VELOCITY ALTERATION STRATEGY FOR COLLSION-FREE TRAJECTORY PLANNING OF TWO ROBOTS

Ming-Yi Ju[*,**]*, Jing-Sin Liu*[*] *and Kao-Shing Hwang*[**]

*Institute of Information Science,
Academia Sinica, Taipei, Taiwan, R.O.C.
Email:liu@iis.sinica.edu.tw

**Department of Electrical Engineering,
National Chung Cheng University, Chiayi, Taiwan, R.O.C.
Email:hwang@ee.ccu.edu.tw

ABSTRACT

Two robots working in a shared workspace can be programmed by planning the trajectory of each robot independently. To account for collision avoidance between them, a real-time velocity tuning strategy based on fast and accurate collision detection is proposed in this paper to determine the step of next motion of slave (low priority) robot for collision-free trajectory planning of two robots with priorities. The effectiveness of the method depends largely on a newly developed method of accurate estimate of distance between links. Under the control of the proposed strategy, the master robot always moves at a constant speed while the slave robot moves at the selected velocity, based on a tradeoff between collision trend index and velocity reduction in one servoing time, to keep moving as long as possible and as fast as possible while avoid possible collisions along the path. The collision trend index is a fusion of distance and relative velocity between links of two robots to reflect the possibility of collision at present and in the future. Graphic simulations of two PUMA560 robot arms working in common workspace but with independent goals are conducted to demonstrate the collision avoidance capability of the proposed approach as compared to the approach based on bounding volumes. It shows that a potential benefit of our approach is less number of speed alterations required to react to potential collisions.

1. INTRODUCTION

Flexible manufacturing systems increasingly catch manufacturers' eyes for recent years. In order to construct a reconfigurable and reprogrammable system, robot arms have been widely used in manufacturing processes to increase productivity, reduce production costs, and improve product quality. However, for some types of complex tasks, the capabilities of single robot arm are insufficient. The use of multiple robot arms in a common workspace is essential to enhance the utilization of robots and improve the versatility of potential applications. As a consequence, it usually leads to the problems of cooperation and collision avoidance in dynamically varying environment since the robot arm may become moving obstacles to each other. Therefore, motion planning must take account of collision avoidance between robot arms. In general, collision-free motion planning for multiple robots can be decomposed into two sub-problems: path planning and trajectory planning [3]. Path planning finds the robots' geometric paths that do not intersect static obstacle, and trajectory planning determines how fast each robot must move along its path to avoid collision with others. Partitioning as the above, the collision avoidance problem of multiple robots is simplified. In this paper, the proposed method focuses on dealing with the problem of collision-free trajectory planning for multiple robots. One of the major features of time/velocity adjusting approaches for trajectory planning is that the number of variables to be considered for collision avoidance does not exceed the number of robots because one variable, usually the time, is enough to express the moving velocity of each robot.

To solve the above problem, plenty of research efforts have been proposed. To premise that the end effectors of robots move along pre-described straight-line paths, Lee and Lee [1] presented a decoupled method for speed reduction/time delay of one robot while the other robot maintains its original trajectory based on space-time collision maps, and compared the effects of delay time and velocity reduction on total traveling time. In their implementation, only the wrist of robot arm is taken into consideration and is modeled simply as a sphere for collision detection. To expand Lee's approach, a simple time delay method based on collision map is proposed in [4] for avoiding collisions between two general robot arms. In his method, robot links are approximated by polyhedron and the danger of collision is expressed by the function of the distance between two robots. Similarly, Basta et al. [2] also proposed an approach mapping the potential collision segment information into the time domain to obtain the space-time collision for planning collision-free motion of two robot arms.

For minimum-time trajectory planning, [5] showed that under certain condition, a time delay in starting one robot on its path leads to motion time optimality with collision avoidance. By considering the limitation of actuator torque and velocity, [6] constructed a two-dimensional coordination space and a maximum velocity curve in terms of parameters defining the position along the paths of two robots to detect the collision region and to plan a

time-optimal velocity curve, respectively. Under the same consideration, Lee [7] converted trajectory planning problem and physical limitations into an optimization problem and applied dynamic programming approach to generate a near minimum-time trajectory for two robots. [8] presented an approach that the collisions between links of robots in 3D are simplified to the different rectangles, called forbidden region, in 2D Space/Time graphs. Through finding the optimal path between the rectangles on the Space/Time graph, robots can avoid collision by means of velocity alteration. All of above approaches consider the trajectory planning problem when both robots are moving along pre-assigned geometric paths.

The proposed approach models robots' links using polyhedra and then computes their enclosing and enclosed ellipsoids for minimum distance estimate and collision detection (details can be found in [13]). Based on the estimated distance for potential collision detection, a velocity alteration strategy is applied to collision-free trajectory planning of two robots. The outline of this paper is as follows. Section 2 describes the approaches for the construction of enclosing and enclosed ellipsoids of convex polyhedra. The method for distance estimates between polyhedra is conducted in Section 3. Collision detection based on distance estimate is introduced in Section 4. Section 5 presents the proposed velocity alteration strategy for trajectory planning of two robots and the simulation results are shown in Section 6. Finally, Section 7 concludes the paper.

2. ELLIPSOID GENERATION

How to model the shapes of robots' links is an important problem to an efficient collision detection algorithm. For a modeling method, it should represent the physical system precisely as possible and must be simple enough to ensure that the algorithms can be solved fast enough to secure the real-time operation of the manipulator. For the proposed algorithm, polyhedra are used to model the geometric objects, and their enclosing and enclosed ellipsoids cooperate with the polyhedral model for collision detection and rapidly distance estimate. The main advantage of the use of ellipsoids is that it is very simple in mathematical representation; therefore it can reduce the complexity of computations to be required. Besides, there is a fullblown approach to compute the closest points between two separate ellipsoids. An ellipsoid is represented as $\varepsilon^n(\mathbf{y},\mathbf{Y})$ in the rest of this paper, where *n* is the dimension, **y** is the center, and **Y** is the characteristic matrix.

