Translating Convex Polyhedra to Contain a Maximum Number of Points

Petru Pau*
RISC-Linz, Austria
e-mail *ppau@risc.uni-linz.ac.at*
West University of Timişoara, Romania
e-mail *ppau@lalescu.uvt.ro*

Abstract

We present a generalization of a problem of optimal placement for convex shapes: *find a translation of a convex* 3*-dimensional polytope such that it contains a maximum number of points from a given set.* For a polyhedron with m vertices and a set of n points in space, the algorithm finds a placement of maximal containment in $O(mnd^2 \log md)$.

1 Introduction

The problem of finding an optimal placement, in certain sense, of a geometric object in a universe populated with other geometric objects, has received a considerable attention. The field of applications of such problems stretches widely; mainly, the robotics required research in such directions.

The goal of this paper is to give some solutions for the problem:

Problem 1 *(***Translation of Max Containment***) Let P be a polyhedron in the three-dimensional space and S a set of points. Determine a translation τ such that $\tau(P)$ contains the maximum number of points in S.*

Barequet, Dickerson & Pau, in [BDP97], dealt with the planar instance of this problem. They presented an $O(nk \log(mk))$ algorithm based on bucketing, and an $O(nk \log(mn))$ plane-sweep algorithm, where k is the size of the output.

The more difficult case in which the convex polygon can be also rotated was solved by Dickerson & Scharstein in [D-S96].

In this paper, we extend one of the algorithms presented in [BDP97] to the three-dimensional case. This algorithm can be also applied to multidimensional instances.

*The author addresses many thanks to Wadham College, Oxford, whose generosity made possible the preliminary work on this paper.

Our paper is organized as follows: Section 2 contains some notations, definitions and theoretical facts necessary for the subsequent discussion. Section 3 contains the algorithm in details and its analysis.

2 Geometric and Algorithmic Preliminaries

In all what follows, P is a convex polyhedron with m vertices in the three-dimensional space. Unless stated otherwise, its vertices are supposed to be in general position: the polyhedron is simplicial (each of its faces is a triangle). Without loss of generality, we consider the vertex p_1 as the origin (i.e. having coordinates (0,0,0)).

S is a set of n points that are also supposed in general position: no four of them are coplanar; no edge or face of P is parallel with the line determined by any two of them, or with the plane determined by any three of them. Usually we denote these point by $q_1, \ldots, q_n$. With some care, we use the same letters in boldface for denoting the vectors $q_k - p_1$. The vector of source p and target q will be denoted $\vec{pq}$.

The letter τ denotes a translation of arbitrary vector; $\tau_{\mathbf{v}}$ is the translation of vector $\mathbf{v}$.

The polyhedron is viewed as a closed set, that is, $P := \text{interior}(P) \cup \text{boundary}(P)$. The standard notation ∂P is used to represent its boundary. The notion of a point *contained* in P is naturally defined.

It is easy to see that if S is contained in P then the number of translations τ such that $\tau(P)$ still contains S is potentially infinite. Unless S is stabbed with some points in vertices, on edges or on faces of P, the polyhedron can be translated with an uncountable number of vectors with S kept inside it. The next definition (a direct generalization of Definition 1 in [BDP97]) individualizes a finite set of such translations, namely, those which keep P anchored on a certain number of points in S.

Definition 1 (Translation Stable Position) *Let $\tau(P)$ be a translated of P containing a set of points S. We say that $\tau(P)$ is in* translation stable position *(shortly TSP) if at least 3 points in S are on $\partial\tau(P)$.*

In fact, we are interested in the number of points in S that lie on faces of P. If a point in S lies on an edge of P, it will be counted two times, because it is contained in two faces; if it lies in a vertex of P, it will be counted three times.

The next result states the existence of such TSPs.

Theorem 1 (Existence of TSP) *If P contains S, then there exists a translation τ such that $\tau(P)$ contains S and $\tau(P)$ is in TSP.*

Intuitively, a proof of this lemma would consider translated copies of P that always keep S inside: First P is translated so that only one point in S is put on the boundary. Then, anchored in this point, T is translated until a second point is reached. Finally, a more complex translational movement, continuously keeping the previous points on the boundary of P, will make it touch a third point.

We do not provide here a formal proof of this theorem; it is a mere generalization of the planar result given by Chazelle in [Ch83].

