

# Fast Algorithms for Solving the Distance between Two Convex Objects in Two- and Three- Dimensional Spaces

Naoyuki Tokuda and Liang Chen

(Computer Science Department, Utsunomiya University, Utsunomiya, Japan 321-8505)

E-mail:tokuda@cc.utsunomiya-u.ac.jp, lchen@alfin.mine.utsunomiya-u.ac.jp

## Abstract

Consider that there is lots of cases that shape of each objects is known first, but we need fast on-line algorithm to determine some relation among some of the objects, this paper introduce fast *log* type algorithm to computer the collision distance between convex objects in two and three dimensional space under this situation.

## Keywords

Distance, Polygon, Polyhedron, Fast Algorithm

## I. INTRODUCTION

The algorithms for estimation the distances between objects have their obvious application background in the area of image processing, 3-dimansional computer vision, collision-free path planning.

Many researches have been carried out on kinds of distances in papers [1], [2], [3], [4], [5].

It is reasonable to think that the time complexity for computing the distance between two polyhedra should always be larger than a linear function of the number of the vertices, since we need to observe the whole image of the objects at least.

But, we should consider the cases with lots of practical backgrounds in which the related data of each of the objects can be thought to have been stored in the memory of a computer, or for the same objects, we need computer lots of times so that the time we spent for the early observation of each objects can always be neglected.

Under such situation, we have reason to early thoroughly observe each objects first and even with reasonable pre-processing. The background of such application could be: the designing of computer games or computer simulating of air planes, the design for movtion of the robot hands. In all these situation, we need to computer the distance between two objects, usually convex objects or the union of convex objects, and priority know the initial intersection points. In all these cases, the pre-processing is always allowed in order to shorten the on-line computing time of the two objects.

For example, paper [1] gave the method for computing the minimum Euclid distance between two convex polygons, each of which is pre-processed as what we do in section 2.

This paper briefly introduces the methods for computing a distance, we call collision distance, between convex polygons in the plane and the distance between convex polyhedra in dimensional space. The collision distance between two convex objects refers to the length of the path that one of the two objects can be moved, in an orientation parallel to a fixed straight line, until it is intersected with the other. The fixed line, will always be called reference line in short in this paper.

In the later part of this paper, the distance is always referred to such collision distance, if it is not specifically mentioned.

This paper, only consider the distance between polygons in the plane and the distance between polyhedra in the space, we mostly explain the method in pictures. The detailed algorithms can be found [6], [7] or more precisely in [8]. The distance between 2D convex objects in the 3D space or the distance one planar convex object and one 3D convex object is also very interesting which can be found in [8].

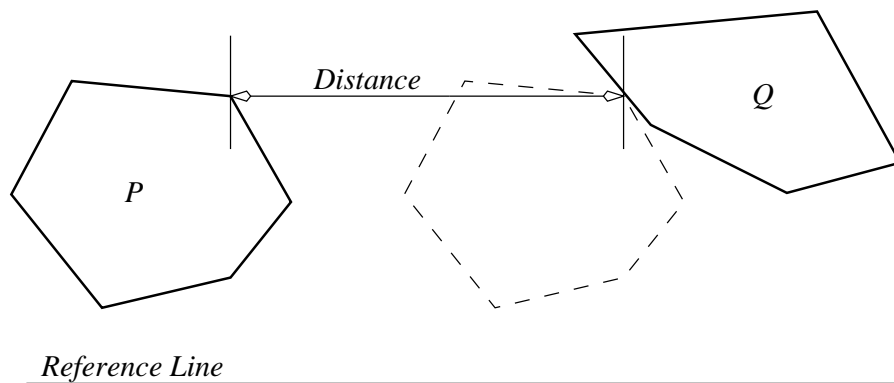


Fig. 1. The Collision Distance

## II. THE DISTANCE IN TWO DIMENSIONAL SPACE

We discuss the problem for solving the distance between two planar convex polygons in two dimensional space.

