ON ENVELOPES OF ARRANGEMENTS OF LINES

by

Eric Guévremont

School of Computer Science
McGill University
Montréal, Québec, Canada

September 1992

A THESIS SUBMITTEDTO THE FACULTY OF GRADUATE STUDIES AND RESEARCH
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS OF THE DEGREE OF
MASTER OF SCIENCE

Abstract

The envelope of an arrangement of lines is the polygon consisting of the finite length segments that bound the infinite faces of the arrangement. In the first part of this thesis, we study the geometry of envelope polygons (simple polygons which are the envelope of some arrangement). We show that envelope polygons are L-convex and derive several geometric properties of envelopes. Also, given an envelope polygon P, we show how to sort by slope in linear time the lines colinear with the edges of P. Using this result, we give a linear time procedure to verify if a given polygon is an envelope polygon. In the second part of this thesis, we introduce a hierarchy of classes of arrangements of lines based on the number of convex vertices of their envelopes. In particular, we look at a class called *sail arrangements* which we prove has properties that enable us to solve a number of problems optimally. Given a sail arrangement consisting of n lines (and $O(n^2)$ vertices), we show how the prune and search technique can be used to determine all the convex vertices of its envelope in O(n) time. This implies that the intersection point with minimum or maximum x-coordinate, the diameter and the convex hull of sail arrangements (problems that have $\Omega(n \log n)$ complexity for arbitrary arrangements) can also be found in O(n) time. We show, in spite of this, that the problem of constructing the full envelope of a sail arrangement still has a lower bound of $\Omega(n \log n)$. Finally, we examine the existence of hamiltonian circuits through the intersection points of a non-trivial subclass of sail arrangements.

Résumé

L'envelope d'un arrangement de lignes est le polygone formé par l'union des segments bornés des faces infinies de l'arrangement. Dans la première partie de cette thèse, nous étudions la géométrie des envelopes simples. Nous démontrons que les envelopes sont L-convexes et nous en donnons plusieurs propriétés. De plus, donné une envelope P, nous montrons comment trier par ordre de pente les lignes colinéaires aux arêtes de P en temps linéaire. Nous utilisons ce résultat afin de déterminer en temps linéaire si un polygone simple est une envelope. Dans la seconde partie de cette thèse, nous introduisons une hiérarchie de classes d'arrangements de lignes basées sur le nombre de sommets convexes de leurs envelopes. En particulier, nous étudions les propriétés de la classe d'arrangements *sail*. Etant donné les n lignes donc $O(n^2)$ points d'intersection) d'un arrangement sail, nous montrons comment la technique *prune-and-search* peut être utilisée pour trouver tous les sommets convexes de son envelope en temps O(n). Ceci nous permet de trouver les points avec les coordonées d'abscisse minimum et maximum, le diamètre et l'envelope convexe des points d'intersection de l'arrangement (ces problèmes ont une complexité de $\Omega(n \log n)$)en temps O(n). Nous montrons, en dépit de ce résultat, qu'il faut au moins $\Omega(n \log n)$ temps pour construire l'envelope d'un arrangement voile. Nous examinons aussi l'existence de circuits hamiltoniens à travers les points d'intersection d'une sous-classe d'arrangements sail.

Acknowledgment

J'aimerais d'abord remercier mes parents, sans le support desquels je n'aurais jamais pû compléter cette thèse. Mes sincères remerciements à Sue Whitesides, qui m'a fait croire en mes capacitésde faire des études graduées et m'a ainsi incité à poursuivre mes études. A Godfried Toussaint, mon superviseur de thèse, pour m'avoir introduit à la géométrie algorithmique et pour sa patience avec moi. A David Eu, pour avoir partagé avec moi tous les hauts et les bas de la vie d'étudiants gradués pour les deux dernières années. Je remercie David Avis, Prosenjit Bose, Elsa Omaña-Pulido, Clark Verbrugge et Binhai Zhu en particulier, et tous les autres membres du département, pour leur amitié et pour avoir fait de McGill un endroit agréable où travailler. Merci à Franca, Lise et Lorraine pour le support administratif et pour avoir gardé le sourire en tout temps. Je remercie également le Fonds FCAR ainsi que SOCS pour leur support financier indispensable.

All the results in this thesis, unless otherwise specified, are original contributions to knowledge. The results of sections 1.3 and 1.4 were obtained in collaboration with Jack Snoeyink of the University of British Columbia. They were accepted under the title "Recognizing an Envelope of Lines in Linear Time" by the Third Annual International Symposium on Algorithms and Computation in Nagoya, Japan (December 16-18, 1992). Most of the other results were obtained in collaboration with David Eu and Godfried Toussaint and presented under the title "On Classes of Arrangements of Lines"to the 4th Canadian Conference in Computational Geometry in St-John's, Newfoundland (August 9-14, 1992). I would like to extend my appreciation to Pinaki Mitra for his helpful suggestions. The proof of lemma 1.2.4 is due to Patrice Belleville.

A special thanks to Fang Fang for bringing sunshine to the past ten months of my life.

Contents

Introduction

The *arrangement* of a set of n lines in the plane ($\Re^2$) is the partition of the plane induced by the lines into $O(n^2)$ vertices, $O(n^2)$ edges and $O(n^2)$ faces[1]. The *vertices* are the intersection points of the lines, the *edges* are the connected components of the lines when the vertices are removed and the *faces* are the connected components of $\Re^2$ when the lines are removed. The *envelope* of an arrangement A, denoted as E(A), is the polygon formed by the union of all the bounded edges of the unbounded faces of A. One can imagine taking the arrangement A, removing segments that extend to infinity, and tracing around the outer boundary of the resulting figure to form a polygon in which adjacent segments that belong to the same line are collapsed into a single edge (see figure I.1 for an example of an envelope). In 1985, Ching and Lee [CL85] established an $\Omega(n \log n)$ lower bound for the problem of computing the envelope of an arrangement of n lines under the algebraic tree model of computation [BO83]. Since then, Suri [Su85], Vegter [Ve87] and Keil [Ke91] have contributed algorithms that achieve this bound. It is worth noting that Keil's simple and elegant algorithm [Ke91] runs in O(n) time if the n lines of the arrangement are given in sorted order of slope.

In this thesis, we are interested in understanding more about the geometry of envelopes. In [To91], Toussaint mentions that the analysis of morphological properties of arrangements of lines (also referred to as line patterns) is of considerable interest to geographers, nuclear physicists and urban planners among others (see [To91] for references). In studying envelopes, which contain a great deal of information about arrangements, we hope to gain a better understanding of the morphology of arrangements and perhaps stim-

1. In this thesis, when there is no ambiguity in the context, we will also use the expression *arrangement of lines* (by abuse of notation) to also mean the set of lines that induces the arrangement.

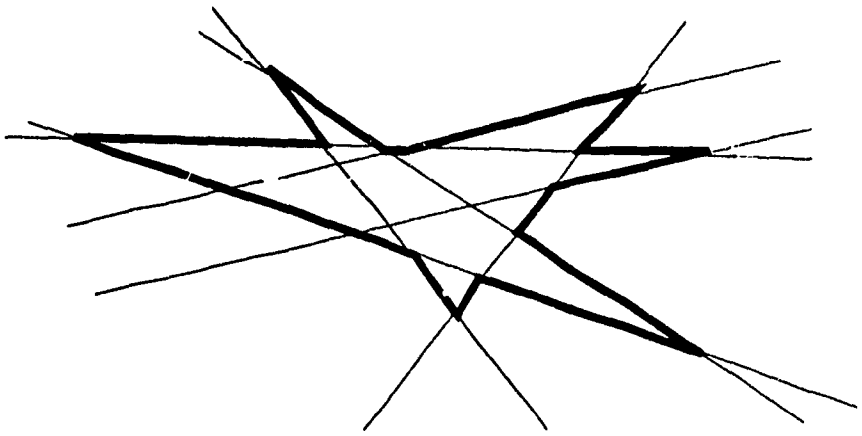


Figure I.1 The envelope of an arrangement of lines

ulate new ideas of research on the subject to the benefit of the aforementioned specialists. At the same time we contribute to the field of computational geometry by introducing and studying a very structured — and exciting — new class of polygons; that of envelope polygons (simple polygons which are the envelope of some arrangement). We also show that properties of envelopes allow us to solve some problems for arrangements of lines of a certain class (defined by the properties of their envelopes) more efficiently than for arbitrary arrangements of lines.

More precisely, we begin chapter 1 by proving several interesting and fundamental properties of envelope polygons. These properties are useful in proving subsequent results. Then, we examine the relationship between the class of envelope polygons and some other well-known classes of polygons: convex, star-shaped and L-convex polygons.

In section 1.3, we present an algorithm to sort by slope in O(n) time the n edges of an envelope polygon (the problem is $\Omega(n \log n)$ for arbitrary polygons). This algorithm, together with the above mentioned algorithm of Keil [Ke91] allows us to recognize in linear time if a given polygon is an envelope or not. Given an arbitrary polygon P of n edges, we run our edge sorting algorithm on it (it is guaranteed to always terminate in O(n) time). If the lines (edges) of the output are not sorted, the input could not have been an envelope polygon. Otherwise, we run Keil's algorithm on the sorted set of lines A to construct the envelope E(A). We can then easily check in linear time that P=E(A), the condition for a polygon to be an envelope. Our sorting algorithm is closely related to the well-known Gra-

ham scan [Gr72]. This characterization of envelope polygons concludes our first chapter.

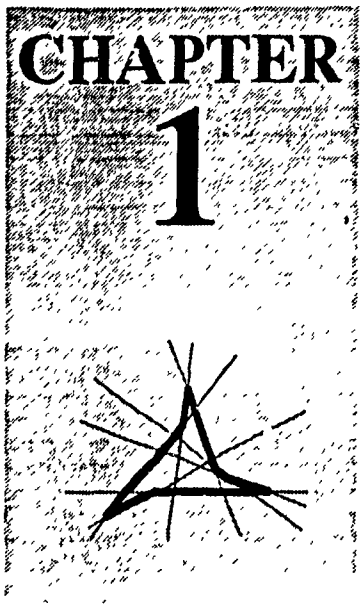
In chapter 2, we study envelopes from a different perspective. We show that given an arrangement of lines of which we know some property of its envelope, we can solve some problems more efficiently than for arbitrary arrangements. In fact, we introduce a hierarchy of classes of arrangements of lines based on the number of convex vertices of their envelopes. This approach has proven to be a productive exploratory strategy in the field of computational geometry as witnessed in [ET89], where much progress was made as a result of defining a hierarchy of polygons that possess more structure than arbitrary simple polygons. The classification of arrangements that we give helps us in understanding more about morphological properties of arrangements of lines and about the complexity of problems concerning arrangements of lines.

We show that a certain class of arrangements, which we call *sail arrangements* (the envelope of a sail arrangement has exactly three convex vertices), has properties that allow us to determine the convex vertices of sail envelopes in O(n) time given a sail arrangement of n lines. Consequently, we can solve several other problems regarding the $O(n^2)$ intersection points of a sail arrangement in O(n) time. In particular we can find the intersection point with minimum or maximum x-coordinate as well as the diameter and convex hull of the intersection points. These problems were shown, in [CSSS89] (in the dual) and [CL85] respectively, to have $\Omega(n \log n)$ lower bounds for arbitrary arrangements under the algebraic tree computation model [BO83]. We show, in spite of this, that even if we know the convex vertices of the envelope of a sail arrangement, computing the complete envelope remains $\Omega(n \log n)$.

We also show that there are classes of arrangements which have other properties that do not hold for arbitrary arrangements. In particular, we show that there exists a nontrivial subclass of sail arrangements which always admits a hamiltonian cycle, and furthermore, we exhibit a polynomial time algorithm to compute a hamiltonian cycle. It was shown by Everett [Ev91] that not all arrangements of lines admit a hamiltonian cycle; refer to figure 2.4.1 (page 46) for an illustration of a counterexample.

The algorithms presented in this thesis use the Real RAM model of computation. All numbers computed have infinite precision. They can be stored in O(1) space, compar-

isons and arithmetic operations on numbers can be accomplished in O(1) time. This implies, for example, that we can compute the intersection of two non-parallel lines in O(1) time, using O(1) space. For the lower bound proofs, we use the algebraic tree model of computation of Ben-Or [BO83]. Throughout the thesis, lines are given as a pair (slope, y-intercept) and edges as (initial vertex, end vertex). Polygons and polygonal chains are stored in doubly-linked lists in clockwise order. Arrangements of lines are given as doubly-linked lists of lines. All the work in this thesis is done in the Euclidean plane.



The Geometry of Envelopes

Consider the following definition. We say that a polygon $P=[p_0,...,p_{n-1}]$ is *simple* if (i) no two non-consecutive edges of P intersect and if (ii) no three consecutive vertices of P are colinear. All polygons considered in this thesis are simple. Most of the work on polygons in computational geometry is done with simple polygons. Perhaps the most important property of simple polygons is that they obey the Jordan Curve Theorem: when a simple polygon P is removed from $\Re^2$, there remains a bounded and an unbounded connected component (respectively called the *interior* and the *exterior* of P). While the formal proof of the Jordan Curve Theorem is complicated for general curves, the reader is referred to [CR41], where a simple proof is given for the case of polygonal curves.

In this chapter, we are concerned with the study of geometric properties of a class of simple polygons, that of envelope polygons. A simple polygon P is an *envelope polygon* if there exists an arrangement of lines A such that P=E(A). It is useful, when discussing envelope polygons, to use the following notation. The *induced arrangement* of a simple polygon P, denoted as IA(P), is the arrangement induced by the set of lines obtained by extending the edges of P to lines. First we give a characterization of envelope polygons.

Proposition 1. A simple polygon P is an envelope polygon if and only if P=E(IA(P)).

Proof: ("Only if" implication) Suppose that simple polygon P is an envelope polygon. Let A denote the arrangement of lines such that E(A)=P. For every edge e of P, there is a line L of IA(P) colinear to e, hence, since P=E(A), L is a line of A (i.e. IA(P) $\subset$ A). Now suppose that there is a line L of A not in IA(P). L is not colinear to any edge of P (=E(A)) and therefore does not contribute any line segment to E(A). L can be removed from A without affecting E(A). It follows that E(IA(P)) = E(A), hence that P=E(IA(P)). The "if" implication of the proposition is obvious.♣

This characterization of envelope polygons will be very useful in determining if a given polygon is an envelope polygon (section 1.4). We begin the chapter by introducing some basic terms we will be using throughout this thesis and by uncovering simple and basic properties of envelope polygons.

1.1. Definitions and Preliminaries

In this section, we present several basic geometric properties of envelope polygons. We start by introducing several definitions and forms of notation about arrangements of lines that will facilitate subsequent discussion.

