Translating Polygons with Applications to Hidden Surface Removal

Mark de Berg

RUU-CS-89-31
December 1989

Utrecht University

Department of Computer Science

Padualaan 14, P.O. Box 80.089,
3508 TB Utrecht, The Netherlands,
Tel. : ... + 31 - 30 - 531454

Translating Polygons with Applications to Hidden Surface Removal

Mark de Berg

Technical Report RUU-CS-89-31
December 1989

Department of Computer Science
Utrecht University
P.O.Box 80.089
3508 TB Utrecht
The Netherlands

Translating Polygons with Applications to Hidden Surface Removal

Mark de Berg*

Abstract

Let S be a set of polygons in the plane. A translation ordering for S (in direction d) is an ordering of the polygons such that, if the polygons are moved one by one to infinity in direction d according to this ordering, no collisions occur between the polygons. We show that, after $O(n \log n)$ preprocessing using $O(n)$ space, it is possible to determine, for any given d, in $O(\log n)$ time whether such an ordering exists and, if so, to compute an ordering in $O(n)$ time.

Translation orderings correspond to valid orderings for hidden surface removal schemes where objects that are closer to the viewpoint are displayed later than objects that are farther away. Thus our technique can be used to generate displaying orderings for polyhedral terrains. One of the advantages of our approach is that it can easily be adapted to handle perspective views within the same time and space bounds.

Keywords Computational geometry, separation problems, hidden surface removal, relative convex hulls.

1 Introduction

In its most general form, the *separability problem* can be stated as follows. Given a set of objects in some space, separate them by a sequence of motions. During the motions, the objects should not collide with each other. (A collision between two objects occurs when their interiors have a non-empty intersection.) These problems come in many different flavors, depending on the objects that are considered, the space they are in, and the type of motions that is allowed. Toussaint [19] gives an extensive survey of such problems.

*Dept. of Computer Science, Utrecht University, P.O.Box 80.089, 3508 TB Utrecht, the Netherlands. Supported by the Dutch Organisation for Scientific Research (N.W.O.). Partially supported by the ESPRIT II Basic Research Actions Program of the EC under contract No. 3075 (project ALCOM).

We consider the following restricted version of the separability problem. Given a set $S = \{P_1, \ldots, P_m\}$ of non-intersecting polygons in the plane, translate them in some direction d to infinity, one at a time. Thus, every polygon has to be translated in the same direction. The problem now is to determine whether the polygons can be ordered such that, if the polygons are translated according to that ordering, no collisions occur and, if so, to compute such a *translation ordering.*

This problem, which is called the *translation problem*, originated in 1980, when Guibas and Yao [5] studied translation orderings for sets of convex polygons. They showed that a translation ordering always exists for a set of convex polygons and gave an $O(n + m \log m)$ algorithm for computing an ordering. (Here, and in the rest of this paper, m is the number of polygons and $n = \sum_{i=1}^{m} |P_i|$ is the total number of vertices of all polygons.). Since then, their work has been extended in several ways. Ottman and Widmayer [16] simplified the method and Nurmi [9] adapted the method to arbitrary polygons, achieving a time bound of $O(n \log n)$. Recently, Nussbaum and Sack [10] gave an optimal $O(n+m \log m)$ algorithm for this problem. Sack and Toussaint [17] showed how to compute, in $O(n \log n)$ time, all directions of separability (i.e., directions for which a translation ordering exists) for *two* arbitrary polygons, which was improved to $O(n \log \log n)$ by Toussaint [20]. Finally, Dehne and Sack [2] studied many of these problems when preprocessing is allowed: after $O(m^2(C_S(p)+\log m))$ time and using $O(m^2)$ space, they are able to answer all kinds of questions on translational orderings. (Here each polygon is assumed to have p vertices and $C_S(p)$ is the time needed to determine all directions of separability of two p-vertex polygons, which varies between $O(\log p)$ and $O(p \log \log p)$ depending on the type of the polygons.) Although their method is efficient when the number of polygons is small, it becomes very costly when there are many polygons. When all polygons have constant size, for example, their preprocessing takes time $O(n^2 \log n)$ and space $O(n^2)$ and computing an ordering for a given direction still takes $O(n^2)$ time.

In this paper it is shown that a set of (arbitrary) polygons can be preprocessed in time $O(n \log n)$ and space $O(n)$, such that it is possible to determine, for any given direction d, in time $O(\log n)$ whether there exists a translation ordering and to compute one (if it exists) in time $O(n)$. We also show that all directions of separability can be computed in $O(n \log n)$ time. Hence, when the number of polygons is large, this improves the results of Dehne and Sack [2] considerably.

One of the main applications of the translation problem is in computer graphics. To render a realistic picture of a scene, hidden surface removal has to be performed. One way to do this is to display the objects in the scene in a 'back to front' (with respect to the viewpoint) order. This way the objects in the front are painted over the objects in the back, thereby achieving the desired overlaying effect. A moment's thought will make it clear that a valid displaying order for this so-called *painter's algorithm* corresponds to a translation ordering for the objects in the direction perpendicular to the viewing plane. However, computing translation orderings in three

dimensions efficiently is difficult. Fortunately, for an important class of three dimensional scenes, the so-called *terrains* (polyhedral scenes in which the projections of the faces of the objects on the xy–plane do not intersect) solutions to the two dimensional translation problem can be used.

The translation ordering for the set of polygonal faces of a scene corresponds to a parallel view of the scene. This is often unwanted. One of the advantages of our method is that it can easily be adapted to yield a valid displaying order for perspective views within the same bounds. Thus we can preprocess a terrain consisting of convex polygonal faces with a total number of n vertices in time $O(n \log n)$ using $O(n)$ space, such that for any viewpoint a displaying order for the faces can be found in time $O(n)$. Notice that this gives a better space bound than the $O(n \log n)$ space that is needed in the binary partition scheme of Paterson and Yao [12].

