Construction of sparse visibility graphs

Mark H. Overmars and Emo Welzl

RUU-CS-87-9
May 1987

Rijksuniversiteit Utrecht

Vakgroep informatica

Budapestlaan 6 3584 CD Utrecht
Corr. adres: Postbus 80.012 3508 TA Utrecht
Telefoon 030-53 1454
The Netherlands

Construction of sparse visibility graphs

Mark H. Overmars and Emo Welzl

Technical Report RUU-CS-87-9
May 1987

Department of Computer Science
University of Utrecht
3508 TA Utrecht
the Netherlands

CONSTRUCTION OF SPARSE VISIBILITY GRAPHS[1)]

by

Mark H. Overmars[2)] and Emo Welzl[3)]

April, 1987

Abstract. Let S be a set of n non-intersecting closed line segments in the plane. The visibility graph Gs of S is the graph that has the endpoints of the segments in S as nodes and in which two nodes (endpoints) are adjacent whenever they "see" each other (i.e., the open line segment joining them either is disjoint from all segments in S or is contained in a segment in S). It is shown that Gs can be constructed in time $O(m \cdot \log n)$, where m is the number of edges in Gs. This result is better than any known result for $m = o(n^2/\log n)$.

keywords and phrases: computational geometry, visibility graph, shortest path, rotational sweep.

[1)] This work was carried out while the first author visited the Institutes for Information Processing, IIG, Technical University of Graz.

[2)] Department of Computer Science, University of Utrecht, P.O. Box 80.012, NL-3508 TA UTRECHT, The Netherlands.

[3)] Institutes for Information Processing, IIG, Technical University of Graz and Austrian Computer Society, Schiesstattgasse 4a, A-8010 Graz, Austria.

Obviously, the visibility graph of S following a direction d does not change if we start to rotate d counter-clockwise until d becomes equal to a direction defined by an edge in this graph. Assume that such a situation occurs (while "rotating" d) and the critical edge points from endpoint p to endpoint q. Even then a further rotation of d will change only the edge outgoing from p (we assume that no two edges have the same slope!). The critical part of our rotational sweep algorithm is now to compute the next endpoint to be "seen" from p. We distinguish two cases.

Case 1. The other endpoint z of the segment s of q lies to the left of the line through p and q (see Figure 1.3): Imagine now that a rubber band is stretched from p to q. Keep the end in p fixed and move the end in q along the segment s until it reaches z. Now the rubber band will form a convex polygonal curve from p to z and it is easily seen that the first vertex x following p on this curve is the next endpoint for p to point to (when d is rotated). Algorithmically, we will see that we can keep and retrieve enough information in z in order to find the point x in logarithmic time.

Case 2. The other endpoint of the segment of q does not lie to the left of the line through p and q: Now let z be the endpoint to which q currently points. It will turn out that z lies to the left of the line through p and q (otherwise this edge would have been treated before). Again we stretch a rubber band from p to q, but now we move it along the edge between q and z. The reasoning continues now like in Case 1; the main thing the reader might be worried about right away is that the endpoint x to which p is pointing next lies "behind" the edge from q to z (seen from p). This will be shown to be impossible, as in a visibility graph

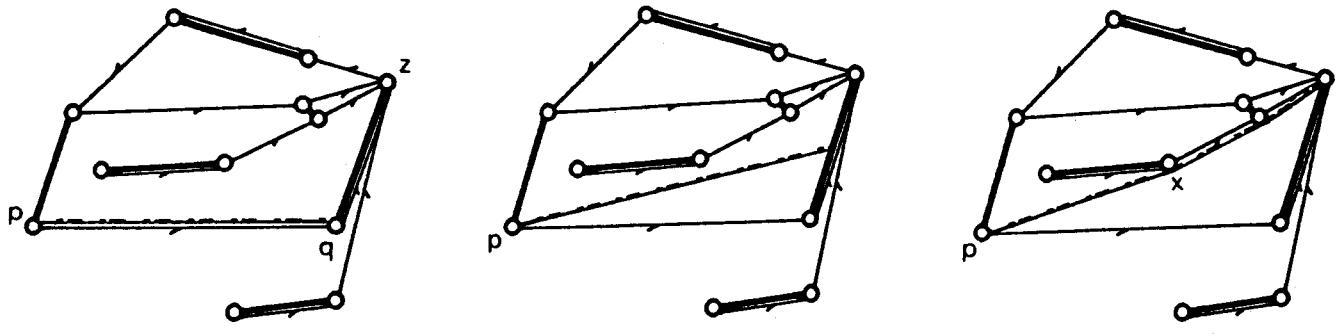

Figure 1.3.
Moving the rubber band along a line segment.

following a direction d two edges never cross (if x would lie behind the edge from q to z, the edges from p to x and the one from q to z would cross in the updated graph!).

Both in Cases 1 and 2 the new edge from p to x is an edge in the visibility graph to be computed. Actually, if we perform the sweep from a direction d to direction d + π, we will meet all edges in the visibility graph and, moreover, every update necessary is associated with an edge in the visibility graph (the edge between p and q in the above terminology). This will eventually lead us to the claimed result.

The paper is organized as follows. In Section 2 we establish the necessary terminology. Section 3 introduces the visibility graph following an edge which basically corresponds to the visibility graph following a direction as discussed above (removing the restriction of no two pairs of points defining parallel lines); moreover, the basic lemmas for the algorithmic treatment in Section 4 are prepared. Finally, in Section 5, a short discussion concludes the paper.

2. Terminology.

The basic terminology necessary for this paper (together with some observations) is introduced in this section.

Definition and Convention. For this and the following section let S be an arbitrary but fixed nonempty set of closed pairwise non-intersecting line segments in the Euclidean plane (a line segment s is the convex hull of two distinct points p and q, which are called the endpoints of s).

V(S) denotes the set of endpoints of the line segments in S. We assume that not all points in V(S) are on a common line.

For $p \in V(S)$, *seg* p denotes the unique line segment s in S with $p \in s$ and *other* p denotes the unique endpoint q in V(S) with $p \neq q$ and *seg* p = *seg* q. □

Definition. Let p and q be two distinct points in the plane and let d be a direction. (We use direction and slope as a synonym for an angle between 0 and 2π.)

(i) pq denotes the open directed line segment from p to q. pq will also be called an edge, where q is the head of pq and p is the tail of pq.

(ii) pq is a view, if pq does not cross any line segment in S, i.e., either pq is disjoint from all line segments in S or pq is included in one of the line segments in S.

