'90 KACC 1990. 10. 26~27

A New Algorithm for Detecting the Collision of Moving Objects

S. M. Hong

Department of Electronics
Kyungpook National University, Taegu, Korea

Abstract

Iterative algorithms for detecting the collision of convex objects whose motion is characterized by a path in configuration space are described. They use as an essential substep the computation of the distance between the two objects. When the objects are polytopes in either two or three dimensional space, an algorithm is given which terminates in a finite number of iterations. It determines either that no collision occurs or the first collision point on the path. Extensive numerical experiments for practical problems show that the computational time is short and grows only linearly in the total number of vertices of the two polytopes.

I Introduction

A common problem in robotics is the detection of collision between two rotating and translating objects, where the rotations and translations are described by a specified path in a suitable configuration space. For example, time histories for the joint variables of two manipulators may be given and it must be determined if and when each pair of constituent parts (links, payloads, obstacles) in the three dimensional work space collides.

Usually, the objects are modelled by polytopes and collision is detected by examining the potential contact of a vertex with a face or an edge with an edge. The nature and difficulty of the computations vary depending on the approach and assumptions. Whatever the approach, the computational time increases as M_1M_2 where M_1 and M_2 represent the complexity of the polytopes (number of vertices) [7,8].

In this paper we describe a new algorithmic approach which has its origins in [1]. It applies to convex objects and uses, as a substep, the computation of the distance between the objects. If at a point on the path the distance is positive, the objects do not intersect and are, in fact, separated by slab of thickness equal to the distance. This means further collision-free motion along the path is possible: the motion can be continued until the thickness of the separating slab shrinks to zero. The algorithmic process consists of successive steps of this type. The main computational effort occurs in two places: the determination of the distance between the pair of convex objects and a root finding problem which corresponds to finding the point on the path where the slab has thickness zero.

The approach is most effective when the objects are convex polytopes. Then the distance computation is well understood. See [2,4]. Furthermore, it is possible to show that a variation of the basic algorithm solves the collision detection problem in a finite number of steps. Results of computational experiments show that the computational times are short and, unlike the prior methods, appear to be proportional to $M_1 + M_2$.

II Problem Statement

To formulate the collision detection problem, it is convenient to represent the spaces occupied by the two potentially colliding objects by point sets K_1, $K_2 \subset R^m$[7]. We keep m general because both $m = 2$ and $m = 3$ occur in practice. The sets represent objects of fixed shape which undergo translations and rotations. The translations and rotations are dependent on a configuration vector q which belongs to a configuration space Q. Thus,

$$K_i = \bar{R}_i(\mathrm{q})C_i + \{\bar{p}_i(\mathrm{q})\} \triangleq \{x = \bar{R}_i(\mathrm{q})w + \bar{p}_i(\mathrm{q}) : w \in C_i\}\ , \tag{2.1}$$

where for $i = 1, 2$: $\bar{p}_i(\mathrm{q})$ is the translation vector, $\bar{R}_i(\mathrm{q})$ is the m by m (orthogonal) rotation matrix, and $C_i \subset R^m$ is the set of points which describes the space occupied by object i in its reference position and orientation.

The configuration vector moves along a path in Q which is specified parametrically by a continuous function $\mathrm{q} : \Theta \to Q$. Here $\Theta = [\theta_b, \theta_e]$, where $\mathrm{q}(\theta_b)$ is the beginning point on the path and $\mathrm{q}(\theta_e)$ is the ending point on the path. We assume that the objects do not interfere at the starting point and wish to determine the value of θ where collision first occurs, or that no collision occurs for all $\theta \in \Theta$.

To be more specific, let

$$R_i(\theta) \triangleq \bar{R}_i(\mathrm{q}(\theta)),\ p_i(\theta) \triangleq \bar{p}_i(\mathrm{q}(\theta)), \tag{2.2}$$

and define

$$K_i(\theta) = R_i(\theta)C_i + \{p_i(\theta)\},\ i = 1,2. \qquad (2.3)$$

Our objective is to solve the following problem.

Collision Detection Problem (CDP) Assume $K_1(\theta_0) \cap K_2(\theta_0) = \phi$. Find the collision point

$$\theta^* = \min\{\theta \in \Theta : K_1(\theta) \cap K_2(\theta) \neq \phi\} \qquad (2.4)$$

or show that

$$K_1(\theta) \cap K_2(\theta) = \phi \quad \text{for all} \quad \theta \in \Theta\ . \qquad (2.5)$$

III Geometric Preliminaries

For $x, y \in R^m$ and $X \subset R^m$: $\langle x, y\rangle$ is the inner product, $|x| = \sqrt{\langle x, x\rangle}$ is the Euclidean norm, *co* X is the convex hull of X. The hyperplane $\{x : \langle \eta, x - y\rangle = 0\}$, $\eta \neq 0$, *separates* $X_1, X_2 \subset R^m$ if $\langle \eta, x - y\rangle \geq 0$ for all $x \in X_1$ and $\langle -\eta, x - y\rangle \geq 0$ for all $x \in X_2$.