The sequel of this paper is organised as follows.

We start in section 2 by presenting some preliminary results on relative convex hulls.

In section 3 our solution to the translation problem is presented. First, we consider sets of convex polygons to illustrate the main idea of our method, namely triangulating the area *in between* the polygons and translating the set of polygons augmented with the triangles of the triangulation. Then we show how to handle arbitrary polygons, using the concept of relative convex hulls.

In section 4, we discuss the application to hidden surface removal. It is shown how perspective views can be handled and how to treat viewpoints above the terrain.

We conclude with a brief summary of our results and by mentioning some open problems in section 5.

2 Relative convex hulls

In this section we present some results on *relative convex hulls*, a generalisation of convex hulls introduced by Toussaint [20]. Relative convex hulls are defined as follows. Define a polygonal circuit to be a closed polygonal path without (proper) self-crossings.

Definition 1 *Let P be a polygon and V a set of polygons. The* convex hull of P relative to V, *denoted $CH(P|V)$, is the polygon whose boundary is the shortest polygonal circuit that includes P but excludes V, i.e., $int(P) \subseteq int(CH(P|V))$ and $int(P') \subseteq ext(CH(P|V))$ for each $P' \in V$.*

(Thus our polygons are a slight generalization of a simple polygons, where we allow some edges and vertices to be used more than once.) Intuitively, if we release an elastic band that is wrapped around P, then it tries to take the shape of the convex hull of P but it can be stopped by the other polygons and it takes the shape of the relative convex hull of P. An example is given in Figure 1. Relative convex hulls exhibit the following useful properties:

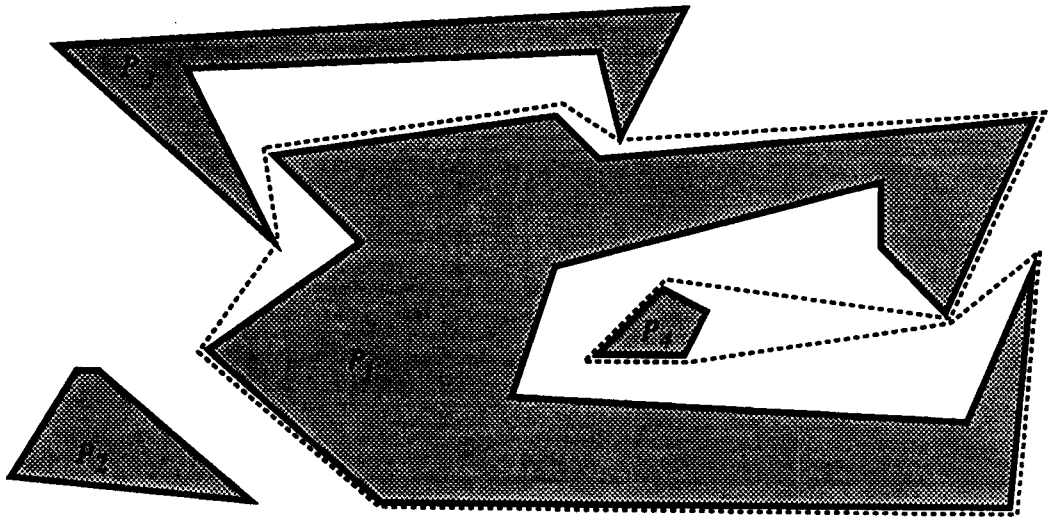

Figure 1: The dashed line is the boundary of the convex hull of P_1 relative to $\{P_2, P_3, P_4\}$. Note that the relative convex hull is not a simple polygon, since there is a vertex that is used twice.

Lemma 1 *Let P be a polygon and V,W be sets of polygons. Then:*

(i) *If v is a convex vertex of $CH(P|V)$, then v is a convex vertex of P.*

(ii) *If v is a reflex vertex of $CH(P|V)$, then v is a convex vertex of P or a convex vertex of some polygon $P' \in V$.*

(iii) *If $V \subseteq W$, then $CH(P|V) \supseteq CH(P|W)$.*

Proof: (i): Let v be a convex vertex of $CH(P|V)$ and let e, e' be the two edges of $CH(P|V)$ incident on v. If v is not a point of P (and clearly this point must be a convex vertex) then there is a small area around v that does not contain any point of P. More specifically, there are points $x \in e$, $x' \in e'$ $(x, x' \neq v)$ such that the triangle determined by x, x' and v does not contain any point of P. But this contradicts $v \in CH(P|V)$, since replacing $\overline{xv} \cup \overline{vx'}$ by $\overline{xx'}$ would yield a polygonal circuit still enclosing P (and excluding V) that is shorter.

(ii): Follows in the same way as (i). Note that if v is a convex vertex of P, then this vertex is used twice by $CH(P|V)$.

(iii): Suppose $V \subseteq W$, but $CH(P|V) \not\supseteq CH(P|W)$. Denote the boundaries of $CH(P|V)$ and $CH(P|W)$ by β and β', respectively. If $CH(P|V) \not\supseteq CH(P|W)$, then there must be some area A enclosed by portions γ of β and γ' of β' such that $A \subseteq CH(P|W)$ and $A \cap CH(P|V) = \emptyset$. (The reader should convince himself that both γ and γ' are connected portions of β and β'.) But γ cannot have a vertex that is convex with respect to A. Such a vertex would be reflex w.r.t. $CH(P|V)$ and thus, by (ii), be a vertex of P or of a polygon $P' \in V$. The first case cannot occur since it contradicts the fact that $CH(P|V)$ contains P. The second case is

impossible since $V \subseteq W$, $A \subseteq CH(P|W)$ and $CH(P|W)$ excludes W. Similarly, γ' cannot have a vertex that is convex w.r.t. A. Such a vertex would be convex w.r.t. $CH(P|W)$ and therefore be a convex vertex of P. This contradicts the fact that $CH(P|V)$ contains P. Of course, it is impossible that neither γ nor γ' contains a convex vertex and, hence, area A cannot exists. □

Let S be a set of polygons. For a polygon $P \in S$ we define $P^* = CH(P|S - \{P\})$ to be the convex hull of P relative to the rest of S and we define $S^* = \{P^*|P \in S\}$ to be the set of these relative convex hulls. In the remainder of this section it is shown how S^* can be computed efficiently. Toussaint [20] has shown how to do this for a set of two polygons. Using ideas similar to his, we show how this can be done for larger sets of polygons.

