[7] G. L. Ritter, H. B. Woodruff, S. R. Lowry, and T. L. Isenhour, "An algorithm for a selective nearest neighbor decision Rule," *IEEE Trans. Information Theory*, vol. IT-21, no. 6, pp. 665–669, Nov. 1975.
[8] I. Tomek, "Two modifications of CNN," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-6, no. 11, pp. 769–772, Nov. 1976.
[9] M. Ichino, "A nonparametric multiclass pattern classifier," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-9, no. 6, pp. 345–352, June 1979.
[10] K. C. Gowda and G. Krishna, "The condensed nearest neighbor rule using the concept of mutual nearest neighborhood," *IEEE Trans. Information Theory*, vol. IT-25, no. 4, pp. 488–490, July 1979.
[11] K. Fukunaga and J. M. Mantock, "Nonparametric data reduction," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. PAMI-6, no. 1, pp. 115–118, Jan. 1984.
[12] Y. T. Chien, *Interactive Pattern Recognition*. New York: Marcel Dekker, Inc., pp. 223–230, 1978.
[13] B. V. Dasarathy and E. B. Holder, "Image characterizations based on joint gray level-run length distributions," *Pattern Recog. Lett.*, vol. 12, no. 8, pp. 497–502, Aug. 1991.
[14] F. Rhodes, "Some characterizations of the chessboard metric and the city block metric," *Pattern Recog. Lett.*, vol. 11, no. 10, pp. 669–675, Oct. 1990.
[15] B. V. Dasarathy, "Fuzzy learning under and about an unfamiliar fuzzy teacher," in *NAFIPS'92, Proc. North American Fuzzy Information Processing Society Conf.*, pp. 368–377, Dec. 1992.

Collision Avoidance of Two General Robot Manipulators by Minimum Delay Time

Cheol Chang, Myung Jin Chung, and Bum Hee Lee

***Abstract*—A simple time delay method for avoiding collisions between two general robot arms is proposed. Links of the robots are approximated by polyhedra and the danger of collision between two robots is expressed by distance functions defined between the robots. The collision map scheme, which can describe collisions between two robots effectively, is adopted. The minimum delay time value needed for collision avoidance is obtained by a simple procedure of following the boundary contour of collision region on collision map. To demonstrate the effectiveness of the proposed time delay method, a computer simulation study is shown where a collision is likely to occur realistically.**

I. Introduction

Industrial robots have made a significant contribution toward automating the manufacturing processes. The efficient use of robots shows productivity increase, production cost reduction, and product quality improvement. However, most robots currently in use perform simple repetitive jobs, such as pick-and-place, machine loading and unloading, spray painting, and spot welding.

Only one robot in a common work space limits the classes of tasks that can be performed. Two or more robots in a work space can improve potential application area of robots. Multiple robots can be used to accomplish a task where each performs its own subtask in parallel, and to save the production time. Also, multiple robots can accomplish complex tasks that can not be performed by a single robot such as transporting an object beyond the payload capability of a single robot. Among the previous applications of multiple robots, the parallel tasking feature is only an example especially in industry aiming to mass production. However, at present, the parallel tasking is not fully utilized since there is no practical methodology that can make several robots operate safely in a common workspace. In the case that more than one robot operate simultaneously in a common workspace, the problem of avoiding potential collisions between the robots should be considered very carefully.

To solve the collision avoidance problem, zone-blocking methods have been proposed. In these methods only one robot operates at a time. So, this semaphore mechanism is not efficient because of not providing the parallel tasking feature. Besides zone-blocking methods some collision avoidance methods [1]–[5] have been proposed for multiple robots. These methods can be divided into two categories: 1) time adjusting methods while maintaining the given geometric path and 2) trajectory modification methods which modifies given geometric path. The former adjusts the time evolution representing the moving speed of robots while the geometrical paths of the robots are fixed. The robot path, which guarantees a robot not to collide with stationary obstacles, can be obtained using some existing methods [6]–[10]. One of the major features of time adjusting approaches 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 speed for each robot. For instance, in the case of two robots, at most two variables are needed for solving the collision avoidance problem. This fact suggests that a collision avoidance problem in multiple robots can be easily solved comparing with a collision avoidance problem for a single robot and stationary obstacles which requires at least three variables in a three dimensional work space.