## A. pre-processing

*Observation 1:* Observe that, given  $n$  points in the plane and *know* that they are the vertex of a convex polygon, the points should decide and only decide *one* convex polygon. But, even though we know that they are the vertex of a polygon, we need at least  $O(n)$  time to decide the sequence of the vertex, e.g. to which two vertices a vertex is adjacent to.

This is obvious from the viewpoint that we need at least check all the vertexes, by noting that if any on the vertexes moves to other position, the sequence of the vertices will be changed.

What we call pre-processing of a convex polygon is as follows: we sort the vertices of the polygon into counter clock sequence, and save them as an array in the memory. From the above observation, we know that it gives extra information than it is need to describe the polygon.

## B. algorithm

The main idea of the algorithm is using binary search method.

### B.1 Movable uni-modal series

*Definition 1:* For a sequential of real numbers  $a_1, a_2, \dots, a_n$ , we call it a movable uni-modal series if and only if there exists two element  $a_i$  and  $a_j$ , such that the series  $a_i, a_{i+1}, \dots, a_{j-1}$  is a strictly decrease series and  $a_j, a_{j+1}, \dots, a_{i-1}$  is a strictly increase series, where  $a_{i+n}$  is denoted as  $a_i$ .

It is easy to know that a subsequent of a moveable uni-modal series is still a movable uni-modal series.

For a movable uni-modal series, it is easy to find the minimum by binary searching. Consider the case  $a_1 > a_n$ , it is easy to know that when  $a_{\lfloor (n+1)/2 \rfloor} > a_1$  (fig.2(1)) or  $a_{\lfloor (n+1)/2 \rfloor} \geq a_{\lfloor (n+1)/2 \rfloor + 1}$  (fig.2(2)), the minimum point should be in the sub-series  $a_{\lfloor (n+1)/2 \rfloor}, a_{\lfloor (n+1)/2 \rfloor + 1}, \dots, a_n$ , otherwise, it should be in the sub-series  $a_1, a_2, \dots, a_{\lfloor (n+1)/2 \rfloor + 1}$  (fig.2(3)). So, it is very easy to construct an algorithm to find the minimum point in a movable uni-

modal series within  $\log$  time cost.

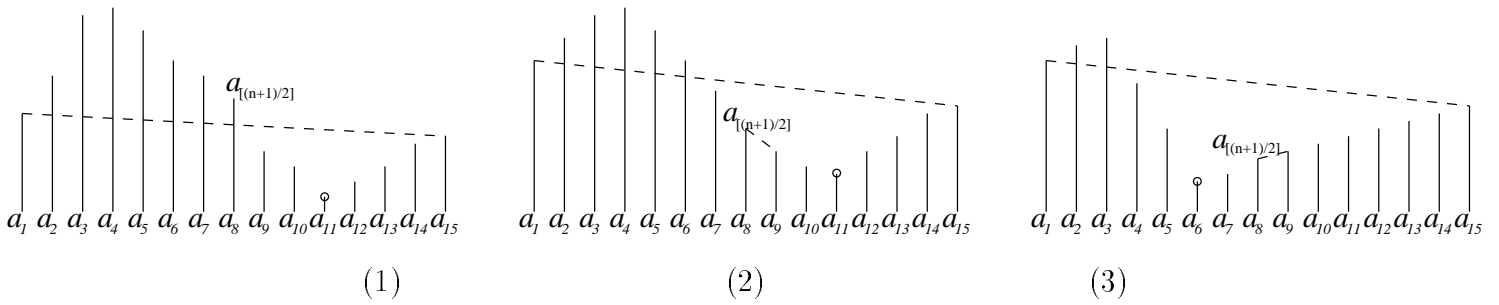


Fig. 2. Movable uni-modal Series

## B.2 The Algorithm

Without losing of generality, we suppose the reference line is parallel to vertical line, and object  $P$  is in the upper side of object  $Q$ , that is, the distance equals to the length of the path that  $P$  can be moved down vertically until it is intersected with  $Q$ .

