Translation queries for sets of polygons

M. de Berg, H. Everett, H. Wagener

RUU-CS-91-30
August 1991

Utrecht University

Department of Computer Science

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

Translation queries for sets of polygons

M. de Berg, H. Everett, H. Wagener

Technical Report RUU-CS-91-30
August 1991

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

ISSN: 0924-3275

Translation Queries for Sets of Polygons

Mark de Berg* Hazel Everett† Hubert Wagener‡

Abstract

Let S be a set of m polygons in the plane with a total of n vertices. A translation order for S in direction d is an order on the polygons such that no collisions occur if the polygons are moved one by one to infinity in direction d according to this order. We show that S can be preprocessed in $O(n \log n)$ time into a data structure of size $O(m)$ such that a translation order for a query direction can be computed in $O(m)$ time, if it exists. It is also possible to test whether a translation order exists in $O(\log n)$ time with a structure that uses $O(n)$ space. These results are achieved using new results on relative convex hulls and on embeddings with few vertices, which are interesting in their own right.

Translation orders correspond to valid orders for hidden surface removal with the painter's algorithm. Our technique can be used to generate displaying orders 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, painter's algorithm, relative convex hulls, embeddings.

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

*Dept. of Computer Science, Utrecht University, P.O.Box 80.089, 3508 TB Utrecht, the Netherlands. Supported by the Dutch Organization 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).

†Département de mathématiques et d'Informatique, Université du Québec à Montréal, Case Postale 8888, succursale A, Montréal, Québec, Canada, H3C 3P8.

‡TU Berlin, FB Informatik, FB 6-2, Franklinstr. 28/29, D-1000 Berlin, Germany.

space they are in, and the type of motion that is allowed. Toussaint [25] gives an extensive survey of such problems.

We consider the following restricted version of the separability problem. Given a set $S = \{P_1, \ldots, P_m\}$ of non-overlapping polygons (i.e. polygons with pairwise disjoint interiors) in the plane, translate them in some direction d to infinity, one at a time. Thus, every polygon has to be translated into the same direction. The problem now is to determine whether the polygons can be ordered such that no collisions occur if the polygons are translated according to that order and, if so, to compute such a *translation order*. This problem, which is called the *translation problem*, originated in 1980, when Guibas and Yao [13] studied translation orders for sets of convex polygons. They showed that a translation order always exists for a set of convex polygons and gave an $O(n + m \log m)$ algorithm for computing an order. 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 [23] simplified the method and Nurmi [16] adapted the method to arbitrary polygons, achieving a time bound of $O(n \log n)$. Recently, Nussbaum and Sack [17] gave an optimal $O(n + m \log m)$ algorithm for this problem. Sack and Toussaint [24] showed how to compute, in $O(n \log n)$ time, all directions of separability (i.e., directions for which a translation order exists) for *two* arbitrary polygons, which was improved to $O(n)$ by Toussaint [26]. Finally, Dehne and Sack [8] 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 orders. (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 $O(p)$ 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 $O(n^2 \log n)$ time and $O(n^2)$ space and computing an order 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 $O(n \log n)$ into a data structure of size $O(n)$, such that it is possible to determine, for any given direction d, in time $O(\log n)$ whether there exists a translation order. Moreover, such an order can be computed (if it exists) in $O(m)$ time using a structure of size $O(m)$. This improves the results of Dehne and Sack [8] considerably. We also show that all directions of separability can be computed in $O(n \log n)$ time.

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 order for the objects in the direction perpendicular to the viewing plane. However, it is difficult to compute translation orders in three dimensions efficiently. Only recently an algorithm has been proposed by de Berg et al. [7] that computes such an order in subquadratic time. The running time of this algorithm is $O(n^{4/3+\epsilon})$, which is considerably worse than the time that can be achieved in the planar case. Fortunately, for an important class of three dimensional scenes, the so-called *polyhedral terrains*—polyhedral scenes in which the projections of the faces of the objects onto the xy–plane do not intersect—solutions to the two-dimensional translation problem can be used.

The translation order 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 m convex polygonal faces with a total number of n vertices in time $O(n + m \log n)$ into a data structure of size $O(m)$, such that for any viewpoint a displaying order for the faces can be found in time $O(m)$. 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 [19].

We achieve our results using new results on *relative convex hulls* and *embeddings*, which are of independent interest.

The convex hull of a polygon P relative to a set S of polygons is the polygon that contains P and excludes S whose boundary has minimal length. This means that the polygon is made 'as convex as possible'; since convex polygons are easier to translate than non-convex polygons, this will help us in solving the translation problem. We show how to compute, in total time $O(n \log n)$, for each polygon in a set S its convex hull relative to the rest of the polygons.

An embedding of a set of non-overlapping polygons is set of non-overlapping polygons such that each polygon in the original set is contained in exactly one polygon of the embedding. For convex polygons it is known that there always exist embeddings with a total number of vertices that is linear in the number of polygons [27]. For non-convex polygons this is not always true. However, we show that there exists an embedding with few vertices that allows us to reduce the space complexity of the data structure for translation queries to linear in the number of polygons. This embedding can be computed in $O(n \log n)$ time.

This paper is organized as follows. We start in Section 2 by presenting our new results on relative convex hulls. In Section 3 we show how to construct small embeddings of sets of non-overlapping polygons. These results are used in Section 4 to obtain an efficient solution to the translation problem. In Section 5 we discuss the application to hidden surface removal and show how perspective views can be handled. We conclude with a brief summary of our results and by mentioning some open problems in Section 6.

