Collision-Free Motion Planning of Two Robots

B. H. LEE, MEMBER, IEEE, AND C. S. G. LEE, SENIOR MEMBER, IEEE

Abstract—**An approach to collision-free motion planning of two moving robots in a common workspace is presented. Each robot is represented by a sphere containing the wrist and the manipulator hand. The results from a strictly straight line trajectory planning method are utilized for planning a path avoiding potential collisions. Due to the distinct nature of the potential collisions between the two moving robots, a new classification of path requirement situations is presented and utilized for planning a collision-free path. Notions of collision map and time scheduling are developed and applied for realizing a collision-free motion planning. A procedure is developed for the time scheduling of the straight line trajectory. An example is shown for the time scheduling of the trajectory, which shows the significance of the proposed approach in collision-free motion planning of the two moving robots.**

I. INTRODUCTION

MOTION PLANNING is composed of path planning and trajectory planning of the robot system. Collision-free motion planning is achieved through collision-free path planning schemes. Most of the path planning schemes concern the problem of avoiding fixed and stationary obstacles in a workspace. Due to the fixed and stationary obstacles, the path planning problem is usually converted to a geometric analysis problem for obtaining a collision-free path.

Ahuja *et al.* [1] discussed two methods for detecting intersections of three-dimensional objects. These methods are based on the projection checking and the octree traversing of stationary obstacles. Chien *et al.* [7] adopted the concepts of state space and developed a rotation mapping graph technique to describe the relationship between the positions and the corresponding collision-free orientations of a robot among obstacles. To simplify the collision avoidance problem, Udupa [25] used a polyhedra model for the obstacles and minimum bounding cylinders in detecting the interference and avoiding collisions among three-dimensional stationary objects. He discretized the three-dimensional space into sectoroids and pascs which were designated free if not occupied by obstacles and robots. A collision-free path is then obtained as lists of free pascs joined together. Lozano-Perez [16], [17] and Brooks [4], [5] extended the polyhedra models and reported systematic approaches with successful simulation results.

Luh *et al.* [18], [19] used the concept of pseudo-obstacles by expanding the real obstacles' edges and faces and viewing the robot as a point located in space. They formulated the forbidden region for the Stanford arm as a union set of polygonal obstacles and pseudo-obstacles. An algorithm was developed to determine the shortest collision-free path given a sequence of edges to be traversed. Pennington *et al.* [22] employed a geometric modeling approach to model a PUMA robot as a sphere, where collisions are detected based on the swept volume of the robot. However, the collision detection was found quite involved in mathematical computation. Petrov *et al.* [23] proposed a learning algorithm by considering the obstacles in the joint space and in the Cartesian space simultaneously and showed a hybrid computer implementation of the algorithm for collision avoidance. Recently, Gilbert *et al.* [10] described a procedure where a collision-free optimal path for a robot is obtained by solving an optimal control problem where the distances between the potentially colliding parts of the robot and the obstacles impose a state space constraint. The state constraints and actuator constraints are handled numerically by a penalty function approach. Khatib [13] proposed a control scheme which is based on the dynamic model of a manipulator in the task space. He used an artificial field and described the obstacles as the repulsive surfaces for the manipulator parts.

It is also notable that there are several experimental works on obstacle avoidance. Gouzenes [11] discussed collision avoidance of manipulators in a flexible assembly cell, using graph-search techniques and Petri nets. Myers *et al.* [20] used a fast static collision check for detecting potential collisions with obstacles. They developed a heuristic method to determine a collision-free path in a reasonable amount of time and demonstrated an application on a VAX-11/780 computer and a microcomputer. In particular, the collision avoidance problem associated with two manipulators in a common workspace can be found in Freund's work [8], [9]. Even though the collision avoidance problem in time-varying environment should be considered in terms of collision-free path and trajectory planning, he developed a useful algorithm for collision

Manuscript received September 15, 1985; revised September 2, 1986. This work was supported in part by the Research Initiation Grant of the SME Manufacturing Engineering Education Program and by the National Science Foundation under Grant ECS-8106954.

B. H. Lee was with the School of Electrical Engineering, Purdue University, West Lafayette, IN 47907, USA. He is now with Department of Control and Instrumentation Engineering, Seoul National University, Seoul, Korea.

