Discrete Comput Geom 4:311-336 (1989)

Discrete & Computational
Geometry
© 1989 Springer-Verlag New York Inc.

The Upper Envelope of Piecewise Linear Functions: Algorithms and Applications*

Herbert Edelsbrunner,[1] Leonidas J. Guibas,[2] and Micha Sharir[3]

[1] Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA

[2] Computer Science Department, Stanford University, Stanford, CA 94305, USA, and DEC Systems Research Center, Palo Alto, CA 94301, USA

[3] Courant Institute of Mathematical Sciences, New York University, New York, NY 10012, USA, and
School of Mathematical Sciences, Tel Aviv University, Tel Aviv, Israel

Abstract. This paper studies applications of envelopes of piecewise linear functions to problems in computational geometry. Among these applications we find problems involving hidden line/surface elimination, motion planning, transversals of polytopes, and a new type of Voronoi diagram for clusters of points. All results are either combinatorial or computational in nature. They are based on the combinatorial analysis in two companion papers [PS] and [E2] and a divide-and-conquer algorithm for computing envelopes described in this paper.

1. Introduction

This paper continues the study, initiated in [PS] and in [E2], of envelopes of piecewise linear functions in two or more variables. The previous papers have established tight lower and upper bounds on the combinatorial complexity of such envelopes. In this paper we provide efficient algorithms for calculating envelopes of this kind, discuss several extensions and special cases of the previous

* Work on this paper by the first author has been supported by Amoco Fnd. Fac. Dev. Comput. Sci. 1-6-44862. Work by the third author has been supported by the Office of Naval Research Grant N00014-82-K-0381, National Science Foundation Grant No. NSF-DCR-83-20085, by grants from the Digital Equipment Corporation and the IBM Corporation, and by a research grant from NCRD, the Israeli National Council for Research and Development.

combinatorial bounds, and give a variety of applications of these results to many problems in combinatorial and computational geometry.

Let us first review the results of [PS] and [E2]. Let $s_1, s_2, \ldots, s_n$ be n d-simplices in $(d+1)$-dimensional space, none of which is vertical (that is, parallel to the $(d+1)$st coordinate axis). We can thus view each s_i as the graph of a partially defined linear function $x_{d+1} = f_i(x_1, x_2, \ldots, x_d)$, whose domain of definition is a d-simplex, namely the orthogonal projection of s_i onto the hyperplane $x_{d+1} = 0$. The *upper envelope*, M, of the given simplices is the pointwise maximum of these functions, that is,

$$M(x_1, x_2, \ldots, x_d) = \max_{1 \le i \le n} \{f_i(x_1, x_2, \ldots, x_d)\},$$

where each f_i is assumed to be $-\infty$ outside its domain of definition. The *lower envelope* of the simplices is defined in a symmetric fashion.

We can associate with the envelope M a polyhedral cell complex, M^*, in d-space such that over each cell of M^* the envelope M is attained by a fixed function f_i. Intuitively, this is the orthogonal projection of the graph of M onto $x_{d+1} = 0$ (see Fig. 2.1 which shows the projection of four triangles in $d+1=3$ dimensions). The combinatorial complexity of M is the complexity of this complex, that is, the total number of faces (of any dimension) composing it. In general, the projection of the faces of M does not yield a convex decomposition of d-space. However, we can obtain a refined convex decomposition by superimposing M^* onto the arrangement of the ($(d-1)$-dimensional) hyperplanes containing the $(d-1)$-faces of the given simplices (see Fig. 2.2 which shows M^* for a set of three triangles in $d+1=3$ dimensions). We measure the complexity of M in terms of this refined decomposition.

If instead of simplices we have a collection of arbitrary piecewise-linear functions of d variables, we can decompose the graph of each of them into a collection of simplices, and then obtain the upper envelope of the given functions as the upper envelope of these simplices.

The two previous papers mentioned above analyse the combinatorial complexity of such envelopes in $d+1$ dimensions. They show that it is $O(n^d \alpha(n))$, where $\alpha(n)$ is the extremely slowly growing inverse of Ackermann's function. Moreover, this bound is tight in the worst case. For $d+1=2$, we face the special case of the envelope of n line segments in the plane. This case has been studied in [HS], [WS], and [S], where it is shown that the complexity of the envelope (in this case the number of subsegments composing it) is $O(n\alpha(n))$, and that this bound is tight in the worst case. The proofs are based on reformulating the problem in terms of Davenport-Schinzel sequences of order 3.

As a matter of fact, in the one-dimensional case, the theory of Davenport-Schinzel sequences yields tight almost linear upper bounds on the complexity of the envelope of any collection of (partially defined) continuous univariate functions, provided that each pair of them intersect in at most a fixed number of points. In contrast, for collections of n functions of $d \ge 2$ variables (which satisfy appropriate conditions on the pattern of their intersections), no comparable tight upper bounds on the complexity of their envelopes is known in general as yet, and the only general bound available so far is the trivial one, $O(n^{d+1})$. We refer to [SS] for a few improved results for certain classes of bivariate functions.

The proofs of the mentioned bounds for piecewise linear functions use induction on the dimension d. The proof in [PS] is based on a divide-and-conquer analysis. We partition the collection of n simplices into two subcollections of roughly $n/2$ simplices each, recursively obtain the envelope of each subcollection, and then obtain the overall envelope by taking the pointwise maximum of the two subenvelopes. Using arguments based on arrangements of hyperplanes, convexity, and on the complexity of the envelope restricted to certain lower-dimensional spaces, we show that the number of additional facets created by superimposing the two subenvelopes is bounded by $O(n^d\alpha(n))$. This leads to a recurrence relation whose solution yields the desired bound on the number of facets (that is, d-dimensional faces) in the envelope. Using Euler's formula for planar maps this yields similar bounds on the total number of faces of M if $d+1=3$. For higher dimensions this proof has been extended in [E2], where the same bound for the overall complexity of the envelope using a different divide step is obtained.

In Section 2 we follow the outline of the proof in [PS] to obtain a divide-and-conquer algorithm for calculating the envelope of triangles in three dimensions. The amount of time needed is at most proportional to the maximum combinatorial complexity of the envelope, namely $O(n^2\alpha(n))$. Because of the lack of convex hull algorithms in four and higher dimensions that run in time proportional to their output, we have not succeeded in generalizing this algorithm so that it runs in time $O(n^d\alpha(n))$ if $d+1\geq 4$. We also discuss several extensions and special cases of envelopes of piecewise linear functions which are needed for the applications that we study.

The major part of the paper is devoted to applications of the combinatorial and computational bounds stated above. These applications include the *hidden line/surface removal problem* which is discussed in Section 3. We give algorithmic results that match and generalize previous results of [Dv] and [M] for $d+1=3$ dimensions. We also obtain algorithms for related problems, such as constructing an image of a solid defined in Constructive Solid Geometry, and obtaining views of a three-dimensional projection of tetrahedra in four dimensions.

Section 4 considers *translational motion planning* for polyhedra in three dimensions. Here, we calculate the space of free placements of a given polyhedron, B, which is free to translate amidst a collection of polyhedral obstacles. We also discuss special cases of the problem, such as where the obstacles form a polyhedral terrain (that is, a piecewise linear surface meeting each vertical line at exactly one point), and the case where B is allowed to translate from its current position along a straight half-line only. The latter special case extends work on movable separability of sets reported in [Tt].

The problem of *stabbing line segments and polytopes* is investigated in Section 5. Using a standard duality transformation, we show that stabbing hyperplanes can be represented by points lying between the upper envelope of one collection of simplices and the lower envelope of another such collection. Our results extend previous work on this problem in two dimensions (see [E3]).

Voronoi diagrams of point clusters are considered in Section 6. For each cluster (that is, finite set of points in d dimensions) its distance from a point is the maximum Euclidean distance from this point to any point in the cluster. The Voronoi diagram of a collection of clusters is then defined as the decomposition of space into maximal domains so that for each domain there is a unique nearest cluster for all points in this domain. By transforming the problem to $(d+1)$-dimensional space, we can reformulate it in terms of envelopes of certain piecewise linear functions.

We do not regard this list of applications as exhaustive, and we expect many more applications to be found. For example, Tamir [Tr] has recently discovered applications of our results to p-center and obnoxious p-center problems for certain trees and graphs.

2. Computing an Envelope

This section presents an algorithm for constructing the upper envelope of a set of n triangles in three dimensions. The algorithm follows the outline of the proof in [PS] that shows that the combinatorial complexity of this envelope is $O(n^2\alpha(n))$. At several points we have to introduce intricate algorithmic tools in order to get a worst-case optimal algorithm. For some of these tools the complexity goes up more than desired when we generalize them to four and higher dimensions. This explains why we do not have an optimal (or even near-optimal) method for computing envelopes in four or higher dimensions yet. After presenting and analysing the algorithm, we study a few extensions of envelope constructions. These will lead to several computational and combinatorial results used in later sections of this paper.

We next present the algorithm that constructs the upper envelope of a set, S, of n triangles in three dimensions. Whenever convenient in the discussion we will make implicit assumptions about the triangles being in general position. The main reason is that we hope to get the point across if we leave out tedious complications. We see the general method, called the "simulation of simplicity," described in [EM] (see also [E1]), as a justification of this sloppy attitude.