A polytope is the convex hull of a finite point set. Thus, e.g., when C_i is a polytope it may be represented by

$$C_i = co\ \{w_{ij} : j = 1, \ldots, M_i\}\ . \qquad (3.1)$$

For a given polytope it is desirable to make M_i as small as possible. In this case, w_{ij}, $j = 1, \ldots, M_i$, are the vertices of the polytope. If C_i is the polytope (3.1), $K_i(\theta)$ is also a polytope:

$$K_i(\theta) = co\ \{z_{ij}(\theta) : j = 1, \ldots, M_i\}, \qquad (3.2)$$

where

$$z_{ij}(\theta) = R_i(\theta)w_{ij} + p_i(\theta)\ . \qquad (3.3)$$

The support function of a compact set $X \subset R^m$, $h_X : R^m \to R$, is defined by

$$h_X(\eta) \triangleq \max\{\langle \eta, x\rangle : x \in X\}\ . \qquad (3.4)$$

A support mapping of X is a function $s_X : R^m \to X$ such that

$$\langle \eta, s_X(\eta)\rangle = h_X(\eta)\ . \qquad (3.5)$$

Thus, $s_X(\eta)$ is a point in X which is farthest in the direction η; see Figure 1. The supporting half space in the direction η is:

$$S_X(\eta) \triangleq \{x : \langle \eta, x\rangle \geq h_X(\eta)\}\ . \qquad (3.6)$$

When $K_i(\theta)$ is the polytope described by (3.2) and (3.3), it is easy to determine the support function and mapping:

$$h_{K_i(\theta)}(\eta) = \max\{\langle \eta, z_{ij}(\theta)\rangle : j = 1, \ldots, M_i\}, \qquad (3.7)$$

$$s_{K_i(\theta)}(\eta) = z_{ij}(\theta),\ j \text{ satisfies } \langle \eta, z_{ij}(\theta)\rangle = h_{K_i(\theta)}(\eta)\ . \qquad (3.8)$$

Numerically, (3.7) and (3.8) are obtained by comparing the M_i inner products $\langle \eta, z_{ij}(\theta)\rangle$.

An application of the above definitions is shown in Figure 2. We wish to know if two compact sets K_1 and K_2 can be separated by a hyperplane with normal η, $\eta \neq 0$. The set

$$L_{K_1,K_2}(\eta) = S_{K_1}(-\eta) \cap S_{K_2}(\eta) \qquad (3.9)$$

is the thickest slab between K_1 and K_2 whose bounding hyperplanes have the normal vector η. A separating hyperplane exists if and only if $L_{K_1,K_2}(\eta)$ is not empty. It is easy to see that the thickness of the slab is given by

$$l_{K_1,K_2}(\eta) = -|\eta|^{-1}(h_{K_1}(-\eta) + h_{K_2}(\eta))\ . \qquad (3.10)$$

If $l_{K_1,K_2}(\eta) = 0$, K_1 and K_2 may or may not intersect, but $l_{K_1,K_2}(\eta) > 0$ implies $K_1 \cap K_2 = \phi$. Of course, $l_{K_1,K_2}(\eta) < 0$ is possible even if $K_1 \cap K_2 = \phi$.

The distance between two compact sets $K_1, K_2 \subset R^m$ is defined by

$$d(K_1, K_2) = \min\{|x_1 - x_2| : x_1 \in K_1, x_2 \in K_2\}\ . \qquad (3.11)$$

If $\nu_1 \in K_1$, $\nu_2 \in K_2$ solve the minimization problem (3.11), i.e., $|\nu_1 - \nu_2| = d(K_1, K_2)$, ν_1 and ν_2 are a pair of *near points*.

The convexity of K_1 and K_2 does imply that $\nu_1 - \nu_2$ is unique [3]. It also leads to the separation of K_1 and K_2 by a slab. See Figure 3. ν_1 is a suitable choice for $s_{K_1}(-\xi)$. Similarly, ν_2 is a suitable choice for $s_{K_2}(\xi)$. Thus, by Figure 2 $L_{K_1,K_2}(\xi) = S_{K_1}(-\xi) \cap S_{K_2}(\xi)$ is a separating slab for K_1 and K_2. Moreover, (3.10) shows that the thickness of the slab is $l_{K_1,K_2}(\xi) = d(K_1, K_2) = |\xi|$. It can be shown that $\eta = \xi$ maximizes $l_{K_1,K_2}(\eta)$. In this sense $L_{K_1,K_2}(\xi)$ is the *best* separating slab.

In subsequent sections, K_1 and K_2 depend on θ in the manner defined by (2.3). We will use $\nu_1(\theta)$, $\nu_2(\theta)$ to denote a near point pair in $K_1(\theta)$, $K_2(\theta)$ and let $\xi(\theta) = \nu_1(\theta) - \nu_2(\theta)$. To avoid a notational morass the following conventions are adopted:

$$d(\theta) = d(K_1(\theta), K_2(\theta))\ , \qquad (3.12)$$

$$L(\theta; \eta) = L_{K_1(\theta),K_2(\theta)}(\eta)\ , \qquad (3.13)$$

$$l(\theta; \eta) = l_{K_1(\theta),K_2(\theta)}(\eta) \qquad (3.14)$$

IV The Basic Algorithm

The preceding separation results motivate the basic algorithm for solving the CDP. The algorithm is an iterative process which produces an increasing sequence of points in Θ. It begins with $\theta_0 = \theta_0$ and after k steps assures that no collision occurs on $[\theta_0, \theta_k]$. A geometric interpretation of the step that produces $\theta_{k+1} > \theta_k$ is shown in Figure 4 and is justified as follows.

Since $K_1(\theta_k) \cap K_2(\theta_k) = \phi$,

$$d_k \triangleq d(\theta_k) > 0 \quad \text{and} \quad \xi_k \triangleq \xi(\theta_k) \neq 0\ . \qquad (4.1)$$

Assume $K_1(\theta_k)$ and $K_2(\theta_k)$ are convex. Then the slab $L(\theta_k; \xi_k)$ separates $K_1(\theta_k)$ and $K_2(\theta_k)$ and has thickness

d_k. Now consider the slab $L(\theta;\xi_k)$ for $\theta > \theta_k$. Its orientation is fixed (the bounding hyperplanes have normal ξ_k) but its thickness and position vary continuously with θ. As long as the thickness is positive $K_1(\theta) \cap K_2(\theta) = \phi$. This suggests that θ_{k+1} be determined by allowing θ to increase until: (1) the thickness is zero or (2) $\theta = \theta_e$. In case (1) either $\theta_{k+1} = \theta^*$ or $\theta_{k+1} > \theta_k$ and there is no collision on $[\theta_k, \theta_{k+1}]$. In case (2) it follows there is no collision on Θ. We now formalize our geometric description. The parameter $\epsilon > 0$ governs the accuracy of the algorithmic solution $\tilde{\theta}$.