The idea is to compute first an area around each polygon P, called the *sleeve* of P, that contains the boundary of its relative convex hull. Then we determine a point of which we know that it is on the boundary of the relative convex hull and we compute a shortest circuit that starts at this point, goes 'around' P and returns to this point. This last part is done using algorithms of Chazelle [1] or Lee and Preparata [8]. They have shown that if the dual tree of the triangulation of a simple polygon is a chain, then a shortest path between two points in such a polygon can be computed in time linear in the number of vertices of the polygon. More precisely, S^* is computed by the following algorithm.

1. Let R be a large rectangle that contains S properly, i.e., $S \subseteq int(R)$. Triangulate $R - S$, the area inside R between the polygons.

2. For each $P \in S$, compute $P^* = CH(P|S - \{P\})$ as follows.

   (i) Add as many triangles that are inside $CH(P)$ to P as possible: while there is a triangle T that shares two edges with P, add T to P. (This step is to ensure that the sleeve that is constructed in the next step does not contain any dead ends, i.e., its dual tree is indeed a chain.) Obviously these triangles will lie completely in P^*. Hence, adding them to P wil not change P^*.

   (ii) Compute $sleeve(P)$ in the following way. Let v_0 be the leftmost vertex of P and T_0 the triangle sharing the edge $\overline{v_0 v_1}$ with P, where v_1 is the next vertex of P in counterclockwise direction. Starting at v_0, walk along the boundary of P, concatenating the triangles that share at least one vertex with P in the natural order to each other, until T_0 is encountered again. More formally, we initialize $T := T_0$ as the triangle just added and $v := v_0$ as and the vertex that is currently treated, and we advance as follows: Let $T_1, \ldots, T_k$ be the counterclockwise enumeration of the triangles that have v as a vertex. If T is the last triangle in this order, set v to the next (in counterclockwise order) vertex along the boundary of P. (Observe

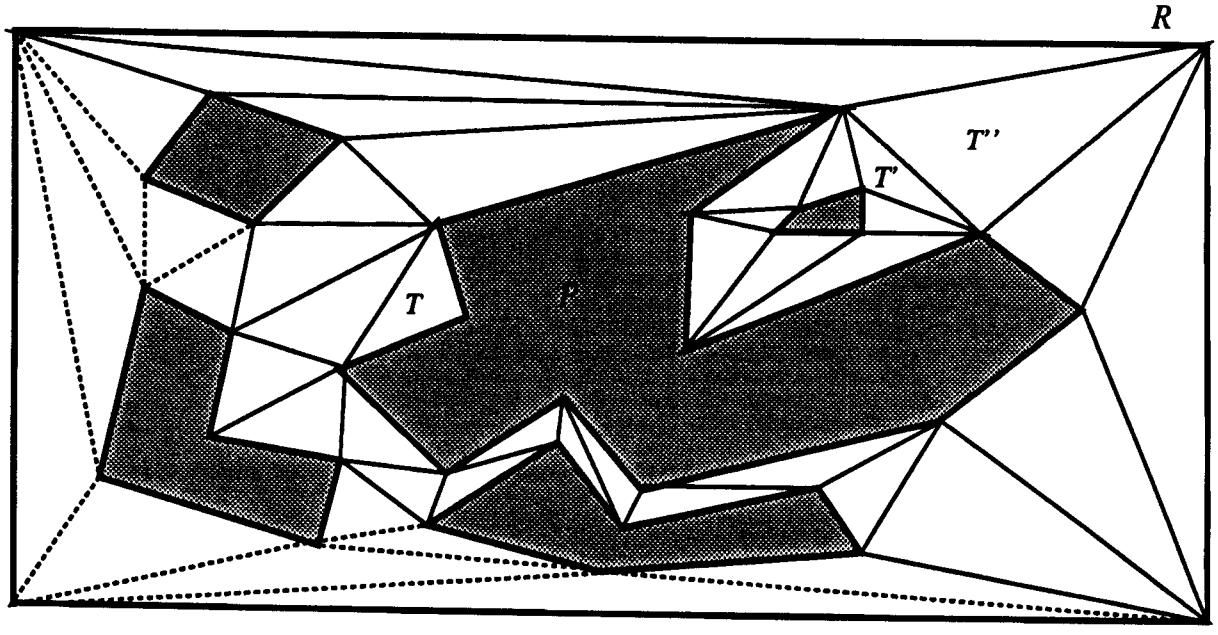

Figure 2: The (non–dotted) triangles around P form the sleeve of P. Triangle T is added in step 2(i) of the algorithm. Observe that triangle T' occurs two times in $sleeve(P)$.

that T also has this vertex as one of its vertices.) Otherwise set T to the next triangle in this order and add T to $sleeve(P)$.

(iii) Compute a shortest path from v_0 in T_0 to (the copy of) v_0 in T_l, the last triangle added to $sleeve(P)$, using the algorithm of [1] or [8]. This path is the boundary of P^*.

See Figure 2 for an illustration.

Lemma 2 *The algorithm given above correctly computes S^* in $O(n \log n)$ time and $O(n)$ space.*

Proof: First we prove the correctness of the algorithm and then we show that it works within the stated bounds.