(iii) The slope of pq, $\sphericalangle pq$, is the counter-clockwise angle between the positive x-axis and pq (in this order).

(iv) The slope of pq relative to d, $\sphericalangle_d pq$, is the counter-clockwise angle between direction d and the edge pq (in this order); so $\sphericalangle pq = \sphericalangle_0 pq$. If e is an edge, then we write $\sphericalangle_e pq$ short for $\sphericalangle_{\sphericalangle e} pq$.

(v) We write $d' < d < d''$ for directions d' and d" if d',d,d" is the counter-clockwise order of these directions; $d' < d$ is short for $0 < d' < d$; $d' \leqslant d < d''$ is short for $(d' < d < d'' \text{ or } d' = d)$; $d' < d \leqslant d''$ is short for $(d' < d < d'' \text{ or } d = d'')$.

(vi) For a point r, we say r lies to the left of (lies to the right of, is collinear to) pq, if r lies to the left of (lies to the right of, lies on) the directed line through pq.

(vii) For a point r not collinear to pq, Δpqr denotes the open triangle with vertices p, q, and r. □

For our purposes it appears advantageous to consider the visibility graph as a directed graph where every undirected edge $\{p,q\}$ is represented twice by the directed edges (p,q) and (q,p). Recall that we call a directed line segment pq also an edge; we will extensively "apply" this abuse of notation by identifying the directed edge (p,q) in the directed visibility graph with the directed line segment (edge) pq.

Definition. The directed visibility graph G^* of S is the directed graph with node set $V(S)$ and edge set

$$E^* = \{pq \mid p,q \in V(S),\ pq \text{ is a view}\}. \quad \square$$

OBSERVATION 2.1. **(i)** If $e,e' \in E^*$, $\sphericalangle e = \sphericalangle e'$ and e and e' intersect, then $e = e'$.
(ii) If $pq,p'q' \in E^*$, and pq intersects $p'q'$ in more than one point, then $pq = q'p'$ or $pq = p'q'$.
(iii) $p \in e$ holds for no $p \in V(S)$ and $e \in E^*$. $\square$

As mentioned in the introduction, the algorithm we present performs a rotational sweep. Starting with some direction d, every endpoint looks in direction d and we start rotating d counter-clockwise. Whenever an endpoint p sees an endpoint q, then pq is an edge in G^*. If $\sphericalangle pq$ is the first direction in which p sees *seg* q then the "view-ray" of p just arrived on *seg* q; this gives an intuitive introduction for the notion of an "arriving edge" as defined now (likewise, the intuition for an "departing edge" should be obvious).

Definition. Let pq be an edge in E^*.
(i) pq is called arriving edge if *other* q does not lie to the right of pq. pq is called departing edge if *other* q does not lie to the

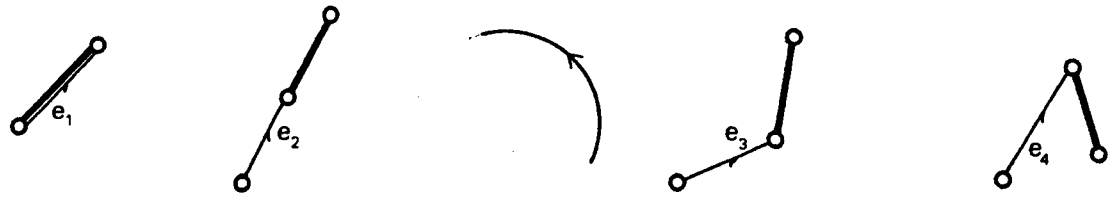

Figure 2.1.
Arriving (e_1, e_2, e_3), departing (e_1, e_2, e_4), only-departing (e_4) and only-arriving (e_3) edges.

left of pq. If pq is departing and not arriving, then pq is called only-departing; if pq is arriving and not departing, then pq is called only-arriving. (See Figure 2.1.)
(ii) We define a total order $\lhd$ on the edges in E*. Let p'q' be an edge in E' distinct from pq. Then

pq $\lhd$ p'q' if $\quad \sphericalangle pq < \sphericalangle p'q'$, or
$\sphericalangle pq = \sphericalangle p'q'$ and p lies to the left of p'q', or
$\sphericalangle pq = \sphericalangle p'q'$, p' is collinear to pq, and p' precedes p on the directed line through pq.

(iii) Let C* denote the sequence $(e_1, e_2, \ldots e_m)$ of all edges in E* sorted according to $\lhd$ (i.e., $e_i \lhd e_j$ for all i,j, $1 \leq i < j \leq m$). We consider C* as a cyclic ordering of E*: The successor of $e \in E^*$ (in C*), *succ* e, is defined by

$$\begin{aligned} \mathit{succ}\ e &= e_{i+1} \quad \text{if } e = e_i,\ 1 \leq i \leq m-1, \\ &= e_1 \quad \text{if } e = e_m. \end{aligned}$$

For $F \subseteq E^*$ and $e \in E^*$, the e-minimal edge in F is e, if $e \in F$, or the first edge in F that follows e in the cyclic order C*; (analogously, the e-minimal edge with a given property is defined). □

The ordering $\lhd$ is basically an ordering by slope. The tedious part of the definition of $\lhd$ concerns only edges of equal slope. These edges are first ordered from left to right. Then edges with equal direction on a common directed line ℓ are ordered in the direction opposite to the direction of ℓ.

Definition. Let p be a point in the plane and let d be a direction. The line segment visible from p in direction d, $vis_d p$, is the first line segment in S intersected by the open ray emanating from p in direction d; if no line segment in S is intersected by this ray, then we set $vis_d p = \infty$. □

OBSERVATION 2.2. Let d' and d" be directions and let p be a point in the plane. There is no point $q \in V(S)$ with pq a view and $d' < \sphericalangle pq < d"$ if and only if $vis_d p$ is the same for all directions d with $d' < d < d"$. □

3. Visibility graph following an edge.

The basic construct of our algorithm, the visibility graph following an edge e, is introduced in this section, together with a combinatorial analysis of its "maintainance" when e changes to its successor in C*. The algorithmic implementation of this maintainance will be postboned to the next section.

