose there is a shattering of size $n + 11$ that is not the described grid. Let ℓ be a grid line that is not present in the new shattering. The pairs that would have been shattered by ℓ must be shattered by two or more lines. The replacement lines must also shatter pairs not shattered by ℓ or they would not be part of the shattering. Each of the replacement lines can only shatter one pair shattered by ℓ and one other pair, by the placement of the grid points. Since each line in the grid shatters at least four pairs, the new shattering must have more than $n + 11$ lines, which is a contradiction.

By a similar argument, the grid must be part of any shattering of size $n + 11 + k_{pc}$ for the constructed instance of PS. The grid cannot shatter the center pairs, and it completely shatters the grid points, so the remaining k_{pc} lines in the shattering serve only to shatter the center pairs. By the definition of the center pairs, each non-grid shattering line ℓ corresponds to a covering line defined by all the left points of pairs shattered by ℓ. This correspondence allows us to recover the related covering for the instance of PC. ∎

The above reduction produces an instance of PS that is not in general position, since there are many horizontal and vertical lines that each pass through more than two points. In addition, the points from the instance of PC may not be in general position. Both of these problems can be eliminated by carefully moving all the points so that they are in general position without changing the resulting grid. However, the details of the correctness proof become rather complex.

The following Corollary follows directly from Theorem 2.

Corollary 3: The Point Shattering Problem in $\Re^d$, $d \geq 2$ is *NP*-Complete.

We now show that if we restrict the shattering lines to be horizontal or vertical, the problem remains *NP*-Complete. The proof is similar to Hassin and Megiddo's *NP*-Completeness proof[HM88] for the problem of stabbing unit horizontal segments with horizontal and vertical lines; the approach was suggested to us by Nimrod Megiddo. Formally, the problem is:

Problem Horizontal–Vertical Point Shattering (HVPS): Given a set of rational points $(x_1, y_1), \ldots, (x_n, y_n)$ in $\Re^2$ and an integer k, does there exist a set of k horizontal and vertical lines that shatter the n points?

Theorem 4: The Horizontal–Vertical Point Shattering problem in $\Re^2$ is *NP*-Complete.

Proof: One easily checks that HVPS is in *NP*. The remainder of the proof is a reduction from a version of 3-SAT with each clause containing three distinct variables, which is *NP*-Complete (see [GJ79]). Let $\{C_1, C_2, \ldots, C_m\}$ be the 3-SAT clauses, with $C_i = x_i \vee y_i \vee z_i$ and $x_i, y_i, z_i \in \{u_1, \bar{u}_1, \ldots, u_n, \bar{u}_n\}$. From this instance of 3-SAT, we construct an instance of HVPS.

For each variable u_j, we build a twelve-point device (see Figure 5a). If the grid lines shown in Figure 5b are included, then it is easy to check that at least three additional lines must be used to shatter the device. Furthermore, there are only two possible sets of three lines that shatter the device; each set contains one horizontal line and two vertical lines. The choice of the horizontal line corresponds to the truth assignment for u_j. The variable devices are placed along a diagonal so that a vertical or horizontal line can intersect only one of them.

For each clause $x_i \vee y_i \vee z_i$, we build a nine-point device with the horizontal lines corresponding to the clause's three variables passing through it (see Figure 5c). When the grid line shown is included, it is easy to check that at least three additional lines must be used to shatter the device, and if only three lines are used, then one of the lines must be horizontal. The clause devices are placed so that a vertical line can intersect only one of them.

In addition to the devices, another $8n + 2m - 1$ points are included to force certain grid lines to be present in the shattering. The grid lines separate the devices from each other, and perform some shattering within the devices. In addition to the lines required to shatter the devices, $6n + m + 1$ grid lines are required.

