

Computing the Minimum Directed Distances Between Convex Polyhedra

CHING-LONG SHIH AND JANE-YU LIU
*Department of Electrical Engineering
National Taiwan Institute of Technology
Taipei, Taiwan 106, R.O.C.*

Given two disjointed objects, the minimum distance (MD) is the short Euclidean distance between them. When the two objects intersect, the MD between them is zero. The minimum directed Euclidean distance (MDED) between two objects is the shortest relative translated Euclidean distance that results in the objects coming just into contact. The MDED is also defined for intersecting objects, and it returns a measure of penetration. Given two disjointed objects, we also define the minimum directed L^∞ distance (MDLD) between them to be the shortest size either object needs to grow proportionally that results in the objects coming into contact. The MDLD is equivalent to the MDED for two intersecting objects. The computation of MDLD and MDED can be recast as a Minkowski sum of two objects and finished in one routine. The algorithms developed here can be used for collision detection, computation of the distance between two polyhedra in three-dimensional space, and robotics path-planning problems.

Keywords: minimum distance, minimum directed distance, Minkowski sum, collision detection, path planning

1. INTRODUCTION

Determining the minimum distance between two convex polyhedra is an important problem in robotics, image processing, CAD systems, computational geometry, and other areas of information processing which deal with geometrical data. All work that is developed for automated path planning requires at its lowest level the ability to detect whether or not collision has occurred. The ability to compute distance and interference efficiently will result in a substantial reduction in the overall time required for most path-planning algorithms. In general, a path planning algorithm needs to ascertain for any position in the workspace not only if a collision has occurred, but also how close it is to occurring if it has not. For instance, a "generate-and-test" type path-planning algorithm requires tests for whether configuration and trajectory are collision-free, and methods for searching intermediate subgoals [2].

Several techniques have been reported for calculation of the minimum distance between convex polyhedra. In particular, Gilbert, et al. [5] defined an object by means of the convex hull of its vertices and provided a quick procedure to compute the distance between 3D objects. Their approach performs an iterative sequence of distance minimizations to obtain elementary subsets of the original shape until a subset containing the global minimum is attained. Bobrow [1] proposed an approach which casts the problem as a constrained

Received January 4, 1997; accepted September 5, 1997.
Communicated by Youn-Long Lin.

nonlinear minimization. His algorithm uses a direct approach to minimize the nonlinear distance function, which generates a sequence of search directions along the surfaces of the objects to obtain the global minimum. The computation time of this algorithm is roughly linear with the number of faces. Lin and Canny [6] proposed an incremental distance calculation starting with a candidate pair of features, one from each polyhedron, that checks the closest points that lie in these features. The algorithm then steps to the next closest pair of features until the closest points are found. The minimum distance between three-dimensional segments was developed in [8]. The computational efficiency of two-dimensional algorithms for polygons was given in [4,10]. The distance between boxes was considered in [9]. The above minimum distance algorithms are asymptotically fast, but they only return a zero value for intersecting objects.

Distance is used as a measure of how far a robot part is from colliding with an obstacle. When the two objects intersect, the distance between them is zero. This gives no information about the intensity of the intersection. The general objective of a penetration measure is to quantify the depth of intersection for object modeling. A minimum directed Euclidean (or translated) distance was proposed in [2,3] to define the intensity of penetration and uses the negative of the minimum Euclidean distance by which the two overlapping objects must be relatively translated so as to have no interior point in common. The minimum directed Euclidean distance is equivalent to the distance between the two objects if the objects are disjointed. Buckley [2] proposed a method to compute the minimum directed Euclidean distance between two-dimensional convex polygons and used this measure in a penalty approach in collision-avoidance robot motion planning. The result shows that the nonintersection constraint provides useful information in the case of body intersection, and that path planning can be done according to the flexible trajectory paradigm. Cameron & Culley [3] applied the Minkowski sum technique to compute the minimum translated distance between two convex polyhedra in a three-dimensional space. However, the complexity and completeness of their algorithm was not analyzed.

As one step toward reducing the number of computations needed for path planning, the aim of this research is to present efficient algorithms to compute the minimum directed distance functions between two convex polyhedra. The polyhedral representation of 3D objects is widely used in robotics and computer graphics research. We define a new minimum directed L^∞ distance (MDLD) between convex polyhedra and derive its relationship with the minimum directed Euclidean distance (MDED) proposed in [2,3]. In its simple form, the measure of the L^∞ distance between a point and a convex polyhedron is the maximum of the distances of the point from the half-space which passes through the faces of the polyhedron. Our computation of the MDLD and MDED functions and the Minkowski sum of two convex polyhedra are based on the boundary model of the polyhedron.

The following is an outline of the paper and its contents. In Section 2, the minimum directed L^∞ distance is defined, and its properties as well as its connection with the minimum directed Euclidean distance are also derived. Efficient algorithms for computing both MDLD and MDED are proposed in Section 3. Several examples and applications are illustrated in Section 4. Finally, Section 5 provides conclusions with regard to the proposed approach.

2. PROBLEM FORMULATION AND PRELIMINARIES

We will assume that two objects P and A are convex polyhedra in R^r space for further discussion. When $r = 2$, objects P and A are two-dimensional convex polygons, and when $r = 3$, objects P and A are three-dimensional convex polyhedra. Some terminology with regard to polyhedral objects used in this paper are introduced here. A convex polyhedron in three-dimensional space is characterized by its faces, edges, and vertices. A face of a three-dimensional polyhedron is a 3D convex polygon. A plane in 3D indicates a set which satisfies the plane equation, $\mathbf{n}^T\mathbf{x} + \mathbf{d} = 0$. A line means a straight line of infinite length passing through two points \mathbf{a} and \mathbf{b} , and line segment \mathbf{ab} indicates a segment of a line connecting two end points \mathbf{a} and \mathbf{b} . An edge is a line segment connecting two vertices. For two-dimensional polygons, the terms face and edge are usually interchangeable. The closest pair of features between two objects is defined as the pair of features which contain the pair of closest points between objects. The pair of closest points means the minimum Euclidean distance points between two objects.

Let \mathbf{p} ($\mathbf{p} \in R^r$) be the translational vector of the reference point of polyhedron P . A convex polyhedron P with m faces can be represented by the set

$$P = \{\mathbf{x} \in R^r \mid \mathbf{N}^T(\mathbf{x} - \mathbf{p}) + \mathbf{d} \leq 0\},$$

where

$$\mathbf{N} = [\mathbf{n}_1, \dots, \mathbf{n}_m] \in R^{r \times m}, \text{ and } \mathbf{d} = [d_1, \dots, d_m]^T \in R^m.$$

Matrix \mathbf{N} is an r by m matrix whose i th column \mathbf{n}_i is the unit outward normal vector of the i th face of polyhedron P . Vector \mathbf{d} is an m dimensional vector, and d_i is the magnitude of the vector from reference point \mathbf{p} perpendicular to the i th face, measured in the negative \mathbf{n}_i direction. Let the vertices of P be represented by

$$\mathbf{U} = [\mathbf{u}_1, \dots, \mathbf{u}_k] \in R^{r \times k},$$

where k is the number of vertices, and $\mathbf{u}_1, \dots, \mathbf{u}_k$ are the vertex translational vectors relative to the reference point of P . A grown convex polyhedron \bar{P} of convex polyhedron P is defined as

$$\bar{P} = \{\mathbf{x} \in R^r \mid \mathbf{N}^T(\mathbf{x} - \mathbf{p}) + \mathbf{d} \leq \mathbf{s}, \mathbf{s} \in R^m \text{ and } s \geq 0\}.$$

and a proportionally grown polyhedron P_ϵ ($\epsilon \in R, \epsilon \geq 0$) of convex polyhedron P is defined by

$$P_\epsilon = \{\mathbf{x} \in R^r \mid \mathbf{N}^T(\mathbf{x} - \mathbf{p}) + \mathbf{d} \leq [1, \dots, 1]^T \epsilon\}.$$