Convention. Throughout this section let e be an edge in E*. □

Definition. (i) For p in V(S), the point visible from p following edge e, $\varphi_e p$, is the head of the e-minimal edge with tail p. $\varphi_e^+ p$ is the head of the e-minimal edge with tail p that is different from e.
(ii) The visibility graph of S following edge e is the subgraph of G* with node set V(S) and edge set $F_e = \{p\varphi_e p \mid p \in V(S)\}$. □

Figure 3.1 shows a visibility graph following an edge e. We establish now a bridge to the intuitive description of the algorithm in the introduction. To this end we observe that if all the edges in E* have different slopes, we could as well use the notation "$\varphi_d p$" instead of "$\varphi_e p$" for a direction d and define $\varphi_d p$ as the first endpoint in V(S) seen from p when initially looking in direction d and then rotating counter-clockwise; then "$\varphi_e p = \varphi_d p$" for $d = \sphericalangle e$. The function φ_d would change only if $d = \sphericalangle e$ for an edge e in E* and this happens for exactly one point p in V(S) (still assuming, of course, that all edges have different slopes).

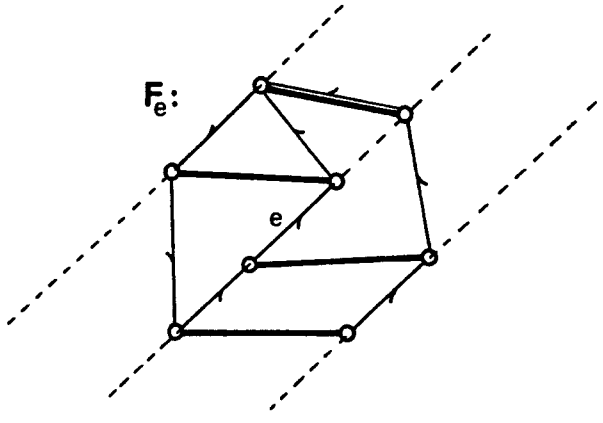

Figure 3.1.
A visibility graph following edge e.

If edges in E^* have common slopes, then φ_e becomes a kind of refinement of φ_d and it ensures that our algorithm has to change only one edge in the visibility graph following the current edge at a time.

The following observation reveals some properties which lead to our algorithm.

OBSERVATION 3.1. Let $p \in V(S)$. Then
(i) $\varphi_e{}^+p = \varphi_e p$ unless p is the tail of e.
(ii) *succ* e is the e-minimal edge in $\{p\varphi_e{}^+p \mid p \in V(S)\}$.
(iii) $F_{succ\ e} = \{p\varphi_e{}^+p \mid p \in V(S)\}$. □

An important property of the graphs F_e for following proofs is that no two edges in F_e intersect, with one exceptional case. In order to identify this exceptional case, we need the following definition.

Definition. An edge e' in F_e is called <u>overwound in</u> F_e if $\sphericalangle_e e' > \pi$. □

LEMMA 3.2. If two edges pq and $p'q'$ in F_e intersect, then $pq = q'p'$ and one of the two edges pq and $p'q'$ is overwound in F_e.

<u>Proof</u>. Let us first consider the case that one of the two edges pq and $p'q'$ is overwound, say, $p'q'$ is overwound. Then no point of $V(S)$ lies to the right of $p'q'$. So if pq and $p'q'$ intersect then p' and q' must be collinear to pq, and it follows that $pq = q'p'$.

Second, let neither pq nor $p'q'$ be overwound in F_e and we assume that pq and $p'q'$ intersect. Hence, $\sphericalangle pq \neq \sphericalangle p'q'$ and we may assume without loss of generality that $\sphericalangle e < \sphericalangle pq < \sphericalangle p'q'$ (see Figure 3.2 for an illustration). As $p' \notin pq$ and $q' \notin pq$ (recall Observation 2.1), we know that p' lies to the right of pq and q' lies to

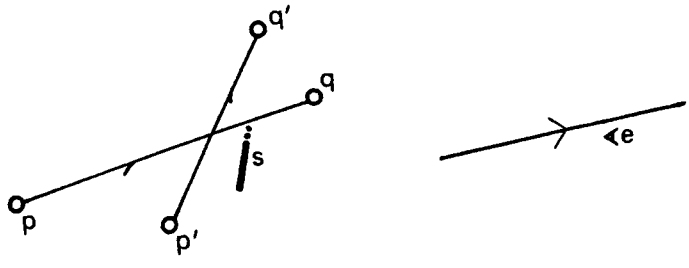

Figure 3.2.
Illustration for proof of Lemma 3.2.

the left of pq (recall that p'q' is not overwound). It follows that $\sphericalangle pq < \sphericalangle p'q < \sphericalangle p'q'$; thus p'q is not a view (otherwise p'q would be e-minimal among p'q and p'q'!). Consequently, $vis_d \cdot p'$ must intersect p'q for $d' = \sphericalangle p'q$. Moreover, $vis_d p'$ cannot change for d, $d' < d < d''$ with $d'' = \sphericalangle p'q'$ (recall Observation 2.2). Thus this line segment $s = vis_d \cdot p'$ intersects the directed line ℓ through p'q' (recall that the line segments in S are closed). Clearly, this intersection must either equal q' or it must succeed q'. In either case s intersects pq; a contradiction. □

A key concept in F_e for our algorithm are so-called chains that are sequences of edges. In order to define these chains, we need to further distinguish properties of edges in F_e (beyond "arriving" and "departing").

Definition. Let V be a finite set of points in the plane. For two points p and q in V, q is called V-hull-successor of p if no point of V lies to the right of pq and no point of V is element of pq.

The hull-sequence of V from p to q is the sequence $(x_1, x_2, \ldots, x_k)$ of pairwise different points in V such that $x_1 = p$, $x_k = q$, and x_{i+1} is V-hull-successor of x_i for all i, $1 < i < k-1$.

If q is V(S)-hull-successor of p, then pq is called hull-edge. □

Note that a point in V may have two V-hull-successors if all the points in V lie on a common line. Since we assume that the points in V(S) do not lie on a common line, it is easily seen that if a point in V(S) has a V(S)-hull-successor, then this V(S)-hull-successor is unique. A straightforward observation is that every hull-edge is a view and so it is an edge in E*. Moreover, only hull-edges can be overwound edges.