The whole algorithm is consisting of the following steps:

1. Find the leftist and rightist vertex of  $P$  and  $Q$  (Figure 3).
2. For the lower part of object  $P$  and the upper part of object  $Q$ , find the distance by binary searching.

The method for the first step is quite directly form the observation that the distances from the vertices of a convex polygon to a straight lines construct a movable uni-modal

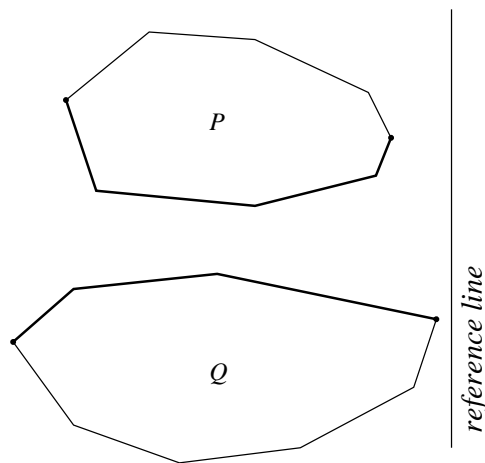


Fig. 3. Computing the Distance between Two Planar Convex Polygons

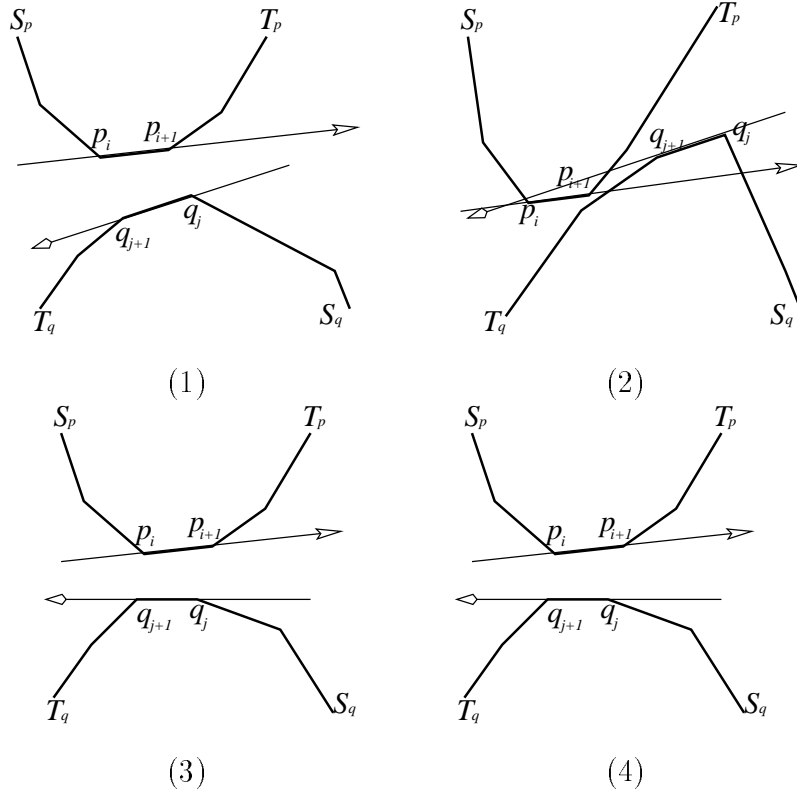


Fig. 4. The Effective Parts for Computing Distance between Polygons

series.