Similarly, a convex polyhedron A with n faces and l vertices is represented by the set

$$A = \{\mathbf{x} \in R^r \mid \mathbf{A}^T(\mathbf{x} - \mathbf{q}) + \mathbf{b} \leq 0\},$$

where

$$A = [a_1, \dots, a_n] \in R^{r \times n}, \quad b = [b_1, \dots, b_n]^T \in R^n.$$

Moreover, q ($q \in R^r$) is the translational vector of the reference point of A , and its vertices are represented by

$$V = [v_1, \dots, v_l] \in R^{r \times l},$$

where v_1, \dots, v_l are the vertex translational vectors of A relative to its reference point.

The following definitions and notations are used in the paper and are defined below.

Definition 1: The maximum value of vector y , $max(y)$, is given by

$$max(y) = max\{y_1, \dots, y_n\}, y \in R^n.$$

Definition 2: The integer function $arg\{max(y)\}$ is the index (or argument) i such that $y_i = max(y)$.

Definition 3: The minimum value of vector y , $min(y)$, is given by

$$min(y) = min\{y_1, \dots, y_n\}, y \in R^n.$$

Definition 4: The integer function $arg\{min(y)\}$ is the index (or argument) i such that $y_i = min(y)$.

$-P$: polygon whose vertex vectors are the negations of the vertex vectors of P when the reference point of P is at the origin.

P_A : Minkowski sum of P and $-A$, and it is the set of translations of the reference point q of Q that bring it into interference with P and has its new reference point at point p , i.e. $P_A = \{q : P \cap A \neq \emptyset\}$.

$d(x, P)$: minimum distance (MD) between point object x and object P , and

$$d(x, P) = \min_{y \in P} \|x - y\|_2$$

$d_\infty(x, P)$: minimum directed L^∞ distance (MDLD) between a point object x and an object P , and

$$d_\infty(x, P) = \max(N^T(x - p) + d) = -\min(-N^T(x - p) - d).$$

A contour of $d_\infty(x, P) = \epsilon$ gives all the boundary faces of P_ϵ for ϵ greater than zero. When $x \notin P$, MDLD $d_\infty(x, P) (= \max(N^T(x - p) + d) > 0)$ is the shortest size that needs to grow proportionally, resulting in point x being on the boundary of P_ϵ , and that resembles an L^∞ -norm distance. When $x \in P$, $d_\infty(x, P) (= -\max(-N^T(x - p) - d) \leq 0)$ is the negative of the shortest distance that x needs to translate, resulting in point x just touching the boundary of P . Note that $d_\infty(x, P)|_{x=q} = \max(N^T(q - p) + d) = \max(-N^T(p - q) + d) = d_\infty(x, P^c)|_{x=p}$, where

$$P^c = \{x \in R^r \mid -N^T(x - q) + d \leq 0\}$$

is the Minkowski sum of point object q and object P .

$d_2(x, P)$: minimum directed Euclidean distance (MDED) between point x and object P ,

and
$$d_2(x, P) = \begin{cases} d(x, P) & x \notin P \\ d_\infty(x, P) & x \in P \end{cases}$$

Examples of MD, MDLD and MDED between point object x and convex polygon P in two-dimensional space are shown in Fig. 1. MDLD $d_\infty(x, P)$ returns the orthogonal distance from point x to the half-space determined by $\arg\{\max(N^T(x - p) + d)\}$, and $d_2(x, P)$ returns the minimum distance from point x to the corresponding closest face. Assume that the orthogonal projection of point x to the closest plane is denoted by x^* . If point x^* is on the nearest face, then $d_2(x, P) = d_\infty(x, P)$; otherwise, $d_2(x, P) > d_\infty(x, P)$. Thus, in general, we have the relationship $d_2(x, P) \geq d_\infty(x, P)$. In other words, $d_\infty(x, P)$ determines only the distance between the closest pair of features of a point and a face. However, $d_2(x, P)$ determines the distance between the closest pair of features in all cases, including a pair of a point and an edge, and a pair of a point and a vertex.

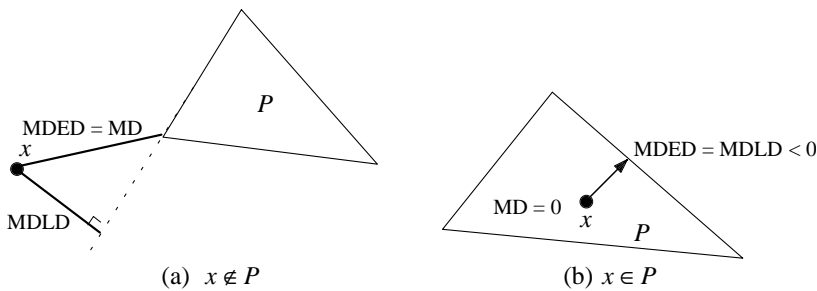


Fig. 1. Geometrical illustration of MD, MDLD, and MDED between a point object x and a convex polygon P in two-dimensional space; (a) point x is outside of P , and (b) point x is inside of P .

We can now define the minimum directed distances (MDD) between two convex polyhedra A and P . In concept, the distance problem for two objects A and P can be reduced to the problem of finding the distance from a point object and the Minkowski sum P_A .

$d(A, P)$: minimum distance (MD) between convex polyhedra A and P , and

$$d(A, P) = \min_{x \in A, y \in P} \|x - y\|_2 = d(x, P_A)|_{x=q}.$$

$d_\infty(A, P)$: The minimum directed L^∞ distance (MDLD) between convex polyhedra A and P , $d_\infty(A, P)$, is given by $d_\infty(A, P) = d_\infty(x, P_A)|_{x=q}$.

Note that

$$d_\infty(A, P) = d_\infty(x, P_A)|_{x=q} = d_\infty(x, A_p)|_{x=p} = d_\infty(P, A), \tag{1}$$

where A_p is the Minkowski sum of A and $-P$. Thus, $d_\infty(A, P)$ is a symmetric distance function. Fig. 2 shows an example of Eq. (1) in 2D. When A and P are disjoint, $d_\infty(A, P)$ is the shortest size either object needs to grow proportionally that results in the objects being in contact and, hence, returns a measure of distance between A and P . In the two-dimensional case, $d_\infty(A, P)$ can be geometrically interpreted as

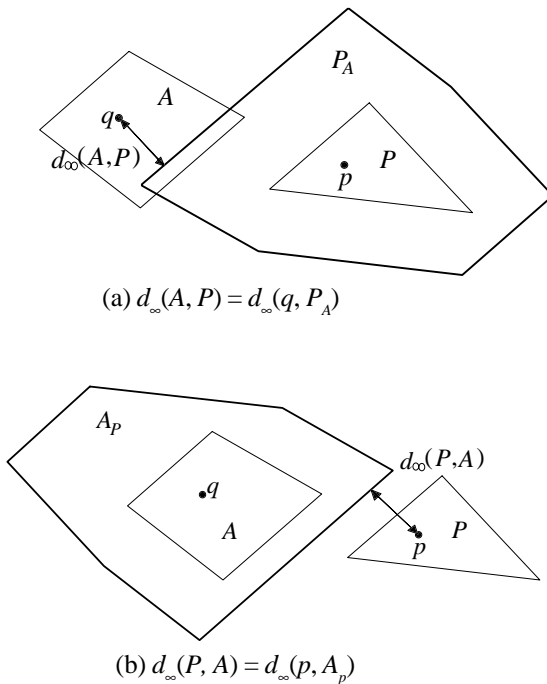


Fig. 2. Illustration of $d_\infty(A, P) = d_\infty(x, P_A)|_{x=q} = d_\infty(x, A_p)|_{x=p} = d_\infty(P, A)$, in 2D: (a) $d_\infty(A, P) = d_\infty(q, P_A)$, and (b) $d_\infty(P, A) = d_\infty(p, A_p)$.