It is evident that the circuit that is computed in step 2(iii) of the algorithm contains P and excludes $S - \{P\}$. We argue that (the boundary of) P^* is confined to $sleeve(P)$. For suppose that it intersected some triangle T not in $sleeve(P)$, then T has a vertex inside P^* that is not a vertex of P. But then it would be a vertex of some other polygon P' and P^* would not exclude P'. Therefore the boundary of P^* must lie in the union of all triangles that share at least one vertex with P. Finally, it is easily seen that (the interior of) the triangles that are added to P in

step 2(i) cannot contain a part of the boundary of P^*. Since v_0 lies on $CH(P)$ it will certainly be a vertex of P^* and it follows that P^* is indeed the shortest path from v_0 in T_0 to v_0 in T_l inside $sleeve(P)$.

By construction the dual of the triangulation of $sleeve(P)$ is a polygonal chain, so we can indeed use the algorithms of [1] or [8]. This is true although $sleeve(P)$ is not necessarily a simple polygon: some triangle could occur more than once in the sleeve. However, Toussaint [20] observed that this is no real problem. If we embed $sleeve(P)$ onto a Riemann surface of several levels ([6]), so that if a triangle occurs for the second (or third) time it lies 'above' its previous occurence, so to speak, the algorithm that computes the shortest path won't know any better than that it is working with a simple polygon and returns the correct path.

To prove the time bound, we note that step 1 takes $O(n \log n)$ time. To perform step 2(i) efficiently, we first make for each polygon P a list of the triangles that share two edges with P. This can easily be done in linear time in total. Then we add these triangles to P and examine the triangles adjacent to them to see if they have to be added too, etc. This way step 2(i) takes only $O(n)$ time for all polygons in total. Steps 2(ii) and 2(iii) take $\sum_{P \in S} O(|sleeve(P)|)$ time. To estimate $\sum_{p \in S} |sleeve(P)|$, we first note that any of the $O(n)$ triangles is added to a sleeve if a vertex that it shares with some polygon P is encountered during the traversal of the boundary of P. Hence, any triangle can occur at most three times in a sleeve (i.e., once in three sleeves, three times in one sleeve, etc.) and the total complexity of all sleeves is $O(n)$. The time bound follows, as well as the space bound. □

Remark: Notice that the time bound in the lemma above is determined by the time needed to compute the triangulation of a polygon (R) with holes (the polygons in S). Therefore, if the number of polygons in S is constant, the algorithm can be implemented to work in $O(n \log \log n)$ time: first remove the holes from the polygons (in linear time per hole) and then triangulate the remaining simple polygon using the algorithm of Tarjan and van Wyk [18].

3 Translating polygons

A translation ordering for a set S of polygons (in direction d) is defined as an ordering such that if the polygons are moved one at a time (in direction d) to infinity according to this ordering, no collisions occur. The computation of translation orderings involves computing some sort of dominance relation between the polygons. A polygon P dominates another polygon P' if P' collides with P when it is moved before P is moved. Thus a translation ordering exists iff the dominance relation between the polygons is free of cycles. It has been shown by Guibas and Yao [5] that it is not necessary to compute all (possibly $\Theta(m^2)$) dominances explicitly, but that it suffices to compute the *immediate* dominances. (P immediately dominates P' if, when P' is moved, some portion of P' intersects some portion of P before it intersects some other polygon.)

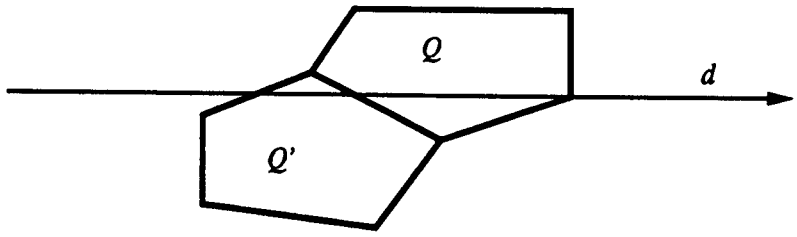

Figure 3: Q is a d-neighbour of Q'.

This immediate dominance relation changes radically, however, when the direction of translation d changes. Hence, if we want to do preprocessing to speed up the computation of a translational ordering for any given d, we have to take a different approach. We will show that a triangulation of the area in between the polygons gives us all the information we need to compute a translation ordering for any given direction.

3.1 Translating convex polygons

Let $S = \{P_1, \ldots, P_m\}$ be a set of convex polygons. The convex hull of S is denoted by $CH(S)$. Furthermore let $\mathcal{T} = \{T_1, \ldots, T_k\}$ be a triangulation of $CH(S) - S$, the area in between the polygons of S. The idea is to translate the set $S \cup \mathcal{T}$. Observe that this new set still contains only convex polygons and, hence, it can still be translated. Surprisingly, translating $S \cup \mathcal{T}$ is an easier task than translating S, as follows from the lemma given next. First we define d-neighbours, a concept that is crucial in our method.

Definition 2 *Let Q and Q' be two polygons. Q is a* d-neighbour *of Q' iff*
(i) Q and Q' share an edge e
(ii) there is a ray in direction d that intersects $int(Q')$ just before it intersects e and $int(Q)$ just after it intersects e.

Notice that if two polygons Q and Q' share an edge e, then either Q is a d-neighbour of Q', or Q' is a d-neighbour of Q, or e is parallel to d. See Figure 3 for an illustration of this definition.