Lee *et al.* [2] presented several time adjusting methods for two robots using a collision map. In their paper, only a wrist of each robot is treated as a possible collision obstacle and modeled by a sphere. A collision map is used to describe potential collisions between two robots efficiently under the condition that given geometrical paths for two robots are fixed. However, the collisions in three-dimensional work space are transformed into collision region(s) in the map and this transformation usually requires an excessive computational effort. Furthermore, the collision region(s) in the map is approximated by box(es) to determine the departure time of one robot. Therefore, this approximation of collision region(s) results in unnecessary extra time delay.

This paper proposes an effective collision avoidance method for two general robot manipulators which are approximated by polyhedra as an extension of Lee *et al.* The proposed method determines the minimum time delay needed for avoiding collisions between two general robot manipulators using distance functions.

Basically the computational scheme for obtaining the delay time adopts the concept of the collision map which represents the region corresponding to collisions between two robots. To obtain the collision-free minimum delay time, a scheme which follows the boundary contour of the collision region in a collision map is proposed. This scheme only checks a part of the boundary contour of the collision region and the TLVST (traveling length versus sampling time) curve conceptually. Due to the computational simplicity, the overall procedure is very simple. Also, this method of avoiding collisions through time delay is relatively easy to implement, because the geometrical paths and time evolution of two robot manipulators

Manuscript received January 9, 1992; revised April 6, 1993.
C. Chang is with the Samsung Advanced Institute of Technology, Suwon, Kyung Ki-do, Korea.
M. J. Chung is with the Department of Electrical Engineering, Korea Advanced Institute of Science and Technology, Taejon 305–701, Korea.
B. H. Lee is with the Department of Control and Instrumentation Engineering, Seoul National University, Seoul, Korea.
IEEE Log Number 9214588.

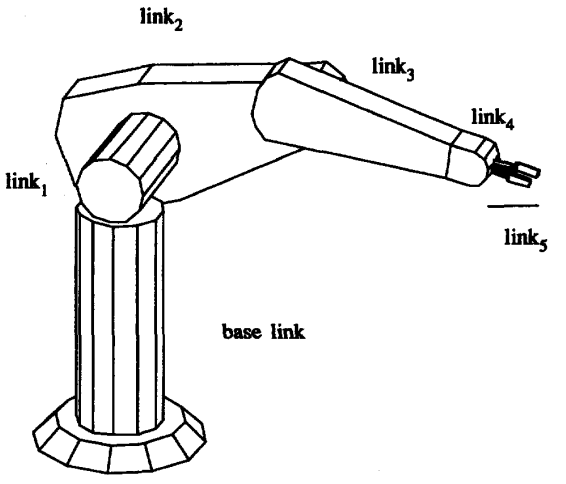


Fig. 1. A PUMA-560 robot approximated by polyhedra.

are not altered at all. Above all, this simple computation characteristic is preserved in any types of robots and modeling schemes.

The structure of this paper is as follows. Section II describes the collision detection between two robots mathematically. For this objective the modeling and its use in the representation of danger of collision between 2–6 degrees-of-freedom PUMA arms are discussed. In section III a boundary contour following method, which provides the minimum delay time for collision avoidance between two robots is presented. A simulation study considering a realistic situation is presented in Section IV to show the effectiveness of the proposed method. Section V draws some conclusions.

II. Collision Detection and Avoidance

A. Distance Measure

In order to detect collision between two robots, the distance among links must be computed and compared every instance of time. To do this, the robot manipulators must be properly modeled. There are many representation schemes for robot arms. Here, we should consider the accuracy of the model as well as the computational burden simultaneously. To compromise these requirements, a convex polyhedra is selected as a model primitive. Each link of a robot is approximated by a convex polyhedron. The gripper is a little more complicated, but it is also approximated by a set of polyhedra. If the robot(s) picks up an object the model of the gripper must be properly changed using the above scheme. As an example, a PUMA 560 robot arm, an articulated manipulator with six degrees of freedom, is modeled by seven convex polyhedra as shown in Fig. 1.

Collision detection between two polyhedra can be done by computing the distance between the polyhedra. Given two objects A and B in Cartesian space, they can be represented by convex polyhedra K_A and K_B, respectively. And, the space occupied by the objects A and B is discussed by sets of points in K_A and K_B respectively. The distance $d(K_A, K_B)$ between K_A and K_B is defined by the closest two points in K_A and K_B as follows:

$$d(K_A, K_B) \triangleq \min\{|x_1 - x_2| : x_1 \in K_A, x_2 \in K_B\} \quad (1)$$

Several algorithms have been proposed to compute the distance $d(K_A, K_B)$ in (1). Among them, a fast algorithm by Gilbert *et al.* [11] is adopted to detect collision between two robots in this paper.