2 Relative Convex Hulls

In this section we present our results on *relative convex hulls*, a generalization of convex hulls, introduced by Toussaint [26]. 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, that is, with $int(P) \subseteq int(CH(P|V))$ and $int(P') \subseteq ext(CH(P|V))$ for each $P' \in V$.*

(Thus our polygons are a slight generalization of 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, where the dashed

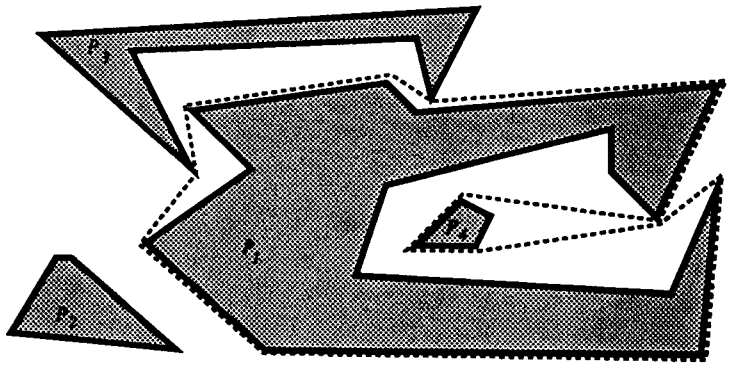

Figure 1: A relative convex hull.

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. Relative convex hulls exhibit the following useful properties:

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 a point of P then clearly v is a convex vertex of P.

If v is not a point of P 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 exist. □

Let S be a set of non-overlapping 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 [26] 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 which we know 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 an algorithm of Chazelle [2] or Lee and Preparata [15]. They have shown that if the dual tree of the triangulation of a simple polygon is a chain, then the shortest path between two points in such a polygon can be computed in time linear in the number of vertices of the polygon. Next, we give a more precise description of the algorithm that computes S^*. See Figure 2 for an illustration.

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.

(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. Meanwhile concatenate the triangles that are incident to each vertex in the same order as they are encountered to each other, until T_0 is reached again. Thus, if a triangle is incident to more than one vertex of P, it is added more than once to the sleeve.

(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 [2] or [15]. This path is the boundary of P^*.

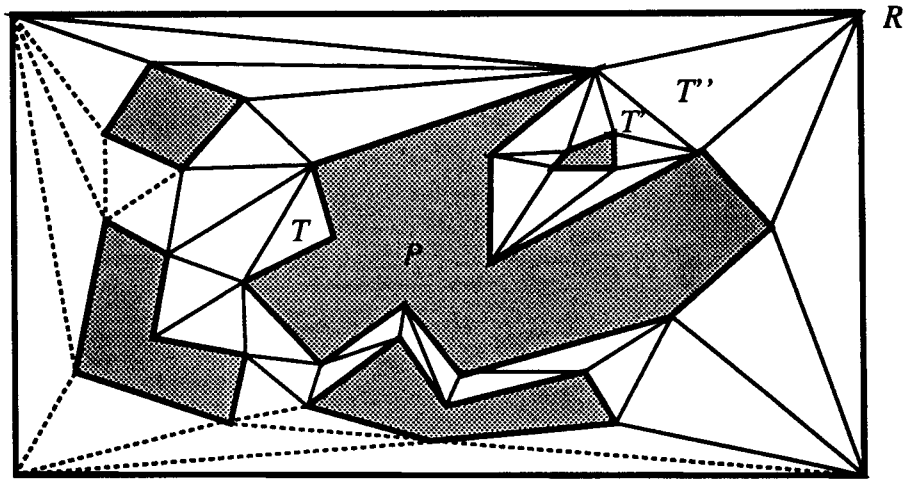

Figure 2: The (non–dotted) triangles around P form the sleeve of P. Triangle T is added in step 2(i) of the algorithm. Observe that triangles T' and T'' occur two times in *sleeve*(P).

Theorem 1 *Let S be a set of non-overlapping polygons with n vertices in total. The set $S^* = \{CH(P|S - \{P\}) : P \in S\}$ of relative convex hulls can be computed 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 intersects 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. Furthermore, 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^*. Hence, these triangles will lie completely in P^* and adding them to P will not change P^*. Finally, 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 $sleeve(P)$.

Because we add in Step (i) all the triangles that share two edges with P to P^*, the dual of the triangulation of $sleeve(P)$ is a chain. Thus we can use the algorithms of [2] or [15]. This is true even though $sleeve(P)$ is not necessarily a simple polygon: some triangle could occur more than once in the sleeve. However, Toussaint [26] observed that this is no real problem: one can embed $sleeve(P)$ onto a surface of several levels, so that if a triangle occurs for the second (or third) time it lies 'above' its previous occurrence. Algorithms that work for simple polygons also work in this case.

To prove the time bound, we note that step 1 takes $O(n \log n)$ time, see for example [20]. 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, etcetera. 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 (that is, once in three sleeves, three times in one sleeve, etcetera) 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 $O(n)$ time [1].

Any order on S naturally corresponds to a unique order on S^* (and vice versa) and this correspondence is also preserved when restricted to translation orders, as the following lemma shows:

Lemma 2 *An order on S is a translation order (in direction d) for S if and only if the corresponding order on S^* is a translation order (in direction d) for S^*.*

Proof: Toussaint has proved in [26] 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. □

3 Embedding Polygons

Let $S = \{P_1, \ldots, P_m\}$ be a set of non-overlapping polygons in the plane. An *embedding* of S is a set $\tilde{S} = \{\tilde{P}_1, \ldots, \tilde{P}_m\}$ of non-overlapping polygons such that $P_i \subseteq \tilde{P}_i$, for $1 \leq i \leq m$. In this section we will show that any set S can be embedded into a set $\tilde{S}$ that has few vertices and, moreover, that can be translated if and only if S can be translated. This embedding will help us to devise a space-efficient solution to the translation problem.