Lemma 3 *A polygon $Q \in S \cup \mathcal{T}$ (possibly a triangle) can be translated to infinity in direction d without collisions if and only if all its d-neighbours already have been translated without collisions.*

Proof: The "only if"-part is trivial. To prove the "if"-part, suppose that all d-neighbours of Q have been translated without collisions, but that Q still collides with some polygon Q'. Consider the moment that Q and Q' first intersect during

the translation. This intersection involves an edge e of Q. But then the d-neighbour of Q that shares e (which must exist since the area in between Q and Q' has been triangulated) would also collide with Q, which contradicts the assumptions. □

Lemma 3 immediately leads to the following simple scheme.

The preprocessing just consists of computing a triangulation $\mathcal{T}$ of $CH(S) - S$ and the dual graph $G(S \cup \mathcal{T})$ of $S \cup \mathcal{T}$. (The nodes in this graph correspond to the polygons in $S \cup \mathcal{T}$ and there is an arc between two nodes iff the corresponding polygons share an edge.)

Now, given a query direction d, we proceed as follows. First we turn $G(S \cup \mathcal{T})$ into a directed graph G_d. Let a be an arc in $G(S \cup \mathcal{T})$ connecting nodes corresponding to polygons Q and Q'. If Q is a d-neighbour of Q' then the arc in G_d corresponding to a, denoted a_d, is directed from Q to Q'. If Q' is a d-neighbour of Q then a_d is directed from Q' to Q. Otherwise (the edge shared by Q and Q' is parallel to d) a has no corresponding arc in G_d. Thus (a node corresponding to) some polygon has incoming arcs from all its d-neighbours and outgoing arcs to all polygons for which it is a d-neighbour.

From Lemma 3 and the definition of G_d it easily follows that a topological ordering of the nodes in G_d corresponds to a translation ordering in direction d for the polygons in $S \cup \mathcal{T}$. (Note that the fact that $S \cup \mathcal{T}$ can be translated guarantees that G_d is acyclic.) Clearly, if the triangles of $\mathcal{T}$ are omitted from of this ordering, we get the desired translation ordering for S. This leads to:

Lemma 4 *A set S of convex polygons can be preprocessed in $O(n \log n)$ time and $O(n)$ space such that, given a direction d, a translation ordering for S in direction d can be computed in time $O(n)$.*

Proof: The convex hull of S as well as the triangulation (and its dual graph) can be computed in time $O(n \log n)$ and $O(n)$ space ([13]). (Note that the total number of edges in $S \cup \mathcal{T}$ (and therefore the number of nodes and arcs in $G(S \cup \mathcal{T})$ as well) is $O(n)$).

Since we can decide in constant time for an arc a in $G(S \cup \mathcal{T})$ what the direction of its corresponding arc a_d in G_d will be, the construction of G_d takes only linear time. Topologically sorting a directed (acyclic) graph can also easily be done in linear time (see, e.g., Knuth [7]). □

3.2 Translating arbitrary polygons

To apply the same idea to a set of arbitrary polygons, some extra preprocessing has to be done. The problem is that if there are non-convex polygons, the triangles of the triangulation might prevent the existence of a translation ordering, i.e., it is possible that a translation ordering for S exists, but not for $S \cup \mathcal{T}$. Consider for example the case where S consist of one U-shaped polygon P. The triangles of

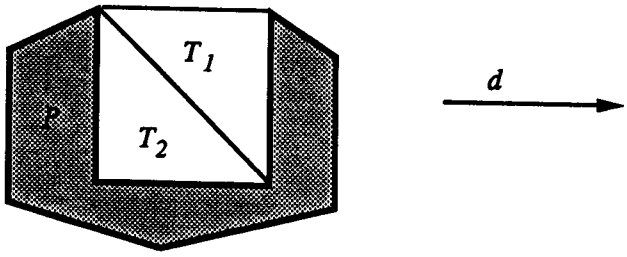

Figure 4: $\{P\}$ can be translated, but $\{P, T_1, T_2\}$ cannot be translated.

the triangulation of $CH(P) - P$ will prevent a translation ordering (which trivially exists for P alone) in a horizontal direction. See Figure 4. This problem can be overcome by using the concept of relative convex hulls (see section 2).

In the remainder we shall need the following lemma, proved by Toussaint in [19].

Lemma 5 ([19]) *A translation ordering for a set of polygons exists if and only if there exists a translation ordering for every pair of polygons in the set.*

Now we are ready to show that the method of the previous section can be used if we first replace every polygon by its convex hull relative to the other polygons. Let $S = \{P_1, \ldots, P_m\}$ be a set of polygons. For a polygon $P \in S$ we define $P^* = CH(P|S - \{P\})$ and we define $S^* = \{P^*|P \in S\}$. Any ordering on S naturally corresponds to a unique ordering on S^* (and vice versa) and this correspondence is also preserved when restricted to translation orderings, as the following lemma shows:

Lemma 6 *An ordering on S is a translation ordering (in direction d) for S if and only if the corresponding ordering on S^* is a translation ordering (in direction d) for S^*.*

Proof: Toussaint has proved in [20] that two polygons P_i and P_j collide if and only if $CH(P_i|P_j)$ and $CH(P_j|P_i)$ collide. Since, by definition of relative convex hulls and by Lemma 1 (iii), $P_i \subseteq P_i^* \subseteq CH(P_i|P_j)$ and $P_j \subseteq P_j^* \subseteq CH(P_j|P_i)$, this implies that P_i and P_j collide if and only if P_i^* and P_j^* collide. □

Once we have replaced the polygons in S by their relative convex hulls, it is safe to triangulate the region between the polygons and add the triangles to the set to be translated. Let $\mathcal{T}$ be a triangulation of $CH(S^*) - S^*$, then we have:

Lemma 7 *There exists a translation ordering in direction d for S^* if and only if there exists a translation ordering in direction d for $S^* \cup \mathcal{T}$.*

Proof: The "if"-part is trivial. The proof of the "only if"-part makes use of Lemma 5. Since there exists (by assumption) a translation ordering for every pair