First, we need a few definitions. We write $M(S)$ for the upper envelope of S, and $M^*(S)$ for the subdivision obtained by projecting the faces of $M(S)$ vertically onto the plane $x_3=0$ (see Fig. 2.1 which is borrowed from [PS]). In general the regions of $M^*(S)$ are not convex. To make them convex we refine $M^*(S)$ by projecting all triangles vertically onto $x_3=0$ and extending the $3|S|$ triangle edges to unbounded lines. The arrangement[1] defined by these lines is denoted by $A(S)$, and $\bar{M}(S)$ denotes the refined subdivision that we get by superimposing $M^*(S)$ and $A(S)$. See Fig. 2.2 taken from [PS]; it shows the projection of three triangles and the extension of their edges yielding an arrangement of nine lines. In order

[1] The *arrangement* defined by a finite set of lines in the plane is the subdivision of the plane that we get by drawing the lines. It consists of vertices (points where lines intersect), edges (pieces of lines that connect vertices), and regions (connected components of the plane reduced by all lines).

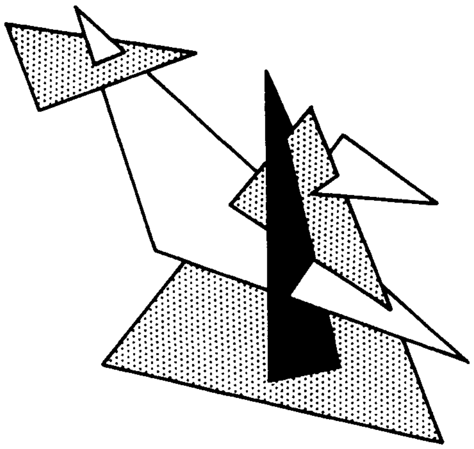


Fig. 2.1

to make $\bar{M}(S)$ a viable representation of $M(S)$ we associate each region of $\bar{M}(S)$ with a pointer to the triangle that assumes the maximum height over this region. Since $\bar{M}(S)$ is a refinement of $M^*(S)$ this triangle is unique in any case. $M^*(S)$ and also $M(S)$ can be obtained from $\bar{M}(S)$ by merging adjacent regions above which the maximum height is assumed by the same triangle. By construction, the envelope vertically above a region of $A(S)$ is a convex function.

Algorithm 1 (constructs $\bar{M}(S)$ as a representation of the upper envelope of S)

```
if $|S| \leq 1$ then Construct $\bar{M}(S)$ directly else
   Step 1. Partition $S$ into sets $S_1$ and $S_2$ of sizes $\lfloor |S|/2 \rfloor$ and $\lceil |S|/2 \rceil$.
   Step 2. Construct $\bar{M}(S_1)$ and $\bar{M}(S_2)$ recursively.
```

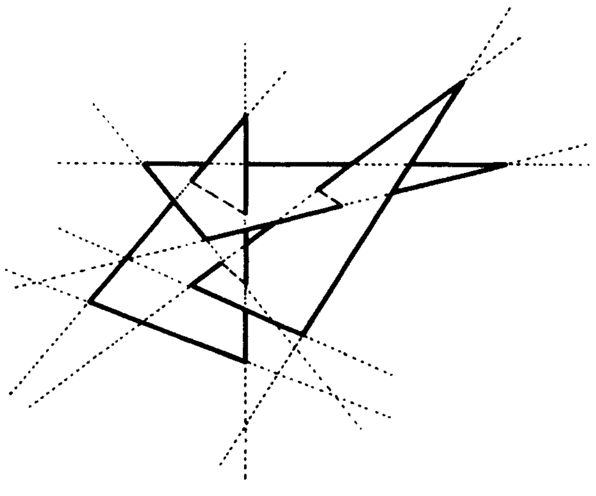


Fig. 2.2

Step 3. Superimpose $\bar{M}(S_1)$ and $A(S_2)$ and, symmetrically, superimpose $\bar{M}(S_2)$ and $A(S_1)$. We denote the thus created subdivisions by $\bar{\bar{M}}(S_1)$ and $\bar{\bar{M}}(S_2)$.

Step 4. Construct $A(S)$ which is $A(S_1)$ and $A(S_2)$ superimposed. Thus, $\bar{\bar{M}}(S_1)$ and $\bar{\bar{M}}(S_2)$ are refinements of $A(S)$.

Step 5. For each region r of $A(S)$ and for $i=1,2$ construct set $S_{i,r} \subseteq S_i$ that contains all triangles of S_i assuming the maximum height over a region of $\bar{\bar{M}}(S_i)$ contained in r.

Step 6. For each region r of $A(S)$ construct $\bar{M}(S)$ restricted to r by intersecting the half-spaces bounded from below by the planes that contain the triangles in $S_{1,r} \cup S_{2,r}$. $\bar{M}(S)$ restricted to r is the vertical projection of the boundary facets of this convex polyhedron clipped at the boundary of r.

endif.

Let us now discuss the various steps of Algorithm 1 in further detail. At the same time we analyse the time-complexity of each individual step which will then lead to the recurrence relation

$$T(n) = 2T\left(\frac{n}{2}\right) + O(n^2\alpha(n))$$

for the amount of time, $T(n)$, the algorithm takes. This recurrence relation solves to $O(n^2\alpha(n))$ (as in [PS]).

The first nontrivial step of Algorithm 1 is step 3 which superimposes $\bar{M}(S_i)$ and $A(S_{3-i})$, $i=1,2$. The combinatorial analysis in [PS] guarantees that the number of faces in the resulting subdivision, $\bar{\bar{M}}(S_i)$, is $O(n^2\alpha(n))$, $n=|S_1|+|S_2|$. Since all regions in both subdivisions, $\bar{M}(S_i)$ and $A(S_{3-i})$, are convex we can use the superimposition algorithm of Guibas and Seidel [GS] which takes time linear in the size of the output. Thus, step 3 takes $O(n^2\alpha(n))$ time.

Step 4 superimposes two arrangements which can be done in $O(n^2)$ time, $n=|S_1|+|S_2|$, using again the same superimposition algorithm. Alternately, we could construct the resultant arrangement, $A(S)$, from scratch which also takes only $O(n^2)$ time (see [E1]).

To understand step 5 it is important to recall that subdivision $\bar{\bar{M}}(S_i)$, for $i=1,2$, as constructed in step 3 is a refinement of arrangement $A(S)$ which is constructed in step 4. Thus, each region of $A(S)$ contains several regions of $\bar{\bar{M}}(S_i)$. Recall also that each region of $\bar{\bar{M}}(S_i)$ records the triangle that assumes the upper envelope above this region. The goal of step 5 is to collect, for each region r of $A(S)$, the triangles associated with regions in $\bar{\bar{M}}(S_1)$ and $\bar{\bar{M}}(S_2)$ contained in r. This can be done by visiting all regions of $\bar{\bar{M}}(S_i)$ using a graph traversal algorithm that first exhausts all regions contained in a common region of the arrangement and then goes to an adjacent arrangement region. This is a straightforward application of depth-first search (see [Tn]) which takes time proportional to the number of regions and edges of $\bar{\bar{M}}(S_i)$. Thus, step 5 also takes time $O(n^2\alpha(n))$.

Finally, we consider step 6 which is the most subtle part of Algorithm 1. For each region r of $A(S)$ it constructs the intersection of the half-spaces that are bounded from below by the planes that contain the triangles in $S_{1,r} \cup S_{2,r}$. If r is the ith region of $A(S)$ and $m_i = |S_{1,r}| + |S_{2,r}|$, then this can be done in $O(m_i \log m_i)$ time (see, e.g., [PrS]). Unfortunately, we only know that the sum of the m_i is $O(n^2\alpha(n))$ which does not imply anything better than that the sum of the $m_i \log_2 m_i$ is $O(n^2\alpha(n) \log n)$. Thus, to achieve $O(n^2\alpha(n))$ running time we have to intersect the half-spaces in a somewhat more intelligent manner. To describe such an alternate algorithm define the *slope* of a triangle as the slope of the line of intersection between the plane spanned by the x_1- and the x_3-axis and the plane that contains the triangle. If the slopes of all triangles in $S_{1,r}$ were smaller than the slopes of all triangles in $S_{2,r}$ we could construct the intersection of the half-spaces by merging the recursively constructed polyhedra for $S_{1,r}$ and $S_{2,r}$ in linear time (see [PrS]). Unfortunately again, the recursion is such that the polyhedra constructed are not exactly those for the regions of $A(S)$ but for the regions of $A(S_1)$ and $A(S_2)$. For example, let r_i be the region of $A(S_i)$ that contains region r, for $i = 1, 2$. It is not advisable to use the polyhedron representing the subenvelope above r_i as a substitute for r's polyhedron since it might have many facets that belong to half-spaces redundant above r. The combinatorial bound on the sum of the m_i does not generalize to these larger numbers; thus it is crucial not to be generous at this point.

The way out of this dilemma is to remember that the subdivision of r_i in $\bar{\bar{M}}(S_i)$ can be viewed as a representation of the polyhedron of r_i. The separation of the triangle slopes implies that the intersection of the boundaries of the two polyhedra, the ones of r_1 and r_2 restricted to the area above region r in $A(S)$, is a connected and piecewise linear curve. Figure 2.3 displays r_1, r_2, r, and the curve without showing the decompositions of the regions. Keep in mind, however, that this curve can merge into the boundary of r and leave it again an arbitrary number of times. Because of the slope condition, this curve has the property that it intersects any plane normal to the x_2-axis in a single point. Using standard methods for merging two subdivisions along a monotone curve (see [PrS] and papers on merging Voronoi diagrams referred to in [PrS]), the total amount of effort is linear in the number of edges of r plus the number of regions that subdivide r in $\bar{\bar{M}}(S_1)$ and $\bar{\bar{M}}(S_2)$. In order to make this all work we have to provide the appropriate subdivisions of the regions of $A(S)$. But these are provided by the superimposition of $\bar{\bar{M}}(S_i)$ and $A(S_{3-i})$ which decomposes the

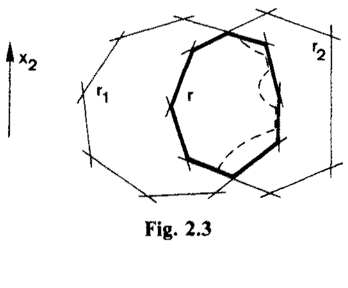


Fig. 2.3

subdivision of r_i into smaller pieces coinciding with regions in $A(S)$. Thus, this superimposition in step 3 implicitly constructs the proper polyhedra (or suitable representations by subdivisions thereof) which can then be merged in linear time each.