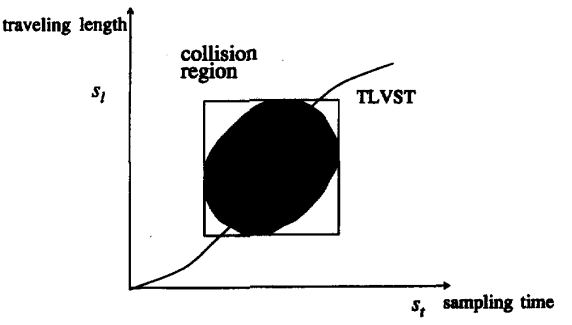


Fig. 2. A collision map and its rectangular approximation.

We assume that there are no collisions between the base links and other links, since it is very rare to place two robots in a workspace so close as collisions may occur. Accordingly the following 25 distance measures $d(link_1^{(1)}, link_1^{(2)})$, $d(link_1^{(1)}, link_2^{(2)}), \cdots, d(link_5^{(5)}, link_5^{(5)})$ are used to detect collisions between two 6 dof robot arms. In the expression of $link_j^{(k)}$, the superscript k indicates the number of robot arms. For convenience, the distance $d(link_i^{(1)}, link_j^{(2)})$ is denoted by d_{ij}. Then, the distance $d(R_1, R_2)$ between two robots R_1 and R_2 is defined by

$$d(R_1, R_2) = \min\{d_{ij} : i, j = 1, \cdots, 5\}. \quad (2)$$

If the distance $d(R_1, R_2)$ is less than zero, there are collisions between two robots [10]. Otherwise, no collision occurs.

B. Collision Map and Collision Detection

A collision avoidance scheme for two robot arms depends on not only the path information but the trajectory information of the robots. In order to incorporate the location and the corresponding time information of the robots into a two-dimensional space simultaneously, a collision map concept is appropriate for our purpose. If a robot R_1 adheres to its original trajectory, the other robot R_2 must change appropriately its original trajectory for collision avoidance. Then a curve which relates the traveled length with the corresponding time instant of robot R_2 can be drawn. Two robot arms have a potential collision under the original trajectory information, if there is a range of collision lengths where the path of robot R_2 is within the colliding range of a point on the path of robot R_1. Then the union of these collision lengths at the collection of time instants can be drawn as a connected region. The detailed construction procedure of the collision map is presented in [2].

As mentioned earlier, the construction procedure requires complex computation and unnecessary approximation for obtaining the collision map. To simplify this procedure we propose a boundary contour following method which is conceptually following the boundary of the collision region until the departure point is determined. A discretized collision map describing the collision between two robot arms R_1 and R_2 is shown in Fig. 2. In the map, s_t on the horizontal axis is the sampling index for time and s_l on the vertical axis is the Cartesian traveling length of the gripper of R_2 at the sampling instant. The real time t corresponding to s_t is the sampling time interval Δt times s_t, and the real traveling length l of R_2 is the sampling length interval Δl times s_l. The TLVST curve represents the traveling length of the gripper of R_2 along the time and it corresponds to the trajectory of R_2. If the TLVST curve is changed, the trajectory of R_2 is also changed. But the trajectory of R_1 is not changed at all. That is, any change on the TLVST curve does not affect the trajectory of R_1.

A collision between R_1 and R_2 occurs at the position in Cartesian space corresponding to any points (s_t, s_l) in the collision region of the collision map. If the TLVST curve passes through the collision region, there are collisions between R_1 and R_2.

The possibility of a collision at a point (s_t, s_l) in the collision map can be examined by computing the distance $d(R_1, R_2)$ between R_1 and R_2 at the point in Cartesian space. Here, we note that the collision map is discretized. Therefore, if the distance at the point (s_t, s_l) is larger than zero, the distance $d(R_1, R_2)$ in the following region must be also larger than zero for collision avoidance. For this purpose, the area of each link can be replaced by the area where the link sweeps out when the corresponding joint moves within its allowable range in the intervals $[s_t - 0.5, s_t + 0.5]$ and $[s_l - 0.5, s_l + 0.5]$ for R_1 and R_2 respectively.

$$\begin{aligned}(s_t - 0.5)\cdot\Delta t \le t \le (s_t + 0.5)\cdot\Delta t \\ (s_l - 0.5)\cdot\Delta l \le l \le (s_l + 0.5)\cdot\Delta l\end{aligned} \tag{3}$$

However, instead of computing the exact swept volume, which is very hard to compute, we use a simple approximation method. First, it is required to compute the upper bound on the largest displacement of any point on the kth link in response to the joint displacement within the allowable angle ε_k for one sampling interval Δt. Denote this bound δ_k. The allowable angle, ε_k, can be estimated through the maximum joint angular velocity of the kth link. Let ω_k be the maximum angular velocity of the kth link; then the following inequality equation holds:

$$\varepsilon_k \le \Delta t \cdot \omega_k \tag{4}$$

Therefore, the upper bound of allowable angle can be approximated by $\Delta t \cdot \omega_k$. This procedure is applied to both robot arms. If the kth link is expanded by the corresponding upper bound δ_k, (3) holds always in the swept area.