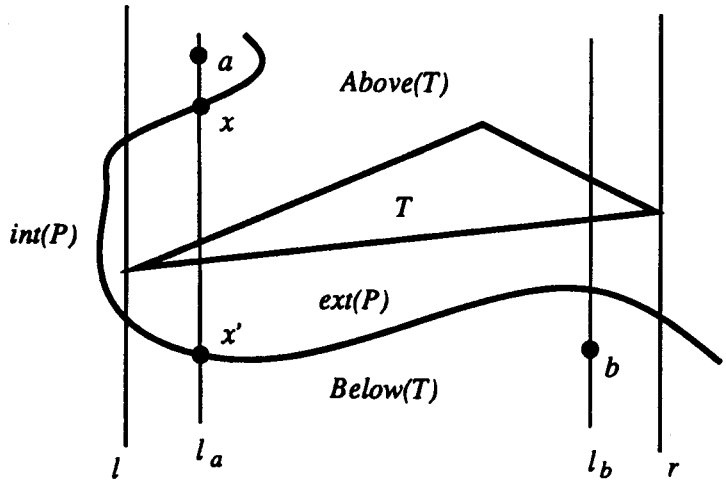

Figure 5: Illustration of the proof of Lemma 6.

of polygons in S^*, and there exists a translation ordering for every pair of triangles (they are convex), it remains to prove that there exists a translation ordering for every pair $P^* \in S^*$, $T \in \mathcal{T}$.

Suppose for a contradiction that some pair P^*, T cannot be ordered and assume w.l.o.g. that d is vertically upward. Let l and r be the two vertical lines tangent to T and denote the area between l and r and above T by $Above(T)$ and the area below T by $Below(T)$ (see Figure 5). If there is no ordering for P^* and T then $Above(T) \cap P^* \neq \emptyset$ and $Below(T) \cap P^* \neq \emptyset$. Let a be a point in the first intersection and b a point in the second intersection and let l_a and l_b denote the vertical lines through a and b. At least one of l_a and l_b, say l_a, must intersect P^* above T as well as below T. Consider x, the first intersection of l_a with P^* above T, and x', the first intersection point below T. Now $\overline{xx'}$ cuts off some portion of $ext(P^*)$ (that has a non-empty intersection with T). Clearly, the part of the boundary of P^* that borders this portion is longer than $\overline{xx'}$. But this means that there must be (a part of) a polygon $P' \in S - \{P\}$ lying in this portion of $ext(P^*)$, contradicting the fact that P^* and $(P')^*$ can be ordered. □

We thus arrive at the following scheme for translating a set S of arbitrary polygons. As a preprocessing step, S^* and a triangulation $\mathcal{T}$ of $CH(S^*) - S^*$ (together with its dual graph $G(S^* \cup \mathcal{T})$) are computed in $O(n \log n)$ time and $O(n)$ space. (See section 2 for the details of the computation of S^*.) Then, given a direction d, we construct G_d as in the previous section. (Note that there might be more than one arc between two nodes in G_d, because two polygons can share more than one edge.) If S^* can be translated in direction d, then G_d does not contain a cycle and we can compute a topological ordering of the nodes in G_d. This ordering corresponds to a translation ordering for S^* (omitting the triangles of the triangulation) which,

by Lemma 6, corresponds to a translation ordering for S. (Note that Lemma 3 is true for non-convex polygons too.) If G_d contains a cycle, then $S^* \cup \mathcal{T}$ cannot be translated. By Lemma's 6 and 7 we can then conclude that no translation ordering for S exists either.

Lemma 8 *A set S of arbitrary polygons can be preprocessed in $O(n \log n)$ time and $O(n)$ space such that, given a direction d, a translation ordering for S in direction d can be computed in time $O(n)$ (if it exists).*

3.3 Computing all directions of separability

In this section it is shown that all directions of separability (i.e., all directions for which a translation ordering exists) can be computed in $O(n \log n)$ time. If this is done as a preprocessing step, then whether or not a translation ordering exists in a given direction can be decided in $O(\log n)$ time.

Toussaint has shown in [20] that there exists a translation ordering for two polygons in direction d if and only if their relative convex hulls are monotonic in direction $d + \frac{1}{2}\pi$. He uses this result to compute all directions of separability of two polygons. Lemma 5 implies that the fact stated above for two polygons also holds for larger sets of polygons:

Lemma 9 *There exists a translation ordering in direction d for a set of polygons if and only if the relative convex hulls of the polygons are monotonic in direction $d + \frac{1}{2}\pi$.*

The proof is not difficult (although some care has to be taken because the relative convex hulls are different if we consider pairs of polygons in isolation) and therefore omitted. Monotonicity of a polygon can be characterized as follows:

Observation 1 *A polygon is monotonic in direction d if and only if it has no reflex vertex v such that the two edges incident to v lie on the same side of the line through v with slope d.*

For a reflex vertex v of some $P^* \in S^*$, let $I_v \subset [0 : 2\pi]$ be the interval such that the two edges incident to v lie on the same side of a line through v with slope d if and only if $d \in I_v$ (in fact, I_v can consist of two disjoint intervals, one starting at 0, the other ending at 2π). Given the two edges incident to v, I_v is easily computed in constant time. Thus, in $O(n)$ time, we can compute $I(S^*) = \{I_v | v \text{ is a reflex vertex of a } P^* \in S^*\}$. By Lemma 9 and Observation 1, a translation ordering for S in direction d exists iff $d \notin \bigcup I(S^*)$. In other words, the set D of directions for which a translation ordering exists is the set $[0 : 2\pi] - \bigcup I(S^*)$. This leads to:

Theorem 1 *All directions for which a translation ordering exists for a given set S of polygons with a total number of n vertices can be determined in time $O(n \log n)$.*

Proof: S^* can be computed in time $O(n \log n)$ (see section 2) and, as we have seen, $I(S^*)$ in linear time. By sorting the endpoints of the intervals in $I(S^*)$ and performing a line sweep keeping track of the number of intervals currently intersected by the sweep point, the set $D = [0 : 2\pi] - \bigcup(S^*)$ can be found in time $O(n \log n)$. □

Observe that D consists of $O(n)$ disjoint intervals. Hence, D can be stored in a search tree which can be built in $O(n \log n)$ time and uses $O(n)$ space. With this tree it can be decided, for a given direction d, in time $O(\log n)$ if $d \in D$ and thus if there exists a translation ordering in direction d.