C. S. G. Lee is with the School of Electrical Engineering, Purdue University, West Lafayette, IN 47907, USA.

IEEE Log Number 8611575.

0018-9472/87/0200-0021$01.00 ©1987 IEEE

avoidance with a computer simulation, which is based on a path planning scheme.

Most of the collision avoidance schemes mentioned emphasize a single robot with a fixed environment of stationary obstacles and achieve various degrees of success. In this paper, we focus on collision-free motion planning, i.e., path and trajectory planning, for two moving robots in a common workspace. We represent each robot by a single sphere at the wrist. It is assumed that no collisions occur at the initial and final locations of the two robots.

In Section II, problem formulation for collision-free motion planning of two robots is presented. Several assumptions on geometric modeling and straight line trajectory planning are also discussed. Different path requirement situations are identified and classified which will be useful for the proposed approach. Then notions of collision map and time scheduling are introduced, and their applications are discussed in Section III. Various collision-free motion planning approaches are proposed in Section IV, where a procedure is developed for obtaining collision-free trajectories. An illustrated example is presented using the proposed approach, and its result is shown in Section V. Finally, the conclusion is summarized in Section VI.

II. Problem Formulation

In the terminology of a potential collision between two robots, we will restrict ourselves to a possible collision between the wrists of two robots, which corresponds to the upper three links of PUMA manipulators. This assumption is reasonable for two robots working closely in a common workspace. Any potential collisions due to the wrist geometry must be considered in planning a geometric collision-free path for the robot movement. Thus geometric modeling of the wrist is discussed first in Section II-A. Various reasons for assuming the straight line motion of the robot are discussed in Section II-B. Then a trajectory planning scheme for the straight line motion is presented in Section II-C [14], [15]. A new classification of path requirement situations is described in Section II-D for further discussions.

A. Geometric Modeling of the Wrist of a Robot

A simplified and appropriate geometric model of the wrist is a good tool for the detection of any potential collisions along the planned trajectory [22]. It is possible to use various geometric primitives instead of a model for the wrist. However, it requires more computational efforts in detection of a potential collision and the subsequent computation of a collision avoidance strategy. Here we shall concentrate on a suitable wrist model for a PUMA robot.

A cylindrical model or a truncated cone model can be used to model the wrist which is composed of the upper three links of the robot. For the detection of potential wrist collisions between two robots, all geometric models should be viewed with respect to a fixed global coordinate frame (x_0, y_0, z_0). The mathematical expression of these models in the global coordinate frame becomes position-dependent and very complicated, thus causing the detection of potential wrist collisions to be computationally intensive.

A sphere can be an alternative in modeling the wrist of a manipulator. The sphere contains links 4, 5, and 6, and includes the workpiece grasped by the manipulator hand. The origin of the sphere, which coincides with the origin of the hand coordinate frame, is uniquely determined from the kinematic equation as follows:

$$\boldsymbol{p}(t) = \left[p_x(t), p_y(t), p_z(t)\right]^T = \left[p_x(\boldsymbol{q}), p_y(\boldsymbol{q}), p_z(\boldsymbol{q})\right]^T \tag{1}$$

where $\boldsymbol{q}$ is the joint-variable vector and the superscript T denotes the transpose operation. The radius of the sphere is determined from the workpiece and the wrist geometry. The main advantage of this model is that the sphere is rotationally invariant. This reduces the complexity of the detection of potential wrist collisions to calculating the distance between the origins of the two spheres. It is notable that a large unnecessary space is included into the sphere so that computational efficiency in detecting the potential collisions is achieved at the expense of the modeling accuracy. The use of the sphere model in obtaining a collision-free path is described in Sections III and IV.

B. Straight Line Path

It is generally known that straight line motion of a manipulator is very difficult to achieve even though it is desirable. However, several advantages exist in planning a collision-free motion of the robots if straight line trajectories are used.

1) Visibility Concept: If one of the manipulators cannot see its destination position, then obstacle(s) exists on the path.

2) Motion Correctness: It is easier for the user to specify straight line Cartesian motion, and the correctness of motion can be verified from the motion of the manipulator hand.