Let A = $\{L_0,L_1,...,L_{n-1}\}$ be an arrangement of n lines. We denote by $I(L_i,L_j)$ the intersection of two non-parallel lines L_i and L_j. We now classify the vertices of arrangement A as follows. Vertex p = $I(L_i,L_j)$ (i, j $\in$ [0,n-1]) is said to be *extreme* on L_i if every vertex lying on L_i lies on one side of p on L_i. The vertex p is said to be *critical* if it is extreme on both L_i and L_j; it is *interior* if it is not extreme on either L_i or L_j. Two non-parallel lines in an arrangement are said to be *adjacent* if they are neighbors in the list of the lines of the arrangement sorted by slope. We now characterize the convex vertices of envelope polygons with the following lemma.

Lemma 1.1.1. Let P be an envelope polygon. A vertex of P is convex if and only if it is a critical vertex of IA(P).

Proof: We first prove the "only if" part of the lemma. Let P = $[p_0,p_1,...,p_{n-1}]$ be an envelope

polygon and let p_i be a convex vertex of P. Let edges e_i and e_{i+1} be the edges that are incident on p_i. Let L_i and L_{i+1} be the lines extended through e_i and e_{i+1} respectively. If p_i is not critical, then p_i is not extreme on at least one of L_i and L_{i+1}. Without loss of generality, suppose that p_i is not extreme on L_i in IA(P). Then there is a vertex p_j of IA(P) on L_i such that p_i lies between p_{i-1} and p_j on L_i. The line segment $[p_{i-1},p_j]$ must lie completely inside P, by the definition of an envelope. Thus, p_i cannot be convex, a contradiction. Hence p_i must be critical.

To see that the "if" part of the lemma holds, suppose that p_i is concave. Let edges e_i and e_{i+1} be incident on p_i. We extend e_i and e_{i+1} to get L_i and L_{i+1} respectively. The vertex chain $C = [p_{i+1},p_{i+2},...,p_{i-1}]$ closes P. By the Jordan curve theorem, L_i and L_{i+1} must each intersect the boundary of P on C at extreme points say, p_a and p_b respectively. The colinearity and orientation of p_{i-1},p_i,p_a and p_{i+1},p_i,p_b imply that p_i is not extreme on either L_i or L_{i+1}. Therefore p_i is not critical.♣

Corollary 1.1.2. If P is an envelope polygon then the convex vertices of P are the intersection points of adjacent lines in the list of lines of IA(P) given in sorted order of slope.

Proof: Ching and Lee [CL85] and independently Atallah [At86] showed that the critical vertices of an arrangement are a subset of the intersections of adjacent lines.♣

The following basic lemma about envelope polygons will be useful in section 1.3.

Lemma 1.1.3. Let P be an envelope polygon. At most two distinct lines of IA(P) are parallel.

Proof: Suppose that there are three distinct parallel lines L_a, L_b, L_c in IA(P) and that L_b is the middle one. Then there is some edge e of P on L_b. The endpoints of e are formed by intersection with two other lines, which form bounded faces of IA(P) together with L_a and L_c on each side of e. This contradicts the fact that e is an edge of envelope polygon P.♣

In [Su85], Suri shows that the size of an envelope of an arrangement of n lines is at most 4n-2. We notice that the recent results of [BEPY91] for the horizon theorem also give tight bounds on the complexity of a zone, hence of an envelope. Since a formal discussion on the relation of the horizon theorem to envelopes would be tedious and technical, we prefer to simply highlight the equivalence.

Lemma 1.1.4. Let P be an envelope polygon such that IA(P) consists of n lines ($n \geq 3$). Then P has at most 3.5n vertices or edges, and this bound is tight up to a constant.

Proof: Envelopes are projectively equivalent to the well-studied horizons or zones of lines in an arrangement of lines. Specifically, the envelope of a set of lines is the zone or horizon of the line at infinity. Thus, the recent bound for the horizon theorem [BEPY91] also gives an upper bound of 3.5n vertices on the complexity of envelope polygons. We present the following example due to Urrutia [Ur91] (figure 1.1.1) to show that the bound is tight up to a constant. ♣

This lemma concludes our presentation of some basic properties of envelope polygons. We will use those properties in the sections to come. Now, we consider an important characteristic of envelope polygons.

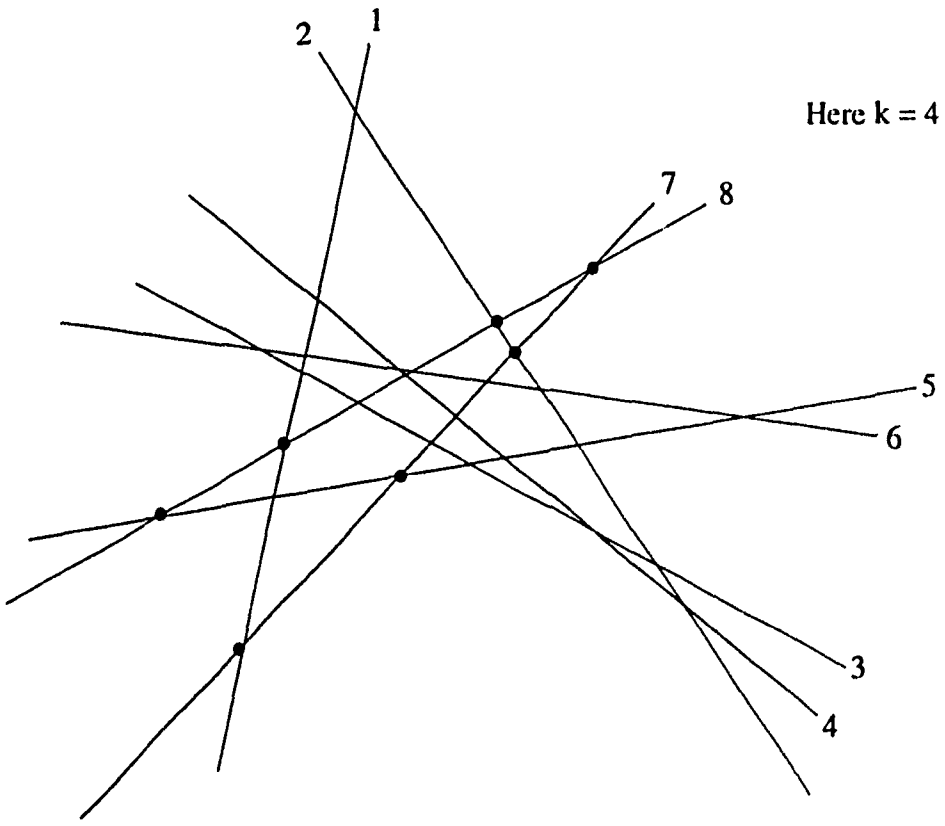


Figure 1.1.1. An arrangement of n lines whose envelope has 3.5n-10 vertices (n even). Lines 2k-1 and 2k add 7 new vertices to the envelope (k>2).

1.2. Envelope Polygons are L-convex

In this section we establish that envelope polygons are L-convex. We strengthen this result by first showing that non-trivial envelopes are not convex.

Lemma 1.2.1. Let P be an envelope polygon of more than four vertices. Then P is not a convex polygon.

Proof: Suppose that $P = [p_0,p_1,...,p_{n-1}]$ is a simple convex polygon of n (> 4) vertices. We first show that P has three consecutive edges, no two of which are parallel. Let e_i denote the edge $[p_i,p_{i+1}]$ and L_i the line colinear with e_i. Suppose, for the sake of deriving a contradiction, that for every triple of consecutive edges of P there are two parallel edges. Since P is simple, then no two consecutive edges of P are colinear. It follows that $(e_0,e_2,e_4,e_6,...)$ are all parallel and that $(e_1,e_3,e_5,e_7,...)$ are parallel. By lemma 1.1.3, at most two distinct lines of IA(P) are parallel since P is an envelope polygon. It follows that IA(P) consists of at most 4 lines, contradicting the assumption that P has more than 4 vertices.

P must thus have three consecutive edges, no two of which are parallel. Let e_i, e_{i+1} and e_{i+2} be three such edges. By lemma 1.1.1 p_i, p_{i+1}, p_{i+2} and p_{i+3} are critical in IA(P) since they are convex in P. Then since e_i and e_{i+2} are not parallel, L_i must intersect L_{i+2} on e_i and on e_{i+2}. This implies that p_i and p_{i+3} are the same vertex, contradicting the assumption that P has more than three vertices. P can thus not be convex.♣

Corollary 1.2.2. The only convex envelope polygons are triangles and parallelograms; they are the envelope of trivial arrangements.

Given a simple polygon P, we say that two points x and y inside P are *visible* if the line segment [x,y] lies completely inside P. A simple polygon P is *star-shaped* if there is a point x inside P such that for every point y in P, x and y are visible. P is a *fan* polygon if it is star-shaped from a vertex. P is a *convex-fan* if it is star-shaped from a convex vertex. In [Za75], Zaslavsky conjectured that envelope polygons are star-shaped; the class of star-shaped polygons subsumes the class of convex polygons. We submit the counterexample below (figure 1.2.1) due to Vegter [Ve87] to show that this is not the case. We can, however,

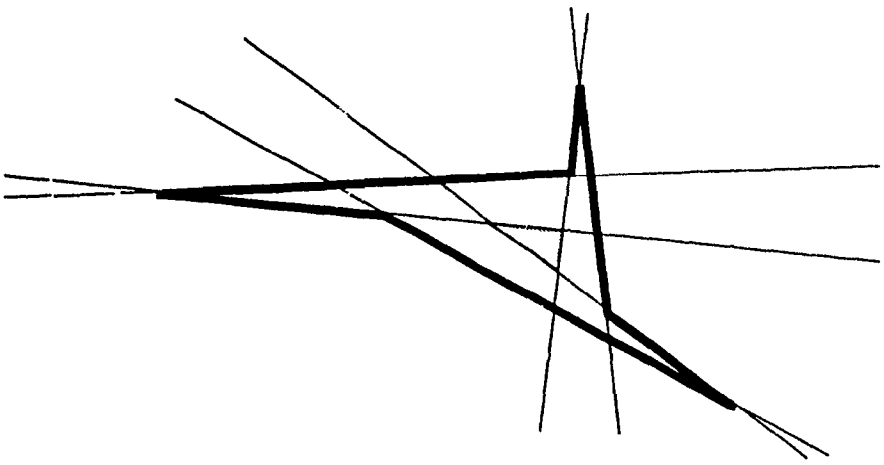


Figure 1.2.1. An envelope polygon which is not star-shaped.

show that envelope polygons are L-convex.

We define L-convexity as an instance of L_k-convexity as follows. A simple polygon P is *L_k-convex* ($k \geq 1$) if for every two points x and y that lie inside P, there exists a set of k-1 distinct points $Q = \{q_1,q_2,...,q_{k-1}\}$ in P such that the k pairs (x,q_1), (q_1,q_2),...,(q_{k-2},q_{k-1}),(q_{k-1},y) are all visible. Note that for $k = 1$, the set Q is empty and we obtain the classical definition of a convex polygon.We can think of Q as a polygonal chain of k line segments that lies in P and has x and y as endpoints. If a set Q exists, we say that x and y have *link distance* k with path Q. We say that x and y have *minimal link distance* k if k is minimal over all possible link paths between x and y. We say that a polygon P is *L-convex* if it is L_2-convex. Before we show that envelope polygons are L-convex, we first recall the following fact.

Lemma 1.2.3. A simple polygon P is L_k-convex if and only if the minimal link distance between every two *vertices* of P is at most k.

Proof: The proof follows directly from lemma D in [LPSSSSTWY88].♣

We can now state the following.

Lemma 1.2.4. If P is an envelope polygon, then P is L-convex.

Proof: By lemma 1.2.3, it is sufficient to show that every two vertices of P have a link dis-

tance at most 2. Let p_a and p_b be two arbitrary vertices of P. If p_a and p_b are on the same line in IA(P), then they are visible and so p_a and p_b have a link distance of one. Otherwise, there is an intersection point q between a line L_a supporting p_a and a line L_b supporting p_b in IA(P). The path $[p_a,q]$, $[q,p_b]$ is interior to P since it is an envelope and so p_a and p_b have a link distance of 2.♣

Whereas envelope polygons have not been studied much, L-convex sets have received considerable attention [HV49]. Properties of L-convex polygons have been exploited in [EAT83] to obtain efficient algorithms for solving a variety of geometric problems. Since envelopes are L-convex, properties of L-convex polygons are useful for answering questions about envelope polygons. In particular, this result implies that there is a simple O(n) time algorithm to triangulate an envelope polygon of n vertices [EAT83].

1.3. Sorting the Edges of an Envelope Polygon in Linear Time

In this section, we consider the problem of sorting by slope the edges of a given polygon. We will show that this can be done in O(n) time if the input is an envelope polygon of n vertices. We will see in section 1.4 that this result has an interesting application for the problem of determining if a given simple polygon is an envelope polygon. We start by showing that the problem of sorting the edges of a polygon is $\Omega(n \log n)$ for monotonic convex-fan polygons [OR87] (hence for arbitrary polygons); the class of monotonic convex-fan polygons has a non-empty intersection with the class of envelope polygons.

Our proof of this lower bound requires the following definitions. A polygonal chain $C=[r_u,r_{u+1},...,r_v]$ is said to be *monotone* with respect to a straight line L if the perpendicular projections of the vertices of C on L are ordered as $(r_u,r_{u+1},...,r_v)$ (note that we do not allow consecutive vertices of C to be projected on the same point on L). A polygon P is *monotone* if it can be split in two chains monotone with respect to a common line. Let L be a line in the plane. Define *slope(L)* as the smallest angle through which L must be rotated clockwise about a point on L so that L is parallel to the x-axis. Note that this definition of slope is not standard and will be used frequently in the sections to come.

Lemma 1.3.1. Given an arbitrary monotonic convex-fan polygon P of n vertices, it takes $\Omega(n \log n)$ time to sort the edges of P in order of slope under the algebraic tree model of computation.