Although intuitive and straightforward, this theorem enables the search of very particular translations. Indeed, if instead of S in the previous theorem we said "the maximal subset of S that can be contained in a translated copy of P", the translation whose existence is stated would be a solution of Problem 1. Moreover, this translation is well defined by three points of S that lie on the boundary of P.

With these considerations, it is sufficient to solve the following:

Problem 2 (Anchored Translation of Max Containment) *Find a translation τ of P such that $\tau(P)$ contains a maximum number of points in S and is in TSP.*

From the theoretical point of view, there are two questions that should be answered:

Question 1 (Find a 3-anchored translation) *Given three points in S, how can one find a translation τ such that $\tau(P)$ contains these points on the boundary, if such a translation exists?*

Question 2 (Number of 3-anchored translations) *Given three points in S, how many translations τ are there, so that $\partial(\tau(P))$ contains these points?*

The answer for both these questions can be done with an approach based on intersection. In [BDP97] a lemma concerning the connection between the TSP's of P and the intersections of two translated copies is given. This lemma generalizes naturally to:

Lemma 1 (3-Anchored Positions) *The points q_1, q_2, q_3 are on $\partial\tau_{\mathbf{v}}(P)$ iff $\mathbf{v} = \vec{p_1 v}$ and $v \in \gamma_{1,2,3}$, where $\gamma_{1,2,3} := \partial\tau_{\mathbf{q_1}}(-P) \cap \partial\tau_{\mathbf{q_2}}(-P) \cap \partial\tau_{\mathbf{q_3}}(-P)$, and $-P := \{p_1 - p | p \in P\}$.*

Proof. Since v is in the intersection of $\partial\tau_{\mathbf{q_1}}(-P)$, $\partial\tau_{\mathbf{q_2}}(-P)$ and $\partial\tau_{\mathbf{q_3}}(-P)$, the vectors of sources q_1, q_2, q_3 and target v join p_1 with three points on the boundary of $-P$. In other words, the point v belongs to $\gamma_{1,2,3}$ iff

$$\begin{array}{rcl} p_1 + \vec{q_1 v} \in \partial(-P) & \Leftrightarrow & p_1 + \vec{vq_1} \in \partial P; \\ p_1 + \vec{q_2 v} \in \partial(-P) & \Leftrightarrow & p_1 + \vec{vq_2} \in \partial P; \\ p_1 + \vec{q_3 v} \in \partial(-P) & \Leftrightarrow & p_1 + \vec{vq_3} \in \partial P. \end{array}$$

If we consider now $P_{\mathbf{v}} := \tau_{\mathbf{v}}(P)$, we have:

$$\begin{array}{rcl} \tau_{\mathbf{v}}(p_1) + \vec{vq_1} & = & v + \vec{vq_1} = q_1, \\ \tau_{\mathbf{v}}(p_1) + \vec{vq_2} & = & v + \vec{vq_2} = q_2, \\ \tau_{\mathbf{v}}(p_1) + \vec{vq_3} & = & v + \vec{vq_3} = q_3. \end{array}$$

This means that q_1, q_2, q_3 belong to $\tau_{\mathbf{v}}(P)$. ■

This lemma provides an answer for Question 1. Indeed, if one wants to find a translation τ of P such that three points q_1, q_2, q_3 are on $\partial\tau(P)$, then one will compute the intersection $\partial\tau_{q_1}(-P) \cap \partial\tau_{q_2}(-P) \cap \partial\tau_{q_3}(-P)$. Now, if one translates P with p_1 in any point in the set determined as before, one can be sure that the copy will contact all three initial points.

If we weaken the conditions, i.e., if we consider only two points, q_1 and q_2, the set of all translations τ such that $\partial\tau(P)$ contains q_1 and q_2 can be also computed using a similar argument. The following result deals with this kind of translations; it will be used both for theoretical purposes – for giving an answer to Question 2 – and for devising the algorithm.

Lemma 2 (2-Anchored Positions) *The set of all points v such that $\partial\tau_{\mathbf{v}}(P)$ contains q_1 and q_2, if not empty, is a simple polygonal line in space whose projection on a plane orthogonal to $\vec{q_1 q_2}$ is a convex polygon.*

Proof. If $\tau_{\mathbf{q_1}}(-P)$ and $\tau_{\mathbf{q_2}}(-P)$ have non-empty intersection, the intersection will be a convex polyhedron (we may impose, as a condition for general position of points, that these two translates of P do not intersect along a face, an edge, or a vertex); we will denote it by $P_{q_1q_2}$.