The algorithm for second step comes from the observation of the following fact: Consider the curve  $\widehat{S_p T_p}$  consisting of the edges of the lower part object  $P$ , and the curve  $\widehat{S_q T_q}$  consisting of the edges of the upper part of object  $Q$ . For any edge  $p_i p_{i+1}$  within  $\widehat{S_p T_p}$  and edge  $q_j q_{j+1}$  within  $\widehat{S_q T_q}$ , when the angle (in counter clock) from  $\overrightarrow{p_i p_{i+1}}$  to  $\overrightarrow{q_j q_{j+1}}$  is not large than  $\pi$ , if  $p_i$  is in the left of  $q_{j+1}$  we could throw  $\widehat{S_p p_i}$  away (figure 4.(1) (2)), otherwise throw  $\widehat{q_{j+1} T_q}$  away when we try to find the distance between  $P$  and  $Q$ . When the angle from  $\overrightarrow{p_i p_{i+1}}$  to  $\overrightarrow{q_j q_{j+1}}$  is not less then  $\pi$ , if  $p_{i+1}$  is in the right of  $q_j$  we could throw  $\widehat{p_{i+1} T_p}$  away (fig 2.(3)(4)), otherwise throw  $\widehat{S_q q_j}$  away when we try to find the distance between  $P$  and  $Q$ .

In the cases when we can throw away some parts in the computation, we call the lefted part as effective part.

Thus, by binary searching, we can reduce the distance of  $P$  and  $Q$  into a pair of curves,

one of which has less than 3 vertices. Then still by binary searching method, it is easy to find the distance.

From the view point of complexity, the time cost is  $O(\log(m + n))$  where  $m$  and  $n$  are the number of the vertices of each polygon.

### III. DISTANCE BETWEEN CONVEX POLYHEDRA

For the case of computing the distance between polyhedra, a complicated pre-processing is needed. For each polyhedron, we firstly arbitrary give a plane  $T$  (Note that,  $T$  has nothing to do with the reference lines, since we pre-process each polyhedron separately), then for each of vertices of the polyhedron, draw a plane parallel to  $T$  and find section of the polyhedron and the plane, which is a convex polygon. We call these sections as pre-processed sections, or sections in short, in the later parts of the paper. Save the all the vertices of all the sections, and the sequence of them, as well as the edges that each vertex connected.

By this way, the polyhedron is cut into a series of "drum"s with two parallel faces (when one of the faces is just a point, it can be regarded as the special case) and several waist faces as in figure 5.

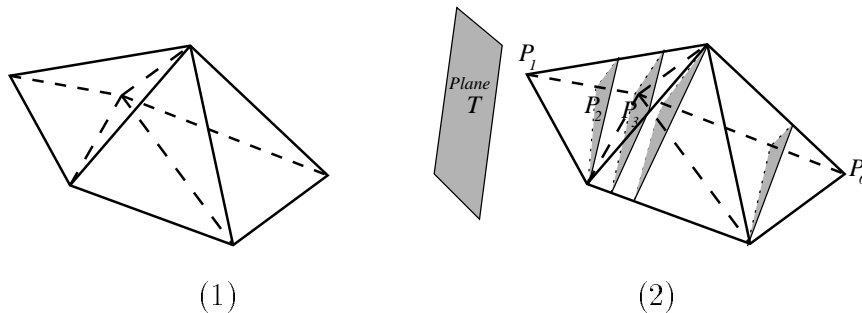


Fig. 5. The Pre-processed Polyhedron

With this pre-processing, firstly we see what it looks like if a polyhedron is "cut" by a plane. This is important, since we will always try to solve sub-problems in two dimensional case which can employ the algorithms of last section.

*Observation 2:* For any plane intersected with a pre-processed polyhedron, it will intersect some of the pre-processed sections. These pre-processed sections which are intersected

with the plane are adjacent (figure 6).

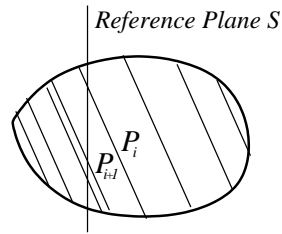


Fig. 6. The Pre-processed Polyhedron

Because of this observation, we can use binary search method to find the first and the last pre-processed sections that the plane intersected. By noting that in the process of the main binary searching, we employ another binary search which intended to determine whether the plane is intersected with a pre-processed section (an convex polygon). Thus, finding the sections intersected with a plane costs  $O(\log^2 N)$  times ( $N$  is the number of vertices of the polyhedron).