Proof: (By a reduction from Integer Sorting) Let $X = \{x_0, \ldots, x_{n-1}\}$, a set of positive integers, be the input to the Integer Sorting problem. We will first show how to build in linear time a comb-like polygonal chain C from X (refer to figure 1.3.1 for an illustration of the construction). For every x_i, produce the two line segments $[(0,2i), (1/x_i,2i+1)]$ and $[(1/x_i,2i+1), (0,2i+2)]$ which respectively have slopes x_i ($\neq 0$) and $\pi - x_i$, and insert them in C. At the same time, extend those edges to lines in order to compute the intersection of those lines furthest from (0,0) with the positive x-axis (suppose that this intersection is (x,0)). To create a star-shaped polygon P, add vertex (x+1,0) to C. C is monotone with respect to the y-axis, and since (x+1,0) lies in the region which is the intersection of all the right half-planes determined by the lines, then it can see all the vertices created and therefore the entire interior of P as well. By this construction the polygon P is simple, monotonic convex-fan, and produced from X in O(n) time. Getting the edges of this polygon sorted by order of slope sorts the x_i's, hence takes $\Omega(n \log n)$ time under the algebraic tree model of computation.♣

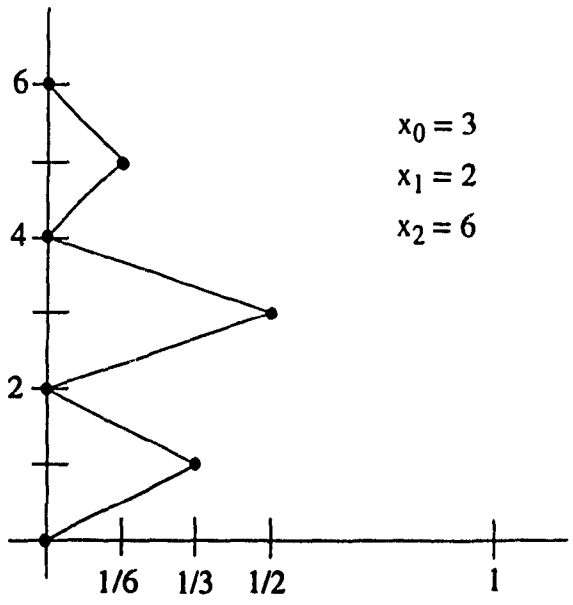


Figure 1.3.1. Constructing the comb-like polygonal chain in the proof of lemma 1.3.1

Now suppose that we are given an envelope polygon P of n vertices. We show how the edges of P can be sorted in O(n) time by order of slope. Our strategy is to represent P as the intersection of two unbounded regions D and U that we define as follows. Choose an edge e of P and let L denote the line colinear with e (assume without loss of generality that L is the x-axis). Then let D be the union of P with the halfplane on and below L and let U be the union of P with the halfplane on and above L.

In what follows, we concentrate on sorting the edges of the polygonal chain delimiting the unbounded region D: $[r_0=(-\infty,0),r_1,...,r_{m-1},r_m=(\infty,0)]$ (r_1 and r_{m-1} are the two extreme vertices of IA(P) on L); sorting the edges of the boundary of U is similar. To sort the edges of the boundary of D (which are edges of P) we use two Graham scans, one clockwise (cw) from r_0, procedure `cwScan()`, and one counterclockwise (ccw) from r_m, procedure `ccwScan()`. The edges output by each scan will be stored in order of slope in linked lists Q_1 and Q_2 respectively; they can easily be merged in linear time afterwards. Q_1 and Q_2 together hold almost all the edges of the boundary of D. The only edges that our algorithm will not detect are those which are parallel to L. However, they can clearly be detected and inserted in our sorted list of edges in O(n) time once L is known. Henceforth, we assume that all edges parallel to e are colinear to L to simplify presentation. We now present procedure `cwScan()`, that we run clockwise on D, starting from r_0; procedure `ccwScan()` is similar, and therefore will be omitted.

Procedure `cwScan(D,`r_0`)` /* in clockwise direction*/

{Input: A polygon D and vertex r_0, obtained from an envelope polygon P as described above}

{Data Structure: A vertex stack with pointers *top* and *next* to the top and next to top elements of the stack.}

{Output: A doubly linked list Q_1 which is a subset of the edges of D in sorted order of slope.}

begin

push(r_0);

push(r_1);

$r_{cur} \leftarrow r_2$;

Repeat

```
    If (next, top, r_cur) form a left turn or are colinear
    then push(r_cur); r_cur ← r_cur+1; /* move forward */
    else /* the 3 points form a right turn */
        /* Insert an edge of D in Q1 when it is popped from the stack */
        if ([next, top] is an edge of D) then insert [next, top] in Q1;
        pop(); /* backtrack */
Until (r_cur = r0)
end;
end {of Algorithm cwScan}
```

Before analyzing the running time and the correctness of this algorithm, let us consider figure 1.3.2. It shows an example of the polygon D obtained as described above and the order that the lines are output by procedures cwScan (1, 2, 3, 4) and ccwScan (a, b, c, d). We first show that the algorithm works in linear time given an arbitrary polygon as input.

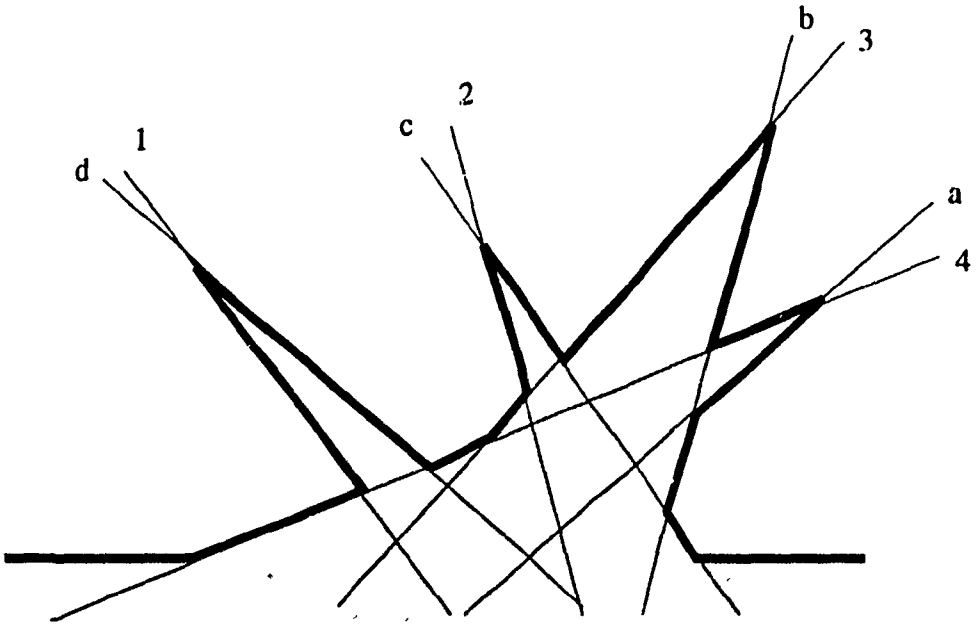


Figure 1.3.2. The unbounded region D.
cwScan outputs lines 1, 2, 3, 4. ccwScan outputs lines a, b, c, d.

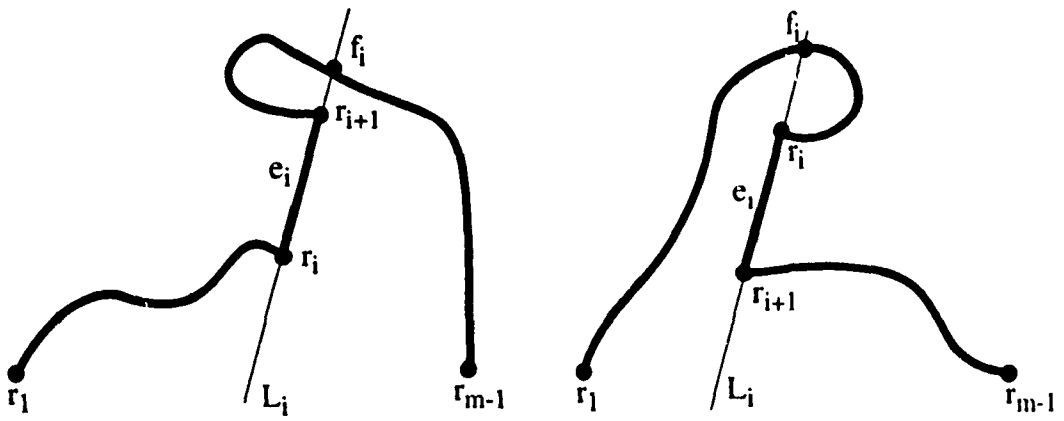
Lemma 1.3.2. The procedure `cwScan()` performs at most O(m) steps given an *arbitrary* polygon of m vertices as input.

Proof: We can determine in O(1) time if [*next, top*] is an edge of D by checking whether the indices of the vertices *next* and *top* are consecutive. The angle test (left or right turn) can be performed in a constant number of operations under the Real RAM model of computation. After each test, we either push a vertex (advance the scan) or pop a vertex (backtrack). Clearly, since it is the stopping condition, the scan advances exactly m times (every vertex is pushed exactly once; they are pushed in clockwise order). Also, since every vertex can be popped at most once from the stack then the algorithm backtracks at most m times.♣

We say that an edge $e_i = [r_i,r_{i+1}]$ of D is *visited* by the scan if at one point in the algorithm r_i and r_{i+1} are consecutive on top of the stack. We say that an edge is *popped* by the scan if it was visited and one of its endpoints is now popped from the stack. We now prove the correctness of the algorithm when it is given as input $D= [r_0,r_1,...,r_m]$ obtained from an envelope, as described above, and vertex r_0. Let L_i denote the line colinear with edge $e_i=[r_i,r_{i+1}]$ of D and let s denote the segment $[r_1,r_{m-1}]$ on L (the line used to split P in two). The proof that the scan correctly sorts some of the edges of an envelope polygon is based on the following property: if P is an envelope and L_i is a line of IA(P), then the intersection of P and L_i is a unique connected line segment (let $s_i=[f_i,g_i]$ denote that segment on line L_i) i.e. every edge e_i of D not parallel to L must satisfy the following two properties: (1) the extension of e_i towards L remains in D and (2) the extension of e_i away from L crosses out of D and then intersects no line of D (without loss of generality, suppose f_i is above L; then clearly g_i must lie below L— the x-axis).

A simple polygon P is *edge-visible* from an edge e of P if for every point x inside P, there is a point y on e such that x and y are visible. Property (1) above of the unbounded region D is similar to edge-visibility from a line at infinity. Edge-visibility was introduced by Toussaint and Avis in [TA82], where they show that a Graham scan gives an easy algorithm to triangulate edge-visible polygons; our algorithm is closely related to theirs. We will use some of their results in our proof of correctness.

The proof of correctness of the sorting procedure will follow from lemmata 1.3.5 and 1.3.9 which establish that every edge of D (not parallel to L) is visited either by the



e_i lies on the chain from r_1 to f_i. e_i lies on the chain from f_i to r_{m-1}.

Figure 1.3.3. Illustrating the statement of lemmata 1.3.3. and 1.3.4.

clockwise or the counterclockwise scan, and that the edges, in both cases, are popped by the scan in order of slope. The following two technical lemmata, which exploit properties (1) and (2) above, are useful in establishing the correctness of the scan. Refer to figure 1.3.3 for an illustration of their statements.

Lemma 1.3.3. Let e_i be an edge of D not parallel to L. Then no vertex of the clockwise chain from r_1 to f_i lies to the right of L_i and no vertex of the clockwise chain from f_i to r_{m-1} lies to the left of L_i.

Proof: By properties (1) and (2) above, $s_i=[f_i,g_i]$ is contained entirely in D (f_i lies above L). The clockwise chain from r_1 to r_{m-1} (entirely above or on L) can only cross L_i at f_i. It follows that no vertex of the chain from r_1 to f_i can lie to the right of L_i and that no vertex of the chain from f_i to r_{m-1} lies to the left of L_i.♣

Lemma 1.3.4. If edge $e_i=[r_i,r_{i+1}]$ (not parallel to L) lies on the clockwise chain from r_1 to f_i, then r_{i+1} lies further from L than r_i. If e_i lies on the clockwise chain from f_i to r_{m-1}, then r_i lies further from L than r_{i+1}.

Proof: By contradiction. Suppose that edge $e_i=[r_i,r_{i+1}]$ lies on the clockwise chain from r_1 to f_i and assume that r_i lies further from L than r_{i+1} on L_i. By lemma 1.3.3 and by the definition of D, no vertex of the chain from r_1 to f_i lies to the right of L_i or below L. In partic-

ular, it must hold for the chain from r_1 to r_i. That chain together with $I(L,L_i)$ forms a closed polygon containing r_{i+1} (on $[r_i,I(L,L_i)]$). Since s_i lies entirely inside D (properties (1) and (2)), then the chain from r_{i+1} to f_i must intersect with the chain from r_1 to r_i, contradicting the simplicity of the clockwise chain from r_1 to r_{m-1}. The symmetric statement follows from the same argument.♣

The following lemma uses lemma 1.3.3 and lemma 1.3.4 to establish that the edges of D not parallel to L are all visited and popped by the clockwise or by the counterclockwise scan. These facts establish that Q_1 and Q_2 together contain all the edges of D (except those colinear with L).

Lemma 1.3.5. Let $e_i=[r_i,r_{i+1}]$ be any edge of D (not parallel to L). Then e_i is visited and popped by the scan started clockwise from r_0 or by the scan started counterclockwise from r_m.

Proof: Suppose, without loss of generality that e_i lies on the clockwise chain from r_1 to f_i. We know from lemma 1.3.3 that no vertex of the chain from r_1 to f_i lies to the right of L_i, hence that e_i lies on the convex hull of the chain from r_1 to r_{i+1}. Since r_{i+1} lies above r_i on L_i (lemma 1.3.4) then when r_i is pushed on the stack, (*next*, *top* = r_i, $r_{cur} = r_{i+1}$) also forms a left turn (*next* must be on the chain from r_1 to r_{i-1} since the scan progresses in clockwise order from r_0) i.e. e_i is visited by a clockwise scan from r_0. Edge e_i will be popped when r_{cur} for the first time, will be on the chain from f_i to r_{m-1}. If e_i lies on the clockwise chain from r_1 to f_i, then by a similar argument it is visited and popped by a counterclockwise scan from r_m.♣

To complete the proof of correctness, we need to argue that in the scan, the edges are inserted in Q_1 in order of slope. The following lemma and its corollaries will serve that purpose.

Lemma 1.3.6. When the scan runs on D, then at every step the content of the stack forms a simple convex polygonal chain monotone (non-decreasing) with respect to the y-axis.

Proof: By property (1) above, the polygon D is edge-visible from a line at infinity. i.e. every vertex of D can be joined to the line $y = -\infty$ by a segment that lies entirely inside D. The

statement of the lemma follows directly from lemma 1 in [TA82]. Since the proof is quite long, it will not be reproduced here.♣

Corollary 1.3.7. The content of the stack forms a convex chain of edges sorted in increasing order of slope after each loop.

Proof: Follows directly from the fact that the stack chain is made of left turns and that it is monotone with respect to the y-axis.♣

Corollary 1.3.8. If edge $e_i=[r_i,r_{i+1}]$ of D lies on the clockwise chain from f_i to r_m, then e_i is not visited by a `cwScan`. If e_i lies on the clockwise chain from r_1 to f_i, then e_i is not visited by `ccwScan`.