With an argument similar to the one used in Lemma 1, we can easily prove that if P is translated so that p_1 lies on any of the points on $\gamma_{1,2} := \tau_{\mathbf{q_1}}(-P) \cap \tau_{\mathbf{q_2}}(-P)$, it will have q_1 and q_2 on the boundary. It remains to show that these points form a closed polygonal curve in space that projects in a convex polygon on the plane orthogonal to $\vec{q_1 q_2}$.

Let π be a plane orthogonal to $\vec{q_1 q_2}$. The convex hull of $\mathrm{proj}_\pi(P_{q_1q_2})$ is a convex polygon $H_\cap$; due to the convexity of $P_{q_1q_2}$, each edge of $H_\cap$ is in fact the projection of one of its edges. These edges form a closed polygonal line; they split the boundary of $P_{1,2}$ in two pieces, one convex-downward – we denote it C_{up}, and the other convex-upward – denoted C_{down}.

We shall prove that $C_{up} \cap C_{down} = \gamma_{1,2}$.

In the following, we consider, w.l.o.g., that the vector $\vec{q_1 q_2}$ is vertical and points downwards: $\tau_{\mathbf{q_2}}(-P)$ is under $\tau_{\mathbf{q_1}}(-P)$ (see Fig.1). We denote by H_P the convex hull of the projection of any of these polyhedra. The edges of $\tau_{\mathbf{q_1}}(-P)$ that project onto edges in H_P form a closed polygonal line; this line splits its boundary into two pieces, one convex-downward (we call it $P_{1,up}$) and one convex-upward ($P_{1,down}$). Similarly, the boundary of $\tau_{\mathbf{q_2}}(-P)$ is split into two pieces, $P_{2,up}$ and $P_{2,down}$.

Now, $P_{2,up} = \tau_{\vec{q_1 q_2}}(P_{1,up})$ and $P_{2,down} = \tau_{\vec{q_1 q_2}}(P_{1,down})$. The boundary of $P_{q_1q_2}$ contains portions of the boundaries of $\tau_{\mathbf{q_1}}(-P)$ and $\tau_{\mathbf{q_2}}(-P)$; in fact, $C_{up} \subset P_{2,up}$ and $C_{down} \subset P_{1,down}$.

This means that $C_{up} \cap C_{down} = P_{2,up} \cap P_{1,down} = \tau_{q_1}(-P) \cap \tau_{q_2}(-P) = \gamma_{1,2}$; since C_{up} and C_{down} were determined by splitting the surface of $P_{q_1q_2}$ after a chain of edges that project onto $H_\cap$, it follows that $H_\cap = \mathrm{proj}_\pi \gamma_{1,2}$. ■

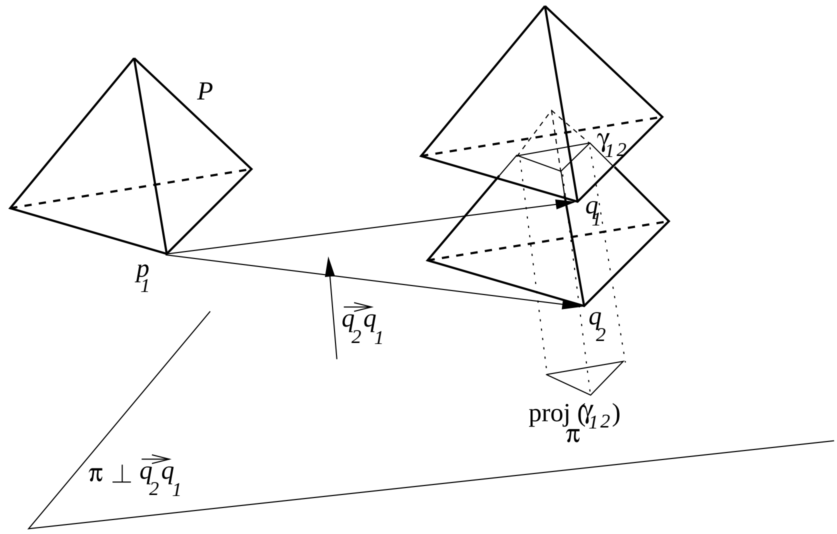

Figure 1: The intersection of the boundaries of two translates of P.