3.1 Embedding Convex Polygons

Let us start by considering a set $S = \{P_1, \ldots, P_m\}$ of disjoint (thus, they are not allowed to touch) *convex* polygons, with n vertices in total. It is known that S can be embedded into a set $\tilde{S}$ with only $O(m)$ vertices in total. We sketch an algorithm due to Wenger [27] that computes such an embedding.

Augment S with three dummy triangles P_{m+1}, P_{m+2} and P_{m+3} such that the convex hull of S consists of one vertex from each of P_{m+1}, P_{m+2} and P_{m+3}. Consider a *triangulation* of S. A triangulation of S is a planar subdivision consisting of the set S with additional line segments, called *triangulation segments*, between vertices of distinct polygons in S. These edges are such that each face which is not a polygon is bounded by exactly three triangulation segments and portions of the boundary of at most three polygons, as in Figure 3. Let G be the planar graph whose nodes are

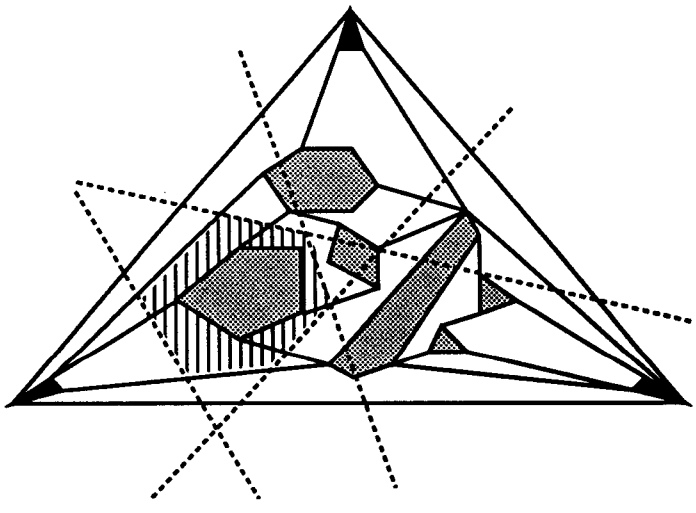

Figure 3: A triangulation of a set of polygons and the embedding for one of them. The black triangles are the dummy triangles.

the $m + 3$ polygons in S and whose edges are the triangulation segments. Since G

is planar the total number of edges in G is $O(m)$. (This is true even though there can be multiple edges between two nodes, because such edges are 'topologically' different.) For each polygon $P_i \in S$, let $N(P_i)$ be a list of all the neighbors of P_i in G. Let $l_{i,j}$ be a line separating P_i from P_j and let $L_1(P_i) = \{l_{i,j} : P_j \in N(P_i)\}$. Furthermore, let l_e be the line containing edge e of G and let $L_2(P_i) = \{l_e : e$ is a triangulation edge connecting two neighbors of P_i, e is on some face with P_i and l_e does not intersect $P_i\}$. For each $1 \leq i \leq m$, let $\widetilde{P}_i$ be the intersection of the half-planes containing P_i that are bounded by lines in $L_1(P_i)$ and $L_2(P_i)$. Now $\widetilde{S} = \{\widetilde{P}_1, \ldots, \widetilde{P}_m\}$. See Figure 3.

Lemma 3 *An embedding $\widetilde{S}$ of S with $O(m)$ vertices can be computed in $O(n + m \log n)$ time.*

Proof: The fact that $\widetilde{S}$ is an embedding with $O(m)$ vertices is not hard to prove. We refer to [27] for a precise proof. It remains to show that $\widetilde{S}$ can be computed in $O(n + m \log n)$ time.

First we compute a triangulation of S. To this end we triangulate (in the usual sense) $CH(S) - S$, the region in between the polygons in $O(n + m \log n)$ time [1]. Note that each (bounded) face which is not a polygon contains either 2 or 3 triangulation segments. To obtain a triangulation of S, we just remove triangulation segments that are not on the convex hull, until each face is bounded by three triangulation segments. Thus, we remove a segment if it is part of a face having only two triangulation segments. Note that the resulting face still has either 2 or 3 triangulation segments. Hence, when no more edges can be removed, we have obtained a triangulation of S. Since there are $O(n)$ segments in the triangulation of $CH(S)-S$, it is straightforward to compute the triangulation of S in $O(n)$ time.

The next step is to compute the separating lines in $L_1(P_i)$, for each P_i. One separating line $l_{i,j}$ can be computed in time $O(\log |P_i| + \log |P_j|) = O(\log n)$ by the algorithm of Edelsbrunner [9] or Chin and Wang [6]. The total time to compute all $O(m)$ separating lines is thus $O(m \log n)$. The intersection of the half-planes bounded by lines in $L_1(P_i) \cup L_2(P_i)$ can be computed in $O((|L_1(P_i)| + |L_2(P_i)|) \log(|L_1(P_i)| + |L_2(P_i)|))$ time [20], which adds up to $O(m \log m)$ time for all P_i's in total. □

Note that $\widetilde{S}$ consists of convex polygons and, hence, is still translatable into any direction. Moreover, any translation order for $\widetilde{S}$ is also valid for S (but not vice versa), since the polygons of S are contained in those of $\widetilde{S}$.

3.2 Embedding Arbitrary Polygons

Let $S = \{P_1, \ldots, P_m\}$ be a set of non-overlapping simple polygons with n vertices in total. Our goal is to find an embedding of S with few vertices that is still translatable. In the convex case it is always possible to find an embedding with $O(m)$ vertices in total. In general this is not always possible. See Figure 4. However, we will be able to compute an embedding that is good enough for our purposes.