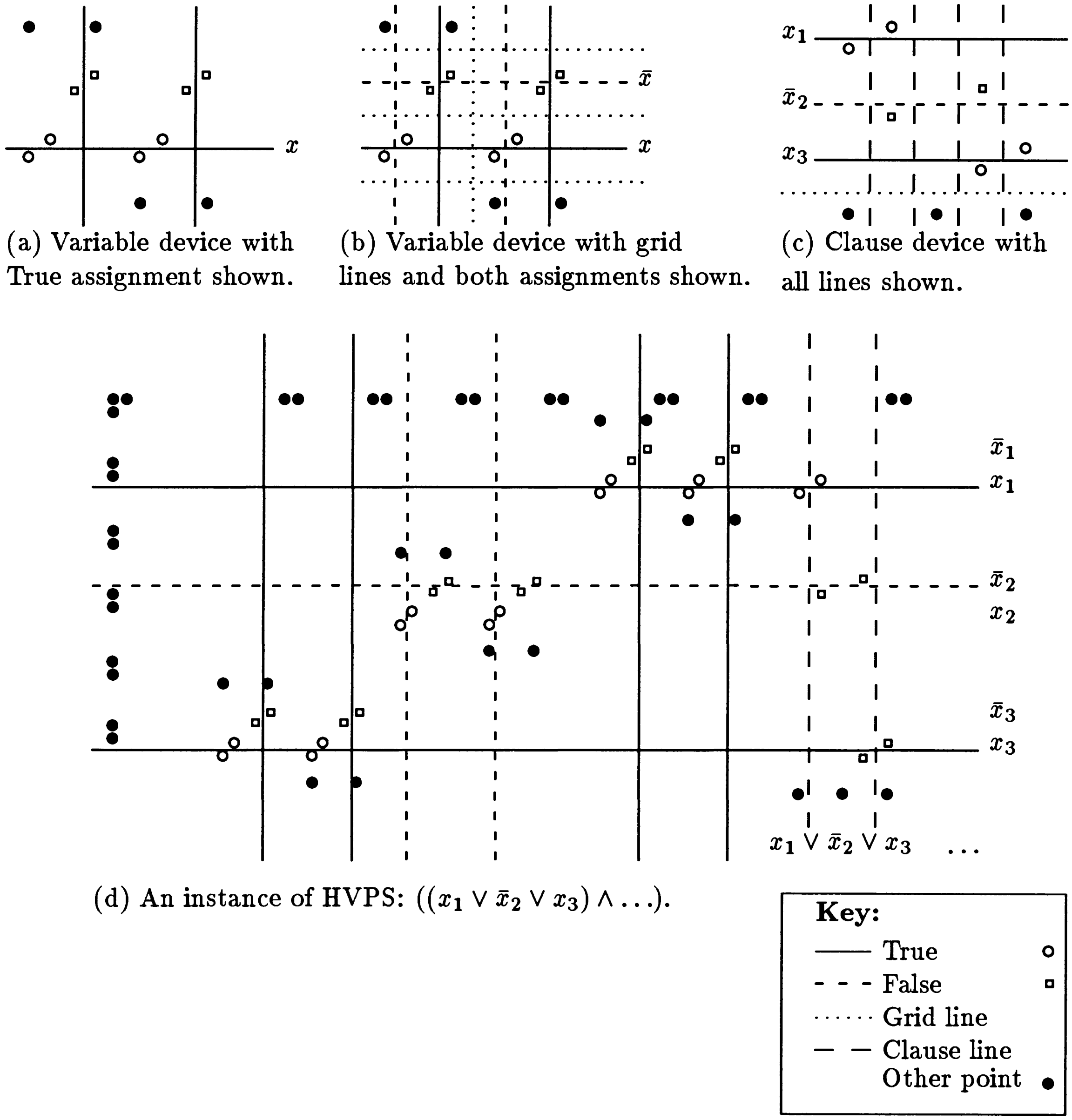
It is easy to check that there is a shattering with $9n + 3m + 1$ lines if and only if the set of clauses are satisfiable. Figure 5d shows the instance $((x_1 \vee \bar{x}_2 \vee x_3) \wedge \ldots)$. ∎

Corollary 5: The following problem is *NP*-Complete: Given a set of n rational points in $\Re^d$, $d \geq 2$ and an integer k, does there exist a set of k hyperplanes, each perpendicular to either the x or y axes, that shatter the points?

We conclude this section by observing that the following variant of point shattering is also *NP*-Complete.

Corollary 6: The following problem is *NP*-Complete: Given a set of n rational points in $\Re^d$, $d \geq 2$ and integers $k, r \geq 2$, does there exist a set of k hyperplanes, whose normals are chosen from r directions, that shatter the points?

Proof: For $r = 2$, the claim follows by a reduction from HVPS, using a device in a corner to force the horizontal and vertical directions to be selected. For $r > 2$, we can expand the device to force $r - 2$ useless directions to be chosen. ∎



(a) Variable device with True assignment shown.

(b) Variable device with grid lines and both assignments shown.

(c) Clause device with all lines shown.

(d) An instance of HVPS: $((x_1 \vee \bar{x}_2 \vee x_3) \wedge \ldots)$.

Figure 5: Construction for the proof of Theorem 4.

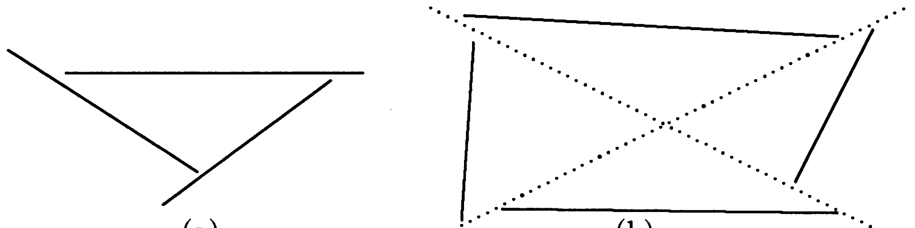


Figure 6: (a) Three line segments that cannot be shattered. (b) Four shatterable line segments that cannot be cut off one by one.