Definition. Let $pq \in F_e$.
(i) For an edge p'q in F_e, pq is the left-neighbor-edge of p'q in F_e if *other* $q \neq p'$ and there is no point y in V(S) with $\sphericalangle pq < \sphericalangle yq < \sphericalangle p'q$ such that $yq \in F_e$ or $y =$ *other* q.
(ii) pq is a rightmost arriving edge in F_e, if pq is not left-neighbor-edge of an arriving edge in F_e; analogously, we define a rightmost only-departing edge in F_e and a rightmost edge in F_e. □

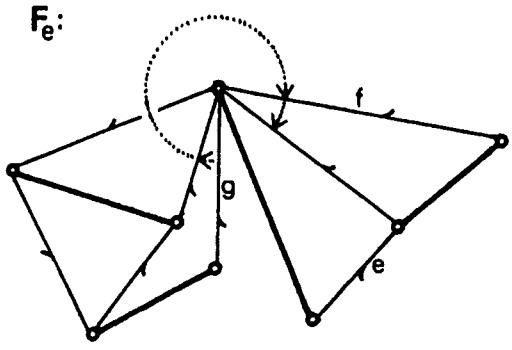

Figure 3.3.
Dotted arcs point from edges to their left-neighbor-edges.
f is a rightmost-arriving edge,
g is a rightmost-only-departing and a rightmost edge.

Definition. (i) Let $e', e'' \in F_e$. e' is called chain-predecessor of e'' in F_e (e'' is called chain-successor of e' in F_e) if the head of e' is the tail of e'', e' is a rightmost arriving edge in F_e, and neither e' nor e'' is a hull-edge.
(ii) A chain in F_e is a sequence $(e_1, e_2, \ldots e_k)$, $k \geq 1$, of edges in F_e such that e_{i+1} is chain-successor of e_i in F_e for all i, $1 \leq i \leq k-1$, e_1 has no chain-predecessor in F_e, and e_k has no chain-successor in F_e.
(iii) A sequence $(e_1, e_2, \ldots e_k)$, $k \geq 1$, of edges in F_e is a chain-suffix in F_e if there are edges $f_1, f_2, \ldots f_j$, $j \geq 0$, in F_e such that $(f_1, f_2, \ldots f_j, e_1, e_2, \ldots e_k)$ is a chain in F_e. □

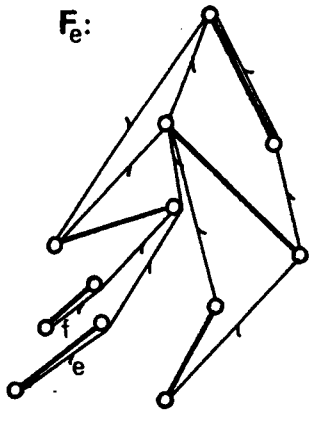

Figure 3.4.
Chains in a visibility graph following an edge e;
e starts a chain of three edges, f starts a chain of two edges,
all other chains consist of one edge only.

In Figure 3.4 the reader can find a graphical display of the chain structure in a visibility graph following an edge e. It should be obvious that every edge in a visibility graph following an edge e lies on exactly one chain. Moreover, it turns out that chains are convex curves:

LEMMA 3.3. Let $(e_1, e_2, \ldots e_k)$ be a chain in F_e. For each i, $1 \leq i \leq k$, no endpoint on this chain lies to the right of e_i.

Proof. The proof begins with two claims that reveal basic properties of chains.

CLAIM 1. If pq is an arriving edge in F_e then $\varphi_e q$ does not lie to the right of pq.

Proof of Claim 1. We assume that $r = \varphi_e q$ lies to the right of pq.

Case 1. $\sphericalangle e \leq \sphericalangle pq < \sphericalangle qr$: Since r lies to the right of pq, we have $\sphericalangle e \leq \sphericalangle pq < \sphericalangle qp < \sphericalangle qr$; moreover, qp is a view, because pq is one. This contradicts the fact that qr is the first edge with tail q that follows e in C^*.

Case 2. $\sphericalangle e \leq \sphericalangle qr < \sphericalangle pq$: Note that $\sphericalangle qr < \sphericalangle pr < \sphericalangle pq$ because r lies to the right of pq. Hence pr is not a view and we can conclude that $vis_d p$ is the same for all d, $\sphericalangle pr < d < \sphericalangle pq$; let s be this line segment (use Observation 2.2). If $s \neq vis_d p$ for $d = \sphericalangle pr$, then p sees an endpoint x in direction $\sphericalangle pr$ with $\sphericalangle e < \sphericalangle px < \sphericalangle pq$; a contradiction, which shows that $s \neq \infty$. Thus s intersects the directed line through pq in a point y succeeding p on this line. If y precedes q, then pq is not a view. If y equals q, then pq is not an arriving edge. If y succeeds q, then qr is not a view. In any case we obtained a contradiction to our assumption and so Case 2 and the claim is settled.

CLAIM 2. For no i, $1 \leq i \leq k-1$, $\sphericalangle e \leq \sphericalangle e_{i+1} < \sphericalangle e_i$.

Proof of Claim 2. Let $e_i = pq$ and $e_{i+1} = qr$. Since r does not lie to the right of pq, we have $\sphericalangle_{qr} pq > \pi$. Hence $\sphericalangle_e pq > \pi$ and pq is overwound in F_e and so pq is a hull-edge; a contradiction to the fact that e_{i+1} is chain-successor of pq in F_e. This proves the claim.

Let $e_i = pq$ be an edge on the chain $(e_1, e_2, \ldots e_k)$ such that an endpoint x on this chain lies to the right of pq.

Case 1. x succeeds q on the chain: By Claim 1, the first endpoint on the chain that succeeds r and that is not collinear to pq must lie to the left of pq. Hence there is an edge $e_j = yz$ that succeeds pq on the chain (i.e., $j > i$), such that z lies to the right of pq

and y does not lie to the right of pq (this holds, because the chain is connected). Note now that by Claim 2, $\sphericalangle e < \sphericalangle e_i < \sphericalangle e_j$; but on the other hand it is easily seen that $\sphericalangle_{pq} e_j > \pi$. Thus $\sphericalangle_e e_j > \pi$ which shows that e_j is overwound in F_e and so it is a hull-edge. However a chain that contains a hull-edge contains only one edge; a contradiction.

Case 2. x preceeds p on the chain: A symmetric reasoning to Case 1 leads to a contradiction. □

We are now ready to state and prove the lemmas which show how to propagate the edge e = pq in F_e to the edge $p\varphi_e^+p$ in $F_{succ\ e}$. We distinguish three cases:

(1) pq is an only-arriving edge.

(2) pq is a departing edge and $\varphi_e q \neq p$.

(3) pq is a departing edge and $\varphi_e q = p$.

Note that Case (1) implies $\varphi_e(q) \neq p$, since whenever pq and qp is in F_{pq} then no point in V(S) lies to the left of pq; in particular, *other* q does not lie to the left of pq and so pq is departing.