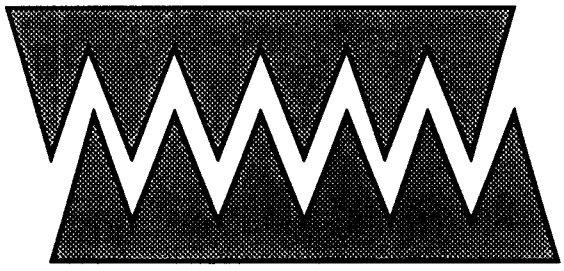

Figure 4: Two polygons for which any embedding has $\Omega(n)$ vertices.

We first compute the set S^* of relative convex hulls, as in Section 2. This way we make the polygons 'as convex as possible', which will help us to apply the same ideas that were used in the convex case. Recall from Section 2 that a polygon $P^* \in S^*$ may not be simple. However, if this is the case then the set of polygons admits no translation order [26] so we will assume that this case does not occur. Two polygons in S^* may share a number of edges as, for example, the relative convex hulls of the two polygons in Figure 4. Let D denote the set of edges that are shared by pairs of polygons in S^*. We will show how to construct an embedding $\tilde{S} = \{\tilde{P}_1, \ldots, \tilde{P}_m\}$ for S^*, and thus for S, such that $\sum_{i=1}^{m} |\tilde{P}_i| - |D| = O(m)$.

As in the convex case, let us augment S^* with three dummy triangles such that the convex hull of S^* consists of one vertex of each of these triangles. Next, we triangulate the region $CH(S^*) - S^*$. We want to remove certain edges from the triangulation of $CH(S) - S$ to obtain a triangulation of S^*. Recall that in a triangulation of a set of polygons it is required that each face is bounded by three triangulation edges. To meet this requirement when the polygons are allowed to touch, we have to add degenerate triangulation segments between touching polygons. More specifically, for each maximal chain of boundary edges that is shared by two polygons we add the first and last vertex of this chain as degenerate triangulation segments. Now the same procedure as in the convex case can be used to obtain a triangulation of S^*: remove non-degenerate triangulation segments that are not on the convex hull until each face is bounded by exactly three triangulation edges. Note that there may be triangulation edges between vertices of the same polygon.

The boundary edges of a polygon in S^* that are not shared with another polygon in S^* form a number of convex chains. Let σ be such a chain. Note that σ does not share vertices with other polygons, except possibly the first and last vertex. We will show how to construct the chain $\tilde{\sigma}$ that replaces σ in the embedding.

Chain σ is on the boundary of a number of faces of the triangulation. Let $f_1, \ldots, f_a$ be an ordered enumeration of these faces, which together form $sleeve(\sigma)$, and let σ_i, $1 \leq i \leq a$, be the portion of σ on the boundary of f_i. Note that it is possible that σ_i consists of just one vertex. Let t_i, $1 \leq i \leq a-1$, be the triangulation

segment separating f_i from f_{i+1}, and let t_0 and t_a be the triangulation segments incident to the first and last vertex of σ. Finally, for a triangulation segment t_i we define $\tau_i = \sigma_i \cup \sigma_{i+1}$ and we define τ_i' to be the union of the two other chains on the boundaries of f_i and f_{i+1} that are incident to t_i. See Figure 5(a) for an illustration of these definitions. The two fat chains in this figure are τ_i and τ_i'. Note that $t_0 = t_a$ in the example of Figure 5. We will construct *separators* for each t_i,

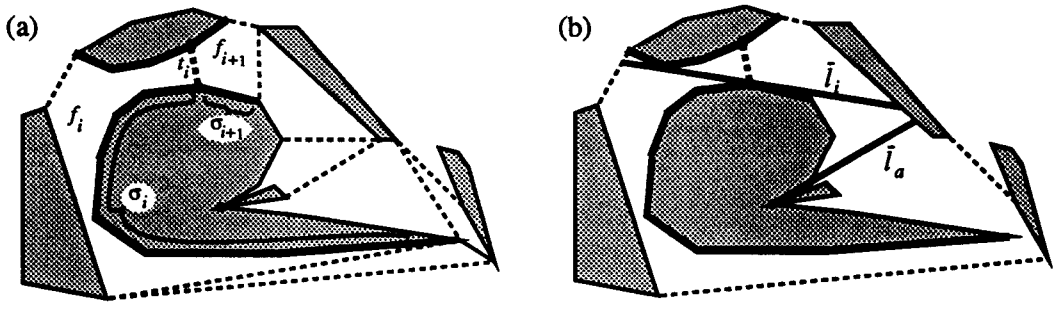

Figure 5: The sleeve of a chain and the two separators for triangulation segments t_i and t_a.

which are segments that lie inside $sleeve(\sigma)$ and separate τ_i from τ_i'. The chain $\tilde{\sigma}$ is constructed from these separators, and some of the triangulation edges between neighbors of σ.

Consider a non-degenerate triangulation segment t_i. Note that, although τ_i and τ_i' are convex chains, it is not always possible to separate them with a line. Therefore we do the following. Let l_i be a line tangent to τ_i at the point where t_i touches τ_i. Let $\overline{l_i}$ be the subsegment of l_i of maximal length inside $sleeve(\sigma)$ that touches τ_i. If $\overline{l_i}$ does not cross τ_i' then $\overline{l_i}$ separates τ_i from τ_i', i.e., $\overline{l_i}$ divides $sleeve(\sigma)$ into two pieces with τ_i on the boundary of one piece and τ_i' on the boundary of the other piece. If $\overline{l_i}$ crosses τ_i', then we rotate l_i such that it remains tangent to τ_i until it becomes tangent to τ_i'. When this happens, $\overline{l_i}$ must separate τ_i from τ_i'. For degenerate triangulation segments t_i (note that only t_0 and t_a are possibly degenerate), we construct $\overline{l_i}$ as follows. Suppose that one of the two polygons involved has a reflex vertex that is incident to t_i; if none of the polygons has a reflex vertex, then we can use the same construction as in the non-degenerate case. Now $\overline{l_i}$ is the maximal extension inside $sleeve(\sigma)$ of the edge of this polygon that is incident to t_i and is on the boundary of $sleeve(\sigma)$. See Figure 5(b). This special construction ensures that we do not introduce a new reflex vertex (one with a greater angle than the old one), which might prevent the separability of the set.