3) Shortest Path: The straight line path is the shortest path between two assembly locations even if the required traveling time may not be minimum.

4) Time-Varying Obstacles: If two robots are not moving in straight line motion, it may be extremely difficult to handle the time-varying environments for mutual wrist collision avoidance.

5) Path Information: After detecting that potential collisions exist on the path, we can utilize the path information of one robot to decide the path or trajectory of the other robot for collision-free motion.

Thus a lot of unnecessary complexities can be alleviated by assuming the straight line path for both robots. The next concern is how to make the robots follow its planned straight line path strictly. A method for this is summarized in the next section.

C. Strictly Straight Line Trajectory Planning

Trajectory planning is a major component of motion planning, which converts the desired path information into a sequence of time-based intermediate configurations of the manipulator starting at the initial location and ending at the final location. For the purpose of controlling the manipulator, the joint trajectory function must be evaluated at each servo instant of time. In evaluating the joint interpolation function, the control set points are assumed to be within the torque constraints of the manipulator. Due to the *approximation* of the straight line path by the joint interpolation function, the control set points from the evaluation process do not satisfy exactly the straight line path in the Cartesian space. This prevents the user from checking any potential collisions easily between two robots even though they follow their own straight line paths.

For example, a desired straight line path with several knot points $A, B, C, \cdots$ is shown in Fig. 1. If an appropriate joint interpolation function is obtained through these knot points and evaluated for the control set points, the resultant control set points will be the points $a_1, a_2, a_3, \cdots$ in the joint-variable space, which correspond to the points $A_1, A_2, A_3, \cdots$ in the Cartesian space, respectively. It is obvious that these points $A_1, A_2, A_3, \cdots$ may not be on the desired straight line path. Thus it is vital to discuss a trajectory planning scheme which results in the control set points exactly on the preplanned straight line path for both robots. Results from Lee [14], [15] will be utilized to develop a collision-free motion planning strategy, and they are briefly summarized here.

In describing the orientation of the manipulator hand, Euler angles (yaw ($\alpha(t)$), pitch ($\beta(t)$), and roll ($\gamma(t)$)) are used instead of the rotation submatrix $[\boldsymbol{n}(t), \boldsymbol{s}(t), \boldsymbol{a}(t)]$. In fact, $[\alpha(t), \beta(t), \gamma(t)]$ are functions of the joint variables. Define the position $\boldsymbol{p}(t)$, orientation $\Phi(t)$, linear velocity $\boldsymbol{v}(t)$, and angular velocity $\Omega(t)$ vectors of the manipulator hand with respect to a global coordinate frame, respectively, as

$$\begin{aligned}
\boldsymbol{p}(t) &\triangleq \big(p_x(t), p_y(t), p_z(t)\big)^T \\
\Phi(t) &\triangleq \big(\alpha(t), \beta(t), \gamma(t)\big)^T \\
\boldsymbol{v}(t) &\triangleq \big(v_x(t), v_y(t), v_z(t)\big)^T \\
\Omega(t) &\triangleq \big(\omega_x(t), \omega_y(t), \omega_z(t)\big)^T. \qquad (2)
\end{aligned}$$

The continuous time index t will be replaced by a discrete servo time index kT, where T denotes the servo time period. Now the equation of a straight line path is described. An initial location $(\boldsymbol{p}(k_0T), \Phi(k_0T))$ and a final location $(\boldsymbol{p}(k_fT), \Phi(k_fT))$ are given to the manipulator. The manipulator hand is required to move from the initial location to the desired final location along the straight line specified by these two end points. The straight line equation that passes through these two points can be described as

$$\boldsymbol{p}(kT) = \boldsymbol{p}(k_0T) + \lambda(kT) \cdot \big(\boldsymbol{p}(k_fT) - \boldsymbol{p}(k_0T)\big) \qquad (3)$$

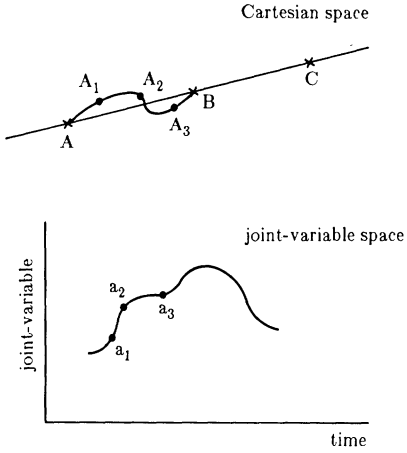


Fig. 1. Joint interpolated trajectory.