We now state our main theorem, which summarizes the results of this section.

Theorem 2 *A set S of polygons, with a total number of n vertices, can be preprocessed in $O(n \log n)$ time and $O(n)$ space, such that, given any direction d, it can be decided in time $O(\log n)$ if there exists a translation ordering for S in direction d and, if so, an ordering can be computed in time $O(n)$.*

4 Application to hidden surface removal

One of the most important applications of translation orderings is in computer graphics when performing hidden surface removal. When an object of a scene is displayed onto a screen, it is painted over the objects that already have been displayed. Therefore, the objects must be displayed in a 'back to front' order. This order corresponds to a translation order perpendicular to the projection plane. Translation orderings for polygons in the plane can be used to obtain displaying orders for so-called (polyhedral) *terrains*. A terrain is a set of polygonal faces in 3–space that do not intersect when projected onto the xy–plane[1]. Observe that this is a very general definition of a terrain: we do not require the scene to be 'connected' (as, e.g., is necessary for the hidden surface removal algorithm of Reif and Sen [15]).

We will now show how our translation algorithm can be used to generate displaying orders for the most general type of views for terrains consisting of convex faces: a perspective view from an arbitrary point. Let $F = \{f_1, \ldots, f_m\}$ be a set of convex polygons in 3–space, the faces of the terrain, and let $\overline{F} = \{\overline{f_1}, \ldots, \overline{f_m}\}$ be the (non–intersecting) set of projections of these faces onto the xy–plane. Let h be the viewing plane and let X be the viewpoint. Thus we want to project the faces of F onto h as seen by an observer at position X. Again, we permit ourselves a preprocessing of $O(n \log n)$ to compute a triangulation $\mathcal{T}$ of $CH(\overline{F}) - \overline{F}$ and its dual graph $G(\overline{F} \cup \mathcal{T})$. After this, given a viewpoint X and a viewing plane h, a correct displaying order can be calculated in linear time, as is shown in the remainder of this section.

Let us assume that $\overline{X}$, the projection of X onto the xy–plane, does not lie in (the

[1]Here and in the sequel, all projections onto the xy–plane are orthogonal.

interior of) the convex hull of $\overline{F}$. (This can easily be accomplished by splitting $\overline{F} \cup \mathcal{T}$ into two sets with a line through $\overline{X}$.)

To find a valid displaying ordering for the faces of F that corresponds to a perspective view all that we have to change in the algorithms of the previous section is the concept of neighbourhood.

Definition 3 *Let Q and Q' be two polygons and $\overline{X}$ be a point in the plane. Q is an* $\overline{X}$-neighbour *of Q' iff*

(i) Q and Q' share an edge e

(ii) There is a ray starting at $\overline{X}$ and intersecting e that intersects $int(Q')$ just before intersecting e and $int(Q)$ just after intersecting e.

The analogon of Lemma 5 is as follows.

Lemma 10 *A face $f \in F$ can be safely displayed (onto h, perspective w.r.t. X) if all the faces corresponding to $\overline{X}$-neighbours of $\overline{f}$ in $\overline{F} \cup \mathcal{T}$ already have been displayed safely.*

Proof: Denote the (perspective) projection of a face f onto h by $proj(f)$ and let f_i, f_j be two faces such that $proj(f_i) \cap proj(f_j) \neq \emptyset$. Thus, there is a ray r starting at X that intersects both f_i and f_j. We argue that f_i and f_j are displayed in the correct order, i.e., the face that is intersect closest to X is displayed last. To see this, consider the projections $\overline{f_i}$, $\overline{f_j}$, $\overline{X}$ and $\overline{r}$ on the xy-plane. Clearly, $\overline{r}$ intersects $\overline{f_i}$ and $\overline{f_j}$ in the same order as r intersects f_i and f_j. Suppose $\overline{r}$ intersects $\overline{f_i}$ first, then either $\overline{f_j}$ is an $\overline{X}$-neighbour of $\overline{f_i}$, in which case f_j is (correctly) displayed first, or there is some $\overline{X}$-neighbour of $\overline{f_i}$ that is intersected by $\overline{r}$. But then this neighbour must have been displayed (safely!) before f_i which implies that also f_j must have been displayed before f_i. □

Of course, there are no real faces corresponding to the triangles that were added when $CH(\overline{F}) - \overline{F}$ was triangulated, and displaying a face corresponding to such a triangle is just a dummy statement. Also (the parts of) the faces that lie on the same side of h as X should not be displayed. Note that, since all faces are convex, we always find an ordering. We conclude:

Theorem 3 *A terrain F consisting of convex polygonal faces with a total number of n vertices can be preprocessed in $O(n \log n)$ time and $O(n)$ space such that, given a viewpoint X and a projection plane h, a valid displaying order for the faces of F to obtain a perspective view can be determined in $O(n)$ time.*

Remark: If the terrain contains non-convex faces, we can always cut up these faces into convex parts without changing the complexity of the scene. The restriction to convex faces is necessary because if there are non-convex faces it is possible that there is a valid displaying order for the faces of the terrain, but no translation order for the corresponding 2-dimensional problem. Consider, e.g., the case where the terrain is completely contained in the xy-plane and the faces are such that they cannot be translated. In spite of this, a valid displaying order exists for viewpoints above the terrain. (In fact, any ordering is valid.)