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. In the case of a planar revolute manipulator the angular displacement of the kth link is the sum of the angular displacements of all the previous joints. Given distance d and angle θ, the magnitude of the displacement is $d\sqrt{2(1-\cos\theta)}$.

Let ε_i be the allowable range for the ith joint and r_i be the maximum distance of a point on the ith link from the ith joint; and let l_i be the distance from the jth joint to the $(j+1)$th joint. Then, the value of δ_k of the k link is

$$\delta_k = \left[\left(\sum_{i=1}^{k-1} l_i\right) + r_k\right] \cdot \sqrt{2(1 - \cos(\sum_{j=1}^{k} \varepsilon_j))} \tag{5}$$

However, in the case of a 3-dimensional articulated robot such as a PUMA arm, the computation of the bound δ_k slightly different from the planar robot case. In what follows, denote d_k as the distance of the end-point on the kth link from the origin of the base coordinates when the robot stretches out. The procedure for approximated δ_k for a PUMA arm is as follows:

The upper bound δ_1 of the first link is

$$\delta_1 = d_1\sqrt{2(1-\cos\varepsilon_1)}. \tag{6}$$

Since the rotational axis of joint 2 is perpendicular to that of joint 1, the displacement of any point on link 2 is also perpendicular to that of link 1. Therefore, δ_2 is

$$\begin{aligned}\delta_2 &= \sqrt{d_2^2(1-\cos\varepsilon_1) + d_2^2(1-\cos\varepsilon_2)} \\ &= d_2\sqrt{2((1-\cos\varepsilon_1) + (1-\cos\varepsilon_2))}\end{aligned} \tag{7}$$

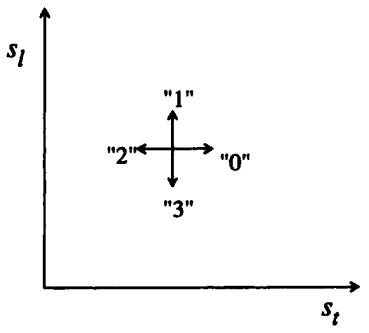


Fig. 3. Four directions of the automaton M.

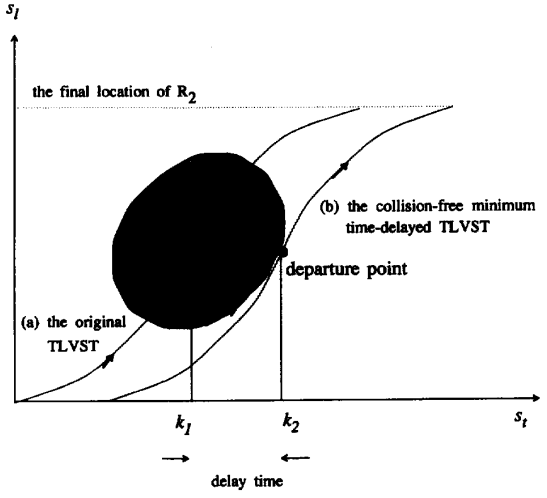


Fig. 4. Simplified description of computing a minimum delay time. (a) The original TLVST. (b) The collision-free minimum time-delayed TLVST.

Similarly, δ_3, δ_4, and δ_5 are given by

$$\delta_3 = d_3\sqrt{2((1-\cos\varepsilon_1) + (1-\cos(\varepsilon_2+\varepsilon_3))} \tag{8}$$

$$\delta_4 = d_4\sqrt{2((1-\cos\varepsilon_1) + (1-\cos(\varepsilon_2+\varepsilon_3))} \tag{9}$$

$$\delta_5 = d_5\sqrt{2((1-\cos\varepsilon_1) + (1-\cos(\varepsilon_2+\varepsilon_3+\varepsilon_5))}. \tag{10}$$

In (10), the distance d_5 includes the length of the gripper as well as link 5.