The only unresolved problem now is how we can guarantee that the slopes of the triangles in $S_{1,r}$ are smaller than the slopes of the triangles in $S_{2,r}$. But this can be achieved if the initial partitioning step of Algorithm 1 constructs S_1 and S_2 intelligently rather than arbitrarily. Just take the $\lfloor |S|/2 \rfloor$ triangles with smallest slopes, call this set S_1, and define $S_2 = S - S_1$.

We thus have an optimal algorithm for constructing the upper envelope of n triangles in three dimensions. This is Algorithm 1 with two changes. First, the partition of the set of triangles takes into account the slopes of the triangles. Second, step 5 is now superfluous and can be removed. This implies the main result of this section.

Theorem 2.1. *The upper envelope of a set of n triangles in three dimensions can be constructed in $O(n^2\alpha(n))$ time and storage. This is optimal in the worst case.*

We remark that Algorithm 1 can be modified so that it constructs the upper envelope of n line segments in two dimensions in $O(n\alpha(n) \log n)$ time and $O(n\alpha(n))$ storage. The amount of storage is optimal since the envelope can consist of $\Theta(n\alpha(n))$ edges; whether or not the time bound is optimal is still an open problem. There is no difficulty in generalizing Algorithm 1 to four and higher dimensions, however, it is still an open problem whether or not this can be done such that the running time is $O(n^d\alpha(n))$ for n d-simplices in $d+1$ dimensions. This would then be optimal since the combinatorial complexity of the envelope is $\Theta(n^d\alpha(n))$ in the worst case. The main obstacle in obtaining this result is step 6 which intersects half-spaces. Currently there is no algorithm available that takes less than $\Omega(m^2)$ time, where m is the number of half-spaces, no matter how many or few faces the resulting polyhedron has. The combinatorial results in [PS] and [E2] only bound the sum of the m_i (where m_i is the number of nonredundant half-spaces above the ith cell of the d-dimensional arrangement) and not the sum of the m_i^2. Indeed, there are cases where the sum of the m_i^2 is $\Omega(n^{d+1})$ and thus contradict the desired $O(n^d\alpha(n))$ upper bound. An approach that might be worth pursuing is to design an algorithm that follows the outline of the divide-and-conquer proof in [E2]. The main difference between Algorithm 1 and such a hypothetical algorithm would be that the latter recurs for a constant number of nondisjoint subsets of d-simplices rather than for two disjoint subsets.

The remainder of this section studies three extensions of the envelope problem which have algorithmic as well as combinatorial applications later in the paper. The first extension considers the region of points that lie above the upper envelope of a finite set S of d-simplices and below the lower envelope of another finite set T of d-simplices in $d+1$ dimensions. From the combinatorial results in [PS] and [E2] we know that both envelopes have complexity $O(n^d\alpha(n))$, with $n = |S|+|T|$, and by Theorem 2.1 we can construct both envelopes in $O(n^2\alpha(n))$ time if $d+1=3$. But how can we be sure that the intersection of the two envelopes does not exceed these complexity bounds? In fact, it does not. One way to see this is to go through the proofs of the $O(n^d\alpha(n))$ upper bounds for upper envelopes and to make one crucial change: rather than constructing the two envelopes above a cell of the d-dimensional arrangement separately, we construct the region of points between the two envelopes. Restricted to the area above a cell r of the arrangement, this region is the intersection of half-spaces and thus convex. It follows that its complexity is bounded by the sum of the complexities of the two regions between the two corresponding upper and lower subenvelopes. This is all we need to get the desired combinatorial result; also Algorithm 1 still works nicely in this extended case if $d+1=3$.

The second extension considers the special case where each d-simplex in $d+1$ dimensions is a ***half-hyperplane***, that is, a portion of a hyperplane h restricted to one side of a $(d-1)$-flat in h. In $d+1=2$ dimensions a half-hyperplane is a half-line. For n such half-lines it is not difficult to show that the number of edges in the upper envelope is at most $2n$ (see [E3]). This two-dimensional result can now be used as the base case of the inductive analysis of upper envelopes in higher dimensions. Recall that the only reason for the $\alpha(n)$ factor in the complexity of general upper envelopes is that the base case considers line segments, and the upper envelope of line segments has worst-case complexity $\Theta(n\alpha(n))$. The reduction for half-hyperplanes in $d+1$ dimensions leads to a linear number of sets of half-hyperplanes in d dimensions whose upper envelopes have complexity $O(n^{d-1})$ by inductive assumption. This leads to an upper bound of $O(n^d)$ for the complexity of n half-hyperplanes in $d+1$ dimensions. Note that this bound also holds for the combinatorial complexity of the region of points above the upper envelope of one set of half-hyperplanes and below the lower envelope of another set of half-hyperplanes. Furthermore, Algorithm 1 takes only $O(n^2)$ time if its input consists of n half-planes in three dimensions. This is because the only step where the $\alpha(n)$ factor sneaks in (step 3) now has complexity $O(n^2)$.

It is interesting to note the similarity between the upper envelope of a set of half-hyperplanes and the so-called zone of a hyperplane in an arrangement of hyperplanes in $d+1$ dimensions (see Chapter 5 of [E1]). In both cases, the combinatorial complexity is $O(n^d)$ but the known proofs of those two results are very different.

Finally, we consider the case where the n d-simplices in $d+1$ dimensions are pairwise disjoint (assuming they are relatively open). In this case, the maximum height above a cell of the d-dimensional arrangement is assumed by only one d-simplex. The combinatorial complexity of this arrangement is $O(n^d)$ which implies the same upper bound for the envelope. The more dramatic effect of the nonintersection assumption is that it simplifies Algorithm 1 significantly and thus allows us to generalize it to higher dimensions without loss of worst-case optimality. Steps 3 and 4 are now the same since $\bar{M}(S_1)=A(S_1)$, $\bar{M}(S_2)=A(S_2)$, and therefore $\bar{\bar{M}}(S_1)=\bar{\bar{M}}(S_2)=A(S)$. In step 5 set $S_{i,r}$ is the singleton set that contains the highest simplex in S_i above the region in $A(S_i)$ containing region r in $A(S)$. Thus, step 6 simplifies to a comparison between the simplex in $S_{1,r}$ and the one in $S_{2,r}$, for every r. The most expensive step of this algorithm is now the superimposition of arrangements $A(S_1)$ and $A(S_2)$ which can be done in quadratic time if $d+1=3$. In arbitrary $d+1$ dimensions, this operation takes $O(n^d)$ time, for $n=|S_1|+|S_2|$ (see [E1]). Thus, we now have an algorithm that runs in arbitrary dimensions and takes $O(n^d)$ time.

A similar effect (namely that the combinatorial complexity of the envelope is $O(n^d)$ rather than $O(n^d\alpha(n))$ can be observed when the n d-simplices in $d+1$ dimensions intersect in a certain restrictive manner. For example, if intersections occur only on the highest level of the recursion (talking in terms of the divide-and-conquer algorithm) then this is true. This proves that the upper envelope has complexity $O(n^d)$ if the set of d-simplices is the union of two sets with the property that any two d-simplices in the same set are pairwise disjoint. Unfortunately though, the computational complexity in this case might deteriorate to $O(n^2 \log n)$, $d+1=3$, if the first divide step separates the union into the two sets rather than discriminating by slope.

We summarize these results.

Theorem 2.2. *Let S be a set of n d-simplices in $d+1 \geq 3$ dimensions and let T be another such set whose cardinality is at most n.*

(i) *The region of points above the upper envelope of S and below the lower envelope of T has combinatorial complexity $O(n^d\alpha(n))$. In $d+1=3$ dimensions it can be constructed in $O(n^2\alpha(n))$ time and storage.*
(ii) *If all d-simplices in S are half-hyperplanes, then the combinatorial complexity of the upper envelope of S is $O(n^d)$, and it can be constructed in $O(n^2)$ time if $d+1=3$. If all d-simplices in T are also half-hyperplanes, then the same complexity bounds hold for the region of points above the upper envelope of S and below the lower envelope of T.*
(iii) *If the d-simplices in S are pairwise disjoint, then the combinatorial complexity of the upper envelope is $O(n^d)$ and it can be constructed in $O(n^d)$ time.*
(iv) *If S is the union of two disjoint sets with the property that any two d-simplices in the same set are disjoint, then the combinatorial complexity of the upper envelope is $O(n^d)$.*

The remainder of this paper considers applications of Theorems 2.1 and 2.2 to several problems in computational and combinatorial geometry.