Remarks.

1. The polyhedron $\tau_{\mathbf{q_1}}(-P) \cap \tau_{\mathbf{q_2}}(-P)$ has, as its boundary, subsets from the boundaries of $\tau_{\mathbf{q_1}}(-P)$ and $\tau_{\mathbf{q_2}}(-P)$; due to the convexity of these two, the number of elements of its boundary (faces, edges and vertices) is $O(m)$.

   Since $\gamma_{1,2}$ is a closed chain of edges on the boundary of $\tau_{\mathbf{q_1}}(-P) \cap \tau_{\mathbf{q_2}}(-P)$, the number of its edges is $O(m)$.

2. Theoretically, this lemma establishes a very important fact: *the convexity of the projection of $\gamma_{1,2}$ on the plane π orthogonal on $\mathbf{q_2} - \mathbf{q_1}$.*

   Consider another closed polygonal line in the space – for instance, the chain $\gamma_{1,3}$, the intersection of the boundaries of $\mathbf{q_1}(-P)$ and $\mathbf{q_3}(-P)$. In general, this polygonal line projects on π into a non-convex (and even non-simple) polygon. However, due to the convexity of $\gamma_{1,2}$, the number of points in $\text{proj}_\pi(\gamma_{1,2}) \cap \text{proj}_\pi(\gamma_{1,3})$ is O(m).

   Now, clearly, if $p \in \gamma_{1,2} \cap \gamma_{1,3}$, then $\text{proj}_\pi(p) \in \text{proj}_\pi(\gamma_{1,2}) \cap \text{proj}_\pi(\gamma_{1,3})$. We have thus proven the following lemma:

   Lemma 3 (Number of 3-anchored translations) *The number of translations of P having three given points on the boundary is $O(m)$.*

3. We use Lemma 2 to justify another technique that will be used in the algorithm: *the local linearization of the space of translations.* For two

points in S, q_1 and q_2, let $\gamma^p_{1,2} := \text{proj}_\pi(\gamma_{1,2})$, where π is a plane orthogonal to $\mathrm{q}_2 - \mathrm{q}_1$.

Since $\gamma^p_{1,2}$ is a convex polygon, it can be parameterized linearly with a parameter t in, say, the interval $[0, 1]$. We again invoke the general position of points in S and vertices of P: no edge in $\gamma_{1,2}$ is orthogonal to π. It follows that $\gamma^p_{1,2}$ is a one-to-one and onto image of $\gamma_{1,2}$, therefore we can see the parameterization as a parameterization of $\gamma_{1,2}$.

Now $[0, 1]$ is a linear space that can be swept.

The intersections of $\gamma_{1,2}$ and $\gamma_{1,3}$ correspond to some particular values of t. They also correspond to translations that put P in TSP with respect to q_1, q_2 and q_3: When translating P anchored with p_1 on $\gamma_{1,2}$, at certain points the boundary of P will touch q_3. These points are exactly $\gamma_{1,2} \cap \gamma_{1,3}$ It is obvious that, between two such intersections, q_3 is either inside P or outside it. In fact, during this anchored sweep, after each intersection, q_3 changes its position from the exterior to the interior of P and reversely.

We can then record the intersections of $\gamma_{1,2}$ with $\gamma_{1,i}$, for $i \neq 1, 2$, $i \in \{1, 2, \ldots, m\}$ by storing the cooresponding values of the parameter t. Then, sweeping the domain of t like in [BDP97], we obtain the translation of P of maximal containment, so that q_1 and q_2 lie on the boundary.

The third point from before contains the essence of our algorithm. However, in order to gain some speed-up, we shall present another result, necessary for reducing the search space.

The algorithm will take pairs of points in S and compute the corresponding chains of straight-line segments γ. If the two points in such a pair are too distant, no translated of P will have them on the boundary – in other words, the two translates of P with p_1 in every of the two points will not intersect.

We use a similar argument as in [BDP97] to reduce the number of pairs that will be considered: *bucketing*. The space containing the points in S is split into rectangular boxes (the buckets) shaped to enclose P. Each point in S will be assigned to the bucket that contains it. When considering pairs of points, they are taken from neighboring buckets. Moreover, only buckets that contain at least one point will be stored.