*Observation 3:* The intersection of a plane and a polyhedron is a convex polygon. The vertices of the polygon are either the intersection of the plane with adjacent waist edges of "drum"s, or the intersection of the plane with the edges of pre-processed sections (figure 7).

#### A. Strategy

As it is in two dimensional case, we suppose the reference line is parallel to vertical line, and polyhedron  $P$  is in the upper side of polyhedron  $Q$ , that is, the distance equals to the length of the path that  $P$  can be moved down vertically until it is intersected with  $Q$ .

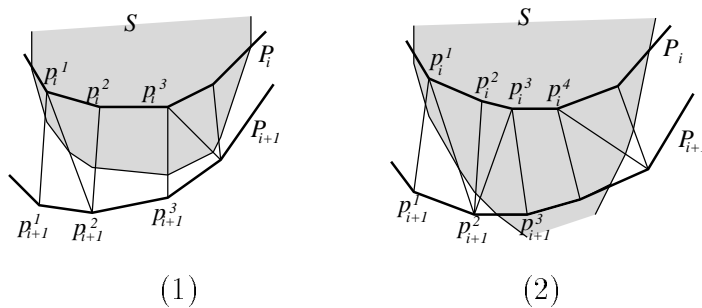


Fig. 7. The Vertices of the Polygon Cut by a Plane

Our main strategy is trying to reduce the search space into a pair of "patch"s of the two polyhedra. To do this, we firstly reduce the effective search space of polyhedron  $P$  into a drum, then reduce it further into a patch with 4 or 3 edges as its boundary. In fact such a patch should be in surface of the waist part of the drum. With the same method, we reduce the effective search space of  $Q$  into such a patch. Then the original problem is transformed into the simple problem of finding the collision distance between such two patches which could be solved just by a kind of enumeration way. What we need to enumerate are, the distances of pairs of vertices, the distances of pairs of edges, or the distances between one patch and one of the vertices of the other patch.

To accomplish the strategy, we need two auxiliary methods. Firstly, we notice that we need a method to determine which side of a pre-processed section in a polyhedron constitutes an effective search space, in the presence of another polyhedron, of course. Only when this method is obtained, we can reduce the effective part into a drum by binary searching. Yet, since the pre-processed section does not always parallel to the reference line, this method is not so direct. For example, consider a pre-processed section  $P_i$  of polyhedron  $P$ . For each edge of  $P_i$  we could have a plane parallel to the reference line, thus we have a three dimensional infinite long column bounded by all these planes. We will try to see if the search space is within the column or not. If the search space is inside the column, than the effective search space of polyhedron  $P$  is the drums below the section  $P_i$ . Otherwise, the effective search space should be in the side of one of the planes going through the edges, say  $T^k$ , which does not include the column. Thus, we could determine which side of  $P_i$  constitutes effective search space by checking which side of  $P_i$  intersects the plane  $T^k$ . We will describe the details of the method in next subsection.

It is worth to notice that, it is true that if we check the bounding planes of the column one by one, we could finally decide if the effective search space is inside the column. Yet, it might takes linear time. Thus, in face, we do not check these bounding planes directly. In stead of this, we arbitrary fix a vertex, say  $a$ , of  $P_i$ , for each other vertex  $b$ , we notice that there could have a plane parallel to the reference line and going through the line  $ab$ . We binary search such planes to determine if the effective search space is inside the column.

To check the search space is inside the column, we should have another auxiliary method

which could be used to decide which side of a plane paralleling to reference line constitutes the effective search space at first. In fact it is also a necessity when we try to reduce further the search space into a patch after we find the effective drum. This method is also described in next subsection.