3 Shattering Polygonal Objects in $\Re^2$

The *NP*-Completeness results in the previous section indicate that it is unlikely that efficient algorithms exist to find a minimum-cardinality shattering for a set of polygonal objects in $\Re^2$. However, it is possible in polynomial-time to find a *minimal* size shattering, if the objects are shatterable (here, minimal means that if any line is deleted the objects are no longer shattered). Unlike points, some sets of polygonal objects cannot be shattered by an arrangement of lines (see Figure 6a). Notice 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 (see Figure 6b). In this section, we present two algorithms that determine if a set of polygonal objects is shatterable by an arrangement of lines, and if so, produce a "small" collection of shattering lines. The notion of "small" is made precise in the following lemma.

Lemma 7: 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.

Proof: Let $\mathcal{H}$ be a collection of k hyperplanes that shatter $\mathcal{O}$. If $k \leq n-1$, we are done. If $k \geq n$, we show that there is a subset of $\mathcal{H}$ with at most $n-1$ hyperplanes that suffices to shatter $\mathcal{O}$.

The proof is by induction on n. The case of $n = 1$ is trivial. Assume that for any set of n or fewer objects, for any set of hyperplanes that shatters the objects, there is a subset of hyperplanes of size $n-1$ or less that also shatters the objects. Now consider a set $\mathcal{O}$ of $n+1$ objects and a set $\mathcal{H}$ of hyperplanes that shatter $\mathcal{O}$. Let $h \in \mathcal{H}$ be a hyperplane such that not all objects of $\mathcal{O}$ are on the same side of ℓ. Thus, ℓ separates $\mathcal{O}$ into two sets $\mathcal{O}_1$ and $\mathcal{O}_2$, of cardinalities $n_1 \geq 1$ and $n_2 \geq 1$, with $n_1 + n_2 = n+1$. Clearly, $\mathcal{H}$ shatters each of $\mathcal{O}_1$ and $\mathcal{O}_2$. By the induction hypothesis, there is a subset of $\mathcal{H}$ of size at most $n_1 - 1[n_2 - 1]$ that shatters $\mathcal{O}_1[\mathcal{O}_2]$. These two subsets, together with hyperplane h, form an arrangement of at most $1 + (n_1 - 1) + (n_2 - 1) = n$ hyperplanes that shatter set $\mathcal{O}$. ∎

Assume that we are given a set $\mathcal{O}$ of n simple polygons in the plane, with a total of N vertices. We describe two algorithms to decide if $\mathcal{O}$ can be shattered by an arrangement of lines, and if so produce a set of $n-1$ or fewer shattering lines. The first algorithm runs in time $O(E + N \log N + n^2 \log n)$, where E is the number of edges in the visibility graph

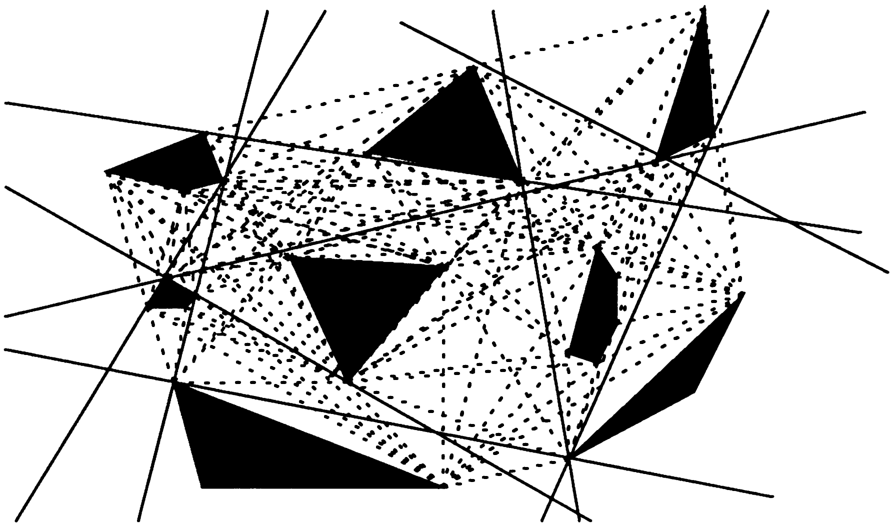


Figure 7: A set of objects with their visibility graph and set of candidate lines shown.

of $\mathcal{O}$. The second algorithm runs in time $O(E + N \log N + C \log n + nC^{.695})$, where C is the number of candidate lines considered (at worst $\min\{E, n^2\}$). The second algorithm is slower than the first when n and E are large (e.g., E may be quadratic in N), but has the advantage that it generalizes to higher dimensions. In Section 5, we present a less efficient algorithm that finds a shattering whose size is within a log factor of optimal.

Both algorithms begin by replacing each object with its convex hull, since a reflex vertex cannot intersect a shattering line. All of the convex hulls can be determined in $O(N)$ total time (see [PS85]). Next, a simple plane-sweep algorithm is used to 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 and convex.