The next lemma (a generalization of Lemma 6 in [BDP97]), enables us to claim that the number of points in a bucket is $O(d)$, where d is the maximum number of points contained in any translated copy of P:

Lemma 4 *Let P be a convex polyhedron. There exists a rectangular box R enclosing P and a rectangular box r enclosed in P so that:*

1. *the elements of r are parallel to the elements of R;*
2. *the ratio $\frac{\text{Vol}(R)}{\text{Vol}(r)}$ is bounded by a constant which does not depend on P.*

When finding these boxes by a brute force approach, the constant that bounds their ratio is quite large; one can certainly find boxes whose volume are in a much lesser ratio, using for instance the algorithm for finding minimal enclosed boxes presented in [O85]; however, for theoretical purposes, this lemma presents the required justification.

We are now in position to give the algorithm and its analysis.

3 The Algorithm

3.1 Informal Description of the Algorithm and Data Structures

We do not discuss any detail related to the storage of the polyhedron P or to the algorithm for computing the intersection of two convex polyhedrons. We also consider here as a black-box the algorithm for determining the position of a point against a convex polyhedron.

We use the following data structures to store the objects that the algorithm deals with:

- the set of points S is stored in an array indexed with the first natural numbers;
- two arrays of points store the vertices of $\gamma_{i,j}$ and $\gamma_{i,k}$, for $i \neq j \neq k \in \{1, \ldots, n\}$;
- a priority queue Q with elements of the form (x, i), where x is a point parameterized by a scalar value t and i is a natural number. The underlying order for the elements of Q is given by the values of the parameter t. The queue may be implemented, for instance, as an AVL tree (see [HS82]);
- a hashing table whose elements represent the *buckets*.

For each point i, we denote by B_i its *neighborhood*; it consists of its own bucket and of the buckets with which its bucket shares a face, an edge or a vertex. It is clear that if a translated of P contains two points on the boundary, these points must lie in neighbor buckets, if not in the same.

For a point p on a chain of edges $\gamma_{i,j}$, parameterized with the value t_0, we will denote the translation of P of vector $\vec{p_1 p}$ by τ_{t_0}. All the intersections of $\gamma_{i,j}$ with some other $\gamma_{i,k}$ will be stored in the priority queue Q according to the value of their parameter.

During the sweep of P anchored in the points q_i and q_j, its boundary will touch another point q_k in S; this position of P corresponds to some element of the form (x, k) stored in Q. When the sweep proceeds further, the point q_k changes its position against P (from outside to inside and vice-versa).

This movement is reflected by the work with the elements of Q. At each moment, the first element in Q corresponds to the first point that will be touched

Preprocessing step - bucketing

Assign the points to the corresponding buckets. Initialize Q.

Iteration

```
max := 0;
FOR each point q_i ∈ S DO
    Q := ∅;
    FOR each point q_j in B_i DO
        compute γ_{i,j}
        c := 0 {c denotes the number of points contained in P}
        IF γ_{i,j} is nonempty
            FOR each point q_k in B_i DO
                Compute γ_{i,k}
                Compute the intersection γ_{i,k} ∩ γi, j
                Insert (x, k) into Q, for each x in the intersection
                IF q_k is contained in τ_0(P)
                    THEN Mark q_k "IN" ; c := c + 1;
                        if c > max then max := c; store translation of vector p_1x
                    ELSE Mark q_k "NOT IN"
                ENDIF
            END FOR
            WHILE Q not empty DO
                delete (x, k) from front of Q
                IF q_k is not "IN"
                    THEN c := c + 1; mark q_k "IN"
                    ELSE c := c − 1; mark q_k "NOT IN"
                ENDIF
                IF c > max
                    THEN max := c; store translation of vector p_1x.
                ENDIF
            END WHILE
        END IF
    END FOR
END FOR
```

Figure 2: Algorithm 1

by P. The change of position of the points q_k when an element (x, k) in Q is processed enables us to keep track of the points that are contained in P at each moment, and thus to store the position of P having a maximum number of points in S inside.

3.2 The Algorithm

The sketch of the algorithm is presented in figure 2.

3.3 Analysis

We will consider in turn each cycle from the algorithm, starting from the innermost.