We are now ready to construct $\tilde{\sigma}$. The construction is done with the following incremental algorithm. At the start of the algorithm $\tilde{\sigma}$ consists of $\overline{l_0}$. The remaining separators $\overline{l_1}, \ldots, \overline{l_a}$ are processed in order and meanwhile we update $\tilde{\sigma}$. Suppose

$\overline{l_0}, \ldots, \overline{l_i}$ already have been added. To process $\overline{l_i}$, remove segments from the 'front' of the current chain $\tilde{\sigma}$ as long as they are on the wrong side of $\overline{l_{i+1}}$ and remove the part of the segment that is intersected by $\overline{l_{i+1}}$ that is on the wrong side. (The wrong side of $\overline{l_{i+1}}$ is the side where τ_{i+1} does not lie.) Add $\overline{l_{i+1}}$ to $\tilde{\sigma}$; if $\overline{l_{i+1}}$ intersects a triangulation segment between neighbors of σ (this intersection is necessarily an endpoint of $\overline{l_{i+1}}$) then add this triangulation segment as well.

Lemma 4 *$\tilde{\sigma}$ is one convex chain with $O(a)$ vertices that does not cross σ. $\tilde{\sigma}$ can be constructed in $O(|sleeve(\sigma)| \log |sleeve(\sigma)|)$ time.*

Proof: The $O(a)$ bound on the size of $\tilde{\sigma}$ and the fact that $\tilde{\sigma}$ does not cross σ follow immediately from the construction. (Recall that a is the number of triangulation segments incident to σ.) To see that $\tilde{\sigma}$ consists of convex chains we note that — assuming that the faces f_i are numbered in clockwise order— we always take a clockwise turn at the intersection between two separators $\overline{l_i}$ and $\overline{l_{i+1}}$ or between a separator and a triangulation segment. So the first part of the lemma follows if we can prove that $\tilde{\sigma}$ is *one* chain, which follows if the separator $\overline{l_i}$ that is processed always intersects the currect chain. To this end, consider $\overline{l_{i-1}}$. We will show that $\overline{l_i}$ either intersects $\overline{l_{i-1}}$, or that $\overline{l_i}$ and $\overline{l_{i-1}}$ intersect the same triangulation segment between two neighbors of σ. It should be clear that it then follows that the current chain $\tilde{\sigma}$ must be intersected by $\overline{l_i}$. So let us assume that $\overline{l_i}$ does not intersect $\overline{l_{i-1}}$. Since $\overline{l_{i-1}}$ separates τ_{i-1} from τ'_{i-1}, and $\overline{l_i}$ separates τ_i from τ'_i, this is only possible if they both intersect the triangulation segment opposite σ_i. Thus σ is indeed one convex chain.

To prove the construction time we note that we spend $O(1 + k)$ time when we process a new separator, where k is the number of separators removed from the chain. This adds up to $O(a)$ time in total. It remains to show that the separators can be computed efficiently. To compute a separator we may have to find a line tangent to two convex chains τ_i and τ'_i, which can be done in time $O(\log |\tau_i| + \log |\tau'_i|)$. Next we have to compute $\overline{l_i}$ from l_i. To this end, we preprocess $sleeve(\sigma)$ for efficient ray shooting as in [5]. Thus, after $O(|sleeve(\sigma)| \log |sleeve(\sigma)|)$ preprocessing we can compute $\overline{l_i}$ from l_i in $O(\log |sleeve(\sigma)|)$ time, which leads to $O(|sleeve(\sigma)| \log |sleeve(\sigma)|)$ time in total. □

To construct the embedding $\tilde{S}$, we replace every chain on the boundary of each relative convex hull, as described above. For each triangulation segment t_i, we only construct one separator, which is used in the construction of the replacements of both chains incident to t_i. In other words, if $\overline{l_i}$ is the separator for $\tau_i \in bd(P_i^*)$ and $\tau'_i \in bd(P_j^*)$ that is created in the construction of $\tilde{P}_i$, then $\overline{l_i}$ is also used in the construction of $\tilde{P}_j$.

Theorem 2 *Let S be a set of m simple polygons with n vertices in total, and D be the set of edges shared by the relative convex hulls of the polygons in S. An embedding $\tilde{S} = \{\tilde{P}_1, \ldots, \tilde{P}_m\}$ of S with $\sum_{1 \leq i \leq m} |\tilde{P}_i| - |D| = O(m)$ can be computed in $O(n \log n)$ time.*

Proof: Consider the construction described above. From Theorem 1 and Lemma 4, the construction time and bound on the size of $\tilde{S}$ easily follow. It remains to prove that $\tilde{S}$ is indeed an embedding.

It is trivial to see that each polygon $P_i \in S$ is contained in $\tilde{P}_i$. To complete the proof we must show that the polygons in $\tilde{S}$ do not intersect each other. Consider a polygon $\tilde{P}_i$. Since boundary chains are only replaced by chains inside the corresponding sleeves, $\tilde{P}_i$ can only intersect another polygon $\tilde{P}_j$ somewhere inside a face that belongs to the sleeves of both $\tilde{P}_i$ and $\tilde{P}_j$. But this cannot happen because $\tilde{P}_i$ is separated from $\tilde{P}_j$ by the separator that is constructed for the triangulation segment between $\tilde{P}_i$ and $\tilde{P}_j$. □

We have seen that we can embed a set of arbitrary polygons into another set of polygons with few vertices. But we also want to be able to translate the new set. To prove that this is possible if it is possible for the original set of polygons, we need the following lemma, proved by Toussaint in [25].