The algorithms continue by computing the visibility graph (VG) of $\mathcal{O}$ in time $O(E + N \log N)$, using Ghosh and Mount's algorithm[GM87]. We assume that the endpoints of each VG edge come from different objects. As a by-product of building VG, we can, within the same time bounds, identify the set $\mathcal{C}$ of all candidate shattering lines (let $C = |\mathcal{C}|$). The candidate lines are those that contain VG edges (not on the convex hull of $\mathcal{O}$) such that no object's interior is intersected by the line (see Figure 7 for an example). They are further restricted to those lines that are "pinned" such that no counterclockwise rotation is possible without intersecting an object. Notice that each candidate line is the counterclockwise internal tangent between two objects. This implies that $C \leq \min\{E, n^2\}$. Observe that in our example, C is much smaller than $\min\{E, n^2\}$. If the set $\mathcal{O}$ is shatterable, then it is shattered by the arrangement of lines in $\mathcal{C}$. 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 first algorithm incrementally builds a set $\mathcal{P}$ of "productive" candidate lines that will be used in our attempt to shatter $\mathcal{O}$. At any given stage during the execution, the arrangement of lines currently in $\mathcal{P}$ partially shatters $\mathcal{O}$. 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 $\mathcal{O}$). We build $\mathcal{P}$ by considering the candidate lines in order of increasing slope. We add a candidate line ℓ to $\mathcal{P}$ if it intersects the convex hull of at least one cluster. Each cluster broken by ℓ is replaced by any resulting smaller clusters. If no clusters remain after processing ℓ, then the algorithm terminates, since $\mathcal{O}$ is shattered by the arrangement of lines in $\mathcal{P}$. If at least one cluster remains after all the lines of $\mathcal{C}$ have been considered, then $\mathcal{O}$ is not shatterable.

We determine if a candidate line, with angle θ from the negative y-axis, intersects the convex hull of a cluster by consulting the *visibility profile* (VP(θ)) of the convex hulls of the current clusters. VP(θ) consists of non-overlapping intervals in sorted order corresponding to the portions of convex hulls visible from infinity, looking in direction θ. Our goal is to efficiently maintain VP(θ) as θ varies over the range $[0, \pi)$. Observe that combinatorial changes in VP(θ) can only occur at θ corresponding to edges of $\mathcal{E}$, the set of VG edges that are tangents between two objects of $\mathcal{O}$. Also, note that the VG edges which define candidate lines are included in $\mathcal{E}$. So, our radial sweep only needs to stop at critical directions θ corresponding to edges of $\mathcal{E}$ to maintain VP(θ) and process $\mathcal{C}$.

The algorithm begins by sorting the edges of $\mathcal{E}$ by increasing angle with the negative y-axis (in time $O(\min\{E, n^2\} \log n)$). Let $\theta = 0$ be the initial "vertical" direction. Initially, VP(θ) is trivial, since there is only one cluster $\mathcal{O}$. Each cluster of objects is represented by its induced subgraph of VG. We also store the extreme left and right points of each cluster, with respect to θ.

As we rotate the vertical direction θ, critical changes occur only at directions corresponding to edges of $\mathcal{E}$. At each such critical direction, we may have a combinatorial change in the current VP, or we may discover a new productive candidate line.

For each VG edge e encountered, we check to see if VP(θ) changes combinatorially. A change occurs when e corresponds to a tangent between two visible clusters (in this case a simple constant-size update is performed on VP(θ)). A cluster is considered to be visible if it appears in VP(θ) either just before or just after the current θ. Since only constant-size changes occur at each such critical direction, the total cost of all updates is only $O(\min\{E, n^2\})$. The extreme left and right points of each cluster are updated by marching along the convex hull of the cluster in the direction of rotation. Since any vertex can only be visited once by a march during the radial sweep, the total cost of all updates is $O(N)$.

When we encounter a VG edge e that corresponds to a candidate line ℓ (see Figure 8), we consult (using a binary search) VP(θ) to detect if ℓ stabs at least one cluster (in query time $O(\log n)$). If it does, then we add ℓ to $\mathcal{P}$, and we update VP(θ), breaking all clusters stabbed by ℓ. We determine, in linear time, the set $\mathcal{S}$ of stabbed clusters by checking if ℓ stabs the segment joining the extreme left and right points of each cluster. The objects of each cluster in $\mathcal{S}$ are separated by ℓ into two groups. Each group containing two or more objects forms a new cluster. For every cluster c of $\mathcal{S}$, we determine the new clusters by splitting c's induced subgraph of VG. We accomplish this by deleting all of c's VG edges that cross ℓ, which can be identified in time proportional to the number of such edges, using Mitchell and Welzl's method[MW90] to update a VG when adding a new obstacle (here, think of the candidate line as the new "obstacle"). We then use a plane sweep to update VP(θ) 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 does not

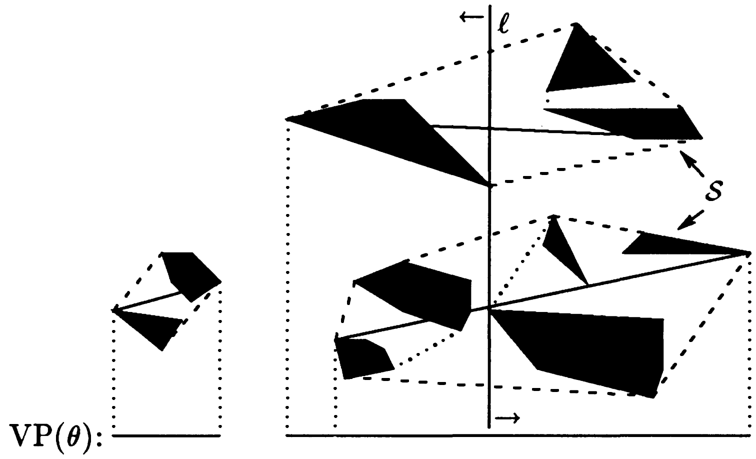


Figure 8: Candidate line ℓ shatters the two clusters in $\mathcal{S}$. The current VP(θ) is shown at the bottom. We have also drawn line segments joining the extreme left and right points of each cluster.