where $0 \leqslant \lambda(kT) \leqslant 1$. It is worth noting that k_fT is not a fixed final traveling time but will be determined from the straight line trajectory planning scheme. Also, since $\Phi(k_0T)$ and $\Phi(k_fT)$ are specified, it is desirable that the manipulator hand reaches its final Euler angles when $\boldsymbol{p}(kT)$ approaches $\boldsymbol{p}(k_fT)$. This can be easily done by interpolating the Euler angles linearly along the given straight line as

$$\Phi(kT) = \Phi(k_0T) + \lambda(kT) \cdot \big(\Phi(k_fT) - \Phi(k_0T)\big) \qquad (4)$$

where $\lambda(kT)$ is the same as in (3).

It is required that $[\boldsymbol{p}(kT), \Phi(kT), \boldsymbol{v}(kT), \Omega(kT), \dot{\boldsymbol{v}}(kT), \dot{\Omega}(kT)]$ be known for the control set points at every servo time instant [14]. Also, since the detection and avoidance of potential wrist collisions must be performed at each servo time instant, it is desirable to have the control servo points (or set points) precisely on the straight line path given by (3) and (4). Thus a discrete time approach has been used to obtain the control set point at each servo time instant. Hereafter, for brevity and simplicity, the servo time period T will be dropped from the rest of the equations if no confusion exists.

In addition to the straight line requirement in the Cartesian space, the control set points must satisfy the smoothness and torque constraints of the robot in the joint-variable space. Since no joint interpolation function is used, the smoothness constraint is necessary for maintaining smooth transition of the control set points. The smoothness constraint on the joint trajectory set points can be stipulated by a velocity bound (VB), an acceleration bound (AB), and a jerk bound (JB). They are given respectively as

$$|\dot{q}_i(k)| \leqslant \epsilon_i^v, \qquad \epsilon_i^v > 0, i = 1, \cdots, 6 \qquad (5)$$

$$|\ddot{q}_i(k)| \leqslant \epsilon_i^a, \qquad \epsilon_i^a > 0, i = 1, \cdots, 6 \qquad (6)$$

$$|w_i(k)| \leqslant \epsilon_i^J, \qquad \epsilon_i^J > 0, i = 1, \cdots, 6 \qquad (7)$$

where ϵ_i^v, ϵ_i^a, and ϵ_i^J are the ith element of six-dimensional bound vectors for the smoothness constraint, respectively, and $w_i(k)$ denotes the jerk of the ith joint at time k. Also, the torques at each joint are constrained by limits which

depend on the joint position (due to the manipulator actuator geometry) and on the joint velocity (due to the back electromotive force terms or other actuator effects) as

$$\tau_{i,\min}(\boldsymbol{q}(k),\dot{\boldsymbol{q}}(k)) \leqslant \tau_i(k) \leqslant \tau_{i,\max}(\boldsymbol{q}(k),\dot{\boldsymbol{q}}(k)), \quad i = 1,\cdots,6. \quad (8)$$

It has been found that there are many difficulties in finding a set of control points analytically which satisfies the straight line requirement, smoothness constraint, and torque constraint [15]. Instead, iterative search algorithms have been developed to determine the control set points on the straight line path. These algorithms generate the control set points exactly on a given straight line path within the smoothness constraint and the torque constraint in the joint-variable space. It is notable that these set points cannot be obtained from the conventional joint-interpolated trajectory. Thus it is not possible to specify a speed profile of a manipulator in Cartesian space if the manipulator should follow the straight line path correctly [6], [12]. For the remaining part of this paper, we will assume that the manipulator follows the straight line path faithfully with a combination of a full acceleration and a full deceleration in Cartesian space. The existence of such combination can be found in detail in [15].

D. Classifiation of Path Requirement Situations

The method in obtaining a collision-free path may vary depending on various path requirement situations especially when two robots are moving on the straight line paths simultaneously. When two robots (we call them robots 1 and 2) are assumed to move on their planned straight line paths with potential collisions, various path requirement situations are identified as follows.