Lemma 5 (Toussaint [25]) *A translation order for a set of polygons exists if and only if there exists a translation order for every pair of polygons in the set.*

Lemma 6 *$\tilde{S}$ can be translated into direction d if and only if S can be translated into direction d, and any translation order for $\tilde{S}$ is also valid for S.*

Proof: Since $S \subseteq \tilde{S}$, the only non-trivial part of the lemma is that $\tilde{S}$ can be translated if S can be translated. By Lemma 2, it suffices to show that $\tilde{S}$ can be translated if S^* can be translated. Suppose for a contradiction that S^* can be translated into direction d but $\tilde{S}$ cannot be translated. By Lemma 5, there are two polygons $\tilde{P}_i$ and $\tilde{P}_j$ that cannot be ordered. Assume w.l.o.g. that d is vertically upward. Let l and r be the two vertical lines tangent to $\tilde{P}_j$ and denote the area between l and r and above $\tilde{P}_j$ by $Above(\tilde{P}_j)$ and the area below $\tilde{P}_j$ by $Below(\tilde{P}_j)$ (see Figure 6). If there is no order for $\tilde{P}_i$ and $\tilde{P}_j$ then $Above(\tilde{P}_j) \cap \tilde{P}_i \neq \emptyset$ and $Below(\tilde{P}_j) \cap \tilde{P}_i \neq \emptyset$. 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 $\tilde{P}_i$ above $\tilde{P}_j$ as well as below $\tilde{P}_j$. Consider x, the first intersection of l_a with $\tilde{P}_i$ above $\tilde{P}_j$, and x', the first intersection point below $\tilde{P}_j$. The part of the boundary of $\tilde{P}_i$ connecting x and x' must contain a reflex vertex v such that both edges incident to v lie on one side of the vertical line through v. But any reflex vertex of $\tilde{P}_i$ is a reflex vertex of P_i^* as well, which by Lemma 1(ii) is also a vertex of some other polygon P_k^*. This implies that P_i^* and P_k^* cannot be ordered, contradicting the fact that S^* can be translated. □

4 Translating Polygons

A translation order for a set S of polygons (in direction d) is defined as an order such that no collisions occur if the polygons are moved one at a time (in direction d)

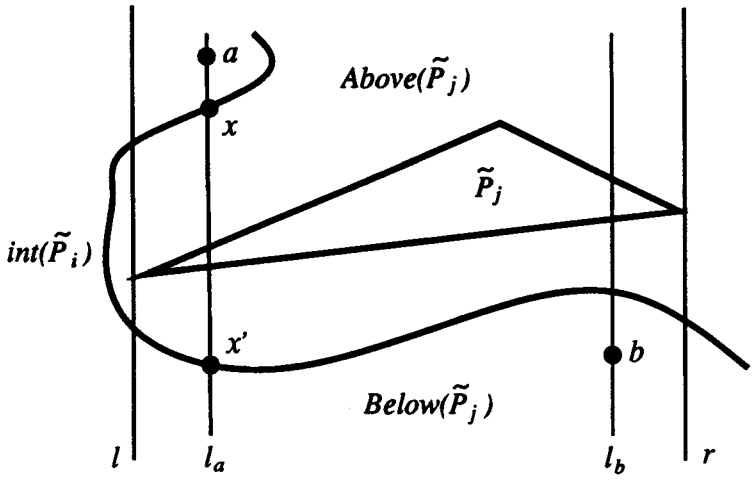

Figure 6: Illustration of the proof of Lemma 6.

to infinity according to this order. The computation of translation orders 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 order exists iff the dominance relation between the polygons is free of cycles. It has been shown by Guibas and Yao [13] that it is not necessary to compute all (possibly $\Omega(m^2)$) 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.)

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 order for any given d, we have to take a different approach. The basic idea is that a triangulation of the area in between the polygons gives us all the information we need to compute a translation order for any given direction. Furthermore, instead of translating the set of polygons itself, we compute an embedding and translate the polygons in the embedding. This reduces the amount of storage and speeds up the queries.

4.1 Translating Convex Polygons

Let $S = \{P_1, \ldots, P_m\}$ be a set of m convex polygons with n vertices in total. The first thing we do is to compute the embedding $\tilde{S}$, according to Lemma 3, and to replace S by $\tilde{S}$. Observe that $\tilde{S}$ consists of convex polygons, so $\tilde{S}$ can still be translated into every direction [13]. Moreover, since the polygons of S are contained in those of $\tilde{S}$, any translation order for $\tilde{S}$ is also a translation order for S.

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

Definition 2 *Let Q and Q' be two polygons. Q is a* d-neighbor *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-neighbor of Q', or Q' is a d-neighbor of Q, or e is parallel to d. See Figure 7 for an illustration of this definition.

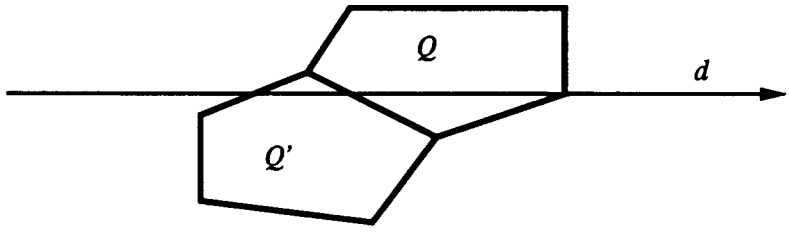

Figure 7: Q is a d-neighbor of Q'.