---

stab a cluster, then it is not productive in refining the current partial shattering.

If the set of clusters becomes empty during our sweep in θ, we stop and conclude that $\mathcal{O}$ is shatterable by the current set $\mathcal{P}$ of productive lines ($|\mathcal{P}| \leq n-1$). Otherwise, if we conclude the sweep in θ and still have one or more clusters of unshattered objects, we conclude that $\mathcal{O}$ is not shatterable. However, notice that $\mathcal{P}$ achieves the maximum possible partial shattering. The total time complexity of this algorithm is $O(E+N\log N+n^2\log n)$.

Our second algorithm differs in the method by which it selects $\mathcal{P}$; it utilizes a divide-and-conquer approach to select productive lines that stab VG edges. For a given VG edge, a line that stabs the edge is determined by consulting the set $\mathcal{C}^*$ of points that are dual (see [CGL83]) to the lines in $\mathcal{C}$. We preprocess $\mathcal{C}^*$ 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. This preprocessing can be done in time $O(P(C))$, yielding a data structure of size $O(C)$ that answers each query in time $O(Q(C))$, where $P(C)$ and $Q(C)$ depend on the method chosen to answer the triangle range queries.

Let e be a VG edge on the convex hull of $\mathcal{O}$ (thus, e does not generate a candidate line). We now call the function $\mathcal{P} = SPLIT(VG, e)$ (see Figure 9 for an example). $SPLIT$ returns with failure if $\mathcal{O}$ is not shatterable.

$SPLIT(G, e)$:

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 $\mathcal{C}^*$ within the double wedge. This step takes time $O(Q(C))$. If no candidate line stabs e, then $\mathcal{O}$ is not shatterable, and $SPLIT$ returns with failure.
2. Remove all edges of G stabbed by ℓ: This can be done in time proportional to the number of edges stabbed, using Mitchell and Welzl's results[MW90] on maintaining visibility graphs under insertions of new obstacles. G is now split into two pieces G_1

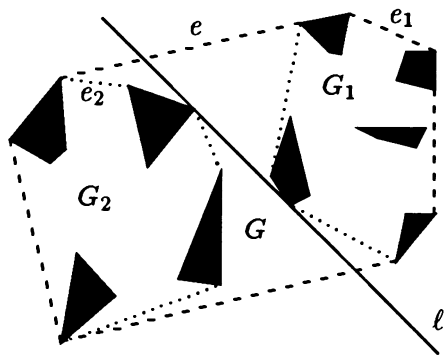


Figure 9: An example for Algorithm 2.

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 $\mathcal{P}_i = SPLIT(G_i, e_i)$. Otherwise, let $\mathcal{P}_i = \emptyset$. Return($\{\ell\} \cup \mathcal{P}_1 \cup \mathcal{P}_2$).

If $SPLIT$ does not return with failure, then the algorithm produces a shattering (by the following lemma), since the set of lines output is constructed in such a way that every visibility graph edge is cut.

Lemma 8: Let $\mathcal{O}$ be a set of disjoint objects in $\Re^d$, and $\mathcal{H}$ be a set of hyperplanes. The hyperplanes in $\mathcal{H}$ stab every edge of the visibility graph (VG) of $\mathcal{O}$ if and only if the arrangement $\mathcal{A}(\mathcal{H})$ of the hyperplanes in $\mathcal{H}$ shatters $\mathcal{O}$.

Proof: Suppose $\mathcal{A}(\mathcal{H})$ does not shatter $\mathcal{O}$. This implies there exists a cell C of $\mathcal{A}(\mathcal{H})$ with at least two objects, and none of the objects in C share any VG edges. Consider the convex hull of all objects in C, and let e be a hull edge that joins two objects. The endpoints of e are vertices, so e must be in VG. This is a contradiction, therefore $\mathcal{A}(\mathcal{H})$ must shatter $\mathcal{O}$. The opposite direction is trivial. ∎

Each call to $SPLIT$ requires $O(Q(C))$ 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 one new productive shattering line. Thus, the algorithm requires total time $O(E+N\log N+P(C)+nQ(C))$.