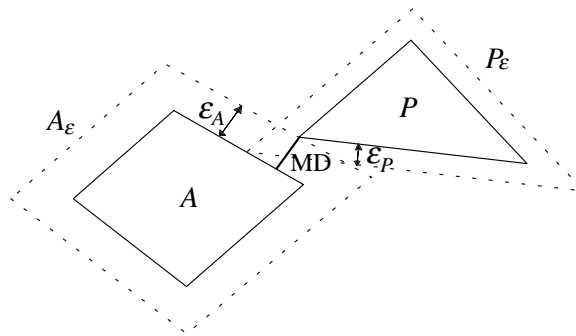
$$d_\infty(A, P) = \max\{\epsilon_A, \epsilon_P\}, \text{ for } A \cap P = \emptyset, \tag{2}$$

where ϵ_A is the smallest value of ϵ such that A_ϵ and P intersect, and ϵ_P is the smallest value of ϵ such that A and P_ϵ intersect. If the objects overlap, $d_\infty(A, P)$ is the negative of the shortest distance needed to translate one object with respect to the other until there is no intersection. Fig. 3 shows the geometrical interpretation of MDLD in 2D.

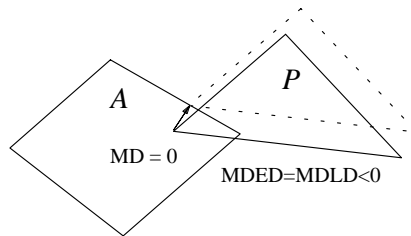
$d_2(A, P)$: minimum directed Euclidean distance (MDED) between convex polyhedra A and P , and

$$d_2(A, P) = \begin{cases} d(A, P) & \text{if } A \cap P = \emptyset \\ d_\infty(A, P) & \text{if } A \cap P \neq \emptyset \end{cases}$$

Examples of MDED are also shown in Fig. 3. One computational difficulty in the above formulation is that the Minkowski sum generally has a very complicated shape in higher dimensions. Nevertheless, not every face of the set P_A needs be considered to find the minimum distance. This is the focus of the following section.



(a) P and A are disjointed



(b) P and A are overlapping

Fig. 3. Geometrical interpretation of MDLD, MD, and MDED between two convex polygons A and P in 2D; (a) disjointed case, and (b) overlapping case.

3. MDD ALGORITHMS

In this section, we shall develop efficient algorithms for computing the minimum directed distance functions for two convex objects. We shall first develop the boundary-model approach to represent the Minkowski sum P_A , which is most suitable for computing the proposed minimum directed distances between convex polyhedra.

If A and P are convex polyhedra, then set P_A is also a convex polyhedron, with points on its boundary representing the configuration with which objects A and P come into contact. The faces of P_A correspond to modes of contact between the objects such that they can slide along each other, and these modes can be classified as (1) face-vertex (FV) contact, (2) vertex-face (VF) contact, and (3) edge-edge (EE) contact. This set of three types of interaction cover all possible types of contact between polyhedra. Other forms of contact are special cases. For instance, an edge-face contact requires at least two contacts from among the above three contact types.

For FV contact cases, a supporting face of P_A is obtained when a face of P slides against a vertex of A . Let

$$\mathbf{n}_i^T(\mathbf{x} - \mathbf{p}) + d_i \leq 0 \tag{3}$$

be the half-space corresponding to a face of P ; then, the vertex of A with a maximum value of $(-\mathbf{n}_i^T \mathbf{v}_j)$ for $j = 1, \dots, l$, can slide against P (see Fig. 4(a)), where l is the number of vertices of A . Thus, the set P_A should satisfy

$$\mathbf{n}_i^T(\mathbf{x} - \mathbf{p}) + d_i \leq e_i, \quad i = 0, 1, \dots, m-1, \tag{4}$$

where

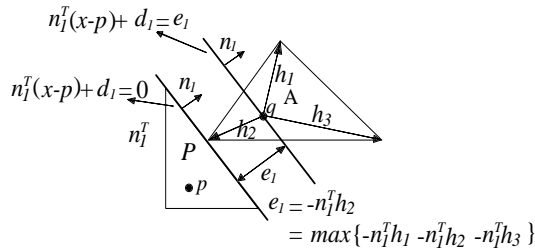
$$e_i = \max\{-\mathbf{n}_i^T \mathbf{v}_j, j = 1, \dots, l\} = \min(\mathbf{V}^T \mathbf{n}_i),$$

and m is the number of faces of P . For each i th face of P , let $j = \arg(\min(\mathbf{V}^T \mathbf{n}_i))$; the i th face of P and j th vertex of A are called a face-vertex (FV) contact pair, and the corresponding minimum distance δ between them is

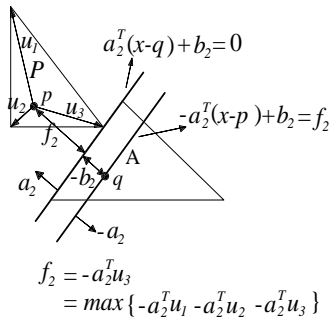
$$\delta = \mathbf{n}_i^T(\mathbf{q} - \mathbf{p}) + d_i + \min(\mathbf{V}^T \mathbf{n}_i) = \mathbf{n}_i^T(\mathbf{q} - \mathbf{p}) + d_i - e_i. \tag{5}$$

The contact pair which has a maximum value of δ in Eq. (5) among all possible face-vertex contact pairs is called a face-vertex closed pair.

For VF contact cases, a supporting face of P_A is obtained when a face of A slides against a vertex of P (see Fig. 4(b)). Let



(a) FV contact



(b) VF contact

Fig. 4. Illustration of the supporting faces of Minkowski sum P_A generated by (a) face-vertex contact, and (b) vertex-face contact.

$$\mathbf{a}_i^T(\mathbf{x} - \mathbf{q}) + b_i \leq 0 \tag{6}$$

correspond to a face of A ; then, the vertex of P with a maximum value of $(-\mathbf{a}_i^T \mathbf{u}_j)$ for $j = 1, \dots, k$, can slide against A , where k is the number of vertices of P . Thus, P_A should also satisfy

$$-\mathbf{a}_i^T(\mathbf{x} - \mathbf{p}) \leq f_i - b_i, \quad i = 0, 1, \dots, n-1, \tag{7}$$

where

$$f_i = \max\{-\mathbf{a}_i^T \mathbf{u}_j, j = 1, \dots, k\} = \min(\mathbf{U}^T \mathbf{a}_i),$$

and n is the number of faces of A . For each i th face of A , let $j = \arg(\min(\mathbf{U}^T \mathbf{a}_i))$; the i th face of A and j th vertex of P are called a vertex-face (VF) contact pair, and the corresponding minimum distance δ between them is

$$\delta = \mathbf{a}_i^T(\mathbf{p} - \mathbf{q}) + b_i + \min(\mathbf{U}^T \mathbf{a}_i) = \mathbf{a}_i^T(\mathbf{p} - \mathbf{q}) + b_i - f_i. \tag{8}$$

The contact pair which has a maximum value of δ in Eq. (8) among all possible vertex-face contact pairs is called a face-vertex closed pair. The three-dimensional case is exactly analogous for the FV and VF contacts in the two-dimensional case.