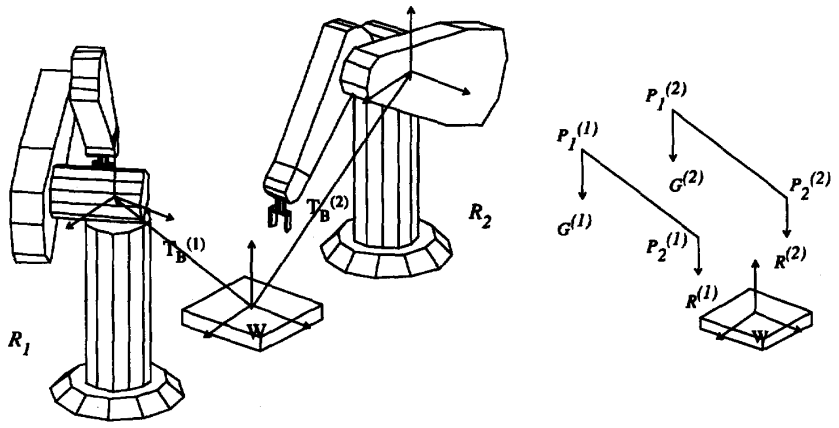
Considering the expanded links, the following inequality equation must hold in order that there is no collision between R_1 and R_2:

$$d(R_1, R_2) > \max\{\delta_i^{(1)} + \delta_j^{(2)} : i, j = 1, \cdots, 5\} \tag{11}$$

where $\delta_i^{(1)}$ indicates the upper bound δ_i of the ith link of robot R_1.

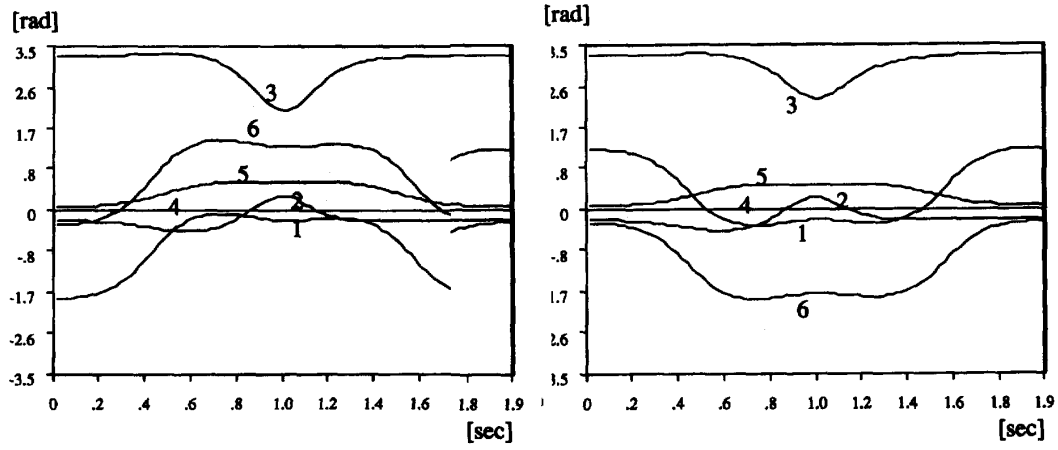
III. Collision Avoidance

In [2], the collision region between the end effectors of two robots is computed under the assumption that each robot follows a straight line trajectory. Then, the collision region is approximated by a box as shown in Fig. 2. However, as mentioned in Section I, in more general type robots and trajectories, it is not easy to compute the whole collision region generally because there is no analytical method to compute the collision region. In extreme case, we must



(a) (b)

Fig. 5. Workcell consisting of two robots, R_1 and R_2, and its given task. (a) Workcell. (b) The given task.



(a) (b)

Fig. 6. (a) Initial joint trajectories of R_1. (b) Initial joint trajectories of R_2.

examine the collision at every point to obtain the whole collision region in the collision map. This task is tedious and requires too expensive computational cost. Also, the rectangle approximation for the obtained collision region may raise extra delay time more than needed.

In this section, we introduce a boundary contour following method for computing the minimum delay time of R_2 for collision avoidance between R_1 and R_2, which is applicable to various robots approximated by a set of polyhedron. It is basically assumed that a trajectory for each robot is given and that the given trajectory meets the dynamical constraints and any robot does not collide with any stationary obstacles in work space.