Proof: Suppose that edge $e_i=[r_i,r_{i+1}]$ lies on the clockwise chain from f_i to r_m. By lemma 1.3.4, r_i lies further from L than r_{i+1}, hence has greater y-coordinate. If e_i is visited by a clockwise scan then r_{i+1} lies above r_i on the stack, contradicting lemma 1.3.6. The argument is the same if e_i lies on the clockwise chain from r_1 to f_i (with respect to a counterclockwise scan).♣

We now complete the proof of correctness with the following.

Lemma 1.3.9. The edges visited by the procedure `cwScan` are inserted in Q_1 in decreasing order of slope. The edges visited by `ccwScan` are inserted in Q_2 in increasing order of slope.

Proof: Suppose that we run `cwScan` on D in a clockwise direction, starting from r_0 (remember that our definition of slope is not standard, see page 14). We show that all edges popped (hence inserted in Q_1) before a given edge e_i is popped have greater slope than e_i. First consider the edges popped after e_i has been visited and before e_i is popped (hence while e_i is on the stack). Such an edge e_j is visited after e_i since it is popped before, hence it is on top of e_i in the stack when e_j is visited. By corollary 1.3.7, e_j has greater slope than e_i.

Now suppose, for the sake of deriving a contradiction, that an edge e_j popped before e_i is visited has smaller slope. Let r_k be the vertex which pops e_j (r_1, r_j, r_{j+1}, r_k, r_i, r_{i+1}, r_m.

$_1$ are in clockwise order on D). We use the following three facts to derive a contradiction: (1) the chain from r_k to r_{i+1} cannot lie to the left of L_j (r_k lies to the right of L_j since (a) r_{j+1} lies above r_j on L_j (corollary 1.3.8 and lemma 1.3.4) and (b) (*next* = r_j, *top* = r_{j+1}, r_{cur} = r_k) must form a right turn to pop r_{j+1}. The fact follows from lemma 1.3.3). (2) the chain from r_k to r_{i+1} cannot lie to the right of L_i (follows from corollary 1.3.8 and lemma 1.3.3 since e_i is visited). (3) r_{i+1} must lie above r_i on L_i (follows from corollary 1.3.8 and lemma 1.3.4).

Line L must lie below both e_j and e_i (it lies below all of the chain from r_1 to r_{m-1}). Since we assumed that L_j has a smaller slope than L_i, we conclude from facts (1) and (2) above that e_i and e_j lie below $I(L_i,L_j)$ (i.e. e_i lies on $[I(L_i,L), I(L_i,L_j)]$ and e_j must lie on $[I(L_j,L), I(L_i,L_j)]$). We thus obtain a triangle T= $(I(L_i,L), I(L_j,L), I(L_i,L_j))$ lying above L. Since P is an envelope, no vertex of P can lie inside T and no edge of P (hence of D) can intersect with the interior of T. The chain from r_k to r_{i+1} cannot lie inside T and must therefore lie on L_i, to the right of L_j, following facts (1) and (2). But since the clockwise polygonal chain from r_1 to r_{m-1} is simple, then r_i must lie above r_{i+1} on L_i, contradicting fact (3). Hence e_j must have greater slope than e_i. The symmetric argument applies for a counterclockwise scan.♣

Theorem 1.3.1. The edges of an envelope polygon P of n vertices can be sorted in order of slope in O(n) time.

Proof: The proof of correctness of our algorithm follows from the previous discussion. Splitting P in D and U can clearly be done in O(n) time since P is stored in a doubly-linked list. Running the scan twice on both D and U also takes linear time (lemma 1.3.2). It follows from lemma 1.3.5 and lemma 1.3.9 that all the edges of D (U) (except those colinear with L) are sorted by the scan. Inserting the edges colinear with L and merging four sorted lists can also be done in O(n) time.♣

1.4. Recognizing an Envelope Polygon in Linear Time

As mentioned earlier, we show in this section how, given an arbitrary simple polygon of n vertices, we can determine in O(n) time if it is an envelope polygon. For this purpose, we rely both on the algorithm for sorting the edges of envelope polygons given in section 1.3 and on Keil's algorithm for computing in O(n) time the envelope of a set of lines given in sorted order of slope [Ke91]. For the sake of completeness, we now outline Keil's algorithm before presenting the general recognition algorithm. The notation we use is different from Keil's, to account for our different definition of slope (see page 14). The only special data structure needed to implement the algorithm is a stack.

Let $A = \{L_0,...,L_{n-1}\}$ be a set of n lines in the plane indexed in increasing order of slope. We show how to construct the upper portion of E(A). For i=0,...,n-1, let B_i be the convex polygonal chain bounding the intersection of the halfplanes lying to the left of the lines $L_0,...,L_i$ and let A_i be the convex polygonal chain bounding the intersection of the halfplanes lying to the right of the lines $L_{i+1},...,L_{n-1}$. The following lemma is the key result that allows us to design our algorithm.

Lemma 1.4.1. (Keil [Ke91]) Let F denote the convex polygonal chain of any unbounded face of A on the upper side of E(A). Then F is the intersection of the convex region to the left of B_i and of the convex region to the right of A_i for some i, i=0,...,n-1. The next unbounded face of A counterclockwise is the intersection of the convex region to the left of B_{i+1} and of the convex region to the right of A_{i+1} (see figure 1.4.1).

To compute the upper envelope of A in linear time using lemma 1.4.1, it is sufficient to show how to compute the following in constant amortized time: (a) the intersection of the convex region to the left of B_i and of the convex region to the right of A_i for a given value of i, (b) B_i from B_{i-1} and (c) A_i from A_{i-1}. Each unbounded face of A consists of a portion of E(A) and of two segments that extend to infinity. It is thus easy to construct the upper part of E(A) in linear time, given the unbounded faces of A in counterclockwise order (for consecutive values of i).

We first outline how to compute the boundary of the unbounded face F determined

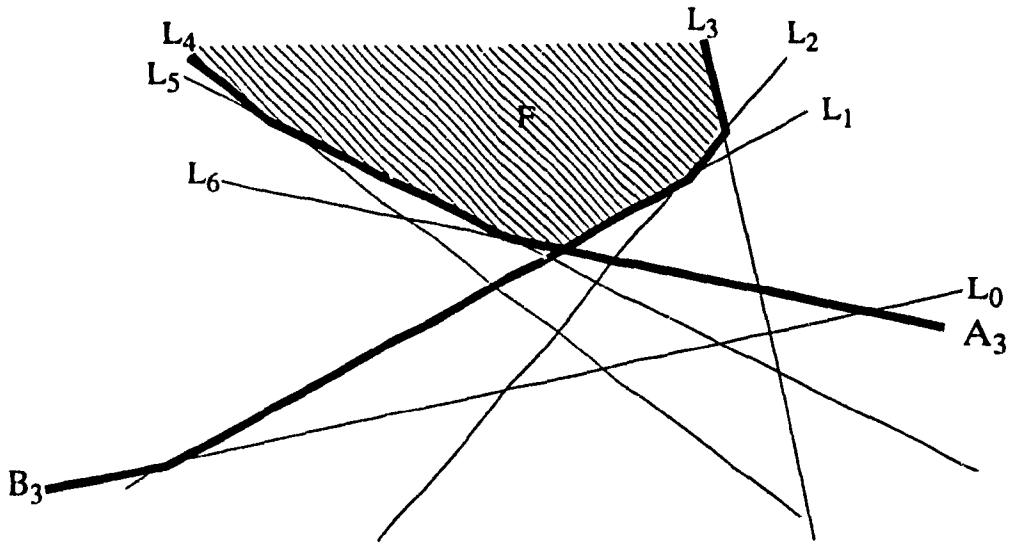


Figure 1.4.1. The polygonal chains B_3 and A_3.

by B_i and A_i. We observe that the lines containing the segments of A_i all have slopes at least as large as the lines containing the segments in B_i. Using this fact, we can compute the boundary of F with an up to down sweepline algorithm. The idea is to start from the two halflines, on from each chain, that extend infinitely upward; they are the unbounded segments of F. We then sweep a horizontal line down, comparing the two segments cut by the sweepline, until the intersection point of the two chains is found. If the segments do not intersect, they are added to the boundary of F. Since we know by lemma 1.1.3 that the size of an envelope of n lines is O(n), then the total time spent sweeping is O(n) (i.e. O(1) amortized time).

We now have to show how B_i can be obtained from B_{i-1} in constant amortized time. We represent B_{i-1} as the contents of a stack such that the j^{th} entry from the bottom of the stack consists of the j^{th} segment of B_{i-1} in order of slope. We observe that B_i is obtained by intersecting the region to the left of B_{i-1} with the region to the left of L_i. We know that the upward unbounded segment of B_i lies on L_i since L_i has a greater slope than any other line forming B_i. Using these two facts, we can simply pop the vertices of B_{i-1} from the stack while they lie to the right of L_i. We can then push the unbounded segment contained in L_i, thereby storing B_i on the stack. Because the lines are given in sorted order of slope, then all the B_i's (i=0,...,n-1) can be computed in O(n) total time (each line is pushed and popped

exactly once from the stack).

However, computing the A_i from A_{i-1} is "tricky". We first observe, as we did before, that A_{i-1} is obtained by intersecting the region to the right of A_i with the region to the right of L_i; this is the reverse of the relation between B_{i-1} and B_i. But to compute F, we need to access A_i and B_i at the same time, while we cannot afford to store A_{i+1} to A_n (we are allowed only O(n) total space). The "trick" is to run the procedure to obtain A_{i-1} from A_i (like for the B_i's) for i = n-1 down to 1, while recording every operation performed to obtain A_0. The transcript of the operations can be obtained in O(n) total time and can thus be stored in O(n) space. Then, we can compute A_i from A_{i-1} by reading the recorded operations in reverse (pushing what was popped and vice versa). With this, we finished outlining the solutions to problems (a), (b) and (c) above. From those three solutions, we can compute the upper portion of E(A) in O(n) total time, as discussed above. This concludes our presentation of Keil's algorithm. We now present the general algorithm to recognize envelope polygons in linear time.

Algorithm Recognize_Envelope(P)

{Input: An arbitrary simple polygon P stored in a doubly-linked list}

{Output: Whether or not P is an envelope.}

begin

Attempt to sort the edges of P by order of slope:

Choose any edge e of P and find the two vertices of P, r_1 and r_{m-1}, which are extreme on L (the line colinear to e).

Split P into D and U. /* O(1) since P is stored in a doubly linked list */

For D (similarly for U):

$Q_1 \leftarrow$ `cwScan(D,` r_0`)`.

$Q_2 \leftarrow$ `ccwScan(D,` r_m`)`.

Verify that the edges of Q_1 and Q_2 are sorted by slope.

If not then return(NO) /* P cannot be an envelope */

else merge Q_1 and Q_2 and add to them the edges colinear with L (they can be found in O(n) time).

Q ← merge the lists of D and U together.

A ← The lines of IA(P) in sorted order of slope obtained from Q.

Check if P = IA(P): /* the edges of P were correctly sorted by slope */

Q ← Compute the envelope of A using Keil's algorithm [Ke91].

Compare P and Q to see if they are identical:

Assume that the vertices of P and Q are given in clockwise order.

Pick any vertex of P and find if there is a vertex of Q with which it corresponds.

If yes, scan the vertices of P and Q in clockwise order, checking that each pair of vertices matches.

Return(Yes) if the vertices of both polygons correspond pairwise.

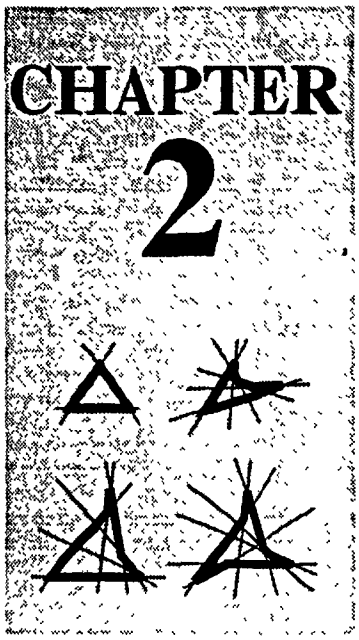
Return(No) otherwise.

end {of **Algorithm Recognize_Envelope**}

Theorem 1.4.1. Algorithm Recognize_Envelope correctly determines in O(n) time if a simple polygon is an envelope.

Proof: Suppose first that the edges of P are correctly sorted in O(n) time (by theorem 1.3.1, this is always the case if P is an envelope polygon). The correctness of Keil's algorithm [Ke91] implies that Q, the envelope of IA(P), is correctly computed in O(n) time. By lemma 1.1.3, Q has O(n) vertices and so can clearly be compared with P in O(n) time. By proposition 1 (page 9), P is an envelope polygon if and only if P = Q.

It can easily be detected from Q_1 and Q_2 if the edges of P are not correctly sorted by the algorithm (by lemma 1.3.2 our sorting procedure is guaranteed to terminate in O(n) time).♣



On Classes of Arrangements

In the previous chapter, we studied geometric properties of envelopes and characterized the class of envelope polygons. In this chapter, we study envelopes from a different perspective, viewing them as a tool with which to characterize arrangements and classify them according to properties of their envelopes.

We now introduce the basic notions required in this chapter. We say that a line of an arrangement A is *exterior* if it contributes at least one edge to E(A). More generally, we say that a line of A is *k-exterior* if it contributes exactly k distinct edges to E(A). We also introduce a similar convention for arrangements of lines. We say that an arrangement is *exterior* if all the lines of the arrangement are exterior. In general, we say an arrangement A is *k-exterior* if (i) it is exterior, (ii) each line of A contributes at most k edges to E(A) and (iii) at least one line is k-exterior. See figure 2.1.2 for an example of a 1-exterior arrangement of lines.

The fact that an arrangement is exterior is very useful when we study arrangements. Since every line of the arrangement contributes to the envelope, properties of the envelope can be used to induce an "ordering" of the lines of the arrangement. It follows from the def-

inition (of exterior) that an arrangement A is exterior if and only if A = IA(E(A)). This implies that there exists a one-to-one correspondence between exterior arrangements and their envelopes.

The notion of exteriority is one of two important notions that introduce. The second one is that of general position. An arrangement of lines A is said to be in *general position* if (i) no two lines of A are parallel and (ii) no three lines are concurrent. It is quite common in computational geometry, when it is question of arrangements of lines, to assume that the lines of the arrangement are in general position (e.g. [Ed89]). This, in general, is to avoid exceptions on important properties of arrangements. Some authors also use general position to simplify presentation. In this chapter, the assumption that we require is that no three lines of an arrangement A intersect on the boundary of E(A). This is to make sure that the extreme vertices on the lines of the arrangements are well-defined, as the intersection of exactly two lines. We will assume general position in order to satisfy this requirement.

We now define E_c as the class of exterior arrangements in general position whose envelope is a simple polygon containing exactly c convex vertices. In particular, we study in this chapter the class E_3, which we also call the class of *sail arrangements*. The name *sail* stems from the fact that simple polygons of exactly three vertices are known as *sail polygons*. Sail polygons have found applications in the design of efficient algorithms for intersecting convex polygons and triangulating point sets [To85]. We begin the chapter by studying sail arrangements.