The relationship of the enclosed ellipsoids, the enclosing ellipsoids and the polyhedral models are depicted in Fig. 1. If the closest points between two enclosed ellipsoids are computed, than a straight line equation can be generated based on the two points. The central idea of the proposed method is to rapidly estimate the closest points between polyhedral objects by means of computing the intersection points of the line equation with the polyhedra or enclosing ellipsoid. Based on all the intersection points, a tight distance estimate between two polyhedra can be derived [12]. Therefore, the approach starts from ellipsoids generation.

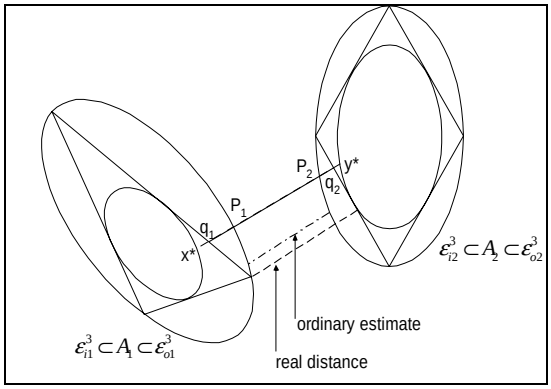

Fig. 1. The distance estimates based on enclosed ellipsoids

2.1 Enclosing Ellipsoid

Löwner-John (L-J) ellipsoid, the minimum-volume enclosing ellipsoid of a body, is an intuitively appealing means to lump the detailed geometry into a single quadratic surface. The computation of the L-J ellipsoid is a convex optimization problem [9] whose solution can be derived by applying the ellipsoid algorithm [10]. It is worth noticing that the L-J ellipsoid is the bounding volume representation of the convex polyhedron and it can be used for collision detection.

2.2 Enclosed Ellipsoid

The generation of an appropriate enclosed ellipsoid for a convex polyhedron is very difficult. Since an unfit enclosed ellipsoid may cause a large error of distance estimate in this proposed approach, a 3-phase approach is proposed in order to generate an enclosed ellipsoid that fits to the polyhedron for minimum distance estimate. Our approach is to derive the enclosed ellipsoid of a convex polyhedron by means of shrinking, stretching, and then scaling an L-J ellipsoid to fit the polyhedron as tight as possible.

Phase 1 – Isotropically shrinking all principal axes

An initial enclosed ellipsoid is given by shrinking the L-J ellipsoid along its principal axes isotropically to be contained in the polyhedron in phase 1. Let $\varepsilon^n(\mathbf{y},\mathbf{Y})$ be the minimum volume *n*-ellipsoid containing a convex polyhedron in *n*-dimensional space. Then, the initial enclosed ellipsoid is given as $\varepsilon^n(\mathbf{y},(n+1)^2\mathbf{Y})$, formed by shrinking $\varepsilon^n(\mathbf{y},\mathbf{Y})$ from its center by a factor of $(n+1)$, to guarantee that the polyhedron contains the initial ellipsoid [10]. Therefore, the ellipsoid $\varepsilon^3(\mathbf{y},16\mathbf{Y})$ is selected to be the initial guess for enclosed ellipsoid computation in 3-dimensional case. The regulation of the shrinking factor is based on the bisection method. The phase terminates with a user-defined error while the ellipsoid cannot extend further without overlapping with the facets of a polyhedron.

Phase 2 - Stretching

The phase 1 terminated while the enclosed ellipsoid is very close to one of the polyhedron's facets; however, it still has some free space to enlarge the enclosed ellipsoid. Stretching operation [11] is applied to expand the enclosed ellipsoid along a given direction in phase 2. Let **s** be the point to adapt to and $\varepsilon^3(\mathbf{c},\mathbf{M})$ be the enclosed ellipsoid generated in phase 1. The idea is to move the ellipsoid's center towards to the point, i.e. **s**, and then stretch the ellipsoid along the movement direction such that the old border point in the opposite direction remains a border point. Therefore the new center is represented as

$$\mathbf{c}' = \mathbf{c} + \beta(\mathbf{s} - \mathbf{c}),$$

where β determines how far to move the ellipsoid's center. With the normalized distance vector

$$\mathbf{a} = \mathbf{M}^{1/2}(\mathbf{s} - \mathbf{c}) / \| \mathbf{M}^{1/2}(\mathbf{s} - \mathbf{c}) \|,$$

the new transformation matrix is given as

$$\mathbf{M}'^{1/2} = (\mathbf{I} + (\alpha - 1)\mathbf{a}\mathbf{a}^T)\mathbf{M}^{1/2},$$

where $\alpha = 1/(1+ \| \beta \mathbf{M}^{1/2}(\mathbf{s} - \mathbf{c}) \|)$.

It is worth to notice that enlarging an ellipsoid means that its transformation matrix makes the vectors shorter, therefore α is always smaller than 1. In the stretching operation, **s** is given as $l \cdot (\mathbf{s}_m - \mathbf{c}) / \| \mathbf{s}_m - \mathbf{c} \|$, where *l* is the distance from the farthest facet of the polyhedron to **c**, the center of enclosed ellipsoid, and $\mathbf{s}_m$ is the mass center of vetices of the farthest facet. In our implementation, β is initialized as 1 and inside the range from 0 to 1. The selection of β is also based on the bisection method. The algorithm terminates while the variation of β is smaller than 0.005.