Basic Collision Detection Algorithm (BCDA)

Step 1 Set $k = 0$ and $\theta_o = \theta_k$.

Step 2 Determine d_k and ξ_k.

Step 3 If $d_k \leq \epsilon$, stop and set $\tilde{\theta} = \theta_k$.

Step 4 Solve, if a root exists, the problem of finding the *smallest* root of $l(\theta;\xi_k) = 0$ on $[\theta_k, \theta_e]$; Equivalently, find, if it exists, the smallest root of

$$F_k(\theta) = h_{K_1(\theta)}(-\xi_k) + h_{K_2(\theta)}(\xi_k) = 0 \qquad (4.2)$$

on $[\theta_k, \theta_e]$

Step 5 If the root finding problem has no solution, stop. There is no collision on Θ.

Step 6 Let θ_{k+1} be the solution of the root finding problem. Increment k by 1 and go to Step 2.

Using methods similar to those used in [1] it is possible to prove that the algorithm performs in a reasonable way [6]

Theorem 4.1 For $i = 1, 2$ assume that C_i is compact and convex and the elements of R_i and p_i are Lipschitz continuous. Then for $\epsilon > 0$, BCDA has the following properties. (i) It stops in a finite number of steps. (ii) If (2.5) holds and ϵ is chosen so that $\epsilon < d(\theta)$ for all $\theta \in \Theta$, the algorithm stops in Step 5. (iii) Suppose collision occurs in Θ. Then the algorithm stops in Step 3. Moreover, $\tilde{\theta} \leq \theta^*$ and $\theta^* - \tilde{\theta}$ can be made as small as desired by a suitably small choice of ϵ.

Result (iii) implies that with $\epsilon = 0$, BCDA generates a convergent sequence $\{\theta_k\}$. Unfortunately, the rate of convergence may be quite slow. The reason for the possible slow convergence is detailed in [5].

Another approach for generating an $\eta(\theta)$ in the direction of $\xi(\theta)$ is possible when $K_1(\theta)$ and $K_2(\theta)$ are polytopes. As discussed in the next section, this approach causes the CDP to be solved in a finite number of steps.

V The Polytope Algorithm

Throughout this section it is assumed that $m = 3$ and $K_1(\theta)$, $K_2(\theta)$ are the polytopes given by (3.2) and (3.3). Generically, $K_1(\theta^*) \cap K_2(\theta^*)$ is a single point with only two types of contact: a vertex-facet contact and an edge-edge contact with edges which are not parallel. Finite convergence can be obtained in the generic cases if the root finding problem corresponding to Step 4 of BCDA is replaced one of three procedures: A, B or C. The choice of the procedure depends on the configuration of $\nu_1(\theta_k)$ and $\nu_2(\theta_k)$. We assume in the following discussions that $\xi_k \neq 0$.

Procedures A and B are intended to give one step convergence for θ_k in the neighborhood of generic vertex-facet contacts.

Procedure A(B) Find, if it exists, the smallest root of $F_k(\theta) = 0$ on $[\theta_k, \theta_e]$, where

$$F_k(\theta) = h_{K_1(\theta)}(-R_i(\theta)R_i^T(\theta_k)\xi_k) + h_{K_2(\theta)}(R_i(\theta)R_i^T(\theta_k)\xi_k) \qquad (5.1)$$

with $i = 1 (i = 2)$.

Procedure C applies when θ_k is in the neighborhood of a generic edge-edge contact. The idea is to capture the behavior of $K_1(\theta)$ and $K_2(\theta)$ for $\theta \in [\theta_k, \theta^*]$ by approximating wedges $TK_1(\theta)$ and $TK_2(\theta)$. See Figure 5.

We now state procedure C.

Procedure C Let Γ be set of roots of the following equations on $[\theta_k, \theta_e]$:

$$\det[(z_{12}(\theta)-z_{11}(\theta))\,(z_{22}(\theta)-z_{21}(\theta))\,(z_{13}(\theta)-z_{11}(\theta))] = 0\,, \qquad (5.1)$$

$$\det[(z_{12}(\theta)-z_{11}(\theta))\,(z_{22}(\theta)-z_{21}(\theta))\,(z_{14}(\theta)-z_{11}(\theta))] = 0\,, \qquad (5.2)$$

$$\det[(z_{12}(\theta)-z_{11}(\theta))\,(z_{22}(\theta)-z_{21}(\theta))\,(z_{23}(\theta)-z_{21}(\theta))] = 0\,, \qquad (5.3)$$

$$\det[(z_{12}(\theta)-z_{11}(\theta))\,(z_{22}(\theta)-z_{21}(\theta))\,(z_{24}(\theta)-z_{21}(\theta))] = 0\,, \qquad (5.4)$$

$$\det[(z_{12}(\theta)-z_{11}(\theta))\,(z_{22}(\theta)-z_{21}(\theta))\,(z_{22}(\theta)-z_{11}(\theta))] = 0\,. \qquad (5.5)$$

Find, if Γ is not empty, the smallest root in Γ.