LEMMA 3.4. Let e = pq be an only-arriving edge. Let z = *other* q and let $x_1, x_2, \ldots x_k$ be the hull-sequence of $V(S) \cap (\Delta pqz \cup pz \cup \{p,z\})$ from p to z. Then:

(i) $x_2 = \varphi_e^+ p$.

(ii) $x_2 = z$ or $(x_2x_3, x_3x_4, \ldots x_{k-1}x_k)$ is a chain-suffix in F_e.

(iii) If $x_2 \neq z$, then $x_{k-1}z$ is a rightmost only-depar-

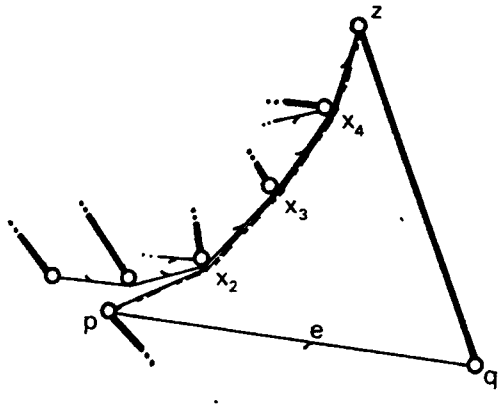

Figure 3.5.
Illustrating Lemma 3.4 for the case $x_2 \neq z$.

ting edge in F_e.

(iv) If $x_2 \neq z$, then px_2 is a rightmost arriving edge in $F_{succ\ e}$. If $x_2 = z$, then pz is a rightmost only-departing edge in $F_{succ\ e}$.

Proof. Note first that z lies to the left of pq and so Δpqz is well-defined. The lemma will now be proved by a sequence of claims.

CLAIM 1. No line segment s in S that intersects $\Delta pqz \cup pz$ intersects $\{q\}$, pq or qz.

Proof of Claim 1. If s intersects pq, then $pq \subseteq s$ (pq is a view!). If s intersects qz then $s = seg\ q = seg\ z$; likewise, if s contains q. In either case s is disjoint from $\Delta pqz \cup pz$, which proves the claim.

CLAIM 2. $x_i x_{i+1}$ is a view for all i, $1 \leq i \leq k-1$.

Proof of Claim 2. Assume $x_i x_{i+1}$ is not a view, i.e. there is a line segment s in S that crosses $x_i x_{i+1}$. Let r be the endpoint of s which lies to the right of $x_i x_{i+1}$. By the definition of a hull-sequence r is not in Δpqz; but this shows, that s intersects $\{q\}$, pq, or qz which is impossible because s has a point in $\Delta pqz \cup pz$, namely the intersection with $x_i x_{i+1}$ (use Claim 1). This shows the claim.

CLAIM 3. $x_2 = \varphi_e^+ p$.

Proof of Claim 3. Let $r = \varphi_e^+ p$. If $r \neq x_2$, then r must lie to the right of px_2 and to the left of pq. As r cannot be in Δpqz, pr must intersect $seg\ q$; a contradiction. Hence the claim holds.

CLAIM 4. $x_i x_{i+1} \in F_e$ for all i, $2 \leq i \leq k-1$.

Proof of Claim 4. Let $r = \varphi_e x_i$ for a fixed i, $2 \leq i \leq k-1$. If $r \neq x_{i+1}$, then $\sphericalangle pq < \sphericalangle x_i r < \sphericalangle x_i x_{i+1}$. As $\sphericalangle pq < \sphericalangle x_i x_{i+1} < qp$, this implies that r must lie to the right of $x_i x_{i+1}$. So r is not in Δpqz and so either $x_i r$ intersects pq, or $x_i r$ intersects qz, or $r = q$. The first two cases are impossible (recall Lemma 3.2). $r = q$ implies $\sphericalangle pq < \sphericalangle x_i x_{i+1} < \sphericalangle x_i r$, which contradicts our initial observation in the proof of this claim. Hence the claim is shown.

Since $q = other\ z$ and q lies to the right of $x_{k-1}z$, the edge $x_{k-1}z$ is departing. Thus, in order to show statement (ii) we have to show that all edges $x_i x_{i+1}$, $2 \leq i \leq k-2$, are rightmost arriving. The fact that these edges are arriving (i.e., $other\ x_{i+1}$ does not lie to the right of $x_i x_{i+1}$) follows easily from Claim 1. That these edges are not hull-edges holds because q lies to the right of all of these edges; that these edges are rightmost arriving follows from the following claim.

CLAIM 5. For no i, $2 < i < k-1$, there is an x to the right of $x_{i-1}x_i$ such that $xx_i \in F_e$.

Proof of Claim 5. If x is to the right of $x_{i-1}x_i$ for an i, $2 < i < k-1$, then xx_i intersects pq, qz, or $x = q$. In the first case, xx_i is not in F_e because of Lemma 3.2. In the second case xx_i is not a view (recall $i < k-1$). If $x = q$, then qz is e-minimal among qz and qx_i and so qx_i is not in F_e; the claim is proven.

In order to show (iii), let y be a point in V(S) that lies to the right of $x_{k-1}z$ and to the left of qz. As y is not in Δpqz, this implies that yz intersects pq which shows that yz is not in F_e and so it is not in $F_{succ\ e}$.

That px_2 is arriving if $x_2 \neq z$ follows again from Claim 1; that px_2 is rightmost arriving is proved in Claim 5 (for $i = 2$). This settles (iv) (the case $x_2 = z$ is treated in the previous paragraph). □

Lemma 3.4(i), (ii), and (iii) show where we have to search for $\varphi_{pq}{}^+p$ if pq is an only-arriving edge: either $\varphi_{pq}{}^+p = \textit{other}\ q$ or a chain is identified where $\varphi_{pq}{}^+p$ can be found; the following definition specifies this point. Lemma 3.4(iv) is important to understand how the chain structure changes from F_e to $F_{succ\ e}$. After the definition we summarize how to propagate an only-arriving edge.

Definition. Let $p \in V(S)$, and let $C = (e_1, e_2, \ldots e_k)$ be a chain in F_e. The target point of p on C is the last head r of an edge e_i in C such that the tail of e_i lies to the left of pr, provided such a point r exists and it is the tail of e_1, otherwise. □

LEMMA 3.5. Let $e = pq$ be an only-arriving edge, let $C = (e_1, e_2, \ldots e_k)$ be the chain in F_e where e lies on (with $e = e_i$), and let f be the rightmost edge with head $z = \textit{other}\ q$, provided such an edge exists.