As mentioned the old border point in the opposite of the stretching direction is still a border point of the new ellipsoid, it implies that perhaps there is free space for the ellipsoid to expand in the opposite side. Therefore, the stretching operation is applied once again for possibly enlarging the ellipsoid. In order to hold the interface point between the facet of a polyhedron and the ellipsoid, the new **s**, which needs to be adapted to, is given as

$$\mathbf{s} = l \cdot (\mathbf{c} - \mathbf{s}_i) / \| \mathbf{c} - \mathbf{s}_i \|,$$

where $\mathbf{s}_i$ is the interface point.

Phase 3 – One by one enlarging each radius

Let $\varepsilon^3(\mathbf{c}',\mathbf{M}')$ be the enclosed ellipsoid generated by means of stretching. Since the matrix $\mathbf{M}'$ is symmetric and positive-definite, it can be diagonalized through a rotational matrix **V**. The relation is expressed as

$$\mathbf{D} = \mathbf{V}^{-1}\mathbf{M}'\mathbf{V}.$$

In fact, matrix **V** is the matrix of eigenvectors of matrix $\mathbf{M}'$ and matrix **D** is the canonical form of $\mathbf{M}'$, a diagonal matrix with $\mathbf{M}'$'s eigenvalues on the main diagonal. Since the inverse square roots of matrix $\mathbf{M}'$'s eigenvalues are equivalent to the length of principal axes of the enclosing ellipsoid, the change of ellipsoid's each radius can be performed individually by means of multiplying matrix **D** with a scaling matrix **S**, which is also diagonal. Therefore, each new radius of the enclosed ellipsoid can be written as $\mathbf{D}' = \mathbf{SD}$; and the enlarged enclosed ellipsoid can be represented as

$$\mathbf{M}'' = \mathbf{V}\mathbf{D}'\mathbf{V}^{-1}.$$

By the use of scaling operations, the length of each principal axis of the enclosed ellipsoid is extended individually until the enlarging process induces the ellipsoid to intersect with facets of the polyhedron.

Since $\varepsilon^3(\mathbf{c}',\mathbf{M}'')$ is generated by stretching along a specified vector and, then, enlarging some axes of $\varepsilon^3(\mathbf{c},\mathbf{M})$ generated in phase 1, the following relationship $\varepsilon^3(\mathbf{c},\mathbf{M}) \subseteq \varepsilon^3(\mathbf{c}',\mathbf{M}'')$ always holds.

3. DISTANCE ESTIMATE

For some applications such as path planning in a tight workspace, an inaccurate or too conservative distance estimate may result in failing to find a solution even though it exists. Therefore, the minimum distance estimate between objects is very important.

3.1 Lower Bound

In general, the minimum distance between the bounding volumes, i.e. the enclosing ellipsoids ε_{o1}^3 and ε_{o2}^3, is set as the lower bound of distance estimate and is used for collision detection. However, due to representation error induced from the difference between a real polyhedron and the ellipsoid model, the intersection points $\mathbf{P}_1$ and $\mathbf{P}_2$, at which the shortest path between the enclosed ellipsoids, ε_{i1}^3 and ε_{i1}^3, intersects with the enclosing L-J ellipsoids, are more suitable points for lower bound distance estimate geometrically. According to [9], the closest points of two separately enclosed ellipsoids $\varepsilon_{i1}^3(\mathbf{a},\mathbf{A})$ and $\varepsilon_{i2}^3(\mathbf{b},\mathbf{B})$ can be computed as:

$$\mathbf{x}^* = \lambda_{\min}(\mathbf{M})[\lambda_{\min}(\mathbf{M})\mathbf{A} - \mathbf{I}]^{-1}\mathbf{A}\mathbf{a},$$

where

$$\mathbf{M} = \begin{bmatrix} \mathbf{A}^{-1} & -\mathbf{I} \\ -\mathbf{A}^{-1/2}\mathbf{a}(\mathbf{A}^{-1/2}\mathbf{a})^T & \mathbf{A}^{-1} \end{bmatrix},$$

$\lambda_{\min}(\mathbf{M})$ is its eigenvalue with minimal real part and $\mathbf{x}^* \in \varepsilon_{i1}^3(\mathbf{a},\mathbf{A})$. Once $\mathbf{x}^*$ is obtained, $\mathbf{y}^* \in \varepsilon_{i2}^3(\mathbf{b},\mathbf{B})$ can be derived in the same way. Therefore, the intersection points $\mathbf{P}_1$ and $\mathbf{P}_2$ of $L(\mathbf{x}^*,\mathbf{y}^*)$, the straight line connecting the closest points $\mathbf{x}^*$ and $\mathbf{y}^*$ on the enclosed ellipsoids, with the L-J ellipsoids $\varepsilon^3(\mathbf{a},\mathbf{A})$ and $\varepsilon^3(\mathbf{b},\mathbf{B})$ are computed respectively based on the coordinate transformations

$\bar{\mathbf{x}}^* = \mathbf{A}^{1/2}(\mathbf{x}^* - \mathbf{a})$ and $\bar{\mathbf{y}}^* = \mathbf{A}^{1/2}(\mathbf{y}^* - \mathbf{a})$.