Another basic step in the collision detection algorithm is the determination $\nu_1(\theta_k)$ and $\nu_2(\theta_k)$. Since $K_1(\theta_k)$ and $K_2(\theta_k)$ are polytopes, it is possible to use the distance algorithm described in [4]. In addition to being efficient, this algorithm provides needed information about the configuration of the near points. With a simple consolidation of its output data, it yields index sets $I_i(\theta)$ and numbers λ_{ij}, $j \in I_i(\theta)$, such that

$$\nu_i(\theta) = \sum_{j \in I_i(\theta)} \lambda_{ij} z_{ij}(\theta),\ \sum_{j \in I_i(\theta)} \lambda_{ij} = 1,\ \lambda_{ij} > 0 \text{ for } j \in I_i(\theta). \qquad (5.6)$$

The number of elements in $I_i(\theta)$, $N_i(\theta)$, has a direct correspondence to the configuration of $\nu_i(\theta)$: if ν_i is a vertex, $N_i = 1$; if ν_i is contained in edge, $N_i = 2$; if ν_i is contained in a facet, $N_i = 3$. This determination is needed if Procedures A, B and C are to be selected appropriately.

Table I lists all of the 9 possible configurations for a near point pair: $\nu_1(\theta_k), \nu_2(\theta_k)$. For θ_k sufficiently close to θ^*, only three configurations are generically possible: (c), (e) and (g). If θ_k is not near θ^*, all 9 configurations may occur. The Table shows how the procedures are selected to satisfy two essential requirements: the results of Theorem 4.1 must be maintained (so $\theta_k \to \theta^*$ or (2.5) can be detected) and that $\theta_{k+1} = \theta^*$ when θ_k is close to θ^*

With the preceding facts in mind, we now state the polytope algorithm

Polytope Collision Detection Algorithm(PCDA)

Step 1 Set $k = 0$, $\theta_0 = \theta_k$, $N_1(\theta_{-1}) = N_2(\theta_{-1}) = 0$ and

Table I Selection of Procedure A, B or C

	Configuration		$m = 3$		
	$N_1(\theta_k)$	$N_2(\theta_k)$	flag=0	flag=1	flag=2
(a)	1	1	B	A	
(b)	1	2	B		
(c)	1	3	B		
(d)	2	1	A		
(e)	2	2	B	A	C
(f)	2	3	B		
(g)	3	1	A		
(h)	3	2	A		
(i)	3	3	B		

flag = 0.

Step 2 Compute d_k, $\nu_1(\theta_k)$, $\nu_2(\theta_k)$, ξ_k, $I_1(\theta_k)$, $I_2(\theta_k)$, $N_1(\theta_k)$ and $N_2(\theta_k)$.

Step 3 If $d_k \leq \epsilon$, stop and set $\tilde{\theta} = \theta_k$.

Step 4 If $N_1(\theta_k) = N_2(\theta_k) = 1$, $I_1(\theta_k) = I_1(\theta_{k-1})$, $I_2(\theta_k) = I_2(\theta_{k-1})$ and flag = 0, increment flag by 1. If $N_1(\theta_k) = N_2(\theta_k) = 2$, $I_1(\theta_k) = I_1(\theta_{k-1})$, $I_2(\theta_k) = I_2(\theta_{k-1})$ and flag = 0 or 1, increment flag by 1. Otherwise, set flag = 0.

Step 5 Enter Table I and implement the indicated procedure.

Step 6 If the procedure in Step 5 does not produce a root, stop. There is no collision on Θ.

Step 7 Let θ_{k+1} be the root provided by the procedure of Step 5. Increment k by 1 and go to Step 2.

The proof of the following theorem is given in [6].

Theorem 5.1 For $i = 1, 2$ assume $K_i(\theta)$ is the polytope defined by (3.2) and (3.3) and the elements of $R_i(\theta)$ and $p_i(\theta)$ are Lipschitz continuous. If a collision occurs on Θ assume it is *regular*, i.e.,

$$K_1(\theta^*) \cap K_2(\theta^*) \text{ is a single point and there is a unique hyperplane which separates } K_1(\theta^*) \text{ and } K_2(\theta^*) . \tag{5.7}$$

Then, for $\epsilon = 0$, the PCDA solves CDP in a finite number of steps. Specifically, it stops either in Step 6 or in Step 3 with $\theta^* = \tilde{\theta}$.

VI The Root Finding Procedure

We now indicate how the root finding problems contained in Procedures A, B and C can be solved.

Because of the complexity of the support functions for $K_1(\theta)$ and $K_2(\theta)$ it is better not to implement Procedures A and B by a direct solution of $F_k(\theta) = 0$. From (3.2), (3.3) and (3.7) it follows that

$$h_{K_i(\theta)}(\eta) = \max\{\langle \eta, R_i(\theta)w_{ij} : j = 1, \ldots, M_i\} + \langle \eta, p_i(\theta)\rangle . \tag{6.1}$$

Using this result in Procedure A gives

$$F_k(\theta) = h_{C_1}(-R_1^T(\theta_k)\xi_k) + h_{C_2}(R_2^T(\theta)R_1(\theta)R_1^T(\theta_k)\xi_k) + \langle R_1(\theta)R_1^T(\theta_k)\xi_k, p_2(\theta) - p_1(\theta)\rangle = 0 . \tag{6.2}$$

Since $h_{C_2}(\eta) = \langle \eta, w_{2j}\rangle$ for some $j \in \{1, \ldots, M_2\}$, the least root of (6.2) on $[\theta_k, \theta_c]$, if it exists, satisfies

$$G_{jk}(\theta) = h_{C_1}(-R_1^T(\theta_k)\xi_k) + \langle R_2^T(\theta)R_1(\theta)R_1^T(\theta_k)\xi_k, w_{2j}\rangle + \langle R_1(\theta)R_1^T(\theta_k)\xi_k, p_2(\theta) - p_1(\theta)\rangle = 0 \tag{6.3}$$

for some j. Let θ^+ be the smallest root of (6.3) on $[\theta_k, \theta_c]$ for $j = 1, \ldots, M_2$. Then we can show that θ^+ is the least root of (6.2) on $[\theta_k, \theta_c]$. Similarly, if (6.3) has no root on $[\theta_k, \theta_c]$ for $j \in \{1, \ldots, M_2\}$ then neither does (6.2).