Case 1: The final arrival time k_f of robot 2 can be relaxed, but its original path cannot be changed for the purpose of collision avoidance because the modification of path may induce potential collisions with other fixed obstacles.

Case 2: The final arrival time k_f of robot 2 can be relaxed, and its original path can be changed for the purpose of collision avoidance.

Case 3: The final arrival time k_f of robot 2 cannot be relaxed, but its original path can be changed for the purpose of collision avoidance.

Case 4: The final arrival time k_f of robot 2 cannot be relaxed, and its original path cannot be changed for the purpose of collision avoidance.

In Case 1, we can only change the speed or delay the motion of robot 2 along the original path to avoid the collision. From now on, the terminology *collision* will be used to denote the terminology *potential wrist collision*. Since the change of robot speed can only be accomplished by modifying the trajectory information, a procedure for a speed change needs to be developed to obtain a collision-free trajectory of robot 2. The delay of robot 2 motion can also be utilized for the purpose of collision avoidance. It then corresponds to the time-coordinated independent motion of the two robots.

In Case 2, collision-free motion planning is considered in the following categories:

- category 1 when speed reduction and/or time delay of the robot 2 motion is applied without any path modification;
- category 2 when only path modification is applied;
- category 3 when path modification with speed reduction and/or time delay of the robot 2 motion are applied simultaneously.

In category 1, a collision-free path can be found exactly in the same way as in Case 1. If a solution from category 1 is not adequate because a fairly large time delay is required or speed reduction is not appropriate for the collision avoidance, then we can consider a solution in categories 2 and 3. In category 2, there are a variety of freedoms in choosing a collision-free path. The choice of the collision-free path depends on the environment of the workspace and various user-designated performance indices. Sometimes, a solution in category 2 corresponds to a solution in category 3 due to various reasons, for example, a robot speed constraint, a path deviation constraint, etc. Clearly, the unnecessary path deviation in category 2 can be reduced by reducing robot speed along a collision-free path.

In Case 3, due to the fixed final arrival time k_f, there is no guarantee that a collision-free path exists for robot 2 satisfying the final arrival time k_f. In Case 4, collision avoidance must be realized by changing the path and/or trajectory of the other moving robot. In this paper, we focus our discussion only on the Case 1 path requirement situation since Case 1 is the most probable situation and the rest will require much further investigation. Thus *speed reduction and/or time delay* of the robot 2 motion will be investigated for the purpose of collision avoidance. In the next section, the general concepts of time scheduling of a straight line trajectory are described using the collision map technique.

III. Time Scheduling and Collision Map

Notions of time scheduling and a collision map are introduced in this section. The *time scheduling* is a procedure to modify and reschedule the traveling speed along a planned trajectory for the purpose of avoiding potential wrist collisions. In a broader sense, time scheduling implies time delaying and speed reduction of one of the two moving robots. Thus the robot motion may slow down or be delayed as a result of the time scheduling of the trajectory. It is notable that the time scheduling concept is different from the time scaling concept in [12].

The *collision map* is a two-dimensional figure in which we can incorporate both the path and trajectory information of two moving robots simultaneously. Time scheduling and collision map concepts are discussed in Sections III-A and -B, and several types of a collision map will be discussed in Section III-C.

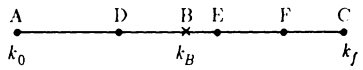


Fig. 2. Initial trajectory planning information.

A. Time Scheduling of Straight Line Trajectory

In Fig. 2, assuming that a straight line path whose trajectory set points are obtained from the trajectory planning scheme discussed in Section II-C, A is the initial location, C is the final location, k_0 is the initial time, and k_f is the final time of the robot trajectory without the relaxation portion. It is further assumed that the given path is planned with an acceleration and a deceleration portion. The boundary point between the acceleration and the deceleration portion is called a break point. It is denoted by point B at time k_B on the path. Then, during the time interval $[k_0, k_B]$, the manipulator hand is in the accelerating motion and during the time interval $[k_B, k_f]$, the manipulator hand is in the decelerating motion.