Then, the problem becomes how to compute the intersection point of a unit ball $\mathbf{B}_O$ centered at the origin and a line $L(\bar{\mathbf{x}}^*, \bar{\mathbf{y}}^*)$. The points on the line $L(\bar{\mathbf{x}}^*, \bar{\mathbf{y}}^*)$ can be described as vector *v* with a parameter *t*:

$\mathbf{v}(t) = (\bar{\mathbf{y}}^* - \bar{\mathbf{x}}^*) \cdot t + \bar{\mathbf{x}}^*$.

The intersection with the unit ball $\mathbf{B}_O$ occurs when

$|\mathbf{v}(t)| = 1$ or $|\mathbf{a}_n|^2 t^2 + |\bar{\mathbf{x}}^*|^2 + 2(\mathbf{a}_n \bullet \bar{\mathbf{x}}^*)t = 1$, where

$\mathbf{a}_n = \bar{\mathbf{y}}^* - \bar{\mathbf{x}}^*$,

and the solution

$$t' = \frac{\sqrt{(\mathbf{a}_n \bullet \bar{\mathbf{x}}^*)^2 - |\mathbf{a}_n|^2 (|\bar{\mathbf{x}}^*|^2 - 1)} - (\mathbf{a}_n \bullet \bar{\mathbf{x}}^*)}{|\mathbf{a}_n|^2}.$$

The intersection point is found as $\mathbf{P}_1 = \mathbf{a} + \mathbf{A}^{-1/2}\mathbf{v}(t')$.

The other intersection point $\mathbf{P}_2$ also can be obtained by the same way. It is worth to notice that $0 < t' < 1$ while the two enclosing L-J ellipsoids are apart. If a L-J ellipsoid intersects with the enclosed ellipsoid of the other polyhedron, t' will be larger than 1.

3.2 Upper Bound

It is intuitive to set the minimum distance between the two enclosed ellipsoids as the upper bound. However, this kind of upper bound still can be improved by taking the polyhedral facets information into consideration. Let the *i*th face of a polyhedron be represented by a plane equation $\mathbf{a}_{ni} \bullet \mathbf{x} = k_i$. Suppose the polyhedron intersects with the line $L(x^*, y^*)$, whose points are described by

$\mathbf{v}(t) = (\mathbf{y}^* - \mathbf{x}^*) \cdot t + \mathbf{x}^*$,

at the point **q**, the intersection point **q** can be found algebraically by solving the minimum and positive *t*, or $t_{\min}$, subject to $\mathbf{a}_{ni} \bullet \mathbf{v}(t) = k_i$. Therefore, the intersection points $\mathbf{q}_1$ and $\mathbf{q}_2$ of both polyhedra (Fig. 1) can be computed respectively. These two points are close to the closest points of the enclosed ellipsoids between the polyhedra. The upper bound of distance estimate is thus apparently improved by replacing the distance between two enclosed ellipsoids with $\overline{\mathbf{q}_1\mathbf{q}_2}$.

3.3 Distance Estimate Error

For one of the polyhedra and its L-J ellipsoid, the distance estimate error is thus computed by $\overline{\mathbf{P}_i\mathbf{q}_i}$. The error varies with the sizes of the ellipsoids, and the orientations and shapes of the polyhedra.

4. COLLISION DETECTION

For collision detection problems, in general, only the bounding volume, e.g. the enclosing ellipsoid, is applied to performing intersection check. However, due to the limit of representation model's precision, such strategies may cause a lot of false alarms. To overcome such kind of problem and to avoid expensive computational expense, a hierarchical strategy based on the proposed distance estimate method is introduced. If all the enclosing ellipsoids are enough apart, the lower bound of the distance estimate between two polyhedra is larger than zero and collision free is guaranteed. The further collision detection needs only be performed when the enclosing ellipsoids intersect with others. In this way, the proposed approach can be used to efficient localize collisions in space.

In the proposed method, the geometrical order of the closest points, which locate on the enclosing and the enclosed ellipsoids of the two polyhedra for distance estimate, is used to tell the intersection of enclosing ellipsoids. A "correct" geometrical order of the set of closet points can be checked easily by using $sign(\overrightarrow{\mathbf{P}_1\mathbf{P}_2} \bullet \overrightarrow{\mathbf{q}_1\mathbf{q}_2})$, the sign of the inner product $\overrightarrow{\mathbf{P}_1\mathbf{P}_2} \bullet \overrightarrow{\mathbf{q}_1\mathbf{q}_2}$. For two enough separate polyhedra, $sign(\overrightarrow{\mathbf{P}_1\mathbf{P}_2} \bullet \overrightarrow{\mathbf{q}_1\mathbf{q}_2})$ always larger than zero. If it is equal to zero, the two enclosing ellipsoid collide at one point. If the result is small than zero, it implies that the geometrical order is violated and there are intersections among these ellipsoids. Therefore, further information is needed to classify the potential collision.

Since the representation of ellipsoid model is not the same as the original polyhedron, a heuristic safe margin is added to guarantee no collision condition is lost in collision detection. The distance estimate error, i.e. $\overline{\mathbf{P}_1\mathbf{q}_1}$ or $\overline{\mathbf{P}_2\mathbf{q}_2}$, are set as the safe margin for the proposed algorithm. Since $\mathbf{q}_1$ and $\mathbf{q}_2$ are derived from the polyhedra and their enclosed ellipsoids, they are not the real closest points between the two polyhedra. In fact, $\overline{\mathbf{q}_1\mathbf{q}_2}$ is larger than the minimal distance between the two polyhedra. Therefore, if the distance, i.e. $\overline{\mathbf{q}_1\mathbf{q}_2}$, between two polyhedra is smaller than $\overline{\mathbf{P}_1\mathbf{q}_1}$ or $\overline{\mathbf{P}_2\mathbf{q}_2}$, the two polyhedra may be in the situation of potential collision. Inspired from the above idea, the criterion for collision detection is therefore given as: if $\overline{\mathbf{q}_1\mathbf{q}_2} > \min(\overline{\mathbf{P}_1\mathbf{q}_1}, \overline{\mathbf{P}_2\mathbf{q}_2})$ is satisfied, it is categorized as collision-free; otherwise, it is judged as potential collision. Since the detection of potential collision is based on the distance estimate error, larger distance estimate error leads to more false warnings for collision detection. It is undesirable for path planning problem in a workspace that is cluttered with obstacles. Hence, an artificial threshold is given to overcome this drawback. The criterion for collision detection is thus replaced by