(i) If f is not defined or if $\sphericalangle f < \sphericalangle pz < \sphericalangle qz$, then $\varphi_e{}^+p = z$ and C is replaced by the chains $(e_1, e_2, \ldots e_{i-1}, pz)$ and $(e_{i+1}, e_{i+2}, \ldots e_k)$ in $F_{succ\ e}$ (the second chain may be empty!); all other chains remain unchanged in $F_{succ\ e}$ (compared to F_e).
The left-neighbor-edge of pz in $F_{succ\ e}$ is f, or, if f is not defined, then pz has no left-neighbor-edge in $F_{succ\ e}$.

(ii) If f is defined and $\sphericalangle pz < \sphericalangle f < \sphericalangle qz$, then let D =

$(f_1, f_2, \ldots f_\ell)$ be the chain in F_e with $f_\ell = f$. Now $\varphi_e{}^+ p$ is the target-point x of p on D. Let j, $0 \leq j \leq \ell$, be the index such that x is the head of f_j, or $j = 0$, if x is the tail of f_1. Then the chains C and D in F_e are replaced by the chains $(f_1, f_2, \ldots f_j)$, $(e_1, e_2, \ldots e_{i-1}, px, f_{j+1}, \ldots f_\ell)$, and $(e_{i+1}, e_{i+2}, \ldots e_k)$ in $F_{succ\ e}$ (two of these chains may be empty!); all other chains remain unchanged in $F_{succ\ e}$ (compared to F_e). The left-neighbor-edge of px in $F_{succ\ e}$ is the rightmost arriving edge with head x in F_e, or, if no such edge exists, then px has no left-neighbor-edge in $F_{succ\ e}$.

Proof. The lemma follows from Lemmas 3.3 and 3.4. □

LEMMA 3.6. Let $e = pq$ be a departing edge such that $z = \varphi_e q$ is not equal to p. Let $x_1, x_2, \ldots x_k$ be the hull-sequence of $V(S) \cap (\Delta pqz \cup pz \cup \{p, z\})$ from p to z. Then:

(i) $x_2 = \varphi_e{}^+ p$.

(ii) $x_2 = z$ or $(x_2x_3, x_3x_4, \ldots x_{k-1}x_k)$ is a chain-suffix in F_e.

(iii) If $x_2 \neq z$, then $x_{k-1}z$ arriving implies that qz is arriving.

(iv) If $x_2 \neq z$, then px_2 is rightmost arriving in $F_{succ\ e}$. If $x_2 = z$, then pz arriving implies that qz is arriving.

Proof. First we verify that $z = \varphi_{pq} q$ lies to the left of pq if it is not equal to p (to make sure that Δpqz is well-defined). Assuming that this is not true, it is easy to see that either z succeeds q on the directed line through pq or z lies to the right of pq. In either case this implies that qp is pq-minimal among qp and qz (recall the ordering $\lhd$ for edges on a common line); a contradiction.

Similar to Lemma 3.4, we perform the proof of this lemma in a sequence of claims.

CLAIM 1. No line segment s in S that intersects $\Delta pqz \cup pz$ intersects $\{q\}$, pq or qz.

Proof of Claim 1. If s intesects pq, then $pq \subseteq s$; if s intersects qz, then $qz \subseteq s$ (pq and qz are views). If s intersects q, then $s = seg\ q$ and so $other\ q$ does not lie to the left of pq. In either case s is disjoint from $\Delta pqz \cup pz$, which proves the claim.

CLAIM 2. $x_i x_{i+1}$ is a view for all i, $1 \leq i \leq k-1$.

Proof of Claim 2. Take over verbatim the proof of Claim 2 in the proof of Lemma 3.4.

CLAIM 3. $x_2 = \varphi_e^+ p$.

Proof of Claim 3. Let $r = \varphi_e^+ p$. If $r \neq x_2$, then r must lie to the right of px_2 and to the left of pq. As r cannot be in $\Delta pqz \cup qz$, pr must intersect qz; a contradiction to Lemma 3.2. Hence the claim holds.

The proof of (ii) can be taken over from the proof of Lemma 3.4(ii) with one exception: we have to show that $x_{k-1}z$ has no chain-successor. If $x_{k-1}z$ is only-departing then this is true. If $x_{k-1}z$ is arriving then we show that qz is also arriving which implies that $x_{k-1}z$ is not rightmost arriving.

Assume that $x_{k-1}z$ is arriving and qz is not arriving and let y = *other* z. Then y does not lie to the right of $x_{k-1}z$ and lies to the right of qz. This implies that y lies to the left of pq and so $\sphericalangle pq < \sphericalangle qy < \sphericalangle qz$. Hence, qy is not a view (otherwise $\varphi_e q = y$ and not $\varphi_e q = z$). Moreover, $vis_d q$ does not change for d, $\sphericalangle qy < d < \sphericalangle qz$ (recall Observation 2.2). Since $s = vis_d q$ is closed, this implies that $s = seg\ z = seg\ y$. As qy is not a view $vis_d q$ has to change for $d = \sphericalangle qy$; but this implies that q sees an endpoint x in direction $\sphericalangle qy$ after all; a contradiction. This concludes the proof of (ii). Note that we have actually also proven (iii). The proof of (iv) for the case $x_2 = z$ is analoguous. The proof of (iv) for the case $x_2 \neq z$ follows from Claim 1. □

LEMMA 3.7. Let e = pq be a departing edge with $\varphi_e q \neq p$, let $ch = (e_1, e_2, \ldots e_k)$ be the chain in F_e where e lies on (with $e = e_i$), and let f be the left-neighbor-edge of qz, $z = \varphi_{pq} q$, provided such an edge exists.
(i) If f is not defined or if $\sphericalangle f < \sphericalangle pz < \sphericalangle qz$, then $\varphi_e^+ p = z$ and C is replaced by the chains $(e_1, e_2, \ldots e_{i-1}, pz)$ and $(e_{i+1}, e_{i+2}, \ldots e_k)$ in $F_{succ\ e}$ (the second chain may be empty!); all other chains remain unchanged in $F_{succ\ e}$ (compared to F_e).
The left-neighbor-edge of pz in $F_{succ\ e}$ is f, or, if f is not defined, then pz has no left-neighbor-edge in $F_{succ\ e}$.
(ii) If f is defined and $\sphericalangle pz < \sphericalangle f < \sphericalangle qz$, then let $D = (f_1, f_2, \ldots f_\ell)$ be the chain in F_e with $f_\ell = f$. Now $\varphi_e^+ p$

is the target-point x of p on D. Let j, $0 < j < \ell$, be the index such that x is the head of f_j, or j = 0, if x is the tail of f_1. Then the chains C and D in F_e are replaced by the chains $(f_1, f_2, \ldots f_j)$, $(e_1, e_2, \ldots e_{i-1}, px, f_{j+1}, \ldots f_\ell)$, and $(e_{i+1}, e_{i+2}, \ldots e_k)$ in $F_{succ\ e}$ (two of these chains may be empty!); all other chains remain unchanged in $F_{succ\ e}$ (compared to F_e). The left-neighbor-edge of px in $F_{succ\ e}$ is the rightmost arriving edge with head x in F_e, or, if no such edge exists, then px has no left-neighbor-edge in $F_{succ\ e}$.