2.1. Characterizing Sail Arrangements

In this section, we characterize the class of sail arrangements that defined above. The property that we establish will help us, in the next section, in finding the convex vertices of a sail arrangement in O(n) time.

We now present a property of simple polygons that is useful in characterizing the class of sail arrangements (please refer to figure 2.1.1 for illustration). Suppose that $C = [p_u, p_{u+1}, \ldots, p_v]$ is a clockwise vertex chain of a simple polygon P. Let the interior of the polygon be the region to the right of C as we move from p_u to p_v on C. Let R_u be the di-

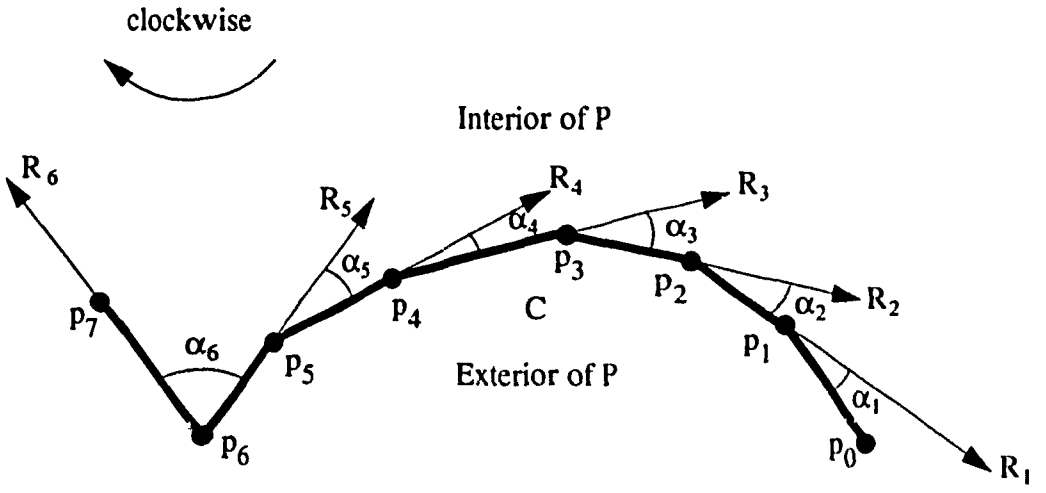


Figure 2.1.1. The walk of directed line R on a simple polygonal chain C.

rected line colinear to $[p_{u+1},p_u]$. Let R_{i+1} be the directed line obtained by rotating R_i counterclockwise about p_{i+1} until it is colinear to $[p_{i+2},p_{i+1}]$, $i \in \{u,...,v-2\}$. Let α_{i+1} be the angle induced by the rotation we make to get R_{i+1} from R_i. Define $Ang(C) = \Sigma \alpha_i$, $i \in \{u+1,...,v-1\}$. If we let the directed line R become successively colinear to $R_u,R_{u+1},...,R_{v-1}$ (by rotating R *counterclockwise* about successive p_i, $i \in \{u+1,...,v-1\}$) then we say that R *travels* on C in *clockwise* fashion about P. If R travels on C from $[p_{v-1},p_v]$ to $[p_u,p_{u+1}]$ in counterclockwise fashion, then R should be rotated in clockwise fashion. Note that the value of Ang(C) is independent of the direction in which R travels.

Similarly, given a simple polygon P, let Ang(P) be the sum of the angles made by the directed line R as it travels in clockwise order once around P (as described above) until it has rotated about every vertex of P. We show that for every simple polygon P, $Ang(P) = (c-2)\pi$, where c is the number of convex vertices of P. We will see later how to use this fact in characterizing sail arrangements.

We first state a well known result dating back to Euclid but offer an inductive proof that uses the Two Ears Theorem [Me75]. Our arguments requires the following definition. A vertex p_i of a simple polygon P is said to be an *ear* if no vertex of P lies in the interior of the triangle $[p_{i-1},p_i,p_{i+1}]$ and if the line segment $[p_{i-1},p_{i+1}]$ lies in P.

Lemma 2.1.1. For every simple polygon P of n vertices ($n \geq 3$), the sum of its interior angles equals $(n-2)\pi$.

Proof: (By induction) For $n = 3$, we have a triangle, for which a proof is given in Proposition 32 of Euclid's *Elements* [Eu300B.C.]. Assume the claim holds for all simple polygons of n-1 vertices and suppose that we are given a simple polygon P of n vertices. By Meisters' theorem [Me75], a polygon with $n > 3$ vertices has at least two non-overlapping ears. Cut off an ear. By the inductive hypothesis, the sum of the interior angles of the resulting polygon is $(n-3)\pi$. The sum of the interior angles in an ear is π since it is just a triangle. These angles all contribute to the sum of the interior angles of the polygon, hence the sum of the interior angles of P equals $\pi + (n-3)\pi = (n-2)\pi$.♣

Lemma 2.1.2. Let P be a simple polygon P of n vertices, c of which are convex. Then $Ang(P) = (c-2)\pi$.

Proof: Let β_i denote the internal angle induced by vertex p_i of simple polygon P. Remember that we defined α_i above as the angle induced by rotating R_{i-1} (colinear to $[p_i,p_{i-1}]$) counterclockwise about p_i until it is colinear to R_i (colinear to $[p_{i+1},p_i]$). R_{i-1} can be pointing in two possible directions: toward p_{i-1} or towards p_i. In both cases, α_i will be the same. If p_i is a convex vertex of P, then α_i is the same as β_i. If p_i is a reflex vertex (hence the corresponding internal angle is strictly greater than π) then α_i consists of $(\beta_i-\pi)$. Thus, for simple polygon P,

$$\begin{aligned} Ang(P) &= (n-2)\pi - (\#\text{reflex vertices})\pi \\ &= (n-\#\text{reflex vertices}-2)\pi \\ &= (c-2)\pi.\clubsuit \end{aligned}$$

Corollary 2.1.3. Let P be a simple polygon P of n vertices, c of which are convex. Then the lines of IA(P) can be sorted in O(cn) time.

Proof: By letting a directed line R travel once around P, we can create c-2 lists of edges sorted by slope. The lists have a total of n edges, so merging them can be accomplished in O(cn) time.♣

This property of simple polygons allows us to characterize sail arrangements as fol-

lows: We observe that the class of sail arrangements is equivalent to another class of arrangements which we call class $\tilde{A}$. We define *class* $\tilde{A}$ to be the set of arrangements A in general position that have the property that the vertices of A are vertices of E(A) if and only if it they are determined by adjacent lines. The properties of arrangements in class $\tilde{A}$ (and hence, of sail arrangements) are exploited in section 2.2, where we determine in O(n) time all the convex vertices of the envelope of a given sail arrangement. The following lemmata, along with lemma 2.1.2, are used in the proof of theorem 2.1.1.

Lemma 2.1.4. Class $\tilde{A} \subset$ 1–exterior arrangements.

Proof: Let arrangement A $\in$ class $\tilde{A}$. It follows from the definition of class $\tilde{A}$ that A must be exterior. Since no two lines are parallel in A, exactly two lines are adjacent (in the list of lines sorted by slope) to every line L in A, and each of those two neighbors of L intersects L at exactly one point. Thus, a line in A contributes exactly two vertices, hence, exactly one edge to E(A). This forces A to be 1-exterior. We can see that class $\tilde{A}$ is a proper subset of 1–exterior arrangements with the example in figure 2.1.2. Here A = $\{L_0,L_1,L_2,L_3,L_4,L_5,L_6\}$ is a 1-exterior arrangement of seven lines. The indices of the lines L_i $(0 \le i \le 6)$ correspond to an increasing order of slope. Consider the points $p_1 = I(L_1,L_4)$, $p_2 = I(L_2,L_5)$ and $p_3 = I(L_3,L_6)$. They are generated by non-adjacent lines but are vertices of the envelope.♣

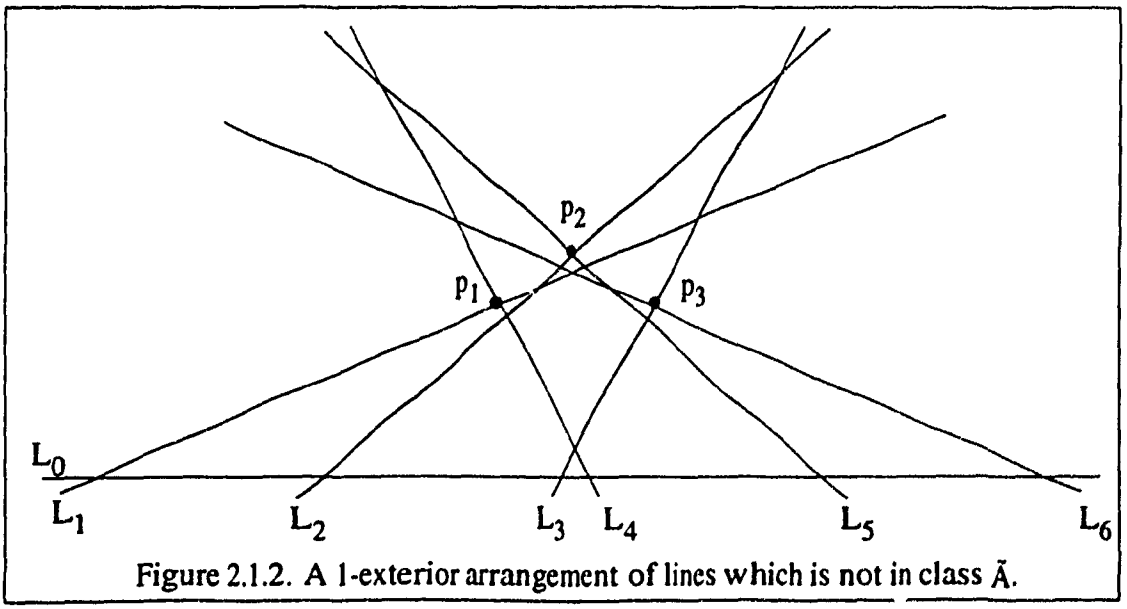


Figure 2.1.2. A 1-exterior arrangement of lines which is not in class $\tilde{A}$.

Theorem 2.1.1. An arrangement of lines A is in class $\tilde{A}$ if and only if A is a sail arrangement.

Proof: ("If" implication) Suppose that we are given a sail arrangement A. Since E(A) has exactly three convex vertices, it follows from lemma 2.1.2 that $Ang(E(A)) = \pi$. i.e. as a directed line R travels around E(A), it will rotate through an angle of exactly π and R becomes successively colinear with the lines of A in order of slope. Since A is exterior (because it is a sail), every line of A will be met by R. It follows that the intersection point of adjacent lines (in order of slope) is a vertex of E(A) and that conversely, a vertex of E(A) is the intersection of adjacent lines. i.e. A is in class $\tilde{A}$.

("Only if" implication, proof by contradiction) Suppose that we are given an arrangement A in class $\tilde{A}$ and that E(A) is a simple polygon of more than three convex vertices. By lemma 2.1.2, $Ang(E(A)) > \pi$. Let directed line R travel clockwise around E(A) starting at edge e_0. Let e_i be the first edge that causes the angle covered by R to exceed π. Then e_i has a slope that lies in between that of some edges e_j and e_{j+1} on the chain from e_0 to e_i, $0 < j < i-1$, so e_j and e_{j+1} are not adjacent in the list of lines sorted by slope.

Clearly, the vertex p_j that is common to both e_j and e_{j+1} is not the intersection of adjacent lines of A, contradicting the assumption that A is in class $\tilde{A}$. We conclude that E(A) must have exactly three convex vertices. Since by definition arrangements in class $\tilde{A}$ are in general position and by lemma 2.1.4 they are 1-exterior, we conclude that A must be a sail arrangement (this also implies that sail arrangements are 1-exterior).♣

2.2. Recognition of the Critical Vertices of a Sail Arrangement

Let X,Y and Z be the three convex vertices of the envelope of a sail arrangement A (E(A) is a sail polygon). Then combinatorially, we obtain four possible geometric situations.

(i) [X,Y], [Y,Z] and [X,Z] are all edges of E(A). E(A) is a triangle.

(ii) Two of [X,Y], [Y,Z] and [X,Z] are edges of E(A). E(A) has one concave chain

of vertices.

(iii) Only one of [X,Y], [Y,Z] and [X,Z] is an edge of E(A). E(A) has two concave chains of vertices.

(iv) None of [X,Y], [Y,Z] and [X,Z] are edges of E(A). E(A) has three concave chains of vertices.

We further classify sail arrangements as k-sail, $0 \leq k \leq 3$, where k indicates the number of concave chains in the envelope. Thus, case (i) above describes a 0-sail, case (ii) a 1-sail, case (iii) a 2-sail and case (iv) a 3-sail. Figure 2.2.1 gives examples of the four subclasses of sail arrangements. Owing to the trivial nature of 0-sails, subsequent discussion shall omit this subclass.