All of this shows that Procedure A can be implemented by solving the M_2 simple root finding problems (6.3). Similarly, Procedure B is equivalent to the solution of M_1 simple root finding problems. Also, Procedure C always requires the solution of exactly five simple root finding problems.

The numerical solution of the root finding problems which arise in the above context will be described. It is crucial that the *smallest* root of each equation be obtained. Many of the standard root finding algorithms do not guarantee this result.

First we put (6.3) into a simpler, more abstract form. Let $\Omega = [\omega_k, \omega_c]$ be a compact interval in R. Also, let F denote the set of functions $f_i : \Omega \to R$ such that

$$F = \{f_i : i \in I\} \tag{6.4}$$

where $I = \{1, \cdots, M\}$. Then, our root finding problem can be stated generally as follows:

Root Finding Problem

Assume that f_i is C^l and $f_i(\omega_k) > 0$, $i \in I$. Let Γ_R be the set of roots of $f_i(\omega) = 0$ on Ω for all $i \in I$. Find, if Γ_R is not empty, the smallest *root ω^* in Γ_R*, or *indicate Γ_R is empty.*

In our algorithm for solving the root finding problem, we approximate the $f_i \in F$ by polynomials on subintervals of Ω and find the smallest root of each polynomial equation. The assumption that the f_i's are C^l is necessary to assure the validity of the polynomial approximations. Suppose K_1 and K_2 are polytopes as represented in Section 3 and all components of R_i and p_i are C^l. Then, the G_{jk} in (6.3) is C^l. In our algorithm, l is 4. In Appendix, the root finding algorithm is given which uses adaptively chosen polynomial fits and has proved to be efficient and highly reliable.

VII Numerical Examples

Several numerical experiments have been carried out to test the performance of PCDA. The computer system was an Apollo DN 4000 with object code produced by the optimizing Fortran compiler. Distance computations were implemented with the subroutine described in [4].

The test objects are shown in Figure 6. The hexagonal cylinders (a) and (c) have axes which are skewed from being orthogonal to the faces (22° for (a) and 45° for (c)). Two polytope approximations were produced for the robot arm shown in (b): one with 44 vertices and the other with 164. The three cases considered are defined in Table II. K_1 was either (a) or a polytope approximation of (b). The object K_2 was the polytope (c). In each case,

Table II Numerical Results

N = number of examples, $\bar{k}$ = average k at termination,
t = average CPU time in seconds

			Collision				No Collision			
	K_1, K_2	M_1, M_2	N	$\bar{k}$	$\bar{t}$	$\bar{n}$	N	$\bar{k}$	$\bar{t}$	$\bar{n}$
PCDA	(a), (d)	12, 12	656	4.4	0.15	12.6	164	4.0	0.16	14.7
	(b), (d)	44, 12	568	4.8	0.30	12.8	189	3.3	0.30	14.9
	(b), (d)	164, 12	567	4.8	0.84	12.8	189	3.3	0.93	14.8

N distinct examples were generated by randomly positioning the polytope (c) initially. K_1 rotates uniformly on $[0, 2\pi]$ about the z axis, K_2 undergoes a uniform rotation about its body z axis on $[0, \pi]$. The value of ϵ was always less than 10^{-6}.

The results are quite promising. Although the complexity, $M_1 + M_2$, of the three cases varies greatly, the iteration number for termination doesn't. The average ranges from about 4.8 in the collision cases to 3.3 in the no collision cases. In a few examples the iteration number reached between 8 and 12. The average CPU times are approximately proportional to $(M_1 + M_2)\bar{k}$. This is to be expected, since the CPU time for the distance algorithm is approximately proportional to $M_1 + M_2$ (see [7]) and the number of root finding problems needed for Procedures A and B is M_1 and M_2. Approximately 35% of the CPU time was spent in the distance algorithm (Step 2) and 55% on Procedures A,B and C..

VIII Conclusion

An iterative algorithm for detecting the collision of two convex polytopes, whose motion is determined by a specified path in configuration space, has been described. It stops in a finite number of iterations and either produces the first collision point or shows that no collision occurs. There is no useful bound on the number of iterations required, but extensive numerical experiments show that an average of 5 to 6 iterations may be expected. Remarkably, this result seems to be independent of the number of vertices in the polytopes. On each iteration the near points in the two polytopes must be computed (time proportional to $M_1 + M_2$) and M_1 or M_2 root finding problems solved. Thus, the computation time grows only linearly in the total number of vertices of the two polytopes.

A heuristic algorithm has been described which finds the smallest root of multiple functions. The algorithm treats all the functions simultaneously which helps to reduce the required number of evaluations of the functions. It also has special features which make it efficient and accurate. The algorithm was used to solve the root finding problems contained in the collision detection algorithms. In this role, it solved successfully all the root finding problems in our numerical experiments.

Appendix: The Root Finding Algorithm

Consider $n+1$ points on Ω which are equally spaced by h, that is, ω_1, $\omega_2 = \omega_1 + h$, $\omega_3 = \omega_2 + h$, etc. Let $p_{n\,i}$ denote the n-th order polynomial interpolating f_i at the $n+1$ points. For $[\omega_1, \omega_{n+1}] \subset \Omega$ we wish to bound

$$e_{n\,i} = \max_{\omega \in [\omega_1, \omega_{n+1}]} |f_i(\omega) - p_{n\,i}(\omega)| \,. \qquad (A.1)$$