Referring to Fig. 2, if the robot starts the decelerating motion from a point D, which is located in the acceleration portion of the straight line $\overline{AC}$, then the maximum decelerating motion will end up at some point E, which results in longer traveling time of the robot from point A to point E than that from the original trajectory information. As a result, the robot will be in the accelerating motion on the portion $\overline{AD}$ and in the decelerating motion on the portion $\overline{DE}$. Thus an immediate objective of time scheduling is to find a point D or a segment $\overline{AE}$ such that the resultant trajectory of the segments can avoid the collision.

Assuming a constant bound on the magnitude of the Cartesian space acceleration, the trajectory planning of a straight line path with one break point results in *near-minimum time* for traversing the path [15]. If we have more segments for a path, then the total traveling time will be greater than that with a break point because the average traveling speed of the manipulator hand will be much slower than before. However, it is notable that the number of all the possible combinations of the acceleration portion and the deceleration portion may be very large. In the limiting case, every two trajectory set points can be composed of an acceleration portion and a deceleration portion. Hence the time scheduling of a straight line trajectory is a procedure for achieving the required *speed reduction* of the robot motion for the purpose of collision avoidance. Be it ever so simple and trivial, *time delay* of the robot motion can be another solution of time scheduling for the purpose of collision avoidance.

As noticed, the collision avoidance of two moving robots is related not only to the path information, but also to the trajectory information of the robots. Thus we shall present a collision map concept which incorporates the location and the corresponding servo time information of the robots into a two-dimensional figure. This will be useful in obtaining a collision-free trajectory using a time scheduling procedure.

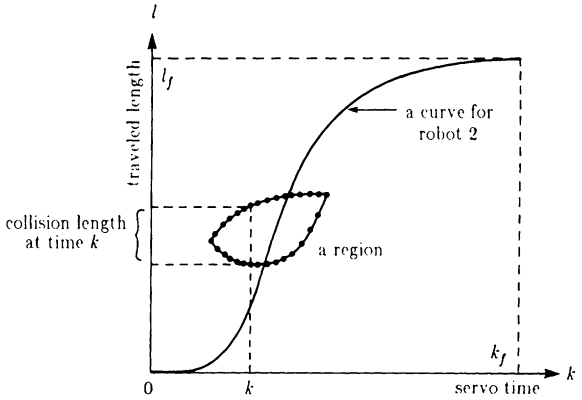


Fig. 3. Primitive collision map.

B. Collision Map

The collision map is obtained from the path and trajectory information of two moving robots. From the trajectory information, the traveled length $\lambda(k) \cdot l_{\text{total}}$ and the corresponding servo time instant $t = kT$ of a robot can be obtained along a straight line path, where l_{total} is the total length of the straight line path. Using this information, a curve can be drawn, which relates the traveled length with the corresponding servo time instant of the robot. We would refer to this robot as robot 2 and the other as robot 1. It will be assumed that robot 1 adheres to its original trajectory, and robot 2 must change its original trajectory for collision avoidance.

Two moving robots have a potential collision under the original trajectory information, if there is a range of collision lengths where the path of robot 2 is within the colliding range of a point on the path of robot 1. The union of these collision lengths at the collection of servo time instants, which determines the points on the path of robot 1, can be drawn as a connected region. We call this a *collision region*. See Fig. 3, where the vertical axis is the traveled length and the horizontal axis is the servo time. More precisely, the collision length at time k corresponds to all points on the robot 2 path that are within $r_1 + r_2$ of the point that lies on the path of robot 1 at time k, where $r_1 + r_2$ is the summation of radii of the two spheres. If the traveled length versus servo time curve (TLVSTC) touches or crosses the region, it indicates that a potential collision between the wrist spheres of both robots exists under the original trajectory information. Note that under our assumptions of maximum Cartesian acceleration and deceleration, the original unmodified TLVSTC takes on the form of two parabolic segments, joined at the break point.