$\overline{\mathbf{q}_1\mathbf{q}_2} > \min(\overline{\mathbf{P}_1\mathbf{q}_1}, \overline{\mathbf{P}_2\mathbf{q}_2}, \text{threshold})$.

5. VELOCITY ALTERATION FOR OLLISION_FREE TRAJECTORY PLANNING

Instead of constructing the configuration space and planning a path in joint domain for collision-free trajectory planning

problem, the proposed method implements collision detection directly in the spatial domain. Through the efficient and accurate distance estimate, the regions where potential collisions occur are clearly specified. Besides, unlike most of the proposed approaches of collision avoidance for robot trajectory planning based on the computation of distance function only at each servoing time [15], the proposed approach also takes the velocity information between links into consideration. For collision detection/avoidance, only the minimum distance estimate is not enough to faithfully reflect the practical situation in a dynamic environment for robotic applications. For example, as shown in Fig. 2, object A has the same distances to object B and to object C; and object A is apart from object B but is heading to object C. If only the closest distance is used for collision avoidance, such two cases have the same imminences. However, the latter, in fact, is more urgent than former. Therefore, more information about the motion of objects such as moving speed or moving direction should also be applied to faithfully reflecting dynamic situation for collision prediction.

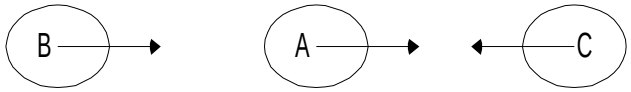

Fig. 2. The right pair is more imminence than the left pair.

Without loss of generality, we first assume that their motions are equivalent to the motions of their mass centers. Then an index to differentiate the objects are approaching or departing to each other is design as

$I(A,B) = v_B \bullet \frac{\overrightarrow{BA}}{|\overrightarrow{BA}|}$, where A is controllable object and B is obstacle.

In fact, the index I is the projection of obstacle's velocity v_B on the unit vector from the obstacle B to the controllable object A. It is noticed that if the obstacle is approaching the controllable object, I is positive; otherwise, I is zero or negative. The index indicates not only the relation of the motion direction but also the magnitude of the relation between moving objects. Since only the approach case will induce an imminent danger, therefore, the combination of the minimum distance and the index to exhibit the practical situation in dynamic environment can be represented as

$$P(A,B) = \max(I(A,B),0) * \frac{1}{1+d_{A,B}},$$

where $d_{A,B}$ is the minimum distance between objects A and B. According to the above equation, if the objects are departure, $P(\cdot)$ is equal to zero. In contrast, if the objects are close and approach to the other one at high speed, than a larger P will be generated. By embedding the information of motion direction and the speed magnitude, the minimum distance can be more correct to reflect the practical situation which is imminent or not in a time-varying environment for many robotic applications.

In the similar way, the collision-trend index P also can be applied to trajectory planning for robot manipulators. Consider two robot arms work in a share workspace with independent goals. In this case, the master robot arm plays a role of moving obstacle to the slave one. Therefore, collision avoidance is required to prevent each link from colliding with the others. The most widely used traditional technique for collision avoidance for two robot systems is semaphore method in which one robot waits before running next path segment till other robot moves out completely from the dangerous region. It is not efficient because of not providing the parallel tasking feature. Therefore, by the use of speed alteration strategy to allow the two robots move concurrently, the main aim of this work is to minimize the delay-time induced form collision avoidance.

In our case, it is assumed that the two robot arms' tips move along their independently pre-planned geometric paths in Cartesian space at predefined speeds. Therefore, the motion of the master robot arm at a constant speed V_{master_tip} can be described as:

$T_{master} = | p_{master\_goal} - p_{master\_initial} | / V_{master\_tip}$,

$N_s = T_{master} / \Delta T$, and

$\Delta p_{master} = (p_{master\_gola} - p_{master\_initial}) / Ns$,

where T_{master} is the total elapsed time of the master robot; $p_{master\_initial}$ and p_{master_goal} are the initial and goal positions of the master robot's tip; N_s is the number of servoing times; ΔT is the time duration of a servoing time; and Δp_{master} is the average tip position variation at a servoing time. Similarly, the motion of the slave robot can be described in the same way.

When a possible collision is detected in the next servoing time, the slave robot is required to slow down its tip's velocity. It is performed by searching the optimal position, which minimizes a given objective function, of the possible positions along the path within the location range where the predefined tip speed can reach in one servoing time without any collision. By considering each link of the master robot as moving obstacle, the collision-trend index between the master robot's link i and the slave robot's link j can be also represented as

$I(i,j) = v_i \bullet \frac{\overrightarrow{link_i link_j}}{|\overrightarrow{link_i link_j}|}$, and

$$P(i,j) = \max(I(i,j),0) * \frac{1}{1+d_{i,j}},$$

where v_i is the end-point velocity of the master robot's link i with respect to the world coordinate. The object function based on the collision-trend index for searching an optimal position is given as

$$f_{obj} = w_1 * \max\{P(i, j),\ i = 1...n \text{ and } j = 1...m\} + w_2 * \frac{V_{tip} - V(t)}{V_{tip}} \quad , \qquad (1)$$

where w_1 and w_2 are weighting factors; V_{tip} is the preprogrammed tip velocity; $V(t) = (p(t) - p(t-1))/\Delta T$ is the selected tip velocity for the slave robot; t is the servoing time index; $p(\cdot)$ is the selected tip position of the slave robot; ΔT is the duration of a servoing time; and n and m are the number of links. The first and second terms at the right hand side of Eq. (1) is designed to force the links to keep a safety distance away from each other while simultaneously demand the slave robot to move at the preprogrammed velocity, respectively.