Lemma 7 *A polygon $Q \in \tilde{S} \cup \mathcal{T}$ (possibly a triangle) can be translated to infinity in direction d without collisions if and only if all its d-neighbors already have been translated without collisions.*

Proof: The "only if"-part is trivial. To prove the "if"-part, suppose that all d-neighbors 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-neighbor 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 7 immediately leads to the following simple scheme. The preprocessing just consists of computing a triangulation $\mathcal{T}$ of $CH(\tilde{S}) - \tilde{S}$ and the dual graph $G(\tilde{S} \cup \mathcal{T})$ of $\tilde{S} \cup \mathcal{T}$. (The nodes in this graph correspond to the polygons in $\tilde{S} \cup \mathcal{T}$ 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 $G(\tilde{S} \cup \mathcal{T})$ into a directed graph G_d. Let a be an arc in $G(\tilde{S} \cup \mathcal{T})$ connecting nodes corresponding

to polygons Q and Q'. If Q is a d-neighbor 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-neighbor of Q then a_d is directed from Q' to Q. Otherwise the edge shared by Q and Q' is parallel to d and a has no corresponding arc in G_d. Thus a node corresponding to some polygon has incoming arcs from all its d-neighbors and outgoing arcs to all polygons for which it is a d-neighbor.

From Lemma 7 and the definition of G_d it easily follows that a topological order of the nodes in G_d corresponds to a translation order in direction d for the polygons in $\tilde{S} \cup \mathcal{T}$ and, consequently, of $S \cup \mathcal{T}$. (Note that the fact that $\tilde{S} \cup \mathcal{T}$ can be translated guarantees that G_d is acyclic.) Clearly, if the triangles of $\mathcal{T}$ are omitted from of this order, we get the desired translation order for S. This leads to:

Theorem 3 *A set S of convex polygons can be preprocessed in $O(n + m \log n)$ time into a data structure of size $O(m)$ such that, given a direction d, a translation order for S in direction d can be computed in time $O(m)$.*

Proof: The embedding $\tilde{S}$ of S can be computed in $O(n + m \log n)$ time, according to Lemma 3. The convex hull of $\tilde{S}$ as well as the triangulation (and its dual graph) can be computed in time $O(m \log m)$ [20, 1]. Note that the total number of edges in $\tilde{S} \cup \mathcal{T}$ (and therefore the number of nodes and arcs in $G(\tilde{S} \cup \mathcal{T})$ as well) is $O(m)$.

Since we can decide in constant time for an arc a in $G(\tilde{S} \cup \mathcal{T})$ 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 be done in linear time (see, e.g., Knuth [14]). □

4.2 Translating Arbitrary Polygons

We will now show how to compute translation orders for a set S of arbitrary polygons. Again we replace S by its embedding $\tilde{S}$, according to Theorem 2. Note that the idea of triangulating the area in between the polygons will not work with an arbitrary set of polygons. The problem is that if there are non-convex polygons, the triangles of the triangulation might prevent the existence of a translation order, i.e., it is possible that a translation order for S exists, but not for $S \cup \mathcal{T}$. Consider for example the case where S consists of one U-shaped polygon P. The triangles of the triangulation of $CH(P) - P$ will prevent a translation order (which trivially exists for P alone) in a horizontal direction. See Figure 8. Fortunately, this problem does not arise for the set $\tilde{S}$. Let $\mathcal{T}$ be a triangulation of $CH(\tilde{S}) - \tilde{S}$, then we have:

Lemma 8 *There exists a translation order in direction d for $\tilde{S}$ if and only if there exists a translation order in direction d for $\tilde{S} \cup \mathcal{T}$.*

Proof: The "if"–part is trivial. The proof of the "only if"–part is similar to the proof of Lemma 6. □

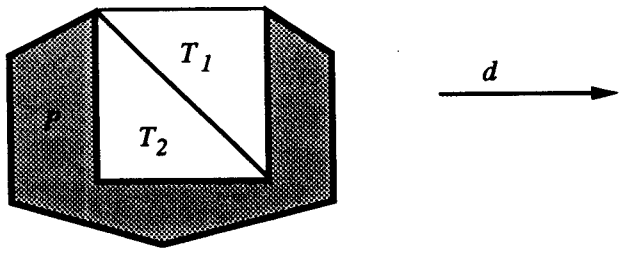

Figure 8: $\{P\}$ can be translated, but $\{P, T_1, T_2\}$ cannot be translated.

We thus arrive at the following scheme for translating a set S of arbitrary polygons. As a preprocessing step, $\widetilde{S}$, and a triangulation $\mathcal{T}$ of $CH(\widetilde{S}) - \widetilde{S}$ together with its dual graph $G(\widetilde{S} \cup \mathcal{T})$ are computed, in $O(n \log n)$ time. Furthermore, we compute for each arc a in G some information that will help us to direct a for a query direction. Recall that a corresponds to two polygons that share an edge or a chain of edges. One of the polygons is a d–neighbor of the other (i.e, it will collide with the other *along the chain* when moved into direction d) for directions d that lie in a certain open interval I_a, and the other polygon is a d–neighbor of the first one in the opposite directions. The two polygons will not collide in the directions corresponding to the endpoints of the intervals, and in all other directions, the polygons are interlocked. We store this interval I_a, which can be computed in linear time in the number of edges of the common chain, with the arc a. Then, given a query direction d, we can construct G_d using the intervals I_a in $O(m)$ time. If we find that two polygons are interlocked, then no translation order exists. Otherwise, we try to sort G_d topologically. A topological order corresponds to a translation order for $\widetilde{S}$ (omitting the triangles of the triangulation) which, by Lemma 6, corresponds to a translation order for $\widetilde{S}$. (Note that Lemma 7 is true for non-convex polygons too.) If G_d cannot be sorted because it contains a cycle, then $\widetilde{S} \cup \mathcal{T}$ cannot be translated. By Lemma's 6 and 8 we can then conclude that no translation order for S exists either.