5 Concluding remarks

In this paper, we have presented an efficient solution to the translation problem. It was shown that, after $O(n \log n)$ preprocessing using $O(n)$ space, a translation ordering for a set of polygons in the plane can be determined in linear time (if it exists). One of the advantages of our method is that it can easily be adapted to yield a valid displaying order for *perspective* views of a terrain (consisting of convex polygonal faces) to be used in hidden surface removal. It should be stressed that the preprocessing of the terrain as well as the algorithm that yields the displaying order are conceptually very simple and a good candidate for efficient implementations.

The main open questions concern dynamization and, more importantly, translation orderings in 3 dimensions.

Using the the methods of [4], edges (or polygons of constant size) can be inserted and deleted in linear time in our structure. A method might exist with better update times. (Note that it takes linear time to compute a translation ordering anyway, so a linear update time is in fact not that bad.)

As yet, little work has been done on the 3–dimensional problem. Nurmi [9] has considered 3–dimensional translation orderings, but his results are theoretically not very strong, since the time complexity of his algorithm depends of the number of intersections in the viewing plane. Nussbaum and Sack [10] have considered the problem of determining all directions of separability for two polyhedra. Egyed [3] has tried to find efficient ways to cut up a scene such that the 2–dimensional translation algorithms can be used. As for the method of this paper, it seems that this method is doomed to fail in 3 dimensions. First of all, there is no efficient way (yet?) to tetrahedralize the space in between a set of polyhedra, let alone to find a tetrahedralization that does not prevent an ordering of the polyhedra. Even worse is the fact the size of such a tetrahedralization may be $\Omega(n^2)$.

Acknowledgement

I would like to thank Peter Egyed for proposing the problem and for useful discussions on the problem. Also, I thank Mark Overmars for giving valuable comments.

References

[1] Chazelle, B., A Theorem on Polygon Cutting with Applications, *Proc. 23rd Annual IEEE Symp. on Foundations of Computer Science*, 1982, pp. 339-349.

[2] Dehne, F., and J.-R. Sack, Separability of Sets of Polygons, *Proc. 12th International Workshop on Graph-Theoretic Concepts in Computer Science*, 1986, pp. 237-251.

[3] Egyed, P., Hidden Surface Removal in Polyhedral–Cross–Sections, *The Visual Computer* **3** (1988), pp. 329-343.

[4] El Gindy, H.A., and G.T. Toussaint, Efficient Algorithms for Inserting and Deleting Edges from Triangulations, *Proc. Int. Conf. on Foundations of Data Organization*, 1985.

[5] Guibas, L.J., and F.F. Yao, On Translating a Set of Rectangles, in: F.P. Preparata (Ed.), *Advances in Computing Research, Vol. I: Computational Geometry*, JAI Press Inc., 1983, pp. 61-77.

[6] Kahn, J., M. Klawe and D. Kleitman, Traditional Galleries Require Fewer Watchmen, *SIAM J. Alg. Disc. Meth.* **14** (1983), pp. 194-206.

[7] Knuth, D.E., *Fundamental Algorithms: The Art of Computer Programming I*, Addison-Wesley, Reading, Mass., 1968.

[8] Lee, D.T., and F.P. Preparata, Euclidean Shortest Paths in the Presence of Rectilinear Barriers, *Networks* **14** (1984), pp. 393-410.

[9] Nurmi, O., On Translating a Set of Objects in Two– and Three–dimensional Space, *Computer Vision, Graphics and Image Processing* **36** (1986), pp. 42-52.

[10] Nussbaum, D., and J.-R. Sack, Translation Separability of Polyhedra, *manuscript*, presented at the 1st Canadian Conf. on Computational Geometry.

[11] Nussbaum, D., and J.-R. Sack, Disassembling Two-dimensional Composite Parts Via Translations, *Proc. Int. Conf. on Optimal Algorithms*, 1989.

[12] Paterson, M.S., and F.F. Yao, Binary Partitions with Applications to Hidden-Surface Removal and Solid Modelling, *Proc. 5th Annual ACM Symp. on Computational Geometry*, 1989, pp. 23-32.

[13] Preparata, F.P., and M.I. Shamos, *Computational geometry, an introduction*, Springer-Verlag, New York, 1985.

[14] Preparata, F.P., and K.J. Supowit, Testing a Simple Polygon for Monotonicity, *Inform. Proc. Letters* **12** (1981), pp.161-164.

[15] Reif, J.H., and S. Sen, An Efficient Output-Sensitive Hidden-Surface Removal Algorithm and its Parallelization, *Proc. 4th Annual ACM Symp. on Computational Geometry*, 1988, pp. 193-200.

[16] Ottman, T., and P. Widmayer, On Translating a Set of Line Segments, *Computer Vision, Graphics and Image Processing* **24** (1983), pp. 382-389.

[17] Sack, J.-R., and G.T. Toussaint, Translating Polygons in the Plane, *Proc. 2nd Annual Symp. on Theoretical Aspects of Computer Science*, 1985, pp. 310-321.

[18] Tarjan, R.E., and C.J. van Wyk, An $O(n \log \log n)$ Time Algorithm for Triangulating Simple Polygons, *SIAM J. Comput.* **17** (1988), pp. 143-178.

[19] Toussaint, G.T., Movable Separability of Sets, in: G.T. Toussaint (Ed.), *Computational Geometry*, North Holland, 1985, pp. 335-376.

[20] Toussaint, G.T., On Separating Two Simple Polygons by a Single Translation, *Techn. Rep. SOCS-88.8*, McGill University, 1988.

[21] Yao, F.F., On the Priority Approach to Hidden-Surface Algorithms, *Proc. 21st Annual IEEE Symp. on Foundations of Computer Science*, 1980, pp. 301-307.