The EE contact mode occurs only in the three-dimensional case. In EE contact cases, not every pair of edges will generate a supporting face of P_A . Let $\mathbf{u}_\alpha \mathbf{u}_\beta$ be an edge of P , and let $\mathbf{v}_\gamma \mathbf{v}_\lambda$ be an edge of A ; then, edges $\mathbf{u}_\alpha \mathbf{u}_\beta$ and $\mathbf{v}_\gamma \mathbf{v}_\lambda$ can come into contact if there exists a normal vector \mathbf{n} (see Fig. 5)

$$\mathbf{n} = \frac{\mathbf{u}_\alpha \mathbf{u}_\beta \times \mathbf{u}_\gamma \mathbf{u}_\lambda}{\|\mathbf{u}_\alpha \mathbf{u}_\beta \times \mathbf{u}_\gamma \mathbf{u}_\lambda\|_2} \tag{9}$$

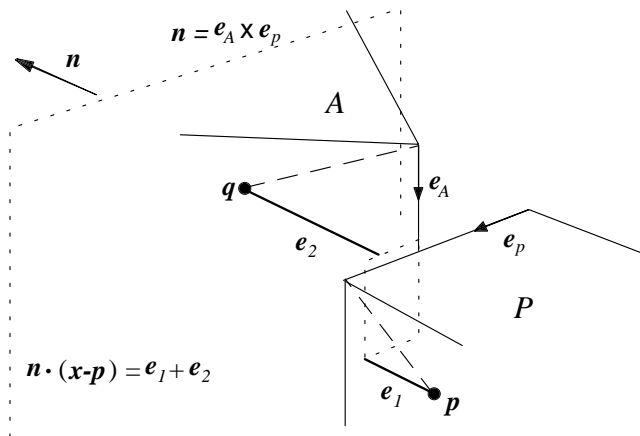


Fig. 5. Illustration of the supporting faces of the Minkowski sum P_A generated by edge-edge contact in 3D.

such that

$$e_1 = \mathbf{n}^T \mathbf{u}_\alpha = \mathbf{n}^T \mathbf{u}_\beta = \max\{\mathbf{n}^T \mathbf{u}_j, j = 1, \dots, k\} \geq 0$$

and

$$e_2 = -\mathbf{n}^T \mathbf{v}_\gamma = -\mathbf{n}^T \mathbf{v}_\lambda = \max\{-\mathbf{n}^T \mathbf{v}_i, i = 1, \dots, l\} \geq 0.$$

In addition, the supporting face generated by the EE contact is the half-space represented by

$$\mathbf{n}^T(\mathbf{x} - \mathbf{p}) \leq e_1 + e_2. \quad (10)$$

These two edges become an edge-edge contact pair, and the minimum distance δ between the two lines $\mathbf{u}_\alpha \mathbf{u}_\beta$ and $\mathbf{v}_\gamma \mathbf{v}_\lambda$ is

$$\delta = \mathbf{n}^T(\mathbf{q} - \mathbf{p}) - e_1 - e_2. \quad (11)$$

The edge-edge pair which has a maximum value of δ among all possible edge-edge contact pairs is called an edge-edge closed pair (named edge i of P and edge j of A).

For two-dimensional convex polygonal objects, we only need to consider the FV contact for all faces of P and the VF contact for all faces of A ; thus, an analytical form of the set P_A can be obtained. For cases of FV contact, we obtain the grown object \bar{P} :

$$\bar{P} = \{\mathbf{x} \in R^r \mid \mathbf{N}^T(\mathbf{x} - \mathbf{p}) + \mathbf{d} - \mathbf{e} \leq 0\},$$

where

$$e_i = \max\{-\mathbf{n}_i^T \mathbf{v}_j, j = 1, \dots, l\} = \max(-\mathbf{V}^T \mathbf{n}_i), \quad \mathbf{e} \in R^m.$$

For the VF contact case, we obtain the grown object \bar{A}^c :

$$\bar{A}^c = \{\mathbf{x} \in R^r \mid -\mathbf{A}^T(\mathbf{x} - \mathbf{p}) + \mathbf{b} - \mathbf{f} \leq 0\},$$

where

$$f_i = \max\{-\mathbf{a}_i^T \mathbf{u}_j, j = 1, \dots, k\} = \max(-\mathbf{U}^T \mathbf{a}_i), \quad \mathbf{f} \in R^n.$$

The set P_A is, thus, the intersection of \bar{P} and \bar{A}^c and is the set

$$P_A = \left\{ \mathbf{x} \in R^r \mid \begin{bmatrix} \mathbf{N}^T \\ -\mathbf{A}^T \end{bmatrix} (\mathbf{x} - \mathbf{p}) + \begin{bmatrix} \mathbf{d} - \mathbf{e} \\ \mathbf{b} - \mathbf{f} \end{bmatrix} \leq 0 \right\}. \quad (12)$$

In addition, the set A_p is, then, the set represented by

$$A_p = \left\{ \mathbf{x} \in R^r \mid \begin{bmatrix} -\mathbf{N}^T \\ \mathbf{A}^T \end{bmatrix} (\mathbf{x} - \mathbf{q}) + \begin{bmatrix} \mathbf{d} - \mathbf{e} \\ \mathbf{b} - \mathbf{f} \end{bmatrix} \leq 0 \right\}. \quad (13)$$

Fig. 6 shows an example of the two-dimensional Minkowski sum P_A obtained by the boundary-model approach. For the three-dimensional case, the expression in (12) is only a good approximation for P_A .

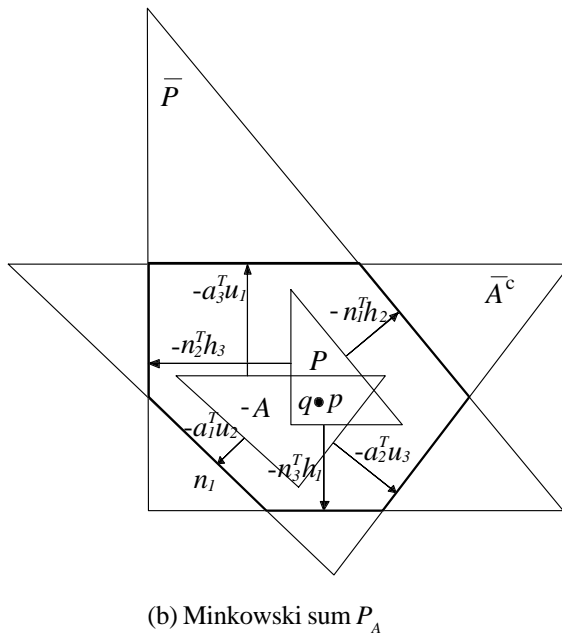
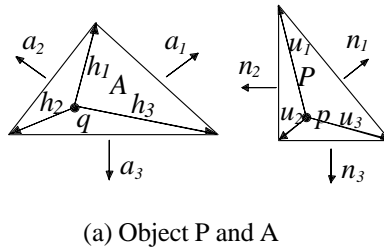


Fig. 6. An example of the two-dimensional Minkowski sum P_A ; (a) the objects A and P, and (b) P_A obtained by the boundary-model approach.