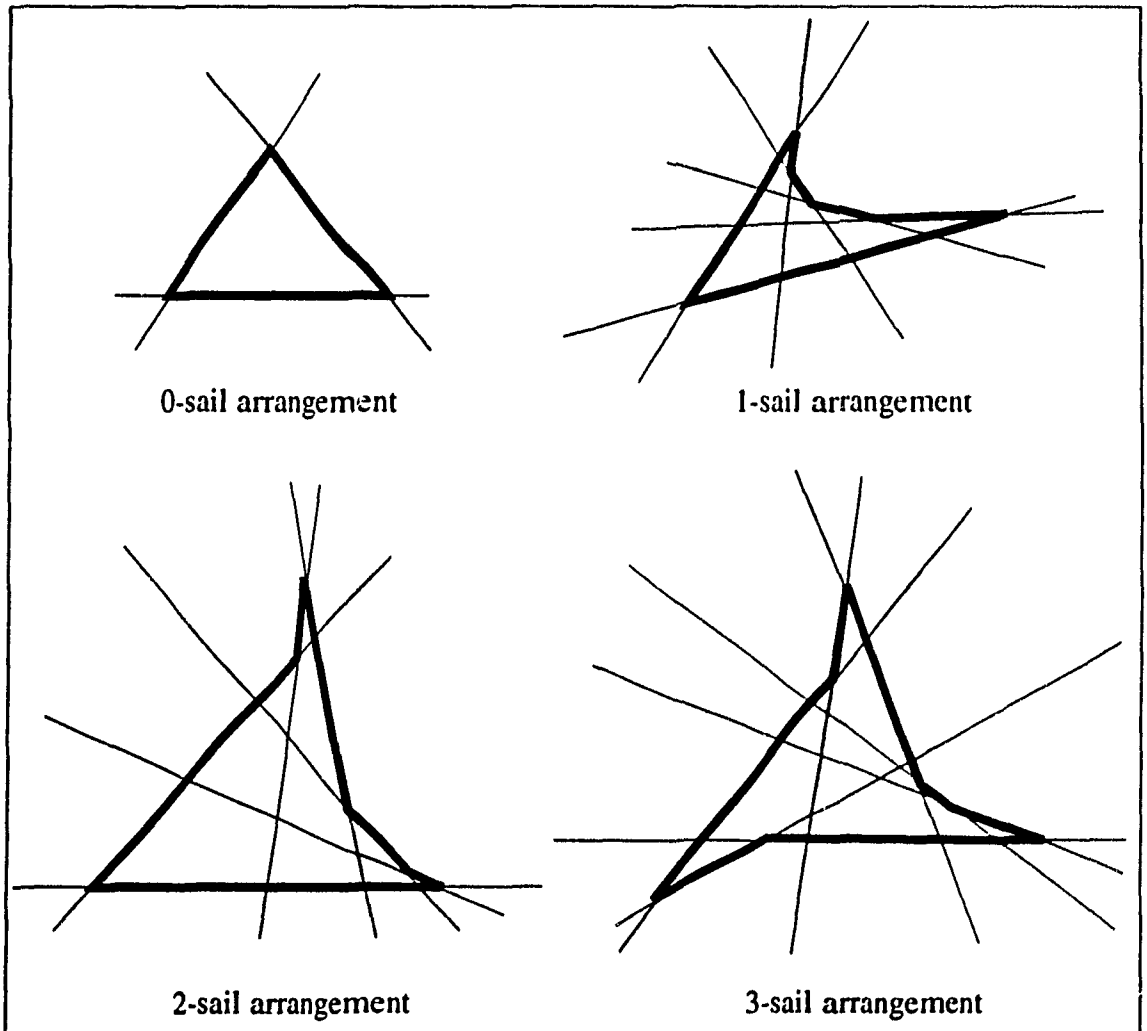


Figure 2.2.1. The family of sail arrangements.

Now we ask the following question. Given a sail arrangement of n lines A, can the three convex vertices X, Y and Z of the resulting envelope be found in linear time (i.e. can we determine which pairs of lines in A realize the three convex vertices of E(A))? It turns out that we can indeed do this in O(n) time. In what follows, we show how to find X, Y and Z in O(n) time for each sail subclass. We later show that we can determine in O(n) time the subclass to which a sail arrangement A belongs. Finding X,Y and Z in linear time allows us to compute in O(n) time the diameter and the convex hull of the $O(n^2)$ points of the arrangements. Furthermore, we can compute the points with minimum or maximum x-coordinate in linear time. All those problems have been shown to have $\Omega(n \log n)$ lower bound [CL85], [CSSS89].

We first define a few terms. A line segment that joins two critical vertices of arrangement of lines A is said to be a *critical edge*. We say that a line of A is a *critical line* if it is colinear to a critical edge. Thus, a k-sail arrangement A has (3 - k) critical edges ($0 \leq k \leq 3$). Remember that in this thesis we use a non-standard definition of slope, given in page 14. We shall frequently make use of the following simple procedures.

Procedure Compute_Extreme(A,L)
{Input: A line L of an arrangement $A = \{L_0,...,L_{n-1}\}$ in general position}
{Output: The two lines L_a and L_b such that $I(L,L_a)$ and $I(L,L_b)$ are the two extreme vertices on L.}
It is clear that the required output of Procedure Compute_Extreme can be obtained in O(n) time.

Procedure Critical_Line(A,L)
{Input: A line L of an arrangement $A = \{L_0,...,L_{n-1}\}$ in general position}
{Output: Determines whether or not the given line L is critical.}
begin
$L_a,L_b \leftarrow$ Compute_Extreme(A,L)
$L_c,L_d \leftarrow$ Compute_Extreme(A,L_a)
$L_e,L_f \leftarrow$ Compute_Extreme(A,L_b)
If (one of $\{L_c,L_d\} = L$ and one of $\{L_e,L_f\} = L$) then return(YES)
else return(NO)

end {of **Procedure Critical_Line**}

Proof of correctness: If L is a critical line, then $I(L,L_a)$ and $I(L,L_b)$ are both critical vertices, so one of $\{L_c,L_d\} = L$ and one of $\{L_e,L_f\} = L$. If L is not a critical line, then at least one of $I(L,L_a)$ or $I(L,L_b)$ is not a critical vertex. It follows that $L \notin \{L_c,L_d\}$ and/or $L \notin \{L_e,L_f\}$. Since procedure Compute_Extreme has O(n) time complexity, procedure Critical_Line exhibits O(n) time complexity.♣

We present the following lemma which applies to all sail arrangements and will be useful for subsequent proofs.

All the vertices of A must lie on one side of L since [X,Y] is an edge of the convex hull therefore the n-1 vertices of A on L are extreme.♣

Lemma 2.2.1. The order of the vertices of A on a critical line L define a slope ordering of the remaining n-1 lines of A.

Proof: Let X,Y and Z be the three critical vertices of A. The convex hull of the vertices of A is the triangle ΔXYZ. Since L is critical, two of the three critical vertices of A lie on L, say X and Y. [X,Y] is a critical edge of E(A). Also, assume that [X,Y] lies on the x-axis, X is to the left of Y on L and Z lies above the x-axis. Suppose that there exists lines L_i and L_j such that $slope(L_i) < slope(L_j)$ but $I(L,L_j)$ lies to the left of $I(L,L_i)$ on L. Then $I(L_i,L_j)$ is forced to lie below the x-axis and thus, outside of the convex hull of the vertices of A. Hence, a contradiction.♣

<u>Finding X,Y and Z when A is 1-Sail</u>

We can apply the above results to find the three convex vertices X,Y and Z of a 1-sail arrangement A in O(n) time with the following algorithm:

Algorithm One_Sail(A)
{Input: A 1-sail arrangement A of n lines.}
{Output: The three pairs of lines that determine the three convex vertices X,Y and Z.}
begin

$L \leftarrow$ Any line of A

$L_a, L_b \leftarrow$ Compute_Extreme(A,L)

$L_c, L_d \leftarrow$ Compute_Extreme(A,L_a)

$L_e, L_f \leftarrow$ Compute_Extreme(A,L_b)

We now have the following intersection points: $I(L_a,L_c)$, $I(L_a,L_d)$, $I(L_b,L_e)$ and $I(L_b,L_f)$.

If (Critical_Line(A,L)) then determine which three of the four points are colinear and delete the point that is not critical.

else determine which two of the four points are identical and delete one of them.

Return the three pairs of lines that determine the three remaining intersection points.

end {of **Algorithm One_Sail**}

Proof of correctness: By definition, a 1-sail envelope has two critical edges which have as endpoints the three convex vertices X,Y and Z. Without loss of generality, let the two critical edges be E_{XY} = [X,Y] and E_{YZ} = [Y,Z]. Note that the intersection points of E_{XY} (E_{YZ}) with every other line in A are extreme (lemma 2.2.1). Now, two cases arise when a line L is chosen. Either (i) the chosen line L is colinear to neither E_{XY} nor E_{YZ} or (ii) the chosen line L is colinear to either E_{XY} or E_{YZ}.

Suppose that the first case is true. Then it follows that lines L_a and L_b are colinear to E_{XY} and E_{YZ}. Computing lines L_c and L_d for L_a gives $I(L_a,L_c)$ and $I(L_a,L_d)$, which must be critical vertices. Similarly, computing lines L_e and L_f for L_b gives $I(L_b,L_e)$ and $I(L_b,L_f)$, which also must be critical vertices. But since Y is common to both E_{XY} and E_{YZ}, two of the points $I(L_a,L_c)$, $I(L_a,L_d)$, $I(L_b,L_e)$ and $I(L_b,L_f)$ must identify vertex Y, meaning that one of the four computed vertices is redundant.

If the second case is true, then without loss of generality, assume that L is colinear to E_{XY}. Then either L_a or L_b is colinear to E_{YZ}. Again without loss of generality, suppose that L_a is colinear to E_{YZ}. L_a will identify the two critical vertices Y and Z. When we compute the extreme vertices for L_b, we get one vertex which is X (= $I(L,L_b)$) and the other which is a non-critical vertex on E_{YZ}, say p. The vertex p lies in between Y and Z on E_{YZ} and hence, is non-critical. It is easy to delete p from the list of the four intersection points.

The algorithm uses procedures Compute_Extreme and Critical_Line which have O(n) time complexity. The deletion of the redundant or non-critical vertex can be done in constant time.♣

Finding X,Y and Z when A is 2-Sail

We can apply the above results, along with the technique of prune and search to find the three convex vertices X, Y and Z of a 2-sail arrangement of lines. We find X,Y and Z in O(n) time with procedure Find_Sail_Tip and algorithm Two_Sail By definition, a 2-sail arrangement A has at exactly one critical edge. Procedure Find_Sail_Tip accepts the sole critical line L and finds the convex vertex that does not lie on L. This is done by applying the prune and search technique. Algorithm Two_Sail finds the sole critical line L (and the two convex vertices that make L critical) and then applies the recursive procedure Find_-Sail_Tip, with L as input, to find the third convex vertex of E(A).

Procedure Find_Sail_Tip(A,L,n)

{Input: A critical line L in a 2-sail arrangement $A = \{L_0, L_1, ..., L_{n-1}\}$ of n lines.}

{Output: The two lines that intersect to give the convex vertex of E(A) that does not lie on L.}

{Assume that L lies on the x-axis and that n is even}

begin

$L_a \leftarrow$ the line $L_i \in A - \{L\}$ such that $(n-2)/2$ intersection points lie on either side of $I(L,L_i)$ on L {This is done using the k-selection algorithm of Blum et al. [BFPRT73]}

$L_b \leftarrow$ Extreme line on L_a other than L

If (L_a is extreme on L_b) then return($L_a$and L_b)

else begin

$A' \leftarrow$ {all lines $L_i \in A$ such that $I(L,L_i)$ lies in between $I(L,L_a)$ and $I(L,L_b)$ on L} $\cup$ {L, L_a, L_b}

return Find_Sail_Tip(A',L,||A'||)

end

end {of Procedure Find_Sail_Tip}

Proof of correctness: Let L_a be any line $\in$ A - {L}. Then we can determine the line L_b other than L such that $I(L_a,L_b)$ is extreme on L_a using procedure Compute_Extreme(). If $I(L_a,L_b)$ is also extreme on L_b then by definition $I(L_a,L_b)$ is critical, in which case we are done. Otherwise, remember from theorem 2.1.1 that as we walk around E(A), we meet the lines of A in sorted order of slope (since A is a sail arrangement). By lemma 2.2.1, the order of the intersection points on L also define a slope ordering of the lines of A, the same as E(A). Now note that L_a and L_b are exterior on different concave chains of E(A). It follows from the previous three statements that the two lines that we seek have corresponding intersection points that lie in between $I(L,L_a)$ and $I(L,L_b)$ on L.

Suppose now that we choose L_a such that there are (n-2)/2 intersection points on either side of $I(L_a,L)$ on L. We can do this in O(n) time using the k-selection algorithm of Blum et al. [BFPRT73]. L_b lies on one of these sides and so we can "throw away" ((n-2)/2 + c) lines at each call to procedure Find_Sail_Tip (c depends on the relative position of $I(L_b,L)$ on L). So we obtain the following recurrence relation:

$$\begin{aligned} T(n) &= T(n - ((n-2)/2 + c)) + O(n) \\ &= T(n/2) + O(n) \\ &= O(n) \end{aligned}$$

Therefore procedure Find_Sail_Tip runs in O(n) time.♣

Algorithm Two_Sail(A)

{Input: A 2-sail arrangement A of n lines.}

{Output: The three pairs of lines that determine the three convex vertices X,Y and Z.}

begin

L ← Any line ∈ A

{Find the sole critical line of A and call it L}

If (not Critical_Line(A,L))

begin

L_a,L_b ← Compute_Extreme(A,L)

If (Critical_Line(A,L_a)) then L ← L_a

else L ← L_b

end

L_a,L_b ← Compute_Extreme(A,L) {Determine the critical vertices on L}

L_c,L_d ← Find_Sail_Tip(A,L,n) {Find the third convex vertex Z}

Return {(L,L_a),(L,L_b),(L_c,L_d)}

end {of **Algorithm Two_Sail**}

Proof of correctness: We only need to show that the first two convex vertices X and Y that define a critical edge of A are correctly determined. Suppose that the line L that we choose is critical. Then procedure Critical_Line will correctly detect this. Otherwise, one of the two lines $L_a$or L_b, which give extreme vertices on L, must be critical. Procedure Critical_Line is applied to L_a to determine if it is critical. If L_a is not critical then L_b must be critical. All procedures used have linear time complexity so algorithm Two_Sail has O(n) time complexity.♣

Finding X,Y and Z when A is 3-Sail

We can apply the previous results for 2 sails to obtain a divide and conquer algorithm for 3-sails. Given a 3-sail arrangement of lines, we first select an arbitrary line L. From this line, we obtain the two lines L_a and L_b that determine the extreme vertices on L. We will show that L_a and L_b define two slope ranges that separate the given arrangement of lines into two or three 2-sail arrangements. We first present the divide and conquer algorithm and then give the proof of correctness.

Algorithm Three_Sail(A)

{Input: A 3-sail arrangement A of n lines.}

{Output: The three pairs of lines that determine the three convex vertices X,Y and Z.}

begin

L ← Any line of A /* without loss of generality, assume that slope(L) = 0 */

L_a,L_b ← Compute_Extreme(A,L)

Re-label L_a and L_b (if necessary) so that slope(L_a) > slope(L_b)

If (I(L,L_a) or I(L,L_b) is critical) then

{The chosen line L passes through a convex vertex. A is split into two 2-sails.}

begin

Split the lines of A into two subsets A_1 and A_2 where
A_1 = {lines $L_i \in A$ such that slope(L_i) ≤ slope(L_b)}
A_2 = {lines $L_i \in A$ such that slope (L_b) ≤ slope(L_i) ≤ slope(L_a)} ∪ {L}
If I(L,L_a) is critical then $L_{Crit} \leftarrow L_a$
else $L_{Crit} \leftarrow L_b$
L_c,L_d ← Find_Sail_Tip(A_1,L,||A_1||)
L_e,L_f ← Find_Sail_Tip(A_2,L,||A_2||)
Return {(L,L_{Crit}),(L_c,L_d),(L_e,L_f)}

end

else begin {A is split into three 2-sails.}

Split the lines of A into three subsets A_1, A_2 and A_3 where
A_1 = {lines $L_i \in A$ such that slope (L_i) ≤ slope(L_b)}
A_2 = {lines $L_i \in A$ such that slope (L_b) ≤ slope(L_i) ≤ slope(L_a)} ∪ {L}
A_3 = {lines $L_i \in A$ such that slope(L_i) ≥ slope(L_a)} ∪ {L}
L_c,L_d ← Find_Sail_Tip(A_1,L,||A_1||)
L_e,L_f ← Find_Sail_Tip(A_2,L,||A_2||)
L_g,L_h ← Find_Sail_Tip(A_3,L,||A_3||)

Return {(L_c,L_d),(L_e,L_f),(L_g,L_h)}

end

end {of **Algorithm Three_Sail**}

Proof of correctness: Let L be any line in A. There are two cases: (i) L passes through a convex vertex of E(A) or (ii) L does not pass through a convex vertex of E(A). We test for these two cases by first applying procedure Compute_Extreme to L to obtain L_a and L_b. The first case is true if and only if L is returned for either Compute_Extreme(A,L_a) or Compute_Extreme(A,L_b). We assume that L is parallel to the x-axis (if not we can rotate the lines of A appropriately). Thus, we can think of L as having slope(L) = 0 or π. Assume slope(L_a) > slope(L_b). Also, let Q be the list of lines of A sorted on increasing slope starting from slope 0. Let X, Y and Z, the three critical vertices of A, appear in clockwise order on E(A).