Suppose that there is an automaton M which moves one of four directions as shown in Fig. 3 in the collision map. When the automaton M is moving along the boundary contour of the collision region, all directions are allowed. The directions themselves have no physical meaning but these directions are needed for computing the minimum delay time, i.e., the direction "0" makes a point at (s_t, s_l) toward $(s_t + \Delta t, s_l)$. Similarly the direction "1" makes the point at (s_t, s_l) toward $(s_t, s_l + \Delta l)$. For constructing a collision-free TLVST curve the directions must be either "0" or "1" because the traveling length and time are not retrogressive. However, "2" and "3" must be considered when M follows the boundary contour of the collision region.

A moving direction of M is determined in the following way. As a candidate for the moving direction, "0" direction is selected. If there is a collision at that direction, the very next direction in clockwise with respect to the previous direction of M is selected. This procedure continues until a collision-free direction is found. The key problem in the computation of delay time is to determine a departure point. A condition to be a departure point on the boundary is that its corresponding direction is either "0" or "1". When a point satisfies this condition, the automaton M moves along the delayed TLVST curve until it arrives at the final point or a new collisions is detected. The departure point denoted by "dp" in Fig. 4 is a point on the boundary of the collision region where M is allowed to proceed to the final desired point along the time-delayed TLVST curve without any collision. If two or more points satisfy the condition while M travels to the final point from the initial point, the last departure point is a real one.

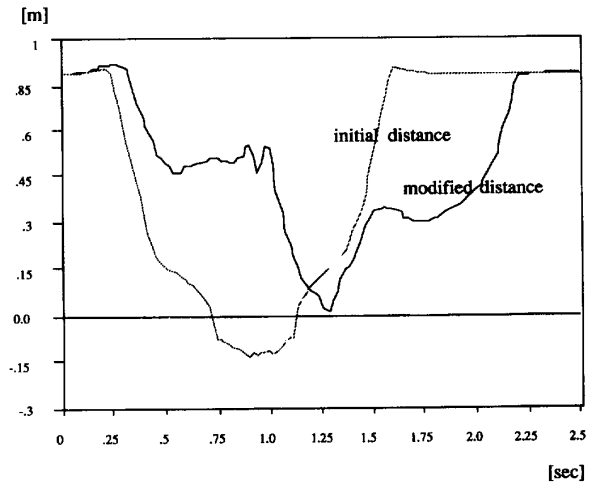


Fig. 7. Distance trajectory between R_1 and R_2.

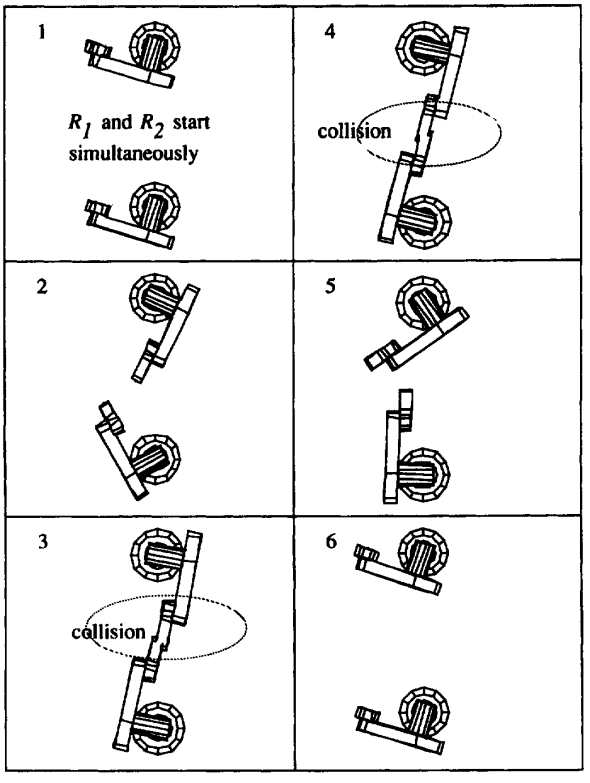


Fig. 8. Graphic simulation of collision situation.