If the f_i, $i \in I$, are C^{n+1}, it follows from the compactness of Ω and the continuity of $d^{n+1}f_i/dx^{n+1}$ that there exists a $L > 0$ such that $|d^{n+1}f_i(\omega)/dx^{n+1}| \le L$ for all $i \in I$ and $\omega \in \Omega$. A standard result on polynomial approximations [9] then shows that

$$e_{n\,i} \le \frac{L}{(n+1)!} \max_{\omega \in [\omega_1, \omega_{n+1}]} |\prod_{j=1}^{n+1} (\omega - \omega_j)| \,, \quad i \in I \,, \quad (A.2)$$

that is, $e_{n\,i} = O(h^{n+1})$. Thus, in principle, $e_{n\,i}$ can be made small by the choice of h. Note

$$f_i(\omega) \ge p_{n\,i}(\omega) - e_{n\,i} \,, \quad i \in I, \ \omega \in [\omega_1, \omega_{n+1}] \,. \quad (A.3)$$

If we can show

$$p_{n\,i}(\omega) > e_{n\,i} \,, \quad i \in I, \ \omega \in [\omega_1, \omega_{n+1}] \,, \qquad (A.4)$$

we know that there is no root of the family F on $[\omega_1, \omega_{n+1}]$. We use this idea in our algorithm. In addition, we try to estimate the roots of $f_i(\omega) = 0$ by obtaining the roots of $p_{n\,i}(\omega) = e_{n\,i}$.

A key issue is choosing h sufficiently small so that the above ideas give acceptable results. Unfortunately, there is no satisfactory procedure for determining L in our root finding problem. Therefore, we estimate $e_{n\,i}$ by a heuristic approach using the polynomial interpolations $p_{n\,i}$ and $p_{n+1\,i}$. Obviously,

$$\max_{\omega \in [\omega_1, \omega_{n+1}]} |f_i(\omega) - p_{n\,i}(\omega)| \le \max_{\omega \in [\omega_1, \omega_{n+1}]} |f_i(\omega) - p_{n+1\,i}(\omega)|$$
$$+ \max_{\omega \in [\omega_1, \omega_{n+1}]} |p_{n+1\,i}(\omega) - p_{n\,i}(\omega)| \,. \qquad (A.5)$$

From (A.1) and (A.2) the first and second terms in the right hand side of (A.5) are functions of $O(h^{n+2})$ and $O(h^{n+1})$, respectively. Thus, if h is small enough, it holds that

$$e_{n\,i} \le \max_{\omega \in [\omega_1, \omega_{n+1}]} |f_i(\omega) - p_{n+1\,i}(\omega)|$$
$$+ \max_{\omega \in [\omega_1, \omega_{n+1}]} |p_{n+1\,i}(\omega) - p_{n\,i}(\omega)|$$
$$\approx \max_{\omega \in [\omega_1, \omega_{n+1}]} |p_{n+1\,i}(\omega) - p_{n\,i}(\omega)| = e_{max\,i} \,. \qquad (A.6)$$

If h is small enough, $e_{max\,i}$ is an approximate upper bound of $|f_i(\omega) - p_{n\,i}(\omega)|$ on $[\omega_1, \omega_{n+1}]$. Moreover, (A.3) and (A.6) imply that to a good approximation the following inequality holds

$$f_i(\omega) \geq p_{n\,i}(\omega) - e_{max\,i} \,. \qquad (A.7)$$

Thus, we can solve $p_{n\,i}(\omega) - e_{max\,i} = 0$ instead of $f_i(\omega) = 0$ to verify $f_i(\omega) > 0$ for $[\omega_1, \omega_{n+1}]$ or to obtain an approximate smallest root of $f_i(\omega) = 0$ on $[\omega_1, \omega_{n+1}]$. Since $p_{n\,i}(\omega) - e_{max\,i} = 0$ is a polynomial equation, it is easy to solve.

In our root finding algorithm, we use $\chi_i \triangleq e_{max\,i}/f_i(\omega_1)$ as the parameter to control h. The parameter χ_i represents the approximation accuracy of $p_{n\,i}$ relative to $f_i(\omega_1)$. In the algorithm, we preset two threshold values, $\chi 1$ and $\chi 2$ ($\chi 1 > \chi 2$), for χ_i. If $\chi_i > \chi 1$ for any i, the algorithm determines that the approximation is not accurate enough and reduces h to $h/3$ to obtain a more accurate approximation; if $\chi_i < \chi 2$ for all i, the algorithm decides that it can increase h to $3h$ while keeping reasonable accuracy; otherwise, the algorithm keeps h without change. Generally speaking, the above choice of χ_i requires smaller h as f_i is closer to 0 on $[\omega_1, \omega_{n+1}]$.

If $\chi_i < \chi 1$ for all $i \in I$, we compute, if it exists, the smallest root of

$$p_{n\,i}(\omega) - \alpha \cdot e_{max\,i} = 0 \qquad (A.8)$$

on $[\omega_1, \omega_{n+1}]$ for each i, where α can be chosen between 1 and $1/\chi_i$. We put $\alpha(\alpha > 1)$ in (A.8) is to make it more likely that (A.3) holds. Let $\bar{\omega}_i$ and ω_i^*, $i \in I$ respectively denote, if they exist, the smallest root of (A.8) and the smallest root of $f_i(\omega) = 0$ on $[\omega_1, \omega_{n+1}]$. Obviously, if $\alpha \cdot e_{max\,i} \geq e_{n\,i}$, $\bar{\omega}_i \leq \omega_i^*$ for each $i \in I$.