The search algorithm utilizes the objective function to evaluate the possible positions on the moving path of the slave robot at each servoing time to select the optimal position for the slave robot. Fig. 3 shows the next possible position of the two robots.

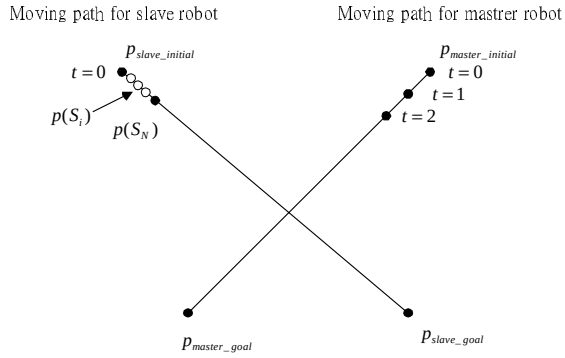

Fig. 3. The moving path of the robots at one servoing time.

Hence, the next possible position of the slave robot's tip can be represented as follows:

$$p_{slave\_tip}(t) = p_{slave\_tip}(t-1) + \Delta p_{slave} \cdot S_i / S_N ,$$

where S_i is the sampling index and S_N is the sampling number/maximum sampling index for a servoing time, and Δp_{slave} is the average tip position variation during a servoing time.

6. SIMULATION EXAMPLE

In order to demonstrate the performance of proposed distance estimate method and velocity alteration strategy for collision-free trajectory planning, two RUMA560 robot arms working in an overlapped working envelope are set up. It is assumed that the two robots stand away from each other so that their link 1 and link 2 may not collide with their counterparts. Therefore, only link 3 to link 6 are involved in collision detection. The parameters for ellipsoids generation for PUMA560 are given in [14]. The base coordinates for the two PUMA560 arms with respect to the world coordinate are

$$B_{master} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \text{ and}$$

$$B_{slave} = \begin{bmatrix} -1 & 0 & 0 & 1080mm \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

The initial and goal positions for the two robots are given as follows:

$$p_{master\_initial} = \begin{bmatrix} 1 & 0 & 0 & 790 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 660.4 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$p_{master\_goal} = \begin{bmatrix} 1 & 0 & 0 & 290 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 660.4 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \text{ and}$$

$$p_{slave\_initial} = \begin{bmatrix} 1 & 0 & 0 & 540 \\ 0 & 1 & 0 & -300 \\ 0 & 0 & 1 & 660.4 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$p_{slave\_goal} = \begin{bmatrix} 1 & 0 & 0 & 540 \\ 0 & 1 & 0 & 200 \\ 0 & 0 & 1 & 660.4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

with respect to their individual base coordinates, respectively.

Since the motion of each joint affects the displacement of all subsequent links, the maximum displacement for each link in Cartesian space depends on both the maximum total distance from a point on the link to the base joint and the maximum angular displacement of all the links. Therefore, the approximated maximum tip's displacement for a PUMA560 can be given as [4]:

$$d = l\sqrt{2((1-\cos\theta_1) + (1-\cos(\theta_2 + \theta_3 + \theta_5)))} ,$$

where l is the distance from joint 1 to tip. In this example, the servoing time is given as 20*ms* and the maximum angular displacement is assumed to be 0.5 *rad/sec*. Based on the constraints, the maximum displacement of the tip of PUMA560 for each servoing time is computed as 60*mm* for a planar motion. The predefined tip velocity for the PUMA560 arms to move along the preplanned geometric paths is selected as 50*mm* for each servoing time in this example. However, collisions between the links will occur if both robot arms move along the given trajectories. The collisions between the two robot arms are shown in Fig. 4(left: 3D display, right: top view).

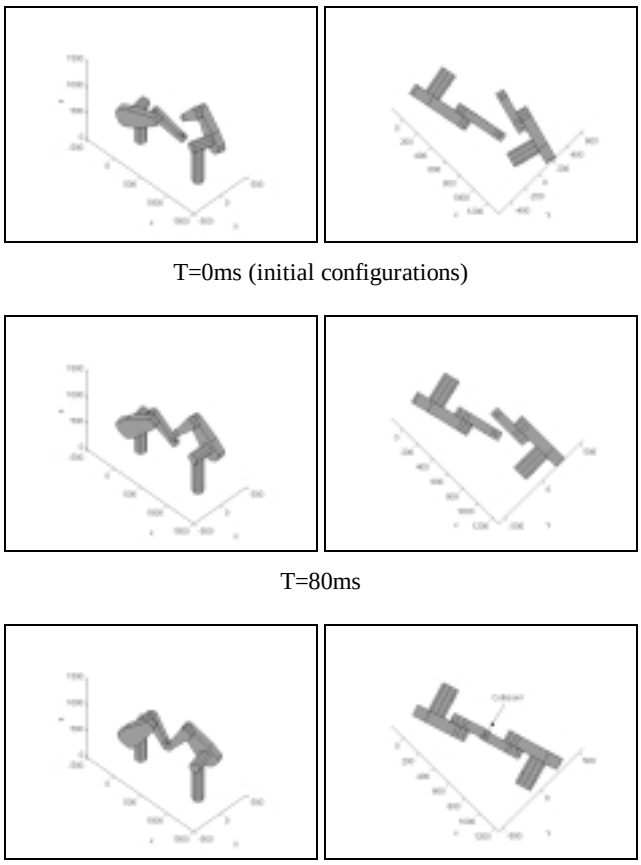

Fig. 4. Two robot arms move along the pre-planned trajectories.