When M moves along a time-delayed TLVST curve, M may be wandering in some area depending on the shape of collision regions. This wandering problem can be resolved by moving M along the time-delayed TLVST curve when a new candidate for a departure point, which is not listed before, is found. Although the current point satisfies the condition, if it is the one that has found before, M moves along the boundary contour of the collision region because the point

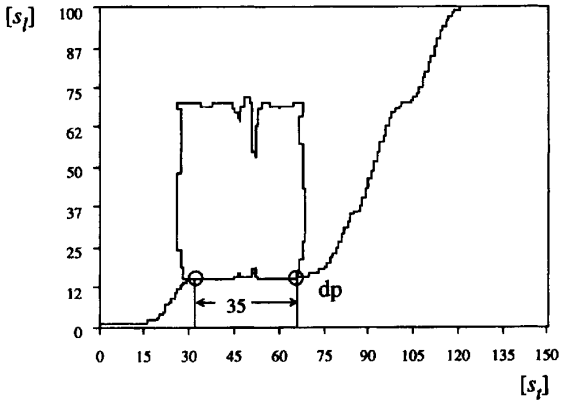


Fig. 9. The minimum delay time and departure point (dp).

is not a true departure point. The delay time is computed using the difference $k_2 - k_1$ in s_t between the original and the collision-free minimum time-delayed TLVST curves as shown in Fig. 4.

Although it is rare, there is a possibility that more than one collision regions may exist in the collision map. In this case, it is required to reexamine the collisions from the new initial point in the delayed TLVST curve to the departure point. If any collision is detected between the new initial point and the departure point, the boundary following procedure is repeated until there is no collision between the initial point and the departure point.

The procedure of computing the minimum value of the delay time is summarized as follows: Step 1: In the collision map, the automaton M proceeds with examining a collision using the distance measure from the origin of the map along the original TLVST curve until any collision is detected. Step 2: If a collision is detected, M proceeds to the departure point using an automaton. Step 3: If the departure point is found, M proceeds with examining a collision along a time-delayed TLVST curve until any collision is detected or until M arrives at the point corresponding to the final location of R_2. If the time-delayed TLVST curve arrives the final location of R_2, this curve becomes the collision-free time-delayed TLVST. Otherwise go to Step 2. Step 4: Compute the minimum delay time TD by $(k_2 - k_1) \cdot \Delta t$.

IV. Simulation Study on the Time Delay Method

In this section, we will show the effectiveness of the proposed collision avoidance scheme based on time delay and discuss the simulation results. Assume that the workcell consists of two PUMA-type robots, R_1 and R_2, and a worktable as shown in Fig. 5(a).

In Fig. 5, W, $T_B^{(1)}$, and $T_B^{(2)}$ are the world coordinates and the base coordinates of R_1 and R_2 with respect to the world coordinates respectively. Let $T_B^{(1)}$ and $T_B^{(2)}$ be given by

$$T_B^{(1)} = \begin{bmatrix} -1 & 0 & 0 & 0.65[m] \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad T_B^{(2)} = \begin{bmatrix} 1 & 0 & 0 & 0.65[m] \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{12}$$

A task for simulation is shown in Fig. 5(b). The description of the task is as follows: 1) Initially the end-effector of R_i is located at $P_1^{(i)}$. 2) It approaches $G^{(i)}$ and grasps an object. 3) It moves to $R^{(i)}$ along straight lines through via points $P_1^{(i)}$ and $P_2^{(i)}$. 4) It releases the object on the table. 5) It moves to $P_1^{(i)}$ along straight lines via $P_2^{(i)}$.

The joint trajectories for robots corresponding to this task are shown in Fig. 6, and these trajectories are assumed to meet the physical constraints. The dotted line in Fig. 7 shows the distance

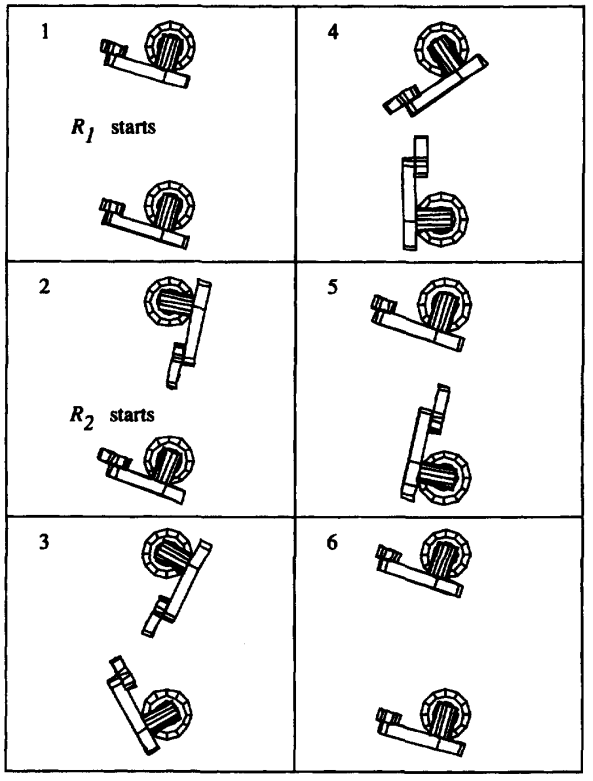


Fig. 10. Graphic simulation of collision avoidance by minimum delay time.

trajectory between R_1 and R_2 along time from 0 sec to 1.92 sec. It is shown that the distance between the time interval [0.76, 1.16] is less than zero. Here, the negative distance represents the estimated degree of the collision between R_1 and R_2 [11]. Consequently, collisions occurs between R_1 and R_2 as shown in Fig. 8.