Proof. The lemma follows from Lemmas 3.3 and 3.6. □

The reader will realize that after preparing the terminology in Lemma 3.7, the statements (i) and (ii) have been taken over from Lemma 3.5; this will be also exploited in the final algorithm in the forthcoming section. There is one of the three cases for propagating an edge which is still open.

LEMMA 3.8. Let e = pq be an edge with $\varphi_e q = p$, let $C = (e_1, e_2, \ldots e_k)$ be the chain in F_e where e lies on. Then $e = e_k$, $\varphi_e{}^+ p$ is the unique V(S)-hull successor x of p, and the chain C is replaced by the chains (px) and $(e_1, e_2, \ldots e_{k-1})$. (The second chain may be empty.) The left-neighbor-edge of px in $F_{succ\ e}$ is the rightmost arriving edge with head x in F_e, or, if no such edge exists, then px has no left-neighbor-edge in $F_{succ\ e}$.

Proof. If $\varphi_e q = p$, then qp is a hull-edge and so e is the last edge on its chain. This also implies, that there is no point to the right of px, $x = \varphi_e{}^+ p$ (for a point r to the right of px we would have $\sphericalangle pq < \sphericalangle pr < \sphericalangle px$; a contradiction). Hence, x is the V(S)-hull successor of p. The change in the chain structure follows from the fact that px is a hull-edge; this also implies that px is rightmost arriving in $F_{succ\ e}$. □

4. Constructing the visibility graph.

The important ingredients for our visibility algorithm have already been prepared in the previous section. The task that remains is to make sure that the necessary operations for an "algorithmic realization" of Lemmas 3.5, 3.7, and 3.8 can be done in logarithmic time. To this end we will need standard search datastructures only (as they are described, e.g., in [Mel]). For basics in computational geometry we refer to [PS] or [Me2].

First we describe four datastructures and their implementation that store the visibility graph following the current edge and the imposed chain structure.

Datastructure FOL.

(i) Description of operations: FOL stores the set of line segments S and its visibility graph following the current edge. It supports the following operations for edges e and g in this graph, for edge e' not in this graph, and for p in V(S).

(1) PHI(p) ..... returns the endpoint visible from p following the current edge.
(2) HEAD(e) ..... returns the head of edge e.
(3) TAIL(e) ..... returns the tail of edge e.
(4) OTHER(p) ..... returns *other* p.
(5) DEP?(e) ..... returns **true**, if e is departing, and **false**, otherwise.
(6) LEFT_NEIGHBOR(e) ... returns the left-neighbor-edge of e, provided it exists, and **empty** otherwise.
(7) RIGHTMOST(p) ... returns the rightmost edge with head p, provided it exists, and **empty**, otherwise.
(8) RIGHTMOST_ARRIVING(p) ... returns the rightmost arriving edge with head p, provided it exists, and **empty**, otherwise.
(9) REPLACE_IN_FOL(e,e',g) ... replaces e by e' in this structure assuming that e has the same tail as e', and that g is the left-neighbor-edge of e' in the new structure, provided e' has a left-neighbor-edge, and that g is **empty**, otherwise.

(ii) Implementation: In the implementation of FOL we will store

V(S) in a linear array and we will associate every edge with its tail. Every point p in V(S) has now seven pointers to the following objects.

- to the point (referred to as φp) visible from p following the current edge.
- to *other* p.
- to the tail of the rightmost arriving edge with head p.
- to the tail of the rightmost edge with head p.
- to the tail of the leftmost edge with head p (an edge is leftmost, if if has no left-neighbor-edge).
- to the tail of the left-neighbor-edge of the edge pφp.
- to the tail of the right-neighbor-edge of the edge pφp (i.e., to the tail of the edge of which pφp is left-neighbor-edge).

Of course, some of the pointers may point to **empty** if the corresponding objects do not exit. Note that the last four pointers listed realize a double linked list of all edges with head p (sorted by slope) with access to both ends of the list. It is now clear that all operations can be performed in constant time; in particular, the REPLACE_IN_FOL-operation can be done in constant time, since the left-neighbor-edge of the edge to insert is provided. The DEP?-operation can be computed directly. □

Datastructure QUEUE.

(i) Description of operations: QUEUE stores the edges of the visibility graph following the current edge. For edge e in this graph and edge e' not in this graph, it supports the following two operations.

(1) DELMIN(e) ..... deletes the edge e and returns the e-minimal edge among the remaining edges.
(2) INSERT(e') .... inserts the edge e'.

(ii) Implementation: A standard balanced tree structure will allow to perform both operations in logarithmic time. Actually, a careful use of a priority queue will do. □

Datastructure CHAIN.