### *B. Two Auxiliary Methods*

METHOD 1: Determine which side of a plane paralleling to the reference line constitutes the effective search space

*Observation 4:* Suppose that  $a$  is a vertex of polyhedron  $P$ ,  $l_a$  is a line pass through  $a$  and has no intersection with the inner parts of  $P$ ,  $L_a$  is a plane that pass through  $l_a$ , then it is easy to prove that the angles from each of the edges of the polygon to the plane  $L_a$  forms a movable uni-modal series.

From this observation, we know that if we have a supporting line of a polyhedron (supporting line means a line pass through the surface of the polyhedron, but not pass through the inner part of it), we can easily find a supporting plane, that is a plane pass through the surface of the polyhedron but not through inner part, with the support line on it.

*Observation 5:* For any two polygons in a plane, and we know a pair of points each of which are on one of the polygons such that the distance between these two points are exactly the distance between the two polygons, then we can find a pair of parallel supporting lines in the plane such that the polygons are on the separate sides of the parallel lines (see figure 8(1)). The time cost is constant. See figure 8. In the worst case that the minimum distance is found within two vertices of the polygons as in figure 8(2), we only need to check 4 pairs of parallel lines at most to find the supporting lines.

Thus we obtain the method to determine which side of a plane paralleling to the reference line is effective:

*Method 1:* (Determine which side of a plane paralleling to the reference line constitutes the effective search space)

1. Employ the algorithm of section 2 to find the distance between the polygons which are

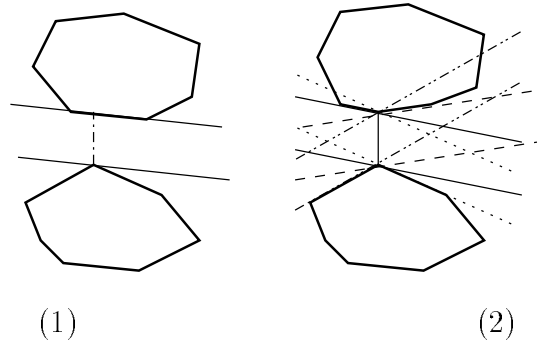


Fig. 8. The Supporting Lines

the intersections of the two convex polyhedra and the plane, according to Observation 3.

2. Find the parallel supporting lines to separate the intersection polygons in the plane, according to Observation 5.
3. Obtain a pair of supporting planes for the pair of polyhedrons, each of which pass through one of the parallel supporting lines obtained in step 2, according Observation 4.
4. Just check which side the supporting planes intersects, we determine which side of the plane  $T$  do not need to be concerned when we try to find the distance of the polyhedra. When the two supporting plane are parallel, the distance obtained.

METHOD 2: Determine which side of a pre-processed section constitutes the effective search space

With Method 1, we can easy construct a method to determine which side of a section constitutes effective search space.

*Method 2* (Determine which side of a pre-processed section constitutes the effective search space)

We explain the polyhedron  $P$  as example.

1. Binary searching of the planes  $T^1, T^2, \dots, T^k$  in figure 9(1), which are parallels to the reference line, and employ the method to determine which side of planes constitutes the effective search space, we can at last reduce the effective search space into the areas illustrated in figure 9(2) or 9(3).