Next, the proposed velocity alteration strategy is applying to tuning the slave robot's velocity to avoid this undesired collision situation without any change of the geometric path. The master robot's tip moves at the same preprogrammed speed as in the previous experiment unless the alteration of the slave robot's speed fails to avoid the occurrence of collision. To decide an alternative velocity for the tip of the slave robot, the objective function Eq. (1) with $w_1 = 50$ and $w_2 = 1$ is used in this simulation example. The sampling number S_N for a servoing time is given as 20. Graphic display of configurations of two robots for several time instants and the resultant tip velocity for the slave robot arm are shown in Fig. 5 and 6, respectively. For comparison study, applying the same speed alteration strategy, the tip velocity generated by using ellipsoids as bounding volume only for distance estimate is also given in Fig. 7. The result shows that the conservative estimate of minimum distance induces more delays of arrival time due to earlier alarm and later relief of potential collisions. Another result generated by the widely used traditional approaches is also depicted in Fig. 8. The traditional approaches stop the slave robot if a collision is detected; otherwise, drive the slave robot at the maximum velocity. However, in transition state, this control strategy will induce undesired jerks for deriving robot arms. In contract, the proposed method provides a smooth transition in velocity profile for robot arm manipulation.

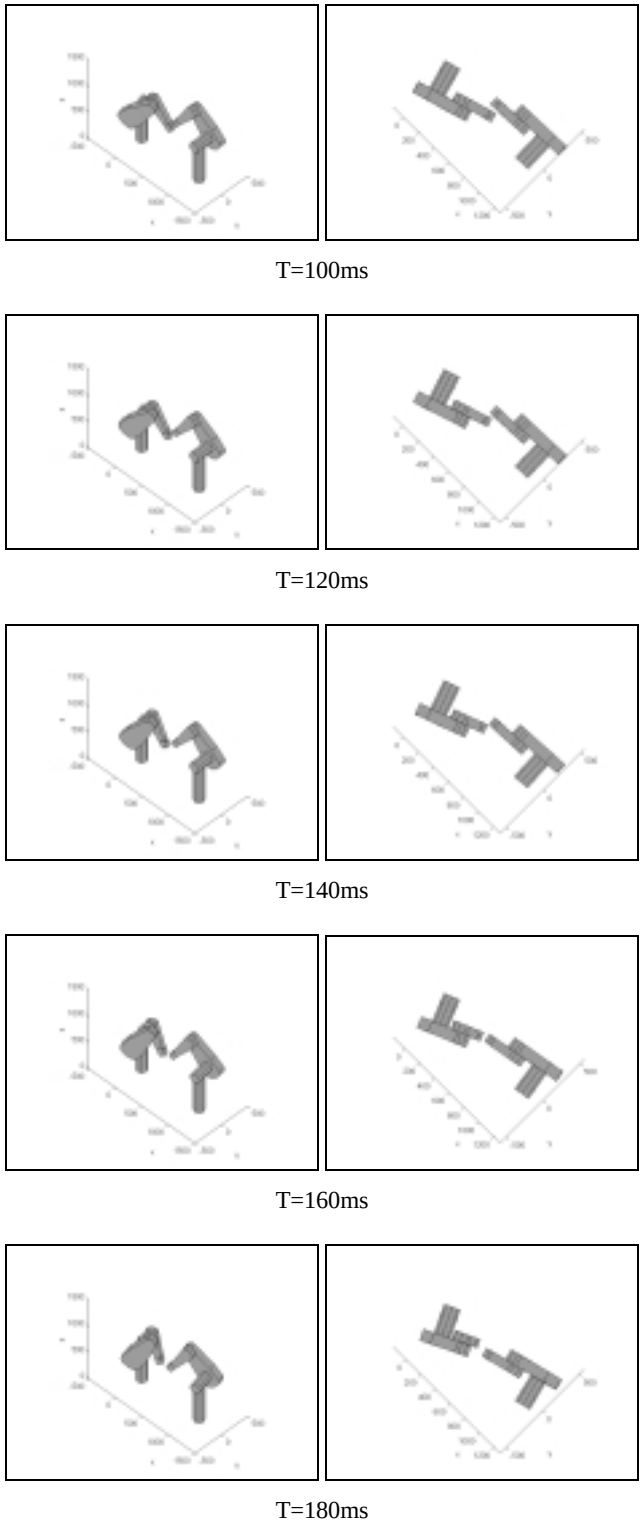

Fig. 5. Two robot arms move without collision along their pre-planned trajectories by velocity alteration strategy.

7. CONCLUSION

The paper has presented a systematic approach for velocity alteration strategy based on fast and accurate collision detection to solve the collision-free trajectory planning problem for multi-robot systems. The minimum distance

between robots' links can be estimated simply and efficiently by using their enclosing and enclosed ellipsoids. Based on the estimated minimum distance for collision detection and avoidance, and the related velocities between links, the proposed speed alteration strategy is able to guide the slave robot arm, with less number of alterations, to move along its predefined geometric path without any collision with the other one. Instead of constructing the whole collision regions in C-space or in collision maps for trajectory planning, the proposed strategy only checks up collision along a part of collision region. Therefore, it is much simpler and then the overall computation can be drastically reduced.