Suppose that the first case is true and that the convex vertex identified is X = $I(L,L_a)$. Note that L, L_a and L_b are exterior on different concave chains of E(A). Since E(A) has exactly three concave chains, this implies that there exist two lines L_c,L_d between L (slope 0) and L_b in Q that determine a second convex vertex of E(A) (Y = $I(L_c,L_d)$) and two lines L_e,L_f between L_b and L_a in Q that determine a third convex vertex of E(A) (Z = $I(L_e,L_f)$). Let A_1 be as defined in the algorithm. Let C_1 be the vertex chain [X,...,Y,..., $I(L,L_b)$] on E(A). A line $L_i \in A$ is in A_1 if and only if it is exterior on C_1. This is because A is in class $\tilde{A}$ (theorem 2.1.1). Hence, C_1 is a portion of $E(A_1)$. Now we will show that L is critical in A_1 and hence, A_1 is a 2-sail arrangement. L passes through the endpoints of C_1 so C_1 lies completely to one side of L, say above L. We claim that all of the vertices of A_1 on L are extreme. Suppose not. Then there exists a vertex p of A_1 below L. $E(A_1)$ must contain p. So there exists a critical vertex of A_1 below L. This critical vertex is the intersection of adjacent lines in A_1 (corollary 2.1.3). But then these two lines are not exterior on C_1, a contradiction to the assumption that they are in A_1. Hence, all of the vertices of A_1 on L are extreme. Thus, by lemma 2.2.1, L is critical. Therefore, A_1 is a 2-sail arrangement. By similar argument, A_2 is 2-sail. So we can apply algorithm Find_Sail_Tip to A_1 and A_2 with L as the sole critical line to obtain Y= $I(L_c,L_d)$ and Z = $I(L_e,L_f)$.

In the second case, L, L_a and L_b are also exterior on different concave chains of E(A). Thus, there exists two lines between L (slope 0) and L_b in Q that determine a convex vertex of E(A), two lines between L_b and L_a in Q that determine a second convex vertex of E(A) and two lines between L_a and L (slope π) in Q that determine a third convex vertex of E(A). We split A into three subsets A_1, A_2 and A_3 as in the algorithm. With arguments similar to that in case (i), A_1, A_2 and A_3 are all 2-sail arrangements with L as the sole critical line. Hence, we can apply algorithm Find_Sail_Tip thrice to A_1, A_2 and A_3 with L to obtain X, Y and Z.

All of the operations have O(n) time complexity, so algorithm Three_Sail() has O(n) time complexity.♣

Now that we are able to find X, Y and Z given that A is k-sail ($0 \le k \le 3$), we observe that we can remove the condition that the value of k be known a priori. The following theorem shows that we need only know that A is a sail arrangement.

Theorem 2.2.1. Given $A = \{L_0, L_1, ..., L_{n-1}\}$, a k-sail arrangement of n lines ($0 \leq k \leq 3$), we can determine the value of k and the three convex vertices of A in O(n) time.

Proof: Let L be any line in A and L_a and L_b be the lines in A such that $I(L,L_a)$ and $I(L,L_b)$ are the extreme vertices on L (L_a and L_b are computed via procedure Compute_Extreme). We can use procedure Critical_Line to determine whether or not L, L_a and L_b are critical. The number of critical lines tell us the number of convex vertices in E(A) and hence, by definition of sail arrangements, the value of k. These tests are sufficient since all critical lines in A (if they exist) are identified by computing the extreme vertices of any line in A. For example, if L was found to be non-critical, then A can be only be 1, 2 or 3 -sail and L_a and L_b can be the only candidates for critical lines since they are the extreme lines on L. Once we know the value of k, we can invoke the appropriate algorithm to find the three critical vertices of A in O(n) time.♣

Corollary 2.2.2. The diameter, the convex hull and the points with minimum or maximum x-coordinate of the $O(n^2)$ vertices of a sail arrangement of n lines can be computed in O(n) time.

2.3. Constructing the Envelope of a Sail Arrangement of Lines.

In [CL85], Ching and Lee showed a lower bound of $\Omega(n \log n)$ for the problems of computing the diameter, convex hull and envelope of an arbitrary arrangement of lines. We have introduced and studied sail arrangements with the hopes that definite characterizations of a non-trivial class of arrangements would allow us to beat these lower bounds. theorem 2.2.1 implies that given a sail arrangement of n lines, the convex vertices can be determined in O(n) time. This allows us to compute the diameter and convex hull of the arrangement in O(n) time even though there are $O(n^2)$ intersection points. We show here that the $\Omega(n \log n)$ lower bound, however, still applies to the envelope construction problem for sail arrangements. This implies that the additional information and structure derived from sail arrangements is insufficient for solving the envelope construction problem for sail arrangements in o(n log n) and ultimately, improves on Ching and Lee's result for arbitrary

arrangements. Remember, for this proof, that our definition of slope is not standard (see page 14).

Theorem 2.3.1. Given a 1-sail arrangement A of n lines, the complexity of computing E(A) is $\Omega(n \log n)$ under the algebraic tree model of computation.

Proof: (By reduction from Integer Sorting) Let $S = \{x_1,..., x_n\} \subset \aleph^+ \cup \{x_0 = 0, x_{n+1} = \infty\}$ be the input to the Integer Sorting problem; assume that the x_i's are distinct and refer to figure 2.3.1 for illustration. Let C be the quarter circle defined by the set of points on the unit circle (centered at the origin) with positive x-coordinate and negative y-coordinate. For every x_i, compute L_i be the unique line tangent to C with slope x_i (let $A = \{L_i\}_{i=0,...,n+1}$). Since C is an arc of a circle, trigonometric operations are required to compute L_i; however they are allowed in the algebraic tree computation model [BO83].

We now show that A thus constructed is a 1-sail arrangement of lines. Without loss of generality, assume that $\text{slope}(L_0) \le \text{slope}(L_1) \le \text{slope}(L_i) \le \text{slope}(L_n) \le \text{slope}(L_{n+1})$, $i = 2,...,n-1$. Let $X = I(L_0,L_{n+1})$, $Y = I(L_0,L_1)$ and $Z = I(L_n,L_{n+1})$. Notice the following two facts:

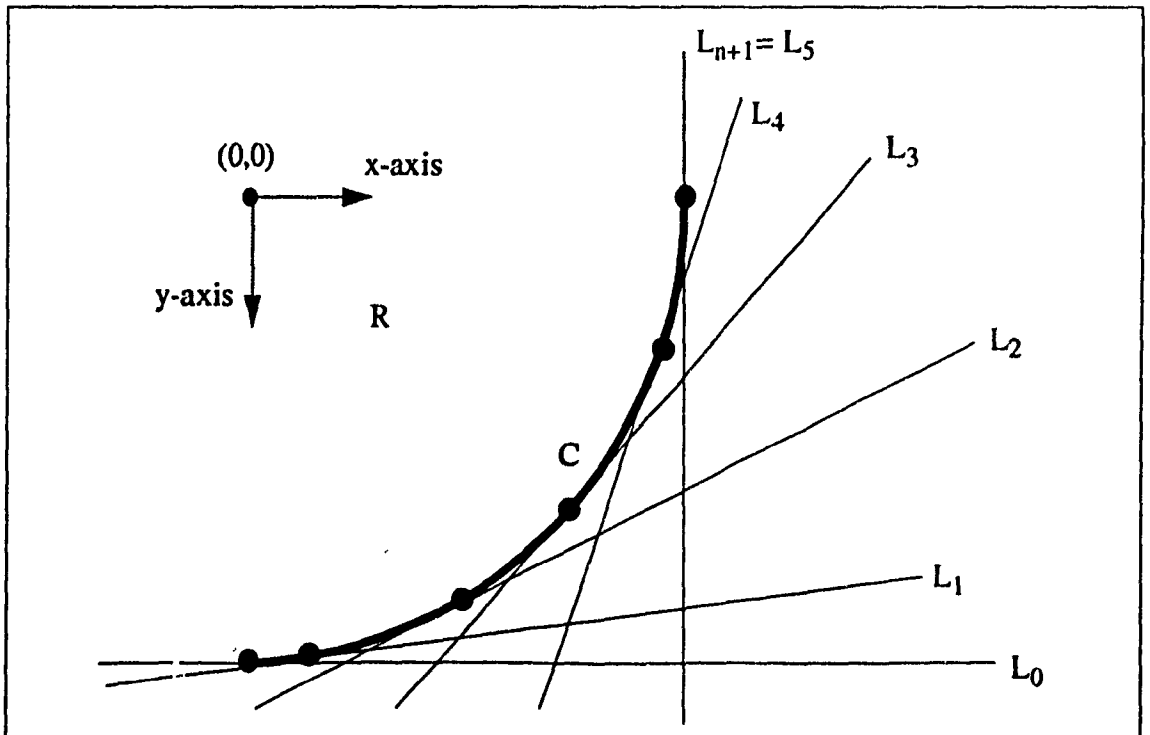


Figure 2.3.1. Constructing a 1-sail arrangement of lines from a set of integers.

(i) C is monotone increasing with respect to both L_0 and L_{n+1} (by naturally extending the definition of *monotone* from a polygonal chain to a smooth curve) and (ii) the order of the intersection points of the lines of A with C on C define a slope ordering of the lines of A. From these two facts, it follows that the order of the intersection points on L_0 (and on L_{n+1}) also define a slope ordering of the lines of A; the same ordering as on C. Therefore no two lines of A intersect below L_0 or to the right of L_{n+1}. i.e. X, Y and Z are all critical vertices. It also follows that no critical vertex can lie on L_i, i= 2,...,n-1 (the extreme vertex of L_i on L_0 (or L_{n+1}) is not extreme on L_0 (or L_{n+1}) because of the slope ordering). We can thus conclude that E(A) has only three convex vertices, X, Y and Z.

To show that A is a 1-sail, we still have to demonstrate that A is exterior and that the lines of A are in general position. First consider the region R to the left of L_{n+1}, above L_0 and to the left of C. Because the lines of A are tangent to C, it follows that (a) R lies to the left of every line of A and that (b) every line of A has a point (the tangent) on the boundary of R. From this, we conclude that A is exterior.

No two lines of A are parallel since we assumed that the x_i's are distinct. Let L_i and L_j be any two distinct lines of A and assume that slope(L_i) > slope(L_j). They both intersect C in a unique point, so clearly $I(L_i,L_j)$ does not lie on C. Let L be any line through $I(L_i,L_j)$. If slope(L) > slope(L_i) or slope(L) < slope(L_j), then L intersects C properly and so is not tangent to C. If slope(L_i) > slope(L) > slope(L_j), then L does not intersect C. In both cases, L is not in A. Therefore, no three lines of A can intersect in a common point, establishing that the lines of A are in general position. We have thus shown that A, constructed in O(n) time from S, is a 1-sail arrangement of lines. Following theorem 2.1.1, in traversing E(A) in clockwise order, the lines of A are met in sorted order of slope, which is the order of the x_i's.

Hence we can solve the Integer Sorting problem by reducing it in linear time to the problem of constructing the envelope of a 1-sail arrangement. Since Integer Sorting is $\Omega(n \log n)$ under the algebraic tree model of computation, it follows that computing the envelope of a 1-sail arrangement has a lower bound of $\Omega(n \log n)$.♣

2.4. Hamiltonian Circuits

An arrangement of lines A is said to be *hamiltonian* or admit a *hamiltonian circuit* if there exists a closed path through some of the bounded segments of A which visits every vertex of A exactly once. Everett showed with the example illustrated in figure 2.4.1 that there exists an arrangement of six lines that is not hamiltonian. One can observe that the critical vertices of A, adjacent to only two bounded segments, force any hamiltonian path to follow those two bounded segments. We can then see that there is no way for any path to reach the vertex p without visiting vertices which have already been forced to be visited. We must conclude that this arrangement does not admit any hamiltonian circuit. It is easy to construct, for any odd value of n ($n > 6$), an arrangement of n lines which do not admit any hamiltonian circuit; this is done by maximizing the number of critical vertices in the arrangement— the envelope has the shape of a "star". This shows that there is an infinite number of arrangements which are not hamiltonian.

We now ask if there are classes of arrangements which always admit a hamiltonian circuit. We show in this section that every 1-sail arrangement of lines is hamiltonian. We also give an O(n log n) time procedure to produce a hamiltonian circuit from a 1-sail arrangement of n lines. Our argument exploits a property of 1-sail arrangements that we establish with the following lemma.

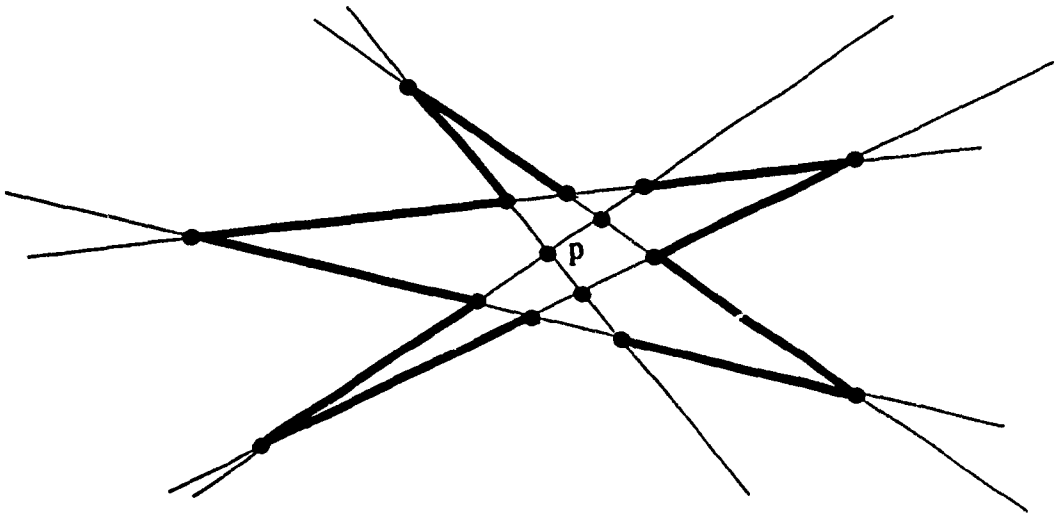


Figure 2.4.1 Example of an arrangement of 6 lines that is not hamiltonian.