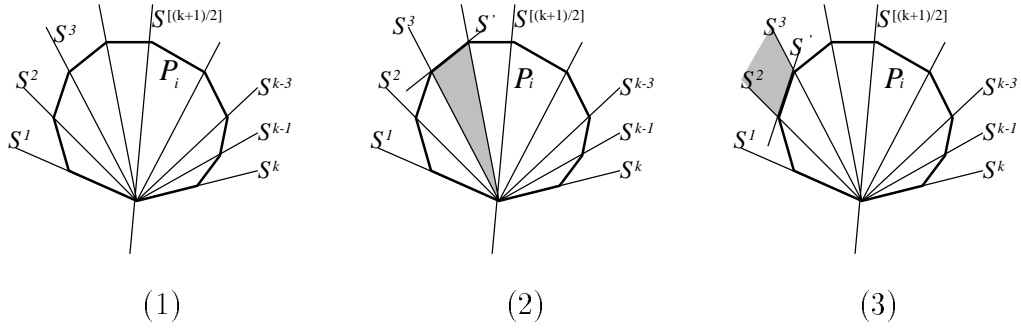


Fig. 9. Search the Effective Parts

2. In case of figure 9(2), it is obvious that the lower side of the polygon constitutes effective search space.
3. In case of figure 9(3), observe that it is impossible that both the upper side of the polyhedron and the lower part of the polyhedron have parts in the effective search side of the plane  $T$ , so if the lower parts of the polyhedron does not go across the plane  $T$ , the effective search space is the upper side of the section (figure 10(3)); otherwise, the effective search space is the lower side (figure 10(1)(2)).

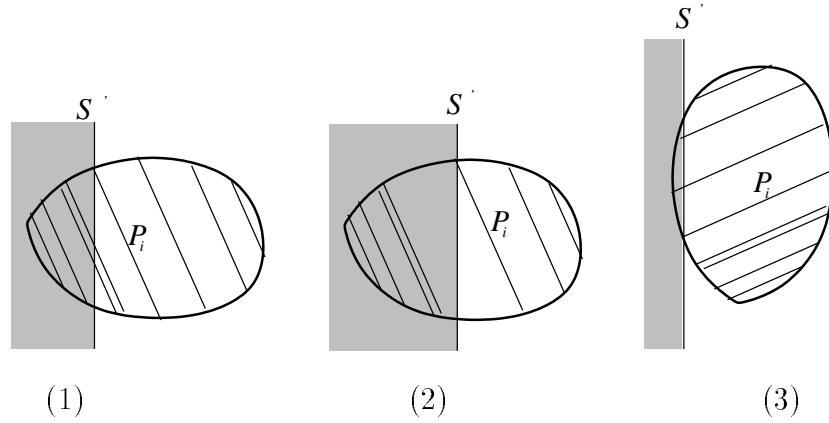


Fig. 10. The Effective Search Side of the Pre-processed Section

### C. Main Algorithms

The main algorithm is consisting of the following steps.

1. With Method 2 (Determine which side of a pre-processed Section constitutes the effective search space), and binary searching, we reduce the effective search space of the polyhedron  $P$  into two adjacent pre-processed sections (figure 11).

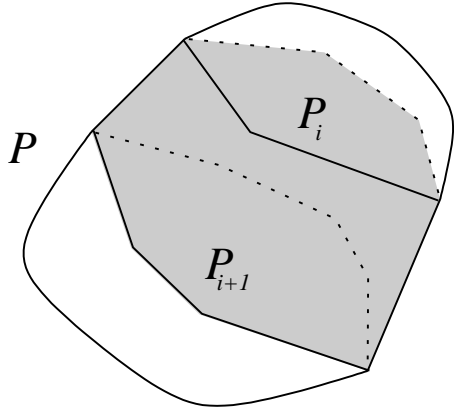


Fig. 11. Two Adjacent Pre-processed sections

2. By using Method 1 (Determine which side of a plane paralleling to the reference line constitutes the effective search space), we can reduce the effective search space of the polyhedron  $P$  within two plane parallel to reference line, as figure 12(1), where the effective area looks like figure 12(2).
3. As what is done in section II, find the curve which is the lower part of  $N_1$  and  $N_2$ , then binary check with method 1 of the planes paralleling the reference line and passing through the segment which are in the lower part of the waist of the drum in figure 12(2), we can reduce the effective search space into a small surface (patch) with 4 or 3 edges as boundary (figure 12(3)).
4. By the similar steps of 1-3, reduce the effective search space of  $Q$  into a small surface (patch) with 4 or 3 edges as boundary.
5. Find the distance between the two small surfaces (patches) directly.