(i) Description of operations: CHAIN stores a set of edge-sequences (the chains in the visibility graph following the current edge).

```
Algorithm SPARSE_VGRAPH.
 1. Input the set S of line segments;
 2. Initialize for S the structures FOL, QUEUE, CHAIN, and HULL;
 3. Let e be the edge such the structures store the visibility
        graph of S following edge e;
 4. p0 := TAIL(e); q0 := HEAD(e);
 5. repeat
 6.     p := TAIL(e); q := HEAD(e);
 7.     output {p,q} as edge of the visibility graph Gs;
 8.     C := CHAIN_OF(e);
 9.     SPLIT_CHAIN(C,q,Cpref,Csuff);
10.     DELETE_LAST(Cpref);
11.     if OTHER(q) = p then
12.         x := HULL_SUCC(p);
13.         CREATE_CHAIN(px);
14.         g := RIGHTMOST_ARRIVING(x)
15.     else
16.         if DEP?(e) then
17.             z := PHI(q);
18.             f := LEFT_NEIGHBOR(qz)
19.         else
20.             z := OTHER(q);
21.             f := RIGHTMOST(z)
22.         endif;
23.         if (f = empty) or (◁f < ◁pz < ◁qz) then
24.             x := z;
25.             CONCATENATE_CHAINS(Cpref,px,empty);
26.             g := f
27.         else
28.             D := CHAIN_OF(f);
29.             x := TARGET_ON_CHAIN(p,D);
30.             SPLIT_CHAIN(D,x,Dpref,Dsuff);
31.             CONCATENATE_CHAINS(Cpref,px,Dsuff);
32.             g := RIGHTMOST_ARRIVING(x)
33.         endif
34.     endif;
35.     REPLACE_IN_FOL(e,px,g);
36.     INSERT(px); e := DELMIN(e)
37. until e = q0p0.     □
```

We still have to take care for the initialization step in SPARSE_VGRAPH. For the datastructure HULL, this amounts to the computation of the convex hull of V(S), which can be done in $O(n \cdot \log n)$ time, $n = |S|$. For the other datastructures we compute F_e for e the first edge in the sequence C*. That is, for every p in V(S), $\varphi_e p$ is the edge in E* with minimal slope.

First $vis_o p$ is computed for all p in V(S). This can be done by a standard sweep technique in $O(n \cdot \log n)$ time, since the line-segments in S are non-intersecting. Then all endpoints in V(S) are processed according to their y-coordinates in decreasing order, endpoints with equal y-coordinates are processed according to their x-coordinates in decreasing order.

For the first point p_o, $\varphi_e p_o$ is the V(S)-hull successor of p_o. Let p be an endpoint to process, that is not p_o, and let $s = vis_o p$.
Case 1. $s = \infty$: Let z be the last endpoint already processed for which $z\varphi_e z$ is an hull-edge and let f be the rightmost arriving edge with head z, provided it exists, and let f = **empty**, otherwise ("rightmost arriving" concerns here edges already computed in F_e). Now we can proceed like in Lemma 3.4(i) and (ii). Note only that $p\varphi_e p$ may be an hull-edge, which changes the processing of the chain structure.
Case 2. $s \neq \infty$. If p sees an endpoint x in direction 0, then $\varphi_e p = x$ and it is obvious how to change the chain structure. If p does not see an endpoint in direction 0, then let z be the enpoint of s with the larger y-coordinate and let f be the rightmost edge with head z (again "rightmost" with respect to the already computed edges). Like in Case 1, we now proceed like in Lemma 3.4(i) and (ii).

A thorough treatment of the initialization step is omitted here; nevertheless, it should be clear that every edge in F_e can be computed in logarithmic time, which implies that the whole "initialization statement" in SPARSE_VGRAPH can be done in $O(n \cdot \log n)$ time.

The correctness of the remaining algorithm follows from Lemmas 3.5, 3.7, and 3.8. From the implementation descriptions for the datastructures we may conclude that every edge propagation needs $O(\log n)$ time. We summarize our result.

THEOREM 4.1. Algorithm SPARSE_VGRAPH computes the visibility graph Gs for a set S of n non-intersecting line segments in $O(m \cdot \log n)$ time and O(m) space, where m is the number of edges in Gs. □

It is worthwhile to mention explicitely, that the algorithm needs only O(n) working space. Thus, for example, the size m of Gs can be computed in $O(m \cdot \log n)$ time and O(n) space.

5. Discussion.

We have presented an algorithm for the efficient computation of sparse visibility graphs. As it appears, the algorithm has a rather complicated (or at least tedious) proof of correctness, however, its implementation seems to be simple and realistic. Still, we are currently working on some simplifications of the algorithm. For example, we can show that chains may be stored in (forward) linked lists with additional pointers from the first edge to the last edge on each chain and vice versa. Then we have to exploit that the edge to propagate is always "basically" the first edge on its chain, and that the TARGET_ON_CHAIN-operation can be implemented by simply walking along chains (starting from the first edge) without a prohibitive time requirement. Actually, this will even speed up all operations on chains. This gives some indication, that the overall running time of the algorithm may still be improvable.

The basic idea of the algorithm works also for sets of linesegments with common endpoints; thus also for simple polygons. An interesting application, although not new, is the triangulation of a set of non-intersecting linesegments (only endpoints of the linesegments may be vertices of the triangulation and the linesegments must be edges of the triangulation). This can be done by running the algorithm and inserting every edge propagated as a linesegment. As a third line of current research, we investigate possible further applications (e.g., visibility graphs for intersecting linesegments or for other objects like circles ...).

References.

[AAGHI] T.Asano, T.Asano, L.Guibas, J.Hershberger, and H.Imai, Visibility of disjoint polygons, *Algorithmica* 1 (1986) 49-63.

[EG] H.Edelsbrunner and L.J.Guibas, Topologically sweeping in an arrangement, *in* "Proc. 18th Ann. ACM Sympos. Theory Comput." (1986) 389-403.

[ES] H.Edelsbrunner and X.Shen, A tight lower bound on the size of visibility graphs, manuscript (1987).

[He] J.Hershberger, Finding the visibility graph of a simple polygon in time proportional to its size, *in* "Proc. 3rd ACM Sympos. on Comput. Geom.", to appear.

[Le] D.T.Lee, Proximity and reachability in the plane, Ph.D. Thesis, Dept. of Electrical Engeneering, University of Illinois at Urbana-Champaign (1979).

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

[Me1] K.Mehlhorn, *Data Structures and Algorithms 1: Sorting and Searching*, EATCS Monographs on Theoretical Computer Science Vol. 1 (1984) Springer Verlag, Berlin-Heidelberg.

[Me2] K.Mehlhorn, *Data Structures and Algorithms 3: Multi-dimensional Searching and Computational Geometry*, EATCS Monographs on Theoretical Computer Science Vol. 3 (1984), Springer Verlag, Berlin, Heidelberg.

[PS] F.P.Preparata and M.I.Shamos, *Computational Geometry -- An Introduction* (1985), Springer Verlag, New York-Berlin.

[SS] M.Sharir and A.Schorr, On shortest paths in polyhedral spaces, *in* "Proc. 16th Ann. ACM Sympos. Theory Comput." (1984) 144-153.

[We] E.Welzl, Constructing the visibility graph for n line segments in $O(n^2)$ time, *Inform. Process. Lett.* 20 (1985) 167-171.