In our algorithm, we choose $n = 2$ and $\alpha = 2$. In detail, we interpolate f_i on $[\omega_1, \omega_3]$ by the quadratic polynomal, denoted by $p_{2\,i}^1$, such that $p_{2\,i}^1(\omega_j) = f_i(\omega_j)$, $j = 1, 2, 3$. Also we interpolate f_i on $[\omega_2, \omega_4]$ by the quadratic polynomal, $p_{2\,i}^2$, such that $p_{2\,i}^2(\omega_j) = f_i(\omega_j)$, $j = 2, 3, 4$. $p_{3\,i}$ is the cubic interpolating polynomial such that $p_{3\,i}(\omega_j) = f_i(\omega_j)$, $j = 1, 2, 3, 4$. It is easy to show

$$e_{max\,i} = \max_{\omega \in [\omega_1, \omega_3]} |p_{2\,i}^1(\omega) - p_{3\,i}(\omega)| = \max_{\omega \in [\omega_2, \omega_4]} |p_{2\,i}^2(\omega) - p_{3\,i}(\omega)|$$

$$= 0.06415\ldots \; |f_i(\omega_1) - 3\, f_i(\omega_2) + 3\, f_i(\omega_3) - f_i(\omega_4)| \,. \qquad (A.9)$$

Thus, the solution of (A.8) can be obtained analytically in terms of $f_i(\omega_1)$, $f_i(\omega_2)$, $f_i(\omega_3)$ and $f_i(\omega_4)$.

Now, we state fully the root finding algorithm. Its objective, if ω^* exists, is to obtain an approximation of ω^*, $\tilde{\omega} \leq \omega^*$. The algorithm has input parameters, β, μ, $\chi 1$ and $\chi 2$: β is the ratio of the initial value of h to $\omega_e - \omega_b$, $\mu > 0$ is the desired bound on $\omega^* - \tilde{\omega}$, $\chi 1$ and $\chi 2$ have already been described. The terminology *trisect* $[\omega_\alpha, \omega_\beta]$ means *set* $h = (\omega_\beta - \omega_\alpha)/3$ *and* $\omega_1 = \omega_\alpha$, $\omega_2 = \omega_1 + h$, $\omega_3 = \omega_2 + h$, $\omega_4 = \omega_3 + h = \omega_\beta$. $\bar{\omega}_i$ denotes the smallest root of $p_{2\,i}^1(\omega) - \alpha \cdot e_{max\,i} = 0$ on $[\omega_1, \omega_3]$ or the smallest root of $p_{2\,i}^2(\omega) - \alpha \cdot e_{max\,i} = 0$ on $[\omega_2, \omega_4]$

Root Finding Algorithm

Step 1 Set $h = \beta\,(\omega_e - \omega_b)$ and $\omega_1 = \omega_b$, $\omega_2 = \omega_1 + h$, $\omega_3 = \omega_2 + h$, $\omega_4 = \omega_3 + h$.

Step 2 If $\omega_1 \geq \omega_e$, stop and report Γ_R is empty.

Step 3 Compute $f_i(\omega_1)$, $i \in I$. If $\min_{i \in I} f_i(\omega_1) < 0$, stop and report error. If $\min_{i \in I} f_i(\omega_1) = 0$, stop with $\tilde{\omega} = \omega_1$.

Step 4 Compute $f_i(\omega_2)$, $i \in I$. If $\min_{i \in I} f_i(\omega_2) \leq 0$ and $\omega_2 - \omega_1 \leq \mu$, stop with $\tilde{\omega} = \omega_1$. If $\min_{i \in I} f_i(\omega_2) \leq 0$ and $\omega_2 - \omega_1 > \mu$, *trisect* $[\omega_1, \omega_2]$ and repeat Step 4.

Step 5 Compute $f_i(\omega_3)$, $i \in I$. If $\min_{i \in I} f_i(\omega_3) \leq 0$ and $\omega_3 - \omega_1 \leq \mu$, stop with $\tilde{\omega} = \omega_1$.

Step 6 Compute $f_i(\omega_4)$, $i \in I$. If $\min_{i \in I} f_i(\omega_4) \leq 0$ and $\omega_4 - \omega_1 \leq \mu$, stop with $\tilde{\omega} = \omega_1$.

Step 7 Compute $e_{max\,i}$, χ_i, $i \in I$ and set $\chi_{max} = \max_{i \in I} \chi_i$. If $\chi_{max} > \chi 1$, *trisect* $[\omega_1, \omega_2]$ and go to Step 4. If $\chi_{max} \leq \chi 1$, compute $\bar{\omega}_i$, $i \in I$ on $[\omega_1, \omega_3]$, if it exists, and obtain $\bar{\omega} = \min_{i \in I} \bar{\omega}_i$. If $\bar{\omega}$ does not exist, go to Step 8; if $\bar{\omega} < \omega_2$, *trisect* $[\bar{\omega}, \omega_2]$ and go to Step 3; if $\omega_2 \leq \bar{\omega} < \omega_3$, *trisect* $[\bar{\omega}, \omega_3]$ and go to Step 3.

Step 8 Compute $\bar{\omega}_i$, $i \in I$ on $[\omega_2, \omega_4]$, if it exists, and obtain $\bar{\omega} = \min_{i \in I} \bar{\omega}_i$. If $\bar{\omega}$ does not exist, go to Step 9; if $\bar{\omega} < \omega_3$, *trisect* $[\bar{\omega}, \omega_3]$ and go to Step 3; if $\omega_3 \leq \bar{\omega} < \omega_4$, *trisect* $[\bar{\omega}, \omega_4]$ and go to Step 3.

Step 9 If $\chi_{max} < \chi 2$, set $h = 3h$; otherwise, leave h unchanged. Set $\omega_1 = \omega_4$, $\omega_2 = \omega_1 + h$, $\omega_3 = \omega_2 + h$, $\omega_4 = \omega_3 + h$ and go to Step 2.

References

[1] R. O. Barr and E. G. Gilbert, "Some efficient algorithms for a class of abstract optimization problems arising in optimal control," *IEEE Trans. Automat. Contr.*, vol. AC-14, pp. 640–652, 1969.

[2] D. P. Dobkin and D. G. Kirkpatrick, "A linear algorithm for determining the separation of convex polyhedra," *Journal Algorithms*, vol. 6, pp. 381–392, 1985.