The time complexity is  $O(\log^4(m + n))$ , where  $m$  and  $n$  are the number of the vertices of  $P$  and  $Q$  separately.

#### IV. CONCLUSION

The pre-process strategy is important and effective in the cases that on-line time complexity is important and the shape of each objects can be obtained firstly. By apply this strategy, the methods for computing the collision distances between convex objects is introduced in this paper. For other kind of distances, it is reasonable to believe that fast,

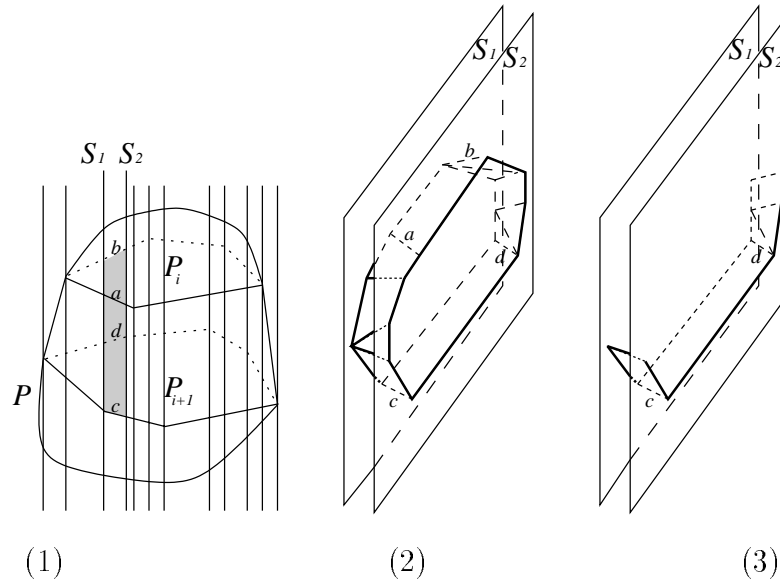


Fig. 12. reduce the Effective Area into A Surface with only 3 or 4 Edges

*log* type algorithms should exist when the objects are appreciatively pre-processed.

#### REFERENCES

- [1] J.T.Schwartz, "Finding the minimum distance Between Two Convex Polygons", *Information Processing Letters*, Vol.13, No.4-5, 1991, pp.168-170.
- [2] M.Shamos, "Problems in Computational Geometry", *Informally distributed lecture notes*, Carnegie-Mellon University, Pittsburgh, PA, 1975.
- [3] P.Widmayer, Y.F.Wu and C.K.Wong, "On Some Distance Problems in Fixed Orientations", *SIAM Journal on Computing*, Vol.16, No.4., 1987, pp.728-746.
- [4] F.Chin and C.A.Wang, "Optimal Algorithms for the Intersection and the Minimum distance Problems Between Planar Polygons", *IEEE Trans. Comput.*, Vol.32, 1993, pp1203-1207.
- [5] H.Edelsbrunner, "On Computing the Extreme Distances Between Two Convex Polygons", *Journal of Algorithms*, Vol.6, 1985, pp213-223.
- [6] L.Chen, E.Song and W.Huang, "Fast Algorithm for Computing the Distance between two Convex Polygons", *Chinese Journal of Computers*, Vol.17, Supplement,1994, pp116-121.
- [7] L.Chen, W.Huang and E.Song, "Fast Algorithm for Computing the Distance between two Convex Polyhedra", *Numerical Mathematics, A Journal of Chinese Universities*, Vol.17, No. 4, 1994, pp.345-359.
- [8] L. Chen, "Fast Algorithms for Computing the Distance between Two Convex Objects in Two- and Three-Dimensional Spaces", *Ph.D thesis*, Software Institute, The Chinese Academy of Sciences, 1994.