For two disjointed convex polygons in 2D, there are three possible types of closest pairs of features: (1) a pair of vertices, (2) a vertex and an edge, and (3) a pair of edges. Case (3) occurs when the closest pair of features is a pair of parallel edges, and the minimum distance between them is easy to obtain. Case (2) (a vertex and an edge) is important

because it covers case (1) (a pair of vertices). Algorithm MDD2, which computes the MDLD, MD, and MDED functions for two-dimensional convex polygons, works in the following way. First, it computes the MDLD functions between point q and objects for \bar{P} and \bar{A}^c , and searches for the closest pair of features from the vertex-face closed pair and the face-vertex closed pair. If objects A and P are overlapping, there is no need to continue. Otherwise, it finds the closest pair of features from the face-vertex closed pair and vertex-face closed pair, and then checks whether the closest pair of features is a pair of vertices or not. Details of algorithm MDD2 are stated below.

Algorithm MDD2: Minimum directed distances for two convex polygons A and P .

Input: convex polygons A and P

Output: MD $d(A, P)$, MDLD $d_{\infty}(A, P)$, and MDED $d_2(A, P)$

Initial: $x = q$; reduce A to a point
Step 1: Compute $d_{\infty}(q, \bar{P})$; face-vertex closed pair
Step 2: Compute $d_{\infty}(q, \bar{A}^c)$; vertex-face closed pair
Step 3: $d_{\infty}(A, P) = \max\{d_{\infty}(q, \bar{P}), d_{\infty}(q, \bar{A}^c)\}$
 If $(d_{\infty}(A, P) \leq 0)$, then ; overlapping case
 $d_2(A, P) = d_{\infty}(A, P)$
 $d(A, P) = 0$
 otherwise, if $(d_{\infty}(q, \bar{P}) > d_{\infty}(q, \bar{A}^c))$, then ; disjointed case
 $i = \arg\{\max(N^T(q - p) + d - e)\}$
 $j = \arg\{\min(V^T n_i)\}$
 $d(A, P) = d(\text{vertex } j \text{ of } A, \text{ edge } i \text{ of } P)$; face-vertex closed pair
 $d_2(A, P) = d(A, P)$
 or if $(d_{\infty}(q, \bar{P}) < d_{\infty}(q, \bar{A}^c))$, then
 $i = \arg\{\max(-A^T(q - p) + b - f)\}$
 $j = \arg\{\min(U^T a_i)\}$
 $d(A, P) = d(\text{vertex } j \text{ of } P, \text{ edge } i \text{ of } A)$; vertex-face closed pair
 $d_2(A, P) = d(A, P)$
 or if $(d_{\infty}(q, \bar{P}) = d_{\infty}(q, \bar{A}^c))$, then ; disjointed case
 $i = \arg\{\max(N^T(q - p) + d - e)\}$
 $j = \arg\{\max(-A^T(q - p) + b - f)\}$
 $d(A, P) = d(\text{edge } i \text{ of } P, \text{ edge } j \text{ of } A)$; parallel edges
 $d_2(A, P) = d(A, P)$

Assume that an operation means an “inner-product” of two vectors in R^r space. The first step in Algorithm MDD2 requires $(m+ml)$ operations, the second step requires $(n+nk)$ operations, and the last step requires four operations to solve the simple problem of comput-

ing the minimum distance between a point and a line segment. Thus, the computational complexity of MDD2 is $O(ml+nk)$. The complexity can be further reduced to $O(m+n)$ as will be discussed next.

The search for m FV contact-pairs in Step 1 of MDD2 can be done in linear time for convex polygons P and A as follows. Assume that the i th face of P and the j th vertex of A is a FV contact-pair; then, it is required that

$$-n_i^T v_j \geq -n_i^T v_t, \quad t = 0, 1, \dots, n-1,$$

or

$$n_i^T (v_t - v_j) \geq 0, \quad t = 0, 1, \dots, n-1,$$

where n is the number of vertices of polygon A . Because of the convexity of polygon P , the above inequality constraints can be reduced to the following two constraints:

$$n_i^T (v_{j-1} - v_j) \geq 0$$

and

$$n_i^T (v_{j+1} - v_j) \geq 0$$

The procedure of algorithm FVCP for searching m FV contact-pairs is given in detail in the following.

Algorithm FVCP:

Input: Two convex polygons P and A , where P and A contain m and n faces, respectively.

Output: m FV contact-pairs.

- Step 1:** {Find the FV contact-pair of n_0 .}
 - 1.1 register $FV = -\infty$
 - 1.2 for $t = 0$ to $n-1$ do
 - 1.3 $FV = \max\{FV, -n_0^T v_t\}$;
 - 1.4 if $FV = -n_0^T v_t$ then $s = t$;
 - 1.5 end for
- Step 2:** {Finding the FV contact-pair of n_i , where $i = 1, 2, \dots, m-1$.}
 - 2.1 $t = s$
 - 2.2 for $i = 1$ to $m - 1$ do
 - 2.3 while $n_i^T (v_{t-1} - v_t) < 0$ or $n_i^T (v_{t+1} - v_t) < 0$ do
 - 2.4 $t = (t + 1) \bmod n$;
 - 2.5 end while
 - 2.6 end for

The Algorithm FVCP for searching m FV contact-pairs consists of two major steps. In the first step, we find the contact-pair of n_0 of P by searching each vertex of A . Thus, this step takes $O(n)$ time to find $\max\{-n_0^T v_t, t = 0, 1, \dots, n-1\}$ for the face n_0 and its corresponding contact-pair v_s , where $s = \arg(\max\{-n_0^T v_t, t = 0, 1, \dots, n-1\})$. In the second step, we utilize the property described above to find the contact-pair of the faces n_1, n_2, \dots, n_{m-1} with their corresponding contact-pairs simply by sequentially searching the vertices v_s, v_{s+1}, \dots , and v_s in ccw order. This step has $O(m)$ time to find the contact-pairs of faces n_1, n_2, \dots, n_{m-1} of P . Therefore,

the algorithm FVCP for computing the FV contact-pairs has $O(n + m)$ time complexity. For the VF contact case, the n VF contact-pairs can be computed in a similar way. Thus, the procedure used to compute the FV and VF contact-pairs of two polygons P and A has $O(n + m)$ time complexity. Therefore, the computational complexity of MDD2 is reduced to $O(n + m)$.

For two nonoverlapping convex polyhedra in three-dimensional space, there are six possible types of closest pairs of features: (1) a pair of vertices, (2) a vertex and an edge, (3) a vertex and a face, (4) a pair of edges, (5) an edge and a face, and (6) a pair of faces. Cases (5) and (6) occur when the closest pair of features is two parallel faces and/or edges. Cases (3) and (4) are most important for computing the MDLD function. Cases (1) and (2) are degenerated cases of Cases (3) and (4) and only need to be considered in computing the MD and MDED functions. The MDD3 algorithm which computes MDLD, MD, and MDED for two three-dimensional convex polyhedra A and P works in a way similar to that of algorithm MDD2 except that it considers additional edge-edge pairs, and it is listed below.