3. Hidden Line and Surface Removal

Imagine that we take a picture of a three-dimensional scene from a point at infinity. To compute what this picture looks like—assuming opaque objects—is commonly known as the hidden line/surface removal problem. Because of the importance of this problem for practical applications there are many algorithms in the literature that were suggested for the problem (see, e.g., [SSS] for a classification of several such algorithms). It is usually assumed that the objects in the scene are determined by their piecewise linear boundaries (they are polytopes) and that they do not intersect. We would like to mention that there is no essential difference between the view from a point at infinity (a *parallel* view) and the view from a finite point (a *perspective view*). Indeed, for every plane through a finite viewpoint there is a projective transformation that maps the plane to the infinite plane and therefore the viewpoint to infinity. The polytopes are mapped to polyhedra such that the parallel view from the new point is equal to the perspective view from the old point on one side of the plane. Note, however, that this transformation moves points to infinity and lets them come back on the other side of space. In other words, polytopes gradually disappear on one side and, at the same time, come back on the other side of space. Thus, in order to get a valid picture we need to make sure that the viewpoint is shielded from the polytopes that come into the picture by traveling through infinity. Such a shielding mechanism is provided if we map the original infinite plane to a finite plane, using the same transformation, and use this plane as a background screen when we take the picture.

In this section we adopt a generalized definition of the hidden line/surface removal problem which is neither restricted to three dimensions nor to nonintersecting objects. We first discuss the more standard case of nonintersecting objects and later we extend the analysis to handle intersecting objects. We also give some applications for this extension.

In $d+1$ dimensions, the objects in the scene are modeled by a collection of d-dimensional simplices; for convenience we assume that they are relatively open. The problem is now the same as computing the upper envelope of the d-simplices assuming that the viewpoint is in the direction of the positive $(d+1)$st coordinate axis. We can thus use the algorithms of Section 2 to solve the hidden line/surface removal problem. Since we presently assume that no two d-simplices intersect (although their relative boundaries might intersect which it is important to allow if we model a polytope by d-simplices), we can use Theorem 2.2(iii) to get the following result.

Theorem 3.1. *Let S be a set of n pairwise nonintersecting relatively open d-simplices in $d+1$ dimensions. The combinatorial complexity of a view is $O(n^d)$ which is best possible in the worst case. Furthermore, it can be constructed in $O(n^d + n \log n)$ time.*

In three dimensions, that is, if $d+1=3$, the same time complexity was previously obtained by [Dv] and [M] who use known algorithms for constructing arrangements in the plane.

Note that the $O(n^d)$ bound for the combinatorial complexity holds even if we make the d-simplices translucent. Rather than computing only the topmost d-simplex above a given point we determine the topmost l, for some constant l, and the "color" at this point is a function of all l simplices.

Consider next generalizations of the hidden line/surface removal problem that arise when the given d-simplices are allowed to intersect. In this case, the problem is exactly the envelope problem studied in Section 2. As an example where this extension is needed we mention an operation that is useful in visualizing a four-dimensional scene given by n pairwise disjoint relatively open tetrahedra. Project these tetrahedra onto three dimensions and compute various views of this three-dimensional scene. Since we lose one dimension when we go from four to three dimensions, the tetrahedra in three dimensions will, in general, intersect. An alternate interpretation of this operation is that we compute views of the four-dimensional scene by moving vertical lines in a given direction until they hit an object. The "view" shows the first object hit by any such line. Note that this visualization of the four-dimensional scene is different from a projection onto a two-dimensional plane along a predefined direction.

Another application where intersections occur is in Constructive Solid Geometry (CSG) where an object is constructed from simple building blocks by means of intersection and union. The object is then represented by the tree whose leaves are the building blocks and each inner node stands either for the union or the intersection of the objects defined in its subtrees. A view of the object can be computed by postorder traversal of the defining tree. A special case in which Algorithm 1 is most effective is when the object is simply the union (or intersection) of many (polyhedral) building blocks, or when its CSG tree has at most two levels. For an arbitrarily defined object, however, Algorithm 1 may not be very efficient.

4. Translating a Polyhedron in Three Dimensions

An object, B, in some space cannot be moved to any arbitrary position if there are obstacles present which it has to avoid. The motion-planning problem for B is to calculate the space of all placements of B, called the *free placements* of B, in which it does not collide with any obstacle (see [HSS] for a recent compendium of work done on motion planning). In this section we consider special cases of motion planning in which B is allowed to translate but not to rotate. The problems that we address make sense in arbitrary dimensions but for simplicity and also because it is the most important setting, we discuss only the three-dimensional case. The much simpler two-dimensional case has been studied in [KLPS], [LS], [PSS], and [GSS]. If the object as well as the obstacles are polyhedra, that is, their boundaries are piecewise linear, then these motion-planning problems lead to certain envelope questions as we will see below.

Let B be an open three-dimensional polyhedron bounded by k facets and let $A_1, A_2, \ldots, A_m$ be closed convex polyhedral obstacles bounded by a total number of n facets. It is not essential that B is open, only the description of our results is slightly easier this way because, otherwise, we have to allow B's boundary to intersect the boundary of an obstacle—only the interiors have to be disjoint. All results are true for B closed if we change the phrasing of the results accordingly. We assume that the A_i are convex; so any nonconvex obstacle is split into convex pieces beforehand (see [C]). Our goal is to calculate the set of all translates of B that avoid the obstacles. The standard approach to solving this problem, initially proposed in [LW], uses Minkowski differences between the A_i and B. A translate B' of B is determined by its translation vector b, that is,

$$B' = B + b = \{x + b \mid x \in B\}.$$

We interchangeably think of b as a vector and as a point. B' intersects obstacle A_i if and only if there is a point $y \in A_i$ and a point $x \in B$ such that $x + b = y$, which is equivalent to $b = y - x$. Another way to say this is that B' and A_i are disjoint as long as b is not in the set

$$K_i = A_i - B = \{y - x \mid y \in A_i \text{ and } x \in B\}.$$

K_i is known as the *Minkowski difference* of A_i and B and is sometimes referred to as the "expanded" or "grown" obstacle. It is clear that B' lies in a free position if and only if b does not belong to $K = \bigcup_{i=1}^{n} K_i$. We can thus represent the set of free positions by its complement, K^c.

To get a handle on the combinatorial complexity of K^c assume that obstacle A_i is bounded by n_i facets; thus, $\sum_{i=1}^{m} n_i = n$. Except for degenerate cases, each facet of K_i is the Minkowski difference of a facet of A_i and a vertex of B, of an edge of A_i and an edge of B, or of a vertex of A_i and a facet of B. The number of such pairs is $O(k \cdot n_i)$, in contrast to the planar case where this number is only $O(k + n_i)$ (see [GRS]). This implies that K_i is bounded by at most $O(k \cdot n_i)$ faces. As a matter of fact, the number of facet-vertex and vertex-facet pairs is $O(k + n_i)$, only the number of pairing edges can be quadratic. Thus, the K_i altogether can be modeled by $O(k \cdot n)$ triangles which implies that the number of faces needed to describe K^c, the set of free placements of B, is $O(k^3 \cdot n^3)$. If B is nonconvex there are cases where the boundary of K^c consists of $\Omega(k^3 \cdot n^3)$ faces (see Fig. 4.1: the "triple fork" of size k can be locked into the "three-sided cage" of size n in $\Omega(k^3 \cdot n^3)$ different ways) which shows that the trivial bound is asymptotically tight. If B is convex, then no such example is known and a plausible conjecture is that K^c is bounded by at most $O(k^2 \cdot n^2 \cdot \alpha(k \cdot n))$ faces. It is rather easy to give examples where the complexity in question is $\Omega(k^2 \cdot n^2)$. Our goal is to show that the complexity of K^c is much lower than proportional to $k^3 \cdot n^3$ in certain important cases, or failing that, to show that the complexity of a single connected component of K^c (which is often all we need to consider) is small.

Consider first the general case. If we are interested in the set of free placements that can be reached by B from its initial position without ever interfering with

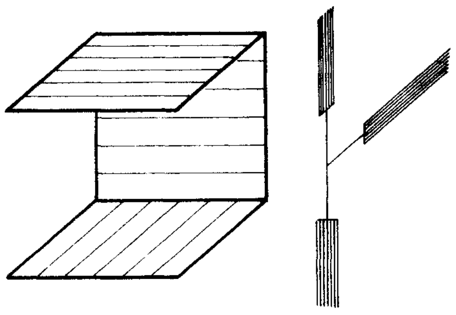


Fig. 4.1

obstacles, then we actually ask for the connected component of K^c that contains the origin—rather than for the entire K^c. To get an upper bound on the combinatorial complexity of this connected component we can use Theorem 4 in [PS] which shows that there are at most $O((k \cdot n)^{3-1/49})$ faces in its boundary. Unfortunately, the proof of this result is nonconstructive and does not lead to an algorithm that constructs the connected component in $o(k^3 \cdot n^3)$ time. Hence, in the general case, even though this result sheds light on the problem structure, no satisfactory solution is yet available. Improvements over these results have recently been obtained in [AS].