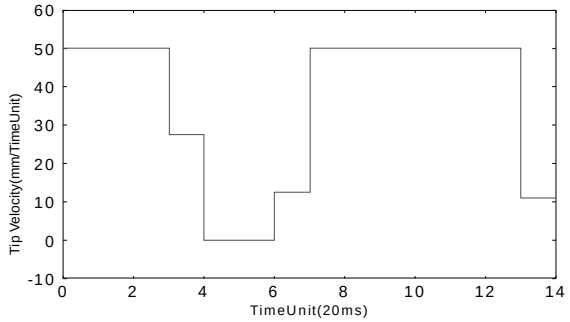

Fig. 6. Tip velocity controlled by velocity alteration strategy.

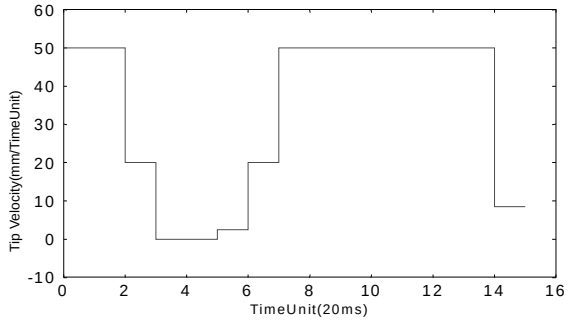

Fig. 7. Tip velocity controlled by velocity alteration strategy using bounding volume for collision detection.

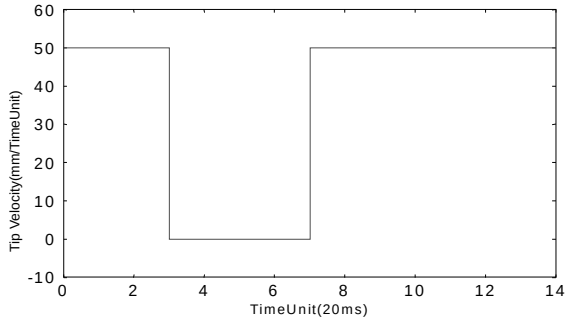

Fig. 8. Tip velocity generated by bang-bang control.

8. REFERENCES

[1] B. H. Lee and C. S. G. Lee, "Collision-free motion planning of two robots," IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-17, No. 1, pp. 21-32, 1987.

[2] R. A. Basta, R. Mehrotra, and M. R. Varanasi, "Collision detection for planning collision-free motion of two robot arms," in Proceedings of the IEEE International Conference on Robotics and Automation, pp. 638-640, April 1988.

[3] K. Kant and S. W. Zucker, "Toward efficient trajectory planning: the path-velocity decomposition," International Journal of Robotics Research, Vol. 5, No. 3, pp. 72-89, 1986.

[4] C. Chang, M. J. Chung, and B. H. Lee, "Collision avoidance of two general robot manipulators by minimum delay time," IEEE Transactions on Systems, Man, and Cybernetics, Vol. 24, No. 3, pp. 517-522, 1994.

[5] K. G. Shin and Q. Zheng, "Minimum-time collision-free trajectory planning for dual-robot systems," IEEE Transactions on Systems, Man, and Cybernetics, Vol. 8, No. 5, pp. 641-644, 1992.

[6] Z. Bien and J. Lee, "A minimum-time trajectory planning method for two robots," IEEE Transactions on Robotics and Automation, Vol. 8, No. 3, pp. 414-418, 1992.

[7] J. Lee, "A dynamic programming approach to near minimum-time trajectory planning for two robots," IEEE Transactions on Robotics and Automation, Vol. 11, No. 1, pp. 160-164, 1995.

[8] K. S. Hwang and H. J. Chao, "Collision-avoidance motion planning admit multiple moving objects", Journal of Information Science and Engineering, Vol. 15, pp.715-736, 1999.

[9] E. Rimon and S. P. Boyd, "Obstacle collision detection using best ellipsoid fit," Journal of Intelligent and Robotic System, Vol. 18, pp. 105–126, 1997.

[10] M. Grotschel, L. Lovasz, and A. Schrijver, Geometric Algorithms and Combinatorial Optimization, 2nd corrected ed., Springer-Verlag, Berlin, 1993.

[11] T. Brychcy and M. Kinder, " A neural network inspired architecture for robot motion planning," in Proceedings of the International Conference on Engineering Applications of Artificial Neural Networks (EANN'95), pp. 103-109, Helsinki, Finland, Aug. 1995.

[12] S. P. Shiang, J. S. Liu, and Y. R. Chien, "Estimate of minimum distance between convex polyhedra based on enclosed ellipsoids," in Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, 2000.

[13] M. Y. Ju, J. S. Liu, S. P. Shiang, Y. R. Chien, K. S. Hwang, and W. C. Lee, "Fast and accurate collision detection based on enclosed ellipsoid," in Proceedings of the International Conference on Control, Automation, Robotics and Vision, 2000.

[14] M. Y. Ju, J. S. Liu, and K. S. Hwang, Ellipsoid Modeling for Articulated Robot Manipulators for Interactive Motion Planning, Tech. Report TR-IIS-00-008, Academia Sinica.

[15] K. S. Hwang and M. D. Tsai, "On-line collision-avoidance trajectory planning of two planar robots based on geometric modeling," Journal of Information Science and Engineering, Vol. 15, No. 5, pp. 131-152, 1999.