Algorithm MDD3: Minimum directed distances for two convex polyhedra A and P .

Input: convex polyhedra A and P

Output: MD $d(A, P)$, MDLD $d_{\infty}(A, P)$, and MDED $d_2(A, P)$

Initial: $x = q$; reduce A to a point
Step 1: Compute $d_{\infty}(q, \bar{P})$; face-vertex closed pair
Step 2: Compute $d_{\infty}(q, \bar{A}^c)$; vertex-face closed pair
Step 3: Compute δ_{max} among edge-edge pairs ; edge-edge closed pair
Step 4: $d_{\infty}(A, P) = \max\{d_{\infty}(q, \bar{P}), d_{\infty}(q, \bar{A}^c), \delta_{max}\}$
 If ($d_{\infty}(A, P) \leq 0$), then ; overlapping case
 $d_2(A, P) = d_{\infty}(A, P)$
 $d(A, P) = 0$
 otherwise, if ($d_{\infty}(q, \bar{P}) > d_{\infty}(q, \bar{A}^c), \delta_{max}$), then ; disjointed case
 $i = \arg\{ \max(N^T(q - p) + d - e) \}$
 $j = \arg\{ \min(V^T n_i) \}$
 $d(A, P) = d(\text{vertex } j \text{ of } A, \text{face } i \text{ of } P)$; face-vertex closed pair
 $d_2(A, P) = d(A, P)$
 or if ($d_{\infty}(q, \bar{A}^c) > d_{\infty}(q, \bar{P}), \delta_{max}$), then
 $i = \arg\{ \max(-A^T(q - p) + b - f) \}$
 $j = \arg\{ \min(U^T a_i) \}$
 $d(A, P) = d(\text{vertex } j \text{ of } P, \text{face } i \text{ of } A)$; vertex-face closed pair
 $d_2(A, P) = d(A, P)$
 or if ($d_{\infty}(q, \bar{A}^c) = d_{\infty}(q, \bar{P}) > \delta_{max}$), then
 $i = \arg\{ \max(N^T(q - p) + d - e) \}$
 $j = \arg\{ \max(-A^T(q - p) + b - f) \}$
 $d(A, P) = d(\text{face } i \text{ of } P, \text{face } j \text{ of } A)$; parallel faces
 $d_2(A, P) = d(A, P)$
 or if ($\delta_{max} > d_{\infty}(q, \bar{P}), d_{\infty}(q, \bar{A}^c)$), then
 $d(A, P) = d(\text{edge } i \text{ of } P, \text{edge } j \text{ of } A)$; edge-edge closed pair
 $d_2(A, P) = d(A, P)$
 or if ($\delta_{max} = d_{\infty}(q, \bar{P}) > d_{\infty}(q, \bar{A}^c)$), then
 $i = \arg\{ \max(N^T(q - p) + d - e) \}$

$$\begin{aligned}
 d(A, P) &= d(\text{edge } j \text{ of } A, \text{ face } i \text{ of } P) && ; \text{ parallel face and edge} \\
 d_2(A, P) &= d(A, P) \\
 \text{or if } (\delta_{\max} = d_{\infty}(\mathbf{q}, \bar{A}^c) > d_{\infty}(\mathbf{q}, \bar{P})), &&& \text{ then} \\
 j &= \text{arg}\{ \max(-\mathbf{A}^T(\mathbf{q} - \mathbf{p}) + \mathbf{b} - \mathbf{f}) \} \\
 d(A, P) &= d(\text{face } j \text{ of } A, \text{ edge } i \text{ of } P) && ; \text{ parallel edge and face} \\
 d_2(A, P) &= d(A, P)
 \end{aligned}$$

The above algorithm first computes the MDLD function between point \mathbf{q} and set \bar{P} and searches for the vertex-face closed pair. Second, it computes the MDLD function between point \mathbf{q} and set \bar{A}^c and searches for the face-vertex closed pair. It then finds the maximum value δ_{\max} of the minimum distances in Eq. (11) among possible c edge-edge pairs which are generated during the first two steps, as shown in Fig. 7. The edges can become a contact pair if each lies in the boundary of its face-vertex closed pair and vertex-face closed pair found during Steps 1 and 2. If objects A and P are overlapping, then it stops. Otherwise, it finds the closest pair of features among the closed pairs found in Steps 1, 2, and 3, and then checks whether the closest pair of features is a pair of vertices or a pair containing a vertex and an edge.

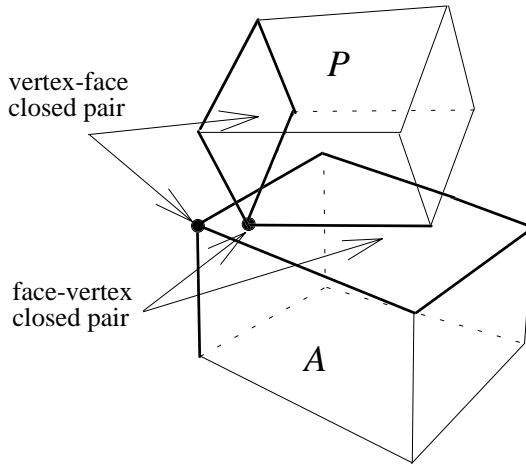


Fig. 7. Possible candidate edge-edge pairs for computing MDLD. The candidate edge-edge pair lies on the boundary of its face-vertex closed pair and vertex-face closed pair.

For the MDD3 algorithm, the first step requires $(m+ml)$ operations, the second step requires $(n+nk)$ operations, and the third step requires around $c(l+k)$ operations. The last step requires many fewer operations than does Step 3 to compute the minimum distance between a point and a 3D convex polygon or between two line segments [8]. The problem of finding the minimum distance between two segments can be transformed into the problem of finding the minimum distance between a point and a parallelogram in 3D. Computing the minimum distance between a point and a convex polygon in 3D is a simple task and can be solved in at most $\max(l, k)$ operations. Thus, the computational complexity of MDD3 is O

$(ml+nk+c(l+k))$. The complexity can be further reduced to $O(m+n+c(l+k))$ when both polyhedra A and P have fixed orientations during motion, and growth obstacles \bar{P} and \bar{A}^c are only computed once.

Although the FV constact pairs in Algorithm MDD3 cannot be done in linear time as we have shown in Algorithm MDD2, the computation of MDLD using the boundary model of the Minkowski sum only requires consideration of cases with FV, VF, and EE contacts. Compared to exhaustive searching, the number of operations required to compute MD and MDED is reduced to about one-third. Moreover, the penetration measure for two intersecting objects can also be obtained using the proposed approach.

4. EXAMPLES AND APPLICATIONS

The examples used in testing algorithm MDD3 include platonic solids listed in Table 1. The data structure of a convex polyhedron has a field for its faces, edges, vertices, the position of the reference point, and its orientation. Each face is represented by its outward normal vector and its distance from the origin. Its data structure also includes a list of vertices which lie on its boundary. Each edge is described by two connecting vertices and its two neighboring faces. Each vertex is characterized by its x, y, and z coordinates with respect to its reference point. Examples were run on an IBM PC which had a 33Mhz 486 CPU. Fig. 8 shows two examples of the minimum directed Euclidean distance $d_2(A, P)$ obtained using the MDD3 algorithm.

Table 1. Platonic solids used in 3D examples.