Things are much improved, however, when we consider the special case in which the obstacles A_i collectively form a so-called *polyhedral terrain*. This is a piecewise linear surface that intersects every vertical line in exactly one point. B is still assumed to be an arbitrary, thus not necessarily convex, polyhedron and we wish to find all free placements of B above this terrain, Σ. Again, we represent such a placement B' by the point b such that $B' = B + b$ and use the preceding analysis to obtain the space of free placements K^c, with K defined as above. Of course, in this restricted case the resulting set K^c is connected: B can be translated from any free placement to any other in a canonical manner by first moving upward to a sufficiently high position, then translating horizontally to a position above the target position, and, finally, descending to the desired position. Nevertheless, the calculation of K^c is significant in certain applications. Such an example occurs when B is required to maintain a fixed maximal vertical distance from Σ, for example, when B surveys Σ from close distance as it flies over it. Also, by preprocessing the boundary of K^c into a data structure which supports fast point location queries, we can decide in logarithmic time whether or not a given placement of B is free (see Chapter 11 of [E1] for an optimal data structure that supports point location queries). This method can also be used to determine the point(s) of contact of B's closure with Σ as it is lowered until it touches Σ. If the obstacles together form a polyhedral domain, then the boundary of K^c is the upper envelope of the expanded obstacles $K_i = A_i - B$. Theorem 1′ in [PS] and the algorithmic results in Section 2 of this paper now imply the following result.

Theorem 4.1. *Let B be a polyhedron bounded by k facets and let Σ be a polyhedral terrain with n facets. Then the number of faces bounding K^c, the set of free placements of B, is $O(k^2 \cdot n^2 \cdot \alpha(k \cdot n))$ and it can be constructed in the same amount of time.*

The preceding arguments can be generalized to cases where we allow B to translate only along a single half-line from its current position. Two-dimensional variants of this problem have been studied extensively by Toussaint and others (see [Tt] for a survey). By applying an appropriate projective transformation (similarly as in Section 3) we can assume that B is initially at infinity and is allowed to descend along vertical lines only. For an arbitrary collection of obstacles A_i we only need to find the upper envelope of the expanded obstacles $K_i = A_i - B$ (again, remembering to add the transformed image of the "background" plane at infinity). For each vertical line, this envelope gives the first obstacle hit by B if it moves along this line. If no obstacle is in the way of B's vertical movement, then the envelope shows the former infinite plane as the obstacle hit first. In the untransformed space this corresponds to B moving to infinity without ever hitting an obstacle. Using our combinatorial and computational knowledge about envelopes we get the following result.

Theorem 4.2. *Let B be a polyhedron bounded by k facets and let $A_1, A_2, \ldots, A_m$ be (possibly intersecting) convex obstacles bounded by a total of n facets. The set of free placements of B that are reachable by translating B along all possible half-lines is bounded by $O(n^2\alpha(n))$ faces and can be constructed in $O(n^2\alpha(n))$ time.*

We conclude this section with an argument that supports our conjecture that the combinatorial complexity of K^c is only nearly quadratic in $k \cdot n$ if B is convex. Here we do not assume that the obstacles form a polyhedral terrain. We show that this is true if B is a line segment. To show this it suffices to demonstrate that there are only $O(n^2)$ free placements of B in which it simultaneously makes contact with three obstacles. These triple contacts correspond to the vertices of K^c. In each such triple contact one of the contacts must be at a point of B different from its two endpoints. This point must touch an edge of an obstacle provided we ignore degenerate cases. Fix such an obstacle edge e and consider the plane h through e that is parallel to B. When B translates within plane h, maintaining contact with e, it can reach at most $O(n)$ placements at which it makes two more contacts with the obstacles (see [KLPS]). From this the claim follows readily.

5. Stabbing Line Segments and Polytopes

Finding transversals of a finite set of objects is the first of two problems discussed in this paper that relate to envelopes by means of a geometric transformation. The second such problem deals with certain Voronoi diagrams for sets of point clusters. In Section 6 we show that this problem is in fact closely related to the stabbing problem of this section.

Let S be a finite set of connected objects in $d+1$ dimensions. (We use $d+1$ to denote the dimensionality, rather than d, in order to be consistent with the notation in Section 2.) A hyperplane is a *transversal* of S if it intersects each object in S; we also say that it *stabs* S. Since a hyperplane intersects a connected object if and only if it intersects its convex hull, we can assume without loss of generality that all objects in S are convex. We consider the problem of finding all transversals of S, or a representation of this set, assuming that S is a collection of convex polytopes. The complexity of a solution will be measured in terms of n, the total number of vertices of the polytopes. In three dimensions, Euler's relation implies that n is proportional to the number of edges and facets bounding the polytopes. This is no longer true in four or higher dimensions. For this reason we restrict most of our discussion to $d+1=3$ dimensions and comment on the difficulties encountered in four and higher dimensions at the end of this section.

Earlier results on this problem can be found in [E3] which gives an $O(n \log n)$-time algorithm for S a set of n line segments in the plane, and in [AD] which gives an $O(mn^d)$-time algorithm for m polytopes bounded by n edges in $d+1$ dimensions. Alternatively, $O(n^{d+1})$-time algorithms are possible for the case of polytopes in $d+1 \geq 3$ dimensions using known algorithms for constructing arrangements of hyperplanes in $d+1 \geq 3$ dimensions (see Chapter 7 of [E1]). We show in this section that this straightforward bound can be improved to $O(n^2)$ in the case of line segments and to $O(n^2\alpha(n))$ in the case of convex polytopes in three dimensions. These results are optimal in the worst case in a sense that will become clear later.

Our development is based on a dual transform, $\mathscr{D}$, that maps a point to a plane and vice versa. If $p = (\pi_1, \pi_2, \pi_3)$ is a point in three dimensions, then we define the plane

$$\mathscr{D}(p): \quad x_3 = 2\pi_1 x_1 + 2\pi_2 x_2 - \pi_3.$$

Notice that $\mathscr{D}(p)$ is nonvertical, that is, it intersects the x_3-axis in a unique point. If h is a nonvertical plane we define $\mathscr{D}(h) = p$ such that $h = \mathscr{D}(p)$. Thus, $\mathscr{D}$ is involutary by definition. It is fairly easy to show that $\mathscr{D}$ preserves incidence relations ($p \in h$ if and only if $\mathscr{D}(h) \in \mathscr{D}(p)$) and order relations ($p$ lies vertically above h if and only if $\mathscr{D}(h)$ lies vertically above $\mathscr{D}(p)$).

We next extend $\mathscr{D}$ to point sets and, in particular, to polytopes. For $\mathscr{P}$ a point set in three dimensions we define

$$\mathscr{D}(\mathscr{P}) = \bigcup_{x \in \mathscr{P}} \mathscr{D}(x),$$

that is, $\mathscr{D}(\mathscr{P})$ is the set of all points that belong to at least one plane dual to a point of $\mathscr{P}$. We call $\mathscr{D}(\mathscr{P})$ the *stabbing region* of $\mathscr{P}$. Since $\mathscr{D}$ preserves incidences we have $p \in \mathscr{D}(\mathscr{P})$ if and only if plane $\mathscr{D}(p)$ intersects $\mathscr{P}$. Figure 5.1 illustrates these definitions. It shows $\mathscr{P}$ as a convex pentagon in two dimensions and displays the stabbing region of $\mathscr{P}$. It also shows a line intersecting the pentagon and its dual point which, of course, belongs to $\mathscr{D}(\mathscr{P})$. If $\mathscr{P}$ is a (connected) polytope, then $\mathscr{D}(\mathscr{P})$ is the set of all points that are neither vertically below all planes corresponding to vertices of $\mathscr{P}$ nor vertically above all such planes. This is because, for such a point x, its dual plane, $\mathscr{D}(x)$, stabs $\mathscr{P}$ and thus must separate at least

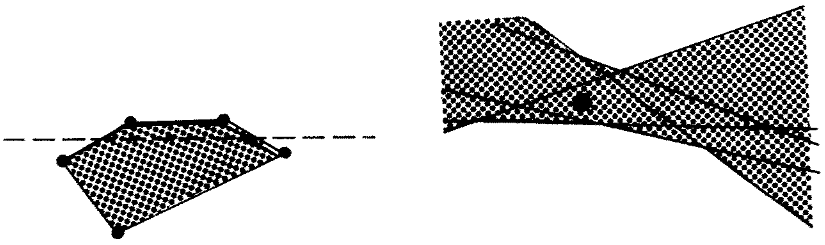


Fig. 5.1

one pair of vertices of $\mathcal{P}$. For $\mathcal{P}$ a line segment there are only two vertices and thus only two dual planes. Consequently, $\mathcal{D}(\mathcal{P})$ is the double wedge of points that lie vertically between the two planes corresponding to the line segment's two endpoints.

By definition, a transversal is a plane that cuts all polytopes in S. It follows that plane h is a transversal of S if and only if its dual point, $\mathcal{D}(h)$, belongs to

$$\mathcal{S}(S) = \bigcap_{\mathcal{P} \in S} \mathcal{D}(\mathcal{P}).$$

This intersection is termed the *stabbing region* of S. It is a representation of all transversals of S. Notice that the transformation, as currently defined, excludes vertical planes which thus have to be treated separately. One way to do this is to vertically project the polytopes onto the plane spanned by the x_1- and x_2-axis and to solve a two-dimensional stabbing problem for this set. Every transversal, which is now a line, corresponds to a vertical transversal, a plane, of the original set, S.

For a given polytope $\mathcal{P}$, $\mathcal{D}(\mathcal{P})$ is the set of points below or on $\mathcal{U}_{\mathcal{P}}$ and above or on $\mathcal{L}_{\mathcal{P}}$, where $\mathcal{U}_{\mathcal{P}}$ (resp. $\mathcal{L}_{\mathcal{P}}$) is the upper (lower) envelope of the planes dual to the vertices of $\mathcal{P}$. These are piecewise linear bivariate functions. Thus, the stabbing region, $\mathcal{S}(S)$, is the set of points below or on the lower envelope of all functions $\mathcal{U}_{\mathcal{P}}$ and above or on the upper envelope of all functions $\mathcal{L}_{\mathcal{P}}$. Using results from Section 2 we can give bounds on the combinatorial complexity of $\mathcal{S}(S)$ and on the amount of time needed to construct it.