To begin with, let i, j be fixed. According to Lemma 3, for each point $q_k \in B_i$ there are $O(m)$ intersections between $\gamma_{i,j}$ and $\gamma_{i,k}$; since q_k can be any of the $O(d)$ points in B_i – where d is the maximum number of points in S that can be contained in a translated of P – the queue contains $O(md)$ elements, coming from the intersections of $\gamma_{i,j}$ with $\gamma_{i,k}$ for at most d points q_k. The time for inserting/removing an element from Q is thus $O(\log md)$. This means that line 11 requires $O(m \log md)$ operations.

The time for the computation in line 9 is $O(m)$ (using the algorithm presented in [Ch92]), and the intersection in line 10 is solved, by an algorithm based on the convexity of the projection of $\gamma_{i,j}$, in $O(m \log m)$. The other operations in the inner cycle – lines 8-17 – are done in constant time; all in all, for this cycle, the number of operations is $O(m \log md)$; since it works for all points in the neighborhood of q_i, that are $O(d)$, the whole cycle works in $O(md \log md)$ operations.

The **WHILE** loop in lines 18-27 mainly consists in processing in turn each element in a priority queue of size $O(md)$, thus the time for this cycle is $O(md \log md)$ too.

Now, the number of pairs q_i, q_j that lead the cycles in lines 2-30, is $O(nd)$ – since, again, q_j is taken from the neighborhood of q_i – therefore the total time of the algorithm is $O(nmd^2 \log md)$.

The space used by the algorithm is $O(md + n)$. Indeed, there should be stored: the set S of points ($O(n)$) and the buckets ($O(n)$); the polyhedron P ($O(m)$); the points in $\gamma_{i,j}$ and $\gamma_{i,k}$ ($O(m)$); the elements of Q, whose size could be as large as $O(md)$.

4 Conclusions and comments

We have presented here the proof of the following theorem:

Theorem 2 *Let P be a convex polyhedron and S a set of points in space; the m vertices of P and the n points in S are supposed in general positions. A translation τ with the property that $\tau(P)$ contains a maximum number d of points in S can be find in $O(nmd^2 \log md)$ time, using $O(mk + n)$ memory.*

There are a few points that require a further research:

1. Although the result seems to be suitable for an immediate generalization to higher dimensional instances, there are though theoretical questions that should be answered first. Therefore, we consider the general case interesting enough to justify the attention of the researcher.

2. Another line of generalization would be to determine a rigid motion ρ of P – i.e., a general isometry – such that $\rho(P)$ contains the maximum number of points in S. We believe that this problem is much more difficult than the translational case, mainly because the space to be swept will multi-dimensional: There are two angles and a vector that define an isometry.

3. Even the algorithm presented here might be subject to improvements. For instance, each intersection $\gamma_{i,j}$ is computed many times – in fact, $O(d)$ times: Obviously, it is computed in line 5. Then, if i and j are sufficiently close, it is computed in line 9 for each point in the neighborhood of q_i that is taken into account by the cycle **FOR** in line 4. And the same occurs when the outmost **FOR** cycle reaches q_j.

   It would certainly be desirable to keep some information from one step of the computation to the other, eventually with a minimum of memory consumption.

References

[BDP97] Barequet, G., Dickerson, M. and Pau, P. *Translating a convex polygon to contain a maximum number of points*, Computational Geometry - Theory and Applications, ISSN 09257721, Volume 8, Issue 4, September 1997, pp. 167-179.

[Ch83] B. Chazelle *The Polygon Placement Problem*, Adv. in Comp. Res. vol. 1, JAI Press 1983, pp. 1-34.

[Ch92] B. Chazelle *An optimal algorithm for intersecting three-dimensional convex polyhedra*, SIAM J. Comp. 21(4):671, 1992.

[D-S96] M.Dickerson and D. Scharstein, *Optimal Placement of Convex Polygons to Maximize Point Containment*, Proceedings of the Seventh Annual ACM-SIAM Symposium on Discrete Algorithm, (1996) 114–121.

[HS82] E.Horowitz, S.Sahni *Fundamentals of Data Structures*, Computer Science Press 1982.

[O85] J. O'Rourke *Finding minimal enclosing boxes*, International J. of Computer and Information Sciences, Vol. 14, No. 3, June 1985.

[PS85] F.P. Preparata and M.I. Shamos *Computational Geometry: an Introduction*, Springer-Verlag (1985).