Lemma 2.4.1. Let $A = \{L_0,L_1,...,L_{n-1}\}$ be a 1-sail arrangement of n lines. Then for each line L of A, the ordering of the n-1 vertices on L corresponds to a slope ordering of the lines in A.

Proof: (By induction) First note that for every arrangement of four lines (only one combinatorial possibility) the vertices are ordered as required. As our inductive hypothesis, assume that the statement of the lemma holds for every 1-sail arrangement of n-1 lines, and suppose that we are given a 1-sail arrangement A of n lines. Let X, Y and Z be the three convex vertices of E(A), such that [X,Y] and [Y,Z] are the two critical edges of E(A). Let L_0 (L_{n-1}) be the critical line through Y and Z (X and Y). L_0 and L_{n-1} are adjacent in the list of lines of A sorted by slope since Y is a critical vertex. To simplify presentation, assume without loss of generality (by rotating A) that L_0 (L_{n-1}) has the smallest (largest) slope of A, and that the indices of the other lines give their order of slope. Thus $X = I(L_0, L_1)$ and $Z = I(L_{n-1},L_{n-2})$.

Let A' be the arrangement of n-1 lines obtained by removing line L_{n-1} from A. We will show by contradiction that A' is a 1-sail arrangement. Suppose that A' is not 1-sail. Then in the arrangement A, there must be a vertex of A inside the triangle $\Delta = (Y, X, I(L_0,L_{n-2}))$. If this is not true then all of the vertices of A' on L_{n-2} are extreme and hence, L_{n-2} is critical and A' is 1-sail. Let $p = I(L_i,L_j)$ be the first vertex inside Δ met by L_{n-1} as we rotate it clockwise about X. Then the triangle $(p,I(L_{n-1},L_i),I(L_{n-1},L_j))$ is empty. Since the order of the vertices of A on L_{n-1} define a slope ordering for the lines of A (lemma 2.1.1), it follows that L_i and L_j are adjacent lines in A. But since p lies inside E(A), its existence contradicts the fact that $A \in$ class $\tilde{A}$. Therefore A' must be a 1-sail arrangement of n-1 lines for which the statement of the lemma holds by the inductive hypothesis.

Line L_{n-1} has the greatest slope in A and $I(L_i,L_{n-1})$ is extreme on L_i (for $i \in \{0,...,n-2\}$) so the position of $I(L_i,L_{n-1})$ on L_i corresponds to the position of L_{n-1} in the order of the lines of A sorted on slope. The statement of the lemma therefore holds for A, a 1-sail arrangement of n lines.♣

The following algorithm uses the above property of 1-sail arrangements to trace a hamiltonian circuit through the vertices of A. See fig. 2.4.2 for an illustration of the resulting hamiltonian circuit. The vertices of A at which left or right turns are made are indicated

by the dots. The direction of the traced hamiltonian circuit is indicated with arrows.

Algorithm One_Sail_Hamiltonian(A)
{Input: A 1-sail arrangement A = $\{L_0,L_1,...,L_{n-1}\}$ of n lines.}
{Output: A hamiltonian circuit denoted as a list H of vertices of A.}
begin
Sort the lines on increasing slope. Relabel the lines appropriately.
Use algorithm One_Sail to find the three convex vertices X, Y and Z of E(A).
Suppose that X = $I(L_0, L_1)$, Y = $I(L_0,L_{n-1})$ and Z = $I(L_{n-1},L_{n-2})$
H ← Y
H ← X
for i = n-2 down to 2 by 2
begin
H ← $I(L_1,L_i)$ {Circuit passes through vertices
on L_i in decreasing order of slope}
if (i - 1 ≥ 2) then H ← $I(L_1,L_{i-1})$ {Move to adjacent line with smaller slope}
if (i - 1 > 2) then H ← $I(L_{i-2},L_{i-1})$ {Circuit passes through vertices
on L_{i-1} in increasing order of slope}
end
H ← Z
{Z to Y is a critical edge and so the hamiltonian circuit returns to Y}
Return H
end {of Algorithm One_Sail_Hamiltonian}

Proof of correctness: Map the vertices of A to an nxn integer matrix M in the following manner. Map each vertex $I(L_i,L_j)$ of A, i < j, to the entry of M with row i and column j (refer to figure 2.4.2). Two vertices that are row or column neighbors in M share an edge in M and this edge corresponds to the line segment that is bounded by the two corresponding vertices of A and lies on a line of A. Note that each line segment $[I(L_i,L_{i+1}),I(L_{i+1},L_{i+2})]$ ($i \in \{0,...,n-3\}$) on the concave chain of E(A) corresponds to the two connected edges in M [(i,i+1),(i+1,i+1)] and [(i+1,i+1),(i+1,i+2)] (these connected edges in M are indicated with dashed lines). Note also that the line L_0 corresponds to the row 0 in M while the line L_{n-1}

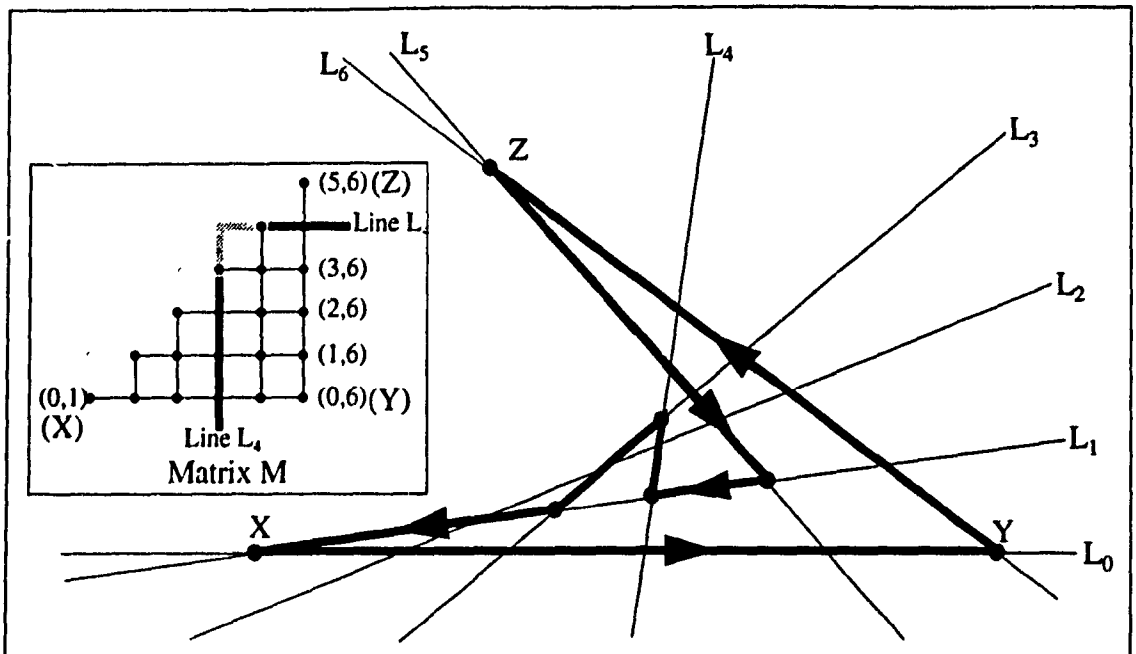


Figure 2.4.2. Tracing a hamiltonian circuit through the vertices of A of a 1-sail.

corresponds to the column n-1 in M. A line $L_i \in A - L_0 - L_{n-1}$ corresponds to the path in M from (0,i) to (i,i) to (i,n-1) (see figure 2.4.2 for an example).

On each line representation in M, the order of coordinate points of M corresponds to the slope ordering of the lines in A. Clearly, the property of 1-sails shown in lemma 2.4.1 allows us to define this mapping. Tracing a hamiltonian circuit through the vertices of A is then transformable to tracing a hamiltonian circuit through the mapped coordinates in M. Algorithm One_Sail_Hamiltonian outputs a hamiltonian circuit in M and thus, a hamiltonian circuit through the vertices of A as required. Even though there are $O(n^2)$ vertices in A (and hence, $O(n^2)$ mapped coordinates in M), we need only specify the vertices of A (mapped coordinates in M) at which the circuit makes a left or right turn. Since there are n lines in A, there are (n-1) + (n-2)/2 vertices in our hamiltonian circuit. We know at which vertices these turns must be made if the lines are sorted. There are n lines so we can perform the traversal in O(n) time. The complexity of our algorithm is bounded, however, by the fact that we first sort the lines. Thus, we can find a hamiltonian circuit for a 1-sail arrangement of n lines in O(n log n) time.♣

Conclusion

In chapter 1, we studied the geometry of envelope polygons. We showed that envelope polygons are L-convex and hence that properties of L-convex polygons are useful in studying envelopes. By combining the results of sections 1.3 and 1.4, we showed that it can be determined in O(n) time if a given simple polygon of n vertices is an envelope.

From this, we can ask whether problems which have superlinear lower bounds or no known tight upper bound for general polygons can be solved in linear time for envelope polygons. A few examples of such problems include finding the longest or shortest diagonal of an envelope, or computing the geodesic diameter and the geodesic center of an envelope.

In chapter 2, we introduced a hierarchy of classes of arrangements of lines based on the number of convex vertices of their envelopes. In particular, we looked at a class called *sail arrangements*: given a sail arrangement A, we can find the three convex vertices of E(A) and therefore the convex hull, the diameter and the points with minimum or maximum x-coordinate of the arrangement in O(n) time. It is $\Omega(n \log n)$, however, to construct the remainder of the envelope of a sail arrangement. We also showed that 1-sail arrangements admit hamiltonian circuits.

It remains, however, an open problem as to whether or not 2 and 3 -sail arrangement graphs admit hamiltonian circuits. Owing to the well defined geometrical structure of sail arrangements, we conjecture that 2 and 3 -sail arrangements are hamiltonian. Another open problem is to see if we can determine in O(n) time if a given arrangement of lines is a sail or not. It would be also of interest to study further the hierarchy E_c, especially for values of c greater than 3, or to define new classes of arrangements for which interesting results can be obtained.

In general, for any problem on arrangements of lines for which a known lower bound is known or for which optimal upper bound is not known, it is interesting to determine if there is a class of arrangements for which the problem can be solved optimally.

References

[At86] Atallah M.J., "Computing the convex hull of line intersections", *Journal of Algorithms*, Vol. 7, 1986, pp.285-288.

[BEPY91] Bern M., Eppstein D., Plassman P., Yao F."Horizon theorems for lines and polygons", *DIMACS Series in Discrete Mathematics and Computer Science*, No. 6, 1991, pp. 45-66.

[BFPRT73] Blum M., Floyd R.W., Rivest V.R., Tarjan, R.E., "Time bounds for selection", *Journal of Computer and System Sciences*, Vol. 7, No. 4, 1973, pp. 448-461.

[BO83] Ben-Or M., "Lower bounds for algebraic computation trees", *Proc. 15th ACM Sympos. on Theory of Comput.*,1983, pp. 80-86.

[CL85] Ching Y.T., Lee D.T., "Finding the diameter of a set of lines", *Pattern Recognition*, Vol. 18, 1985, pp. 249-255.

[CR41] Courant R., Robbins H., "What is mathematics?", *Oxford University Press*, 1941.

[CSSS89] Cole R., Salowe J., Steiger W., Szemerédi E., "An optimal-time algorithm for slope selection", *SIAM Journal on Computing*, Vol. 18, No. 4, 1989, pp. 792-810.

[Ed87] Edelsbrunner H., "Algorithms in combinatorial geometry", *Springer-Verlag*, 1987.

[EAT83] ElGindy H., Avis D., Toussaint G.T., "Applications of a two-dimensional hidden-line algorithm to other geometric problems", *Computing*, Vol. 31, 1983, pp. 191-202.

[ET89] ElGindy H., Toussaint G.T., "On geodesic properties of polygons relevant to linear time triangulation", *The Visual Computer*, Vol. 5,

1989, pp. 68-74.

[Eu300B.C.] Euclid, "Proposition 32", *Elements*, 300 B.C.

[Ev91] Everett H., personal communication.

[Gr72] Graham R., "An efficient algorithm for determining the convex hull of a planar set", *Information Processing Letters*, Vol. 1, 1972, pp. 132-133.

[HV49] Horn A., Valentine F.A., "Some properties of L-sets in the plane", *Duke Mathematics Journal*, Vol. 16, 1949, pp. 131-140.

[Ke91] Keil, M., "A simple algorithm for determining the envelope of a set of lines", *Information Processing Letters*, Vol. 39, 1991, pp. 121-124.

[LPSSSSTWY88] Lenhart W., Pollack R., Sack J.R., Seidel R., Sharir M., Suri S., Toussaint G.T., Whitesides S., Yap C.K., "Computing the link diameter of a simple polygon", *Discrete and Computational Geometry*, Vol. 3, No. 3, 1988, pp. 281-293.

[OR87] O'Rourke J., "Art gallery theorems and algorithms", *Oxford University Press*, 1987.

[Me75] Meisters G.H., "Polygons have ears", *American Mathematical Monthly*, 1975, pp. 648-651.

[Su85] Suri S., "Computing the envelope of a set of lines", *Unpublished manuscript*, Johns Hopkins University, 1985.

[TA82] Toussaint G.T., Avis D., "On a convex hull algorithm for polygons and its application to triangulation problems", *Pattern Recognition*, Vol. 15, No. 1, 1982, pp. 23-29.

[To85] Toussaint G.T.,"A simple linear algorithm for intersecting convex polygons", *The Visual Computer*, Vol. 1, 1985, pp. 118-123.

[To91] Toussaint G.T., "Computing morphological properties of arrangements of lines", *Proc. SAMPE Japan 1991*, Dec. 11-14, 1991, Tokyo.

[Ur91] Urrutia J., personal communication.

[Ve87] Vegter G., "Computing the bounded region determined by finitely many lines in the plane", *Technical Report 87-03*, Dept. of Mathematics and Computing Science, Rijkuniversiteit Groningen, 1987.

[Za75] Zaslavsky T., "Facing up to arrangements: face-count formulas for partitions of space by hyperplanes", *Mem. Amer. Math. Soc.*, Vol. 154, 1975.