In order to analyse $\mathcal{S}(S)$ we model each function $\mathcal{U}_{\mathcal{P}}$ and $\mathcal{L}_{\mathcal{P}}$ by a collection of triangles in three dimensions. If m is the number of vertices of $\mathcal{P}$, then $\mathcal{U}_{\mathcal{P}}$ and $\mathcal{L}_{\mathcal{P}}$ can be decomposed into $O(m)$ triangles. This puts us into the situation described in Theorem 2.2(i). The upper bound on the combinatorial complexity can be improved from $O(n^2\alpha(n))$ to $O(n^2)$ if S is a set of n line segments. This is because each $\mathcal{U}_{\mathcal{P}}$ and $\mathcal{L}_{\mathcal{P}}$ is composed of two half-planes that meet at a common line. The improvement now follows from Theorem 2.2(ii).

Theorem 5.1. *Let S be a set of convex polytopes in three dimensions and let n be the total number of vertices.*

(i) *The number of faces bounding $\mathcal{S}(S)$ is $O(n^2\alpha(n))$ and so is the amount of time needed to construct $\mathcal{S}(S)$. Both bounds are tight in the worst case.*
(ii) *The number of faces bounding $\mathcal{S}(S)$ is $O(n^2)$ if all polytopes in S are line segments. In this case, $O(n^2)$ time suffices to construct $\mathcal{S}(S)$. Both bounds are tight in the worst case.*

Using the lower bound examples indicated in [PS] it is not difficult to prove that all bounds are asymptotically tight in the worst case. In this context it is interesting to note that examples with $\Omega(n^2\alpha(n))$ faces can be modeled even with the restriction that all polytopes in S are triangles in three dimensions.

If we specialize the computational results of Theorem 5.1 to $d+1=2$ dimensions, we get an algorithm that constructs the stabbing region of polygons with a total of n vertices in $O(n\alpha(n)\log n)$ time, and in $O(n \log n)$ time if all polygons are line segments. The former result is new although it follows easily from the combinatorial analysis of two-dimensional envelopes in [HS] and the algorithmic techniques in [E3]. The latter result dates back to [E3]. Note the $\log n$ term in the time-complexity that comes up in two dimensions. The reason for this extra term is that the homogeneous solution of the recurrence relation that describes the time-complexity is essentially the same as the additive term (see Section 2). In three dimensions the additive term is significantly larger than the homogeneous solution which explains why the $\log n$ term disappears.

We conclude this section with a few remarks about the generalization of our methods to $d+1 \geq 4$ dimensions. The first difficulty that arises is combinatorial and concerns the decomposition of the $\mathcal{U}_{\mathcal{P}}$ and $\mathcal{L}_{\mathcal{P}}$ into d-simplices. The total number of vertices of the input polytopes, n, is proportional to the number of facets of the $\mathcal{U}_{\mathcal{P}}$ and $\mathcal{L}_{\mathcal{P}}$, but it might very well be that the number of lower-dimensional faces of the $\mathcal{U}_{\mathcal{P}}$ and $\mathcal{L}_{\mathcal{P}}$ by far exceeds $O(n)$. Indeed, in $d+1=4$ dimensions their number is $\Omega(n^2)$ if S consists of a constant number of cyclic polytopes. Such constellations need more than $O(n)$ simplices to model the boundary of the stabbing region by two envelopes of simplices. This weakens our bounds on the combinatorial complexity of stabbing regions which use envelope bounds. Even if we had a method that circumvents the sketched difficulty, there is no algorithm known that constructs the stabbing region in time $o(n^{d+1})$ because of reasons explained in Section 2. But $O(n^{d+1})$ is straightforward if we use arrangement algorithms as mentioned above.

6. Voronoi Diagrams of Point Clusters

In this section we consider applications of envelopes to a certain generalization of Voronoi diagrams. This generalization can be defined in any number of dimensions, and we will do so, but our discussion of the combinatorial and computational complexity is mostly confined to the plane. The diagram that we have in mind bears close relationship to the notion of complete linkage clustering (see, e.g., [H]). For this clustering method, the distance between any two clusters is defined as the maximum distance between any two points, one of each cluster. We come back to this clustering method at the end of this section.

Let δ denote the Euclidean distance function. For a set of points, C, called a *cluster*, and for a point p, define

$$\delta(p, C) = \max\{\delta(p, x) \mid x \in C\}$$

as the *distance* between p and C. In most cases we let C be finite and, whenever it is convenient to have C infinite, it will be the convex hull of a finite number of points in which case the maximum of the distances between p and points of C is well defined. The definition of $\delta(p, C)$ implies that the closed ball with center p and radius $\delta(p, C)$ contains cluster C. In fact, it is the smallest ball centered at p for which this is true. Now let S be a finite set of clusters. The

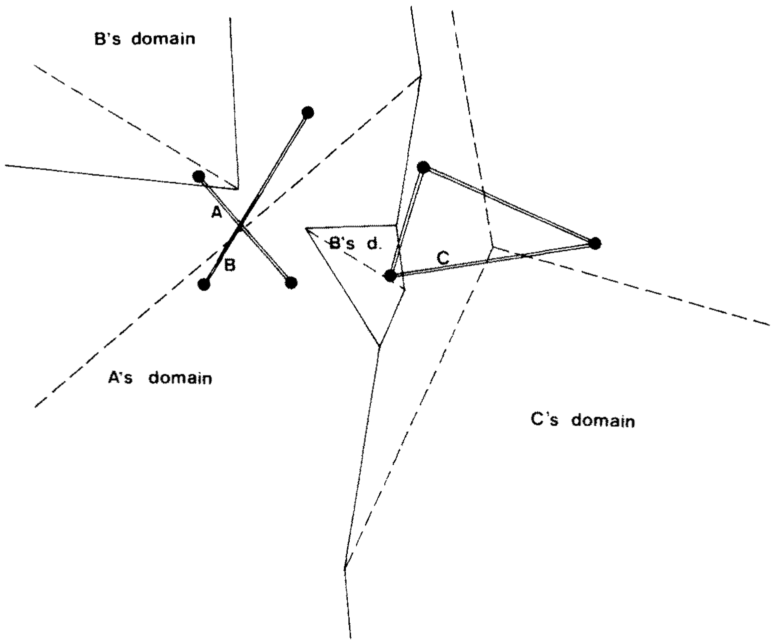


Fig. 6.1

Voronoi diagram of S, $\mathcal{V}(S)$, is a partition of space into maximal (but not necessarily connected) *domains*, one for each cluster, such that a point p belongs to the domain associated with cluster C if $\delta(p, C) < \delta(p, D)$ for all clusters $D \neq C$ in S. See Fig. 6.1 for an example. It is convenient to replace a cluster C by its convex hull which is all right since the distance of a point p from C is the same as its distance from the convex hull of C. The clusters in Fig. 6.1 are A (two points), B (two points), and C (three points). The domain of B consists of two connected components which we call *regions*. The solid lines show the Voronoi diagram and the dashed lines decompose each region using the furthest point Voronoi diagram of the cluster. This is the diagram that associates with each point the part of the domain for which the point is the furthest point of the cluster. The significance of this decomposition is that it shows which point of the cluster attains the distance to the cluster and where it does so.

Below, we discuss some properties of this kind of Voronoi diagram. First, we demonstrate that they are closely related to envelopes and thus derive general upper bounds on their complexity. Second, we study the special case where the convex hulls of the clusters are disjoint. It turns out that this condition reduces the combinatorial complexity of the diagrams dramatically. For simplicity, we restrict our attention to the two-dimensional case. Three- and higher-dimensional cases can be treated in the same way as the stabbing problem in four and higher dimensions (see Section 5); we thus omit all details pertaining to these extensions. Specifically, we prove the following theorem.

Theorem 6.1. *Let S be a set of clusters in the plane and let n be the sum of the cardinalities of the clusters.*

(i) *The number of faces of $\mathscr{V}(S)$ is $O(n^2\alpha(n))$.*
(ii) *If each cluster consists of one or two points, then the number of faces of $\mathscr{V}(S)$ is $O(n^2)$, and this is tight in the worst case.*
(iii) *If the convex hulls of any two clusters are disjoint, then $\mathscr{V}(S)$ contains at most $|S|$ regions. The number of edges and vertices in this case is $O(n)$.*

Proof. The proof consists of three fairly independent steps. First, we demonstrate the upper bounds in (i) and (ii) by means of envelopes in three dimensions. Second, we construct an example that proves the lower bound on the maximum complexity stated in (ii). Finally, we show that the domain of a cluster is connected if the convex hulls of any two clusters are disjoint. This leads to a proof of (iii).