[3] E. G. Gilbert and D. W. Johnson, "Distance functions and their application to robot path planning in the presence of obstacles," *IEEE J. Robotics Automat.*, vol. RA-1, pp. 21–30, 1985.

[4] E. G. Gilbert, D. W. Johnson, and S. S. Keerthi, "A fast procedure for computing the distance between complex objects in three-dimensional space," *IEEE J. Robotics Automat.*, vol. RA-4, pp. 193–203, 1988.

[5] E. G. Gilbert and S. M. Hong, "A new algorithm for detecting the collision of moving objects," in *Proc. IEEE Conf. on Robotics and Automat.*, Scottsdale, AZ, pp. 8–14, May 1989.

[6] S. M. Hong, "New algorithms for detecting the collision of moving objects," Ph.D. Dissertation, Dept. Aerospace Engineering, Univ. of Michigan, 1989.

[7] S. Kawabe, A. Okano, and K. Shimada, "Collision detection among moving objects in simulation," in

Proc. 4th Int. Symp. on Robotics Research, pp. 489–496, MIT Press, 1988.

[8] T. Lozano-Pérez, "A simple motion-planning algorithm for general robot manipulators," *IEEE J. Robotics Automat.*, vol. RA-3, pp. 224–238, 1987.

[9] M. J. D. Powell, *Approximation Theory and Methods*. Cambridge, London: Cambridge Univ. Press, 1981.

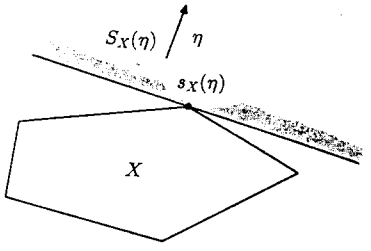

Figure 1. The support mapping and the supporting half space.

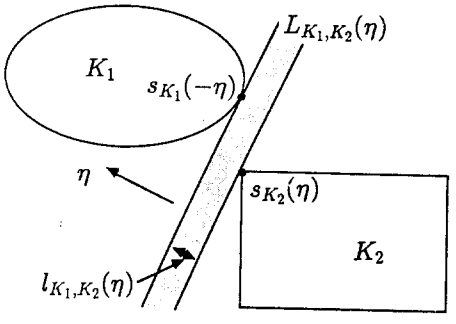

Figure 2. The separating slab with normal vector η.

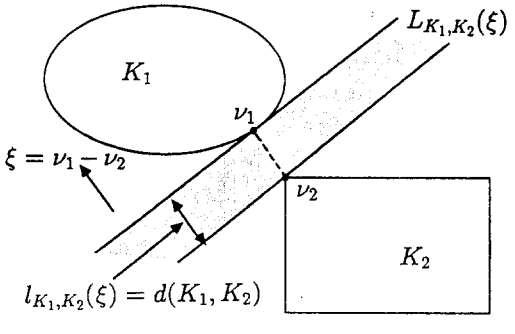

Figure 3. The separating slab generated by the near points ν_1 and

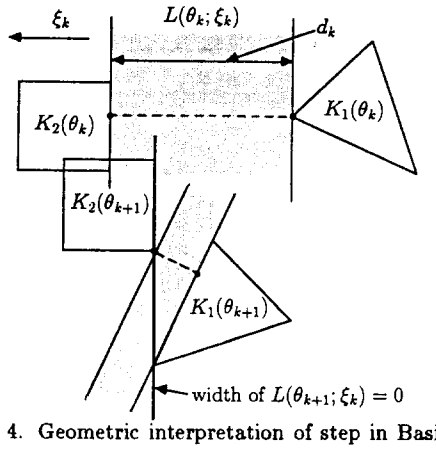

Figure 4. Geometric interpretation of step in Basic Algorithm for solving the CDP.

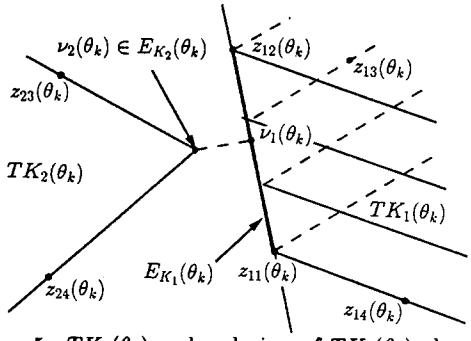

Figure 5. $TK_1(\theta_k)$ and end view of $TK_2(\theta_k)$ along the line connecting $z_{21}(\theta_k)$, $z_{22}(\theta_k)$.

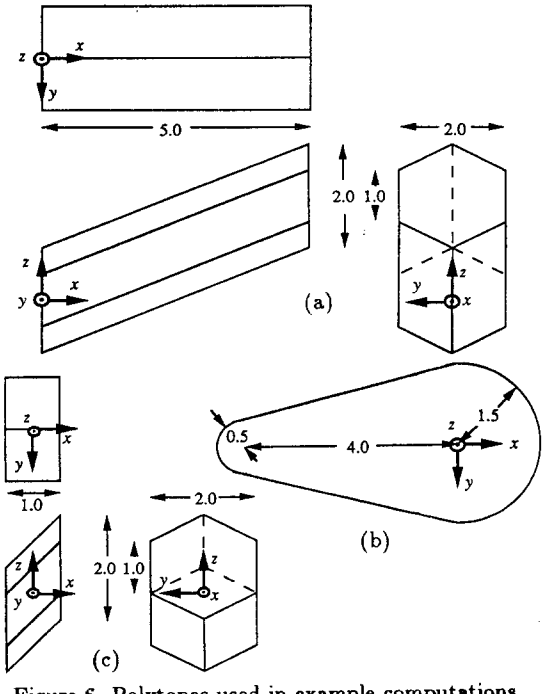

Figure 6. Polytopes used in example computations.