Currently, the best method for triangle range queries that supports reporting a point using only $P(C) = O(C\log C) = O(C\log n)$ preprocessing is due to Edelsbrunner and Welzl[EW86,Ede87] and achieves $Q(C) = (C^{.695})$. This yields an $O(E+N\log N+C\log n+nC^{.695})$ time bound for our algorithm. A lower bound by Chazelle[Cha87] shows that $Q(C) = \Omega(\sqrt{C})$, when only $P(C) = O(C\log C)$ preprocessing and $O(C)$ space is allowed. If a comparable upper bound is ever achieved, then our second algorithm would run in time $O(E + N\log N + C\log n + n\sqrt{C})$, which is superior to the first algorithm. Although currently the worst-case running time of our second algorithm is inferior to our first, it has the advantage of generalizing to higher dimensions (see Section 4).

We summarize with the following theorem:

Theorem 9: 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 a minimal set of lines of size at most $n-1$ in time $O(E+N\log N+n^2+\min\{n^2\log n, C\log n+nC^{.695}\})$, where E is the size of the visibility graph induced by $\mathcal{O}$, and C is the number of candidate lines considered. The space required by the algorithm is $O(E)$.

Proof: Build the set $\mathcal{C}$ of candidate lines. Choose the more efficient algorithm based on C, and use it to construct a set $\mathcal{P}$ of $n-1$ or fewer shattering lines. $\mathcal{P}$ is reduced to a minimal set of shattering lines by building the arrangement of lines in $\mathcal{P}$ in $O(n^2)$ time (see [Ede87]). For each object in $\mathcal{O}$, locate the cell in the arrangement that contains it and label the cell occupied ($O(n\log n)$ total time). For each shattering line ℓ, search the zone of ℓ for a pair of occupied cells that share an edge that is a subset of ℓ ($O(n)$ time per line). If no such pair exists, then ℓ is redundant. In this case, ℓ is deleted from $\mathcal{P}$, and the arrangement is updated ($O(n)$ time per update). At the conclusion of this procedure, every remaining line in $\mathcal{P}$ is necessary. ∎

4 Shattering Bounded Polyhedra in $\Re^d$

The two shattering algorithms in the previous section handle polygonal objects in the plane. In this section, we generalize the divide-and-conquer algorithm to determine if a set $\mathcal{O}$ of n bounded polyhedra in $\Re^d$, $d \geq 3$ can be shattered by an arrangement of hyperplanes, and if so produce a set of $n-1$ or fewer shattering hyperplanes ($n-1$ hyperplanes are always sufficient by Lemma 7). Again, we let N be the total number of vertices.

We replace each object with its convex hull, since the set of shattering planes defines an arrangement with convex cells. The convex hulls can be determined in $O(N^{\lfloor (d+1)/2 \rfloor})$ time (see [Ede87]). Since overlapping objects cannot be shattered, we next check for intersection among the objects; in $\Re^3$ this is done in time $O(N\log N)$ by a space sweep (see [PS85]), and, in general, can be done in time $O(N^{\lfloor d/2 \rfloor +1})$. Thus, we assume from now on that all the objects are disjoint convex polytopes.

Define the visibility graph, $VG(\mathcal{O})$, to be the graph with the vertices of $\mathcal{O}$, and edges between vertices of different objects that can see each other. We compute $VG(\mathcal{O})$ by brute force in $O(N^{\lfloor d/2 \rfloor +2})$ time. ($VG(\mathcal{O})$ has at most $O(N^2)$ edges).

We build a set $\mathcal{C}$ of candidate hyperplanes consisting of all hyperplanes that lie in free space and are tangent to d vertices of $\mathcal{O}$ ($|\mathcal{C}| = O(n^d)$). Observe that if $\mathcal{O}$ is shatterable, then the arrangement of hyperplanes in $\mathcal{C}$ shatters $\mathcal{O}$. We determine $\mathcal{C}$ by starting with the set $\mathcal{H}$ of all hyperplanes defined by d mutually visible vertices. We assume that the edges of VG have been labeled using a total ordering. For each VG edge e, we select all the hyperplanes in $\mathcal{H}$ that lie in free space and are defined by e and $(d-2)$ other VG edges, each with a larger label than e. The hyperplanes are selected by calling the function $\mathcal{C}_e = SELECT(\mathcal{O}, e, VG, d)$. Let $\mathcal{C} = \bigcup_e \mathcal{C}_e$. Observe that $|\mathcal{C}| = O(N^d)$, since we do not generate any duplicate hyperplanes. Each top-level call to $SELECT$ requires $O(N^{d-2}\log N)$ time, so the overall time complexity is $O(N^d \log N)$.

$SELECT(\mathcal{X}, (v_1, v_2), G, k)$:

1. Project the objects of $\mathcal{X}$ in $\Re^k$ onto a hyperplane π perpendicular to the line ℓ containing the edge (v_1, v_2) ($O(N^{\lfloor k/2 \rfloor})$ time). The edge (v_1, v_2) projects onto a point p. Let $\mathcal{X}^P$ be the set of projected objects. If any member of $\mathcal{X}^P$ contains p (this can be checked in $O(N^{\lfloor (k-1)/2 \rfloor})$ time), then ℓ is not feasible; Return($\emptyset$). Let G^P be the projected visibility graph obtained from G, with the restriction that an edge $(p, v) \in \mathrm{G}^P$ only if both (v_1, v) and $(v_2, v) \in \mathrm{G}$ (this assures that v, v_1, v_2 are mutually visible). For such edges, keep the smaller of the two labels.
2. If $k = 3$, then for each object $X_i^P \in \mathcal{X}^P$, determine the set $\mathcal{A}_i$ of angles for which a line through p avoids X_i^P ($O(N)$ time). The set $\mathcal{A}$ of angles for which p can see infinity in both directions is given by $\bigcap_i \mathcal{A}_i$ ($O(N \log N)$ time). Each edge $(p, v) \in \mathrm{G}^P$ defines a line ℓ_v through p. Sort the lines by their angle ($O(N \log N)$ time). Let $\mathcal{S}^P$ be the set of lines ℓ_v whose angle is in $\mathcal{A}$ ($O(N)$ time). Observe that $|\mathcal{S}| = O(N)$. This step takes $O(N \log N)$ time.
3. If $k \geq 4$, then for each edge $(p, v) \in \mathrm{G}^P$ with a larger label than e, determine any feasible hyperplanes that it defines by calling $\mathcal{S}_e^P = SELECT(\mathcal{X}^P, (p, v), \mathrm{G}^P, k-1)$ ($O(N^{k-2}\log N)$ total time). Let $\mathcal{S}^P = \bigcup_e \mathcal{S}_e^P$. Observe that $|\mathcal{S}| = O(N^{k-2})$. This step takes $O(N^{k-2}\log N)$ time.
4. Construct $\mathcal{S}$ by sweeping every hyperplane in $\mathcal{S}^P$ along ℓ, and adding the resulting hyperplanes to $\mathcal{S}$ ($O(N^{k-2})$ time). Return($\mathcal{S}$).

Now that $\mathcal{C}$ has been constructed, let e be a VG edge on the convex hull of $\mathcal{O}$. We call the function $\mathcal{P} = SPLITD(VG, e)$ to determine a set $\mathcal{P}$ of $n-1$ or fewer hyperplanes that shatter $\mathcal{O}$. $SPLITD$ returns with failure if $\mathcal{O}$ is not shatterable.

$SPLITD(G, e)$:

1. Find a candidate hyperplane h that stabs e, by checking each hyperplane in $\mathcal{C}$ ($O(N^d)$ time). If no candidate hyperplane stabs e, then $\mathcal{O}$ is not shatterable, and $SPLITD$ returns with failure.
2. Remove all edges of G stabbed by h, by checking each edge ($O(N^2)$ time). 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 $\mathcal{P}_i = SPLITD(G_i, e_i)$. Otherwise, let $\mathcal{P}_i = \emptyset$. Return($\{h\} \cup \mathcal{P}_1 \cup \mathcal{P}_2$).

If $SPLITD$ does not return with failure, then the algorithm produces a shattering (by Lemma 8), since $\mathcal{P}$ is constructed in such a way that every visibility graph edge is cut. Each call to $SPLITD$ requires $O(N^d)$ time. There can be at most $n-1$ calls to $SPLIT$, since each call yields one new productive shattering hyperplane. So, $O(nN^d)$ time is spent to determine $\mathcal{P}$.

We reduce $\mathcal{P}$ to a minimal set of shattering hyperplanes in $O(n^d)$ time, by building the arrangement of hyperplanes in $\mathcal{P}$ in time $O(n^d)$ (see [Ede87]). For each object in $\mathcal{O}$, locate the cell in the arrangement that contains it and label the cell occupied ($O(dn)$ total time).

For each hyperplane h of $\mathcal{P}$, search the zone of h for a pair of occupied cells that share a $(d-1)$-face that is a subset of h (there are $O(n^{d-1})$ such pairs, so $O(n^{d-1})$ time is required per hyperplane). If no such pair exists, then h is redundant. In this case, h is deleted from $\mathcal{P}$, and the arrangement is updated ($O(n^{d-1})$ time per update). At the conclusion of this procedure, every remaining hyperplane in $\mathcal{P}$ is necessary. We summarize with the following theorem:

Theorem 10: Given a set $\mathcal{O}$ of n bounded polyhedral polygonal objects in $\Re^d$, $d \geq 3$, with a total of N vertices, one can detect if $\mathcal{O}$ can be shattered by a set of hyperplanes and, if so, output a minimal set of hyperplanes of size at most $n-1$ in time $O(N^d(n+\log N))$.