In order to relate $\mathscr{V}(S)$ to an envelope of triangles in three dimensions we use two geometric transforms. Let $p=(\pi_1, \pi_2)$ be a point in the two-dimensional plane which we associate with the plane $x_3=0$ in three dimensions. The first transform, U, projects p vertically onto the paraboloid of revolution given by $x_3=x_1^2+x_2^2$, that is,

$$U(p)=(\pi_1, \pi_2, \pi_1^2+\pi_2^2).$$

The second transform, $\mathscr{E}$, maps p to the unique plane that touches the paraboloid in point $U(p)$, that is,

$$\mathscr{E}(p): \quad x_3=2\pi_1x_1+2\pi_2x_2-(\pi_1^2+\pi_2^2).$$

These transforms can be used to express distance information in two dimensions as combinatorial information in three dimensions. The crucial property here is that $\delta^2(p, x)$, the square of the distance between points p and x in the plane, is equal to the vertical distance between point $U(x)$ and the vertical projection of x onto plane $\mathscr{E}(p)$ (see Chapter 1 of [E1]). The distance from x to a point p is thus the square root of the vertical distance from point $U(x)$ down to plane $\mathscr{E}(p)$. Consider now a cluster of points, C, instead of a single point. The distance from x to C, $\delta(x, C)$, is equal to the square root of the largest vertical distance from $U(x)$ to any of the planes $\mathscr{E}(p)$, $p\in C$, that is, to the lower envelope, $\mathscr{L}_C$, of all these planes. If C contains m points, then this envelope is the boundary of a convex polyhedron consisting of at most m facets which can be decomposed into $O(m)$ triangles. Let M be the upper envelope of all the surfaces $\mathscr{L}_C$, $C\in S$. Then a point x lies in the domain of cluster C exactly when $\mathscr{L}_C$ is vertically nearest to $U(x)$ among all $\mathscr{L}_D$, $D\in S$. Since $U(x)$ lies above all surfaces $\mathscr{L}_D$, this is equivalent to $\mathscr{L}_C$ attaining the upper envelope M at x. Thus, we get $\mathscr{V}(S)$ by constructing the upper envelope of the surfaces $\mathscr{L}_D$, or more specifically of the triangles composing these surfaces, and then projecting the faces of the envelope vertically onto the plane $x_3=0$. The upper bound of (i) now follows immediately from Theorem 1′ in [PS]. If each cluster consists of at most two points each, the $\mathscr{L}_D$ are either single planes or two half-planes glued together along a common line. For these functions we have an upper bound of $O(n^2)$ for

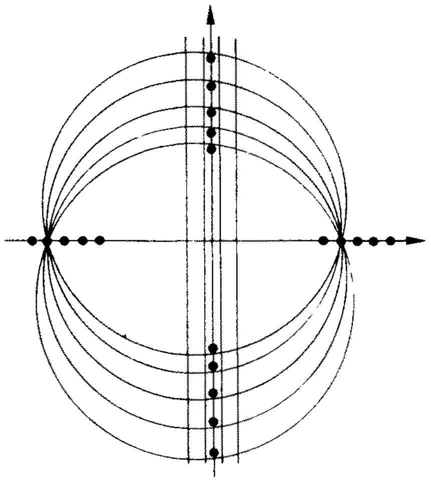


Fig. 6.2

the combinatorial complexity as argued in Section 2. This proves the upper bound in (ii).

We next show the lower bound in (ii). Assume without loss of generality that $n = 4k$ for some integer k. We describe a set S of $2k$ clusters, each containing two points, such that $\mathcal{V}(S)$ has $\Omega(n^2)$ regions. This example is illustrated in Fig. 6.2. It consists of k point pairs on the x_1-axis such that the ith pair can be obtained by moving the $(i-1)$st pair a distance ε to the right, for $2 \leq i \leq k$ and $\varepsilon > 0$ sufficiently small. More specifically, we choose the first pair of points at locations $(-1, 0)$ and $(1-(k-1)\cdot\varepsilon, 0)$, with $k \cdot \varepsilon < \frac{1}{2}$. Note that this implies that the x_2-axis is a symmetry axis of the $2k$ points. The Voronoi diagram of these clusters only consists of $k-2$ vertical slabs of width ε, one for each pair except for the first and the last, and two half-planes, one for the first point pair and one for the last. Now add another k point pairs on the x_2-axis. To describe how these points are to be chosen, we take a point p and move it upward inside one of the slabs. For each location of p we consider the smallest disk with center p that contains the horizontal point pair corresponding to the slab. By construction, the disk contains no other horizontal cluster. Since the horizontal clusters are almost identical, the wandering disk is almost the same for point p moving in any other slab. Thus, we can choose point pairs on the x_2-axis such that the disk alternates between containing one vertical cluster and containing no vertical cluster. Each slab is then decomposed into $2k+1$ pieces, $k+1$ of which define the domain of the corresponding horizontal cluster. The lower bound in (ii) follows.

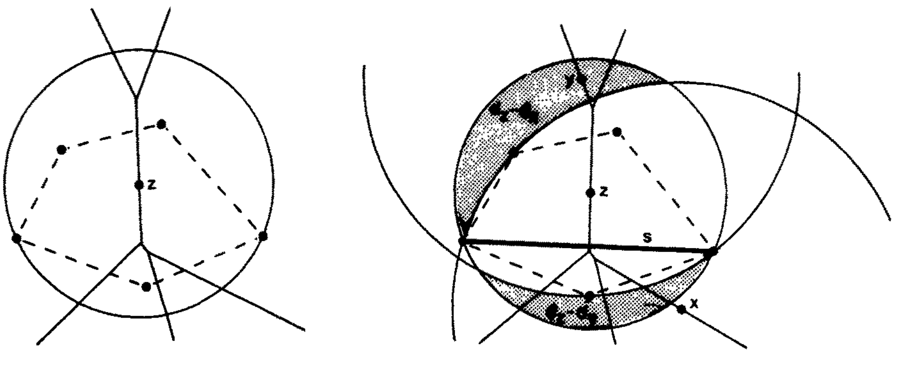
Finally, we prove that each domain is connected if the convex hulls of the clusters are pairwise disjoint. The proof takes two steps. First, it verifies that the skeleton of the furthest point Voronoi diagram[2] of a cluster, for short the skeleton

[2] The *skeleton* of a cluster C is the set of points x such that there are at least two points $c \in C$ that maximize $\delta(x, C)$. It is a straight line tree with at most $m-2$ vertices and $2m-3$ edges, if m is the number of points in C. For convenience we define the skeleton to be the point c itself if $C = \{c\}$.

of the cluster, intersects the domain of the cluster in a connected tree. Second, it shows that an arbitrary point of the cluster can be connected to this tree by a straight line segment that lies entirely within the domain of the cluster. We do the second step first because it is simpler than the first step. Let p be a point that belongs to the domain of a cluster C (see Fig. 6.1). By definition, C is the only cluster that is fully contained in the closed disk with center p and radius $\delta(p, C)$. Let c be the point in C such that $\delta(p, c) = \delta(p, C)$. If c is not unique, then p already belongs to the skeleton of C. Otherwise, move p straight toward c until it runs into an edge of the skeleton. The disk at every intermediate location of p lies strictly inside p's original disk which implies that it contains no other cluster besides C. It is clear that p must run into an edge of the skeleton for, otherwise, the disk of p would eventually vanish, which can only mean that c is the only point in C and thus is equal to the skeleton of C by definition.

We now prove that if two points, x and y, on the skeleton of C belong to the domain of C, then all points between x and y on the skeleton of C also lie in this domain. Note that the set of points between x and y is well defined since the skeleton is a tree. Let us introduce some definitions. For z an arbitrary point of the skeleton of C we write d_z for the smallest closed disk with center z that contains C, and we let ∂_z denote the circle bounding d_z. Since z belongs to the skeleton of C there are at least two points in C that maximize the distance from z; by definition these points lie on ∂_z. The line segment connecting such two points in $C \cap \partial_z$ is called a *C-chord* of ∂_z (see Fig. 6.3(b) which shows the C-chord of ∂_z).

We now come back to proving that if x and y are two points on the skeleton of C that belong to the domain of C, then any point z between x and y belongs to the domain of C. By definition, the only cluster contained in d_x is C and similarly C is the only cluster contained in d_y. Consider d_z, the smallest disk around z that contains C. If d_z is to contain another cluster, D, at least one point of D must lie in $d_z - d_x$ and at least one point of D must lie in $d_z - d_y$; otherwise, D is contained in d_x or d_y too. We prove below that $d_z - d_x$ and $d_z - d_y$ are



(a) (b)

Fig. 6.3

separated from each other by a C-chord of ∂_z (as in Fig. 6.3(b)). This implies that the convex hulls of C and D intersect, a contradiction.

We first consider the slightly simpler case that z is not a vertex of the skeleton and thus belongs to an edge of the skeleton. In this case, ∂_z has only one C-chord which we call s; the endpoints of s cut ∂_z into two arcs. Both d_x and d_y contain s (as they contain C) which implies that s separates $d_z - d_x$ from $d_z - d_y$ unless both ∂_x and ∂_y intersect ∂_z in the same arc. We now prove that this is not possible.

Think of a point ζ continuously moving on the skeleton from x to y, and let z be this point at some instance of time. When ζ moves away from z its circle, ∂_ζ, moves too. It still goes through the endpoints of s, the common C-chord of ∂_z and ∂_ζ, but d_ζ grows along one arc of ∂_ζ and it shrinks along the other arc. We call the former the *growing arc* and the latter the *shrinking arc* of ∂_ζ. As long as ζ moves on the same edge of the skeleton the growing arc and the shrinking arc do not change, that is, they are always determined by the endpoints of the same line segment, s. The shrinking arc hits another point, c, of C at the same instance of time when ζ encounters a vertex of the skeleton. (Note that the growing arc cannot hit a point of C since all points of C are inside the circle.) Now ∂_ζ contains three points of C, the endpoints of s and point c. When ζ moves on, one of the two line segments connecting c with the endpoints of s becomes the unique C-chord of ∂_ζ. Since c lies on the shrinking arc of ∂_ζ when ζ encounters the vertex, the new shrinking arc can only be smaller than before, and the new growing arc can only be larger than before. This proves that $d_\zeta - d_x$ strictly grows, that is, if z is a point between x and y and z' is a point between z and y, then $d_z - d_x \subseteq d_{z'} - d_x$. By a symmetric argument, $d_\zeta - d_y$ strictly shrinks as ζ moves from x to y. Consequently, ∂_x cannot intersect the shrinking arc of ∂_z and ∂_y cannot intersect the growing arc of ∂_z. This implies the claim for z being a point of an edge of the skeleton.