Lemma 9 *A set S of m arbitrary polygons can be preprocessed in $O(n \log n)$ time into a data structure of size $O(m)$ such that, given a direction d, a translation order for S in direction d can be computed in time $O(m)$, if it exists.*

4.3 Computing All Directions of Separability

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

Toussaint has shown in [26] that there exists a translation order for two polygons in direction d if and only if their relative convex hulls are monotone 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 10 *There exists a translation order in direction d for a set of polygons if and only if the relative convex hulls of the polygons are monotone 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 monotone in direction $d+\frac{1}{2}\pi$ 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 10 and Observation 1, a translation order for S in direction d exists iff $d \notin \bigcup I(S^*)$. In other words, the set D of directions for which a translation order exists is the set $[0 : 2\pi] - \bigcup I(S^*)$. This leads to:

Theorem 4 *All directions for which a translation order 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 order in direction d.

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

Theorem 5 *A set S of m polygons, with a total number of n vertices, can be preprocessed in $O(n \log n)$ time into a data structure of size $O(n)$ such that, given any direction d, it can be decided in time $O(\log n)$ if there exists a translation order for S in direction d. If an order exists it can be computed in $O(m)$ time with a structure that uses $O(m)$ space.*

5 Application to Hidden Surface Removal

One of the most important applications of translation orders is in the performance of hidden surface removal in computer graphics. 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 orders 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 [22]).

We next show how our translation algorithm can be used to generate displaying orders for *perspective* views for terrains consisting of convex faces. Let $F = \{f_1, \ldots, f_m\}$ be a set of convex polygons in 3–space, the faces of the terrain, and let $F' = \{f'_1, \ldots, f'_m\}$ 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 $O(n + m \log n)$ to compute an embedding $\overline{F}$ of F', 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 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 order 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 neighborhood.

Definition 3 *Let Q and Q' be two polygons and $\overline{X}$ be a point in the plane. Q is an* $\overline{X}$-neighbor *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 analog of Lemma 7 is as follows.

Lemma 11 *A face $f \in F$ can be safely displayed if all the faces corresponding to $\overline{X}$-neighbors 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

[1] In this section all projections onto the xy–plane are orthogonal.

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 intersected 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 $\overline{X}$-neighbor of $\overline{f_i}$, in which case f_j is (correctly) displayed first, or there is some $\overline{X}$-neighbor of $\overline{f_i}$ that is intersected by $\overline{r}$. But then this neighbor 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 order. We conclude:

Theorem 6 *A terrain F consisting of m convex polygonal faces with a total number of n vertices can be preprocessed in $O(n + m \log n)$ time into a data structure of size $O(m)$ such that, given a viewpoint X and a projection plane h, a valid displaying order for the faces of F can be determined in $O(m)$ time.*

Remark: If the terrain contains non-convex faces, we can always cut 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 order is valid.)

6 Concluding Remarks

In this paper, we have presented an efficient solution to the translation problem for a set of m polygons in the plane. It was shown that there exists a structure of size $O(m)$ such that a translation order for a query direction can be determined in $O(m)$, if it exists. It is also possible to test in $O(\log n)$ time whether an order exists for the query direction with a structure that uses $O(n)$ space. 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 good candidates for efficient implementations.

One of the open problems concerns translation queries in three dimensions. The recent algorithm of de Berg et al. [7] computes a depth order, but it does not give a

structure for answering queries. See also [4, 11, 17] for results on three-dimensional problems that are related to our work.

A second interesting open problem (in both two and three dimensions) is the following. Suppose that a translation order in a given direction does not exist for some set S of polygons. Then we would like to cut the polygons in S into smaller pieces to achieve translatability. It is unknown how to compute a minimum (or small) number of cuts.

Acknowledgement

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

References

[1] R. Bar-Yehuda and R. Grinwald, Triangulating Polygons with Holes, *Proc. 2nd Canadian Conference on Computational Geometry*, 1990, pp. 112–115.

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

[3] Chazelle, B., Triangulating a Simple Polygon in Linear Time, *Proc. 31st IEEE Symp. on Foundations of Computer Science*, 1990, pp. 220–230.

[4] B. Chazelle, H. Edelsbrunner, L. Guibas, R. Pollack, R. Seidel, M. Sharir and J. Snoeyink, Counting and Cutting Cycles of Lines and Rods in Space, *Proc. 31st IEEE Symp. on Foundations of Computer Science*, 1990, pp. 242–251.

[5] B. Chazelle and L. Guibas, Visibility and Intersection Problems in Plane Geometry, *Discr. & Computational Geometry* **4** (1989) pp. 551–581.

[6] Chin, F., and C.A. Wang, Optimal Algorithms for the Intersection and the Minimum Distance Problems Between Planar Polygons, *IEEE Trans. on Computers* **C-32** (1983), pp. 1203–1207.

[7] de Berg, M., M. Overmars and O. Schwarzkopf, Computing and Verifying Depth Orders, *manuscript*, 1991.

[8] 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.

[9] Edelsbrunner, H., Computing the Extreme Distances between Two Convex Polygons, *J. of Algorithms* **6** (1985), pp. 213–224.

[10] Edelsbrunner, H., and Robson, A.D., Covering Convex Sets with Non-Overlapping Polygons, *Discrete Mathematics* **81** (1990), pp. 153–164.

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

[12] 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.

[13] 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.

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

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

[16] 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.

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

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

[19] 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.

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

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

[22] 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.

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

[24] 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.

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

[26] Toussaint, G.T., On Separating Two Simple Polygons by a Single Translation, *Discr. & Computational Geometry* **4** (1989), pp. 265–278.

[27] Wenger, R., Upper Bounds on Geometric Permutations for Convex Sets, *Discr. & Computational Geometry* **5** (1990), pp. 27–33.

[28] 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.