	Tetrahedron	Cube	Octahedron	Icosahedron
Number of vertices	4	8	6	12
Number of faces	4	6	8	20

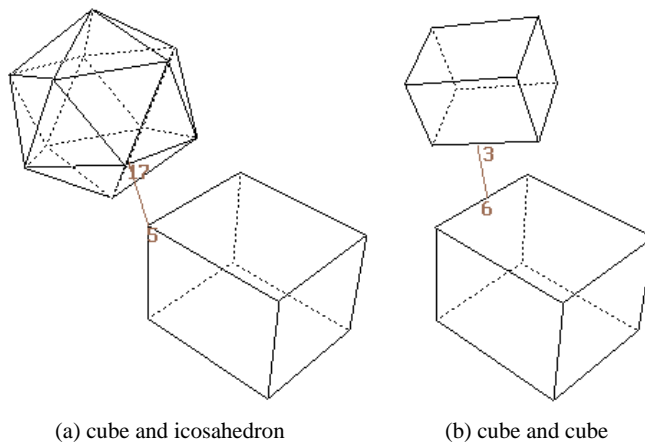


Fig. 8. Example results of algorithm MDD3 for the minimum distance between two platonic solids; (a) cube and icosahedron, and (b) cube and cube.

The resulting minimum distance has been verified by using the traditional optimization (reduced gradient) routine. The computational time needed for the MDD3 algorithm for objects listed in Table 1 is summarized in Table 2. If sets \bar{P} and \bar{A}^c were computed in advance, the subroutine could generally be run in several milliseconds (about 3 to 8 milliseconds). Numerical data in Table 2 indicate that algorithm MDD3 roughly increased linearly in computation time with the total number of vertices and faces.

Table 2. Running time (in milliseconds) of MDD3 between two platonic solids A and P with /without computing \bar{P} and \bar{A}^c .

$A \setminus P$	Tetrahedron	Cube	Octahedron	Icosahedron
Tetrahedron	7.44 / 2.4	10.98 / 3.4	10.25 / 4.8	19.38 / 6.1
Cube	11.24 / 3.2	15.42 / 5.4	15.44 / 5.1	28.12 / 7.2
Octahedron	9.78 / 4.9	14.35 / 5.2	14.89 / 4.9	29.13 / 7.0
Icosahedron	18.25 / 6.0	26.48 / 7.4	28.15 / 6.8	57.65 / 8.4

In the first application, we will illustrate the procedure used in applying the proposed MDLD function to detect the intersection between a line segment and a convex polygon P in a two-dimensional x - y space. The line segment may represent a line path of a point object while the polygon represents either a polygon obstacle or a configuration-space obstacle. The line segment in two-dimensional space can be represented by a degenerated polygon as shown in Fig. 9(a). Thus, the intersection detection problem becomes a standard problem which algorithm MDD2 can solve. Consider a line segment with two end points p_0 and p_1 , and a unit vector u which points to point p_1 from point p_0 , as shown in Fig. 9(a). Line segment p_0p_1 in 2D can be represented by a polygon with two vertices, p_0 and p_1 , and satisfying two planar half-spaces as below:

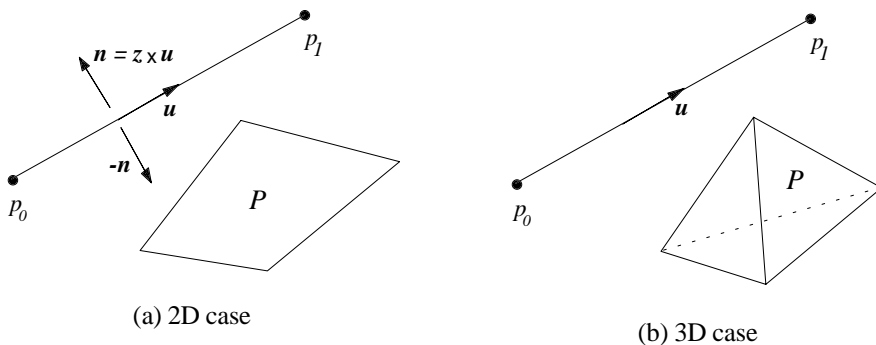


Fig. 9. Collision detection between a line segment and a convex polyhedron P ; (a) 2D case, and (b) 3D case.

$$\mathbf{n}^T(\mathbf{x} - \mathbf{p}_0) \leq 0 \text{ and } -\mathbf{n}^T(\mathbf{x} - \mathbf{p}_1) \leq 0,$$

where $\mathbf{n} = \mathbf{z} \times \mathbf{u}$, “ \times ” is the vector cross-product operation, and \mathbf{z} is the z-axis unit vector. Therefore, the collision intersection problem between a line segment and a convex polygon is reduced to that of finding the MDLD. This methodology can be directly applied to three-dimensional cases, where a line segment in three-dimensional space is represented by a convex polyhedron having one edge, $\mathbf{p}_0\mathbf{p}_1$, and two vertices, \mathbf{p}_0 and \mathbf{p}_1 , as shown in Fig. 9(b).

In the second application, we will illustrate a “generate-and-test” 2D path-planning algorithm for a point robot by using the proposed MDLD as a key function. The input is a set of stationary convex polygonal obstacles, the “initial” position, and the “final” position. If the robot is not a point object, then we can reduce the problem to a point robot problem by working on the configuration space. The procedure is as follows.

Algorithm 2DPATH: 2D path planning

Input: “initial” and “final” positions

Output: Collision-free line-segment path passes the subgoals in the subgoal list
 $\{S_0 = \text{“initial”}, S_1, \dots, S_N = \text{“final”}\}$

Initial: starting point $S = \text{“initial”}$, and target point $G = \text{“final”}$,
subgoal list = $\{S_0 = \text{“initial”}\}$

Step 1: Detect intersection with a line segment path.

Connect starting S point and target G point.

Identify the obstacles intersecting the straight line path between S and G . If no obstacle is found, then go to Step 2; otherwise go to Step 3.

Step 2: Update subgoal list.

If target point G is equal to “goal”, then place “goal” into the subgoal list and finish. Otherwise, place target G into the subgoal, let the new starting point $S = G$ and the new target point $G = \text{“goal”}$, and go to Step 1.

Step 3: Search the nearest obstacle.

Search the nearest obstacle among colliding obstacles to the starting point and go to Step 4. The nearest obstacle is the one which has the minimum distance to starting point S .

Step 4: Group connecting obstacles.

Does the nearest obstacle intersect other obstacles? If so, add these obstacles to the nearest obstacle list. Repeat adding of those obstacles which intersect obstacles in the list to the nearest obstacle list. Go to Step 5.

Step 5: Select midpoint.

Straight line L , which passes through S and G , divides all the vertices in the nearest obstacle list into two groups. Search the farthest vertex from the group which has the smaller average distance from vertices to the line L . The farthest point is the one with the largest distance to straight line L . Assign the selected farthest vertex as the new target point G and go to Step 1.

In the above algorithm, Steps 1, 3, and 4 are easily solved by the MDLD function developed here. It is noted that Step 4, which connects overlapping obstacles, can be done in advance and requires very little time during path planning. An example of the result of the proposed algorithm in selecting one subgoal is shown in Fig. 10. The 2DPATH algorithm is quite fast, and the planned trajectories are near the shortest paths. Fig. 11 shows a result of path planning done by the 2DPATH algorithm with only seven vertices visited and fifteen line segments generated. Algorithm 2DPATH was tested in the environment as shown in Fig. 11 with 100 randomized “initial” and “final” points. The average running time on an IBM 486-33 PC was about 45 milliseconds. The algorithm proposed here differs from previous approaches in three important ways, (1) detection of the trajectory in admissibility, (2) the means of generating alternative immediate points, and (3) the generated subgoal in forward order. The above algorithm can be modified for use in the 3D path-planning problem. 3D path planning can be reduced to 2D path-planning in a 2D path plane that passes “initial” and “final” points. The 2D polygonal obstacle’s vertices are obtained from the intersection of the edges of obstacles and the 2D path plane.