Finally, if z is a vertex of the skeleton then ∂_z contains three points of C which cut ∂_z into three arcs. One arc is growing when ζ is immediately before and immediately after z, and one arc is shrinking both times. The third arc changes from shrinking to growing. Since $d_\zeta - d_x$ is strictly growing ∂_x intersects neither the shrinking nor the status-changing arc, and since $d_\zeta - d_y$ is strictly shrinking ∂_y neither intersects the growing nor the changing arc.[3] Thus, $d_z - d_x$ and $d_z - d_y$ are, again, separated from each other, in this case by two C-chords of ∂_z.

Thus we have shown that the domain of each cluster in S is connected (it can be empty, though) if their convex hulls are pairwise disjoint. The first part of claim (iii) follows. This does not imply that the number of edges and vertices of $\mathcal{V}(S)$ is also $O(|S|)$. By Euler's relation for planar graphs this is, however, true for vertices of $\mathcal{V}(S)$ that are incident to at least three edges. A vertex of degree 2 must also lie on an edge of the skeleton of the cluster in one of the two adjacent domains. Since each edge of the skeleton meets the boundary of its cluster's domain in at most two points (this follows from the above argument)

[3] In degenerate cases ∂_z can contain $k > 3$ points of C. These k points cut ∂_z into k arcs, $k-2$ of which change from shrinking to growing and thus intersect neither ∂_z nor ∂_y.

we conclude a linear upper bound on the number of degree 2 vertices and thus of edges of $\mathcal{V}(S)$. □

We remark that $\mathcal{V}(S)$ can be further refined by decomposing each region of $\mathcal{V}(S)$ into subregions by the corresponding skeleton. This yields a refined convex subdivision of the plane so that a point x belongs to a subregion corresponding to point c of cluster C if and only if x is nearest to C and its distance to C is attained by c. In the special case where the convex hulls of the clusters are disjoint, the domain of cluster C is thus decomposed into at most $|C|$ subregions. Euler's relation implies that the numbers of edges and vertices of the refined subdivision are $O(n)$, n the total number of points in all clusters.

To construct $\mathcal{V}(S)$ for an arbitrary collection S of clusters we can use the three-dimensional envelope algorithm of Section 2 which implies the following result.

Theorem 6.2. *Let S be a set of clusters where n is the sum of the cardinalities of the clusters.*

(i) *$\mathcal{V}(S)$ can be constructed in $O(n^2\alpha(n))$ time.*
(ii) *If each cluster is of size one or two, then $O(n^2)$ time suffices and this is optimal in the worst case.*

There are two major open problems concerning cluster Voronoi diagrams that remain. Is $O(n^2\alpha(n))$ for the combinatorial complexity of $\mathcal{V}(S)$ tight? A better upper bound (maybe $O(n^2)$) would also improve the time bound in Theorem 6.2(i). Second, can $\mathcal{V}(S)$ be constructed in less than quadratic time (maybe $O(n \log n)$) if the convex hulls of the clusters are pairwise disjoint? An affirmative answer to the second question could also be relevant to complete linkage clustering of n points in the plane. In this method, the points are considered to be individual clusters initially, and at each stage the two nearest clusters are merged until all points belong to the same cluster. As mentioned at the beginning of this section, the distance between two clusters is defined as the maximum distance between any two points, one from each cluster. The most efficient algorithm known for this problem takes $O(n^2)$ time and $O(n)$ storage for the entire sequence of merges (see [Df]). At each stage, the set of points is partitioned into a collection of clusters. The Voronoi diagram of these clusters is potentially useful since the two nearest clusters also have a common edge in the diagram. The problem is now to maintain the diagram through a sequence of $n-1$ cluster merges. In the case of single linkage clustering, where the distance between two clusters is the minimum distance between any two points, one from each cluster, a similar approach yields an $O(n \log n)$-time algorithm (see [E1]). The latter clustering method is intimately related to the notion of the minimum spanning tree of the points.

Acknowledgment

The authors would like to thank Emo Welzl for valuable and stimulating discussions on some of the problems presented in this paper. In particular they

would like to mention that Emo Welzl independently derived parts of Theorem 6.1(iii).

References

[AS] Aronov, B. and Sharir, M., Triangles in space, or: Building (and analyzing) castles in the air, *Proc. 4th Ann. ACM Sympos. Comput. Geom.*, 1988, pp. 381-391.

[AD] Avis, D. and Doskas, M., Algorithms for high dimensional stabbing problems, Report SOCS-87.2, School of Computer Science, McGill University, Montreal, Quebec, 1987.

[C] Chazelle, B., Convex partitions of polyhedra: a lower bound and worst-case optimal algorithm, *SIAM J. Comput.* **13** (1984), 488-507.

[Df] Defays, D., An efficient algorithm for a complete link method, *Comput. J.* **20** (1977), 364-366.

[Dv] Devai, F., Quadratic bounds for hidden line elimination, *Proc. 2nd Ann. ACM Sympos. Comput. Geom.*, 1986, pp. 269-275.

[E1] Edelsbrunner, H., *Algorithms in Combinatorial Geometry*, Springer-Verlag, Heidelberg, 1987.

[E2] Edelsbrunner, H., The upper envelope of piecewise linear functions: tight bounds on the number of faces, Report UIUCDCS-R-87-1396, Department of Computer Science, University of Illinois, 1987.

[E3] Edelsbrunner, H., Maurer, H. A., Preparata, F. P., Rosenberg, A. L., Welzl, E., and Wood, D., Stabbing line segments, *BIT* **22** (1982), 274-281.

[EM] Edelsbrunner, H. and Mücke, E. P., Simulation of simplicity: a technique to cope with degenerate cases in geometric algorithms, *Proc. 4th Ann. ACM Sympos. Comput. Geom.*, 1988, pp. 118-133.

[GRS] Guibas, L. J., Ramshaw, L., and Stolfi, J., A kinematic framework for computational geometry, *Proc. 24th Ann. IEEE Sympos. Found. Comput. Sci.*, 1983, pp. 100-111.

[GS] Guibas, L. J. and Seidel, R., Computing convolutions by reciprocal search, *Discrete Comput. Geom.* **2** (1987), 175-193.

[GSS] Guibas, L. J., Sharir, M., and Sifrony, S., On the general motion planning problem with two degrees of freedom, *Proc. 4th Ann. ACM Sympos. Comput. Geom.*, 1988, pp. 289-298.

[HS] Hart, S. and Sharir, M., Nonlinearity of Davenport-Schinzel sequences and of generalized path compression schemes, *Combinatorica* **6** (1986), 151-177.

[H] Hartigan, J. A., *Clustering Algorithms*, Wiley, New York, 1975.

[HSS] Hopcroft, J., Schwartz, J., and Sharir, M. (eds.), *Planning, Geometry and Complexity of Robot Motion*, Ablex, Norwood, NJ, 1987.

[KLPS] Kedem, K., Livne, R., Pach, J., and Sharir, M., On the union of Jordan regions and collision-free translational motion amidst polygonal obstacles, *Discrete Comput. Geom.* **1** (1986), 59-71.

[LS] Leven, D. and Sharir, M., Planning a purely translational motion for a convex object in two-dimensional space using generalized Voronoi diagrams, *Discrete Comput. Geom.* **2** (1987), 9-31.

[LW] Lozano-Pérez, T. and Wesley, M. A., An algorithm for planning collision-free paths among polyhedral obstacles, *Comm. ACM* **22** (1979), 560-570.

[M] McKenna, M., Worst-case optimal hidden-surface removal, *ACM Trans. Graphics* **6** (1987), 19-28.

[PS] Pach, J. and Sharir, M., The upper envelope of piecewise linear functions and the boundary of a region enclosed by convex plates: combinatorial analysis, *Discrete Comput. Geom.*, to appear.

[PSS] Pollack, R., Sharir, M., and Sifrony, S., Separating two simple polygons by a sequence of translations, *Discrete Comput. Geom.* **3** (1988), 123-136.

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

[SS] Schwartz, J. T. and Sharir, M., On the two-dimensional Davenport-Schinzel problem, Report 193 (revised), Computer Science Department, Courant Institute, New York, 1987.

[S] Shor, P., Private communication.

[SSS] Sutherland, I. E., Sproull, R. F. and Shumacker, R. A., A characterization of ten hidden surface algorithms, *Comput. Surveys* **6** (1974), 1-55.

[Tr] Tamir, A., Improved complexity bounds for center location problems on networks by using dynamic data structures, Manuscript.

[Tn] Tarjan, R. E., Depth-first search and linear graph algorithms, *SIAM J. Comput.* **2** (1972), 146-160.

[Tt] Toussaint, G., Movable separability of sets, in *Computational Geometry*, G. T. Toussaint, ed., North-Holland, Amsterdam, 1985, pp. 335-375.

[WS] Wiernik, A. and Sharir, M., Planar realization of nonlinear Davenport-Schinzel sequences by segments, *Discrete Comput. Geom.* **3** (1988), 15-47.

Received November 30, 1987, and in revised form February 3, 1988.