5 Approximation Algorithm

The theorems of Section 2 show that finding a minimum-cardinality shattering (if one exists) for a set of objects in $\Re^d$ is *NP*-Complete. Using Lemma 8, we will reformulate this problem as a minimum set cover problem: Given a collection $\mathcal{X}$ of subsets of a finite set $\mathcal{S}$ and an integer $k \leq |\mathcal{X}|$, does there exist a subset $\mathcal{X}' \subseteq \mathcal{X}$ such that $\mathcal{S}$ equals the union of the members of $\mathcal{X}'$ and $|\mathcal{X}'| \leq k$? While the set cover problem is *NP*-Complete (see [GJ79]), both Johnson[Joh74] and Lovász[Lov75] showed that the natural greedy heuristic achieves a set cover whose size is at worst $(1 + \log |\mathcal{S}|)$ times the minimum. If $\mathcal{O}$ is a set of n disjoint polyhedral objects with N vertices, then let $\mathcal{S} = \{\text{edges of } VG(\mathcal{O})\}$, and $\mathcal{X} = \{\{\text{edges of } \mathcal{S} \text{ stabbed by } h\} | h \text{ is a candidate hyperplane}\}$. Let $M(\mathcal{O})$ be the size of the minimum-cardinality shattering of $\mathcal{O}$. Hence, the following greedy procedure produces a set $\mathcal{P}$ of $(1 + \log N)M(\mathcal{O})$ or fewer hyperplanes whose arrangement shatters $\mathcal{O}$.

GREEDY($\mathcal{O}$):

1. Build the visibility graph $VG(\mathcal{O})$, by determining which pairs of vertices see one another. (This step requires $O(E + N \log N)$ time in $\Re^2$, and $O(N^{\lfloor d/2 \rfloor + 2})$ time in higher dimensions).
2. Determine the set $\mathcal{C}$ of candidate hyperplanes. (This step requires $O(N^d \log N)$ time for $d \geq 3$).
3. For each edge e of $VG(\mathcal{O})$, build a list of all candidate hyperplanes that stab e. Also label each hyperplane of $\mathcal{C}$ with the number of VG edges that it stabs. (This step requires $O(E|\mathcal{C}|)$ time).
4. Let $\mathcal{P} = \emptyset$.
5. Let $\mathcal{E} = \{\text{edges of } VG(\mathcal{O})\}$.
6. While there exists an edge in $\mathcal{E}$,
   1. Select a hyperplane h in $\mathcal{H} \backslash \mathcal{P}$ that stabs a maximum cardinality subset of $\mathcal{E}$.
   2. Add h to $\mathcal{P}$.
   3. For each edge in $\mathcal{E}$ stabbed by h, remove e from $\mathcal{E}$ and subtract one from the label of each hyperplane which stabs e. (This step requires $O(E|\mathcal{C}|)$ time, in total).

The algorithm runs in $O(E|\mathcal{C}|)$ time. It is open if there is a approximation algorithm that produces a $cM(\mathcal{O})$ size shattering for some constant c.

6 Conclusion and Open Problems

In this paper, we have shown that finding a minimum-cardinality shattering using an arrangement of hyperplanes is *NP*-Complete for all but the simplest versions of the problem. In addition, we have also presented algorithmic results for finding small shatterings of bounded polyhedral objects. However, there still remain many interesting questions to answer regarding shattering. We are currently working on the following extensions and related open problems:

1. Efficiency – Can we obtain an $O(n^d)$ time shattering algorithm for $\Re^d$? Such an improvement might provide insight for an improved algorithm for the "disjoint projections" problem in $\Re^2$, which has eluded an $O(n^2)$ time algorithm.
2. Approximation algorithms – Are there provably good approximation algorithms to find a small shattering whose size is at most a constant times larger than optimal?
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 addressed by Edelsbrunner, Robison, and Shen[ERS87] and Wenger[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 cell of the subdivision contains exactly one object. Figure 6a gives an example of three line segments for which such a shattering does not exist.
4. Constructive Solid Geometry – What is the complexity of the shattering problem mentioned in the introduction?

References

[CGL83] B. Chazelle, L. J. Guibas, and D. T. Lee. The power of geometric duality. In *Proc. 24th Annual Symposium on Foundations of Computer Science*, pages 217–225, October 1983. Also, *BIT* 25(1):76–90, 1985.

[Cha87] B. Chazelle. Polytope range searching and integral geometry. In *Proc. 28th Annual Symposium on Foundations of Computer Science*, pages 1–10, October 1987. To appear in *J. American Mathematical Society.*

[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.

[Fre91] R. Freimer. Shattering configurations of points with hyperplanes. Technical Report 91-1199, Dept. of Computer Science, Cornell University, April 1991.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability: A guide to the theory of NP-Completeness.* Freeman, New York, 1979.

[GM87] S. Ghosh and D. Mount. An output sensitive algorithm for computing visibility graphs. In *Proc. 28th Annual Symposium on Foundations of Computer Science*, 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.*

[Joh74] D. S. Johnson. Approximation algorithms for combinatorial problems. *J. Computer and System Sciences*, 9:256–278, 1974.

[Lov75] L. Lovász. On the ratio of optimal integral and fractional covers. *Discrete Mathematics*, 13:383–390, 1975.

[McK87] M. McKenna. Worst-case optimal hidden-surface removal. *ACM Transactions on Graphics*, 6:19–28, 1987.

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

[Mit90] J. S. B. Mitchell. Computing visibility graphs of intersecting obstacles. Manuscript, 1990.

[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.

[PS85] F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction.* Springer-Verlag, New York, 1985.

[SV90] V. Shapiro and D. Vossler. B-rep→CSG conversion I: Construction and optimization of CSG representations. Technical Report CPA89-3a, Cornell Programmable Automation, Cornell University, September 1990.

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