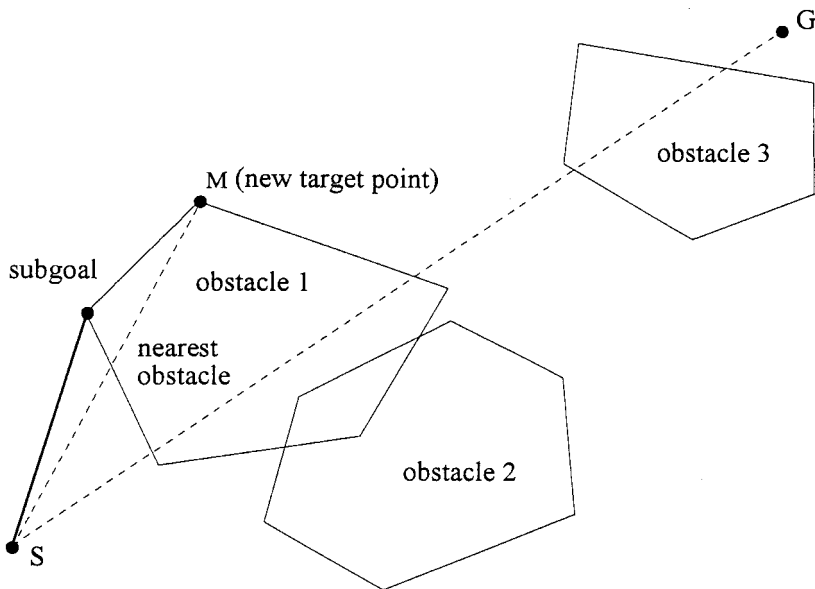


Fig. 10. Example of decision made by algorithm 2DPATH.

5. CONCLUSIONS

Computation of the proposed minimum directed distance has been recast as the Minkowski sum of a two convex object problem. The proposed boundary-model approach can generate analytical expressions for the Minkowski sum of two convex objects 2D; hence, MDLD can be expressed in analytical form. For the 3D case, only a few edge-edge pairs need be considered during computation of the minimum distance between two convex polyhedra. The algorithm developed here can be used for collision detection, computation of the dis-

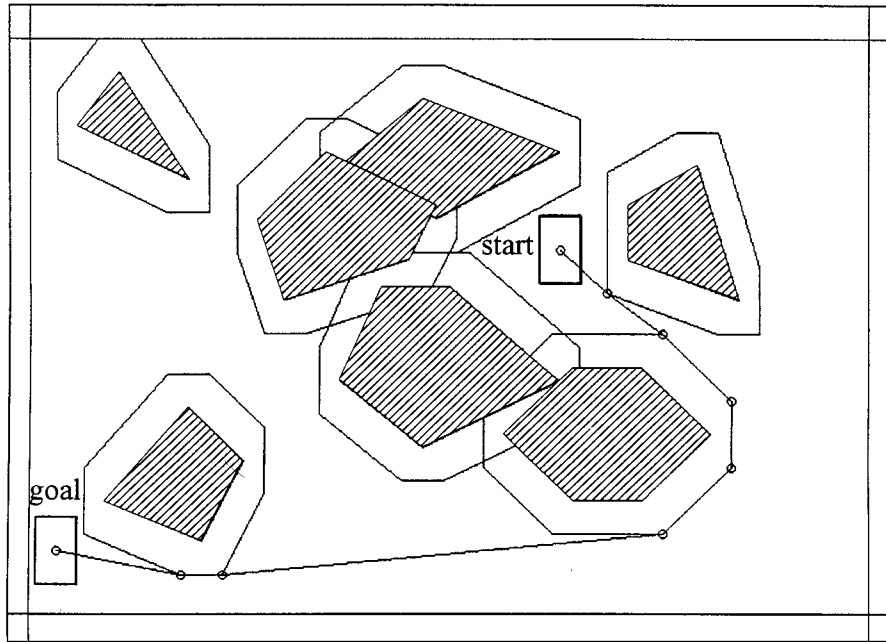


Fig. 11. A path planned by the 2DPATH algorithm in 2D.

tance between two polyhedra in three-dimensional space, and robotics path-planning problems. The advantages of using the Minkowski sum in computing distance measures between two convex polyhedra are: (1) the measure of the separating distance or degree of penetration can be considered; (2) MDLE, MD and MDED functions can be computed in one routine; and (3) computation complexity is linear in 2D, and computation is fast in 3D.

REFERENCES

1. J.E. Bobrow, "A direct minimization approach for obtaining the distance between convex polyhedra," *International Journal of Robotics Research*, Vol. 8, No. 3, June 1989, pp. 65-76.
2. C.E. Buckley, "A foundation for the flexible-trajectory approach to numeric path planning," *International Journal of Robotics Research*, Vol. 8, No. 3, June 1989, pp. 44-64.
3. S.A. Cameron and R.K. Culley, "Determining the minimum translational distance between two convex polyhedra," *IEEE International Conference on Robotics and Automation*, 1986, pp. 591-596.
4. F. Chin, and C.A. Wang, "Optimal algorithms for the intersection and minimum distance problems between planar polygons," *IEEE Transactions on Computers*, Vol. C-32, 1983, pp. 1203-1207.
5. E.G. Gilbert, D.W. Johnson and S.S. Keerthi, "A fast procedure for computing the dis-

- tance between complex objects in three-dimensional space," *IEEE Transactions on Robotics and Automation*, Vol. 4, No. 2, 1988, pp. 193-203.
6. M.C. Lin and J. Canny, "A fast algorithm for incremental distance calculation," *IEEE International Conference on Robotics and Automation*, April, 1991, pp. 1008-1014.
 7. T. Lozano-Perez, "Spatial planning: a configuration space approach," *IEEE Transactions on Computers*, Vol. 32, No. 2, 1983, pp. 108-120.
 8. V.J. Lumelsky, "On fast computation of distance between line segments," *Information Processing Letters*, Vol. 21, August, 1985, pp. 55-61.
 9. W. Meyer, "Distances between boxes: applications to collision detection and clipping," *IEEE International Conference on Robotics and Automation*, 1986, pp. 597-602.
 10. J.T. Schwarz, "Finding the minimum distance between two convex polygons," *Information Processing Letters*, Vol. 13, 1981, pp. 168-170.



Ching-Long Shih (施慶隆) received the B.S. and M.S. degrees from National Chiao Tung University, and the Ph. D. degree in electrical engineering from the Ohio State University in 1988. Currently, he is a professor in the Department of Electrical Engineering, National Taiwan Institute of Technology. His research interests include mobile robot motion planning and legged robot locomotion.



Jane-Yu Liu (劉建渝) received the B.S. degree in electronic engineering from Chung-Yuan Christian University in 1992 and the M.S. degree in electrical engineering from the National Taiwan Institute of Technology in 1994. The subject of his masters thesis was minimum directed distance and path planning in 2D space. He currently is employed by the Winbond Electronics Corp., working on device reliability.