The collision region of the given task is shown in Fig. 9. Graphical description of this collision region in the collision map is not required to determine the minimum delay time. However, to explain the concept and procedure, we obtained it through complex computation. The sampling time is 20 ms, and the traveling length of R_2 is uniformly divided. The maximum angular velocity of joints is assumed to be 0.5 rad/sec. In this case, the maximum angular displacement of joints within one sampling time is 0.01 rad. The distance parameters d of links are

$$d_1 = 0.2, \quad d_2 = 0.457, \quad d_3 = 0.85, \quad d_4 = 0.88, \quad d_5 = 1.0. \tag{13}$$

Using (6)–(10), δ_i are

$$\delta_1 = 0.2, \quad \delta_2 = 0.65, \quad \delta_3 = 1.9, \quad \delta_4 = 2.0, \quad \delta_5 = 3.0. \tag{14}$$

Also, Fig. 9 shows the delay time by the proposed scheme. In this example, the delay time is 0.7 sec, and the computation for obtaining the delay time requires about 90 sec on a SPARC workstation. The distance trajectory between R_1 and R_2, when R_2 is delayed by 0.7 sec, is shown as a solid line in Fig. 7. As shown in Fig. 7, the distance is always greater than zero through the whole execution time interval [0, 2.6]. Fig. 10 shows the collision-free movements of two robot arms graphically when a delay time is considered.

V. Conclusion

A simple and efficient time delay method for avoiding collisions between two general robot manipulators was proposed. Each robot was approximated by a set of convex polyhedra and the degree of collision between robots was represented by distance between two robots.

The proposed time delay method adopted the collision map scheme and the minimum delay time was determined by following the boundary contour of the collision region in the collision map. This method only checks up collisions along a part of the boundary contour of the collision region for a given TLVST curve. Due to the simplicity, the overall computation procedure becomes very simple and is applicable to any types of robots and modelings.

References

[1] E. Freund and H. Hoyer, "Path finding in multi-robot systems including obstacle avoidance," *Int. J. Robotics Res.*, vol. 7, no. 1, pp. 42–70, Feb. 1988.

[2] B. H. Lee and C. 5. G. Lee, "Collision-free motion planning of two robots," *IEEE Trans. Syst., Man, Cybern.*, vol. 17, no. 1, pp. 21–32, Jan.–Feb. 1987.

[3] C. Chang and M. 1. Chung, "A collision-free motion planning for two articulated robot arms using minimum distance functions," *Robotica*, vol. 8, pp. 137–143, 1990.

[4] B. H. Lee, "Constraint identification in time-varying obstacle avoidance for mechanical manipulators," *IEEE Trans. Syst., Man, Cybern.*, vol. 19, no. 1, pp. 140–143, Jan.–Feb. 1989.

[5] Z. Bien and I. Lee, "A minimum-time trajectory planning method for two robots," *IEEE Trans. Robot., Automat.*, vol. 8, no. 1, pp. 414–118, Jun. 1992.

[6] T. Lozano-Perez, "A simple motion-planning algorithm for general robot manipulators," *IEEE J. Robot., Automat.*, vol. 3, no. 3, Jun. 1987.

[7] W. E. Red and H. V. Truong-Cao, "Configuration maps for robot path planning in two dimensions," *Trans. ASME, Dynamic Syst., Measur., Contr.*, vol. 107, pp. 292–298, Dec. 1985.

[8] V. I. Lumelsky, "Dynamic path planning for a planar articulated robot arm moving amidst unknown obstacles," *IEEE J. Robot., Automat.*, vol. 1, pp. 21–30, Mar. 1985.

[9] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *Int. J. Robotics Res.*, vol. 5, pp. 90–98, 1986.

[10] I. S. Singh and M.D. Wagh, "Robot path planning using intersecting convex shapes : Analysis and Simulation," *IEEE J. Robot. Automat.*, vol. 3, no. 2, pp. 101–108, Apr. 1987.

[11] 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. Robot. Automat.*, vol. 4, no. 2, pp. 193–203, Apr. 1988.