Now, we present a method to obtain the collision region in more detail. Assume that two robots are moving in a straight line fashion as shown in Fig. 4, where robot 1 has to move from point A_1 to point B_1 and robot 2 has to move from point A_2 to point B_2. Note that paths of A_1B_1 and A_2B_2 are generally not on a common plane. Since robot 2 moves on the straight line path A_2B_2, a collision between the robots may occur if the sphere of the wrist of robot 1 intersects the sphere of the wrist of robot 2 during their motion. The distance between two trajectory set

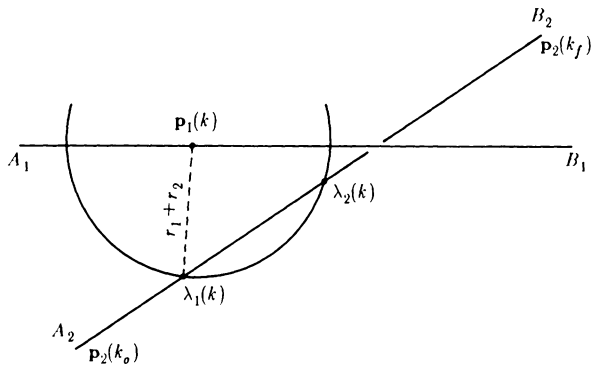


Fig. 4. Two robot path.

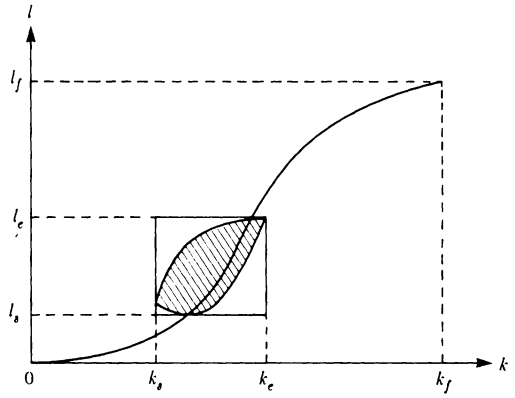


Fig. 5. Collision map.

points on the robot 1 and the robot 2 paths must be greater than $r_1 + r_2$ for collision avoidance. The portion that must be avoided by robot 2 at time k can be determined from the intersection of the sphere of radius of $r_1 + r_2$, centered at the point on the path of robot 1 at time k, and the original robot 2 path. The intersected length, however, must be computed for all the servo time instants of potential collisions.

With reference to Fig. 4, point $\boldsymbol{p}_1(k)$ is a point on the robot 1 path at time k. The equation of the robot 2 path is denoted as

$$\boldsymbol{p}_2 = \boldsymbol{p}_2(k_0) + \lambda \cdot \big(\boldsymbol{p}_2(k_f) - \boldsymbol{p}_2(k_0)\big). \tag{9}$$

Since the collisions between the robots occur at time k when the distance between point $\boldsymbol{p}_1(k)$ on the robot 1 path and the robot 2 path in (9) is less than or equal to $r_1 + r_2$, we solve the following equation:

$$(r_1 + r_2)^2 = \|\boldsymbol{p}_1(k) - \boldsymbol{p}_2\|^2. \tag{10}$$

Using (9) for $\boldsymbol{p}_2$, we have

$$\begin{aligned}(r_1 + r_2)^2 &= \big\{\boldsymbol{p}_1(k) - \boldsymbol{p}_2(k_0) - \lambda \cdot \big(\boldsymbol{p}_2(k_f) - \boldsymbol{p}_2(k_0)\big)\big\}^T \\ &\quad \cdot \big\{\boldsymbol{p}_1(k) - \boldsymbol{p}_2(k_0) - \lambda \cdot \big(\boldsymbol{p}_2(k_f) - \boldsymbol{p}_2(k_0)\big)\big\}.\end{aligned} \tag{11}$$

More explicitly,

$$\begin{aligned}(r_1 + r_2)^2 &= \|\boldsymbol{p}_1(k) - \boldsymbol{p}_2(k_0)\|^2 - 2\lambda \cdot \big(\boldsymbol{p}_1(k) - \boldsymbol{p}_2(k_0)\big)^T \\ &\quad \cdot \big(\boldsymbol{p}_2(k_f) - \boldsymbol{p}_2(k_0)\big) + \lambda^2 \cdot \|\big(\boldsymbol{p}_2(k_f) - \boldsymbol{p}_2(k_0)\big)\|^2.\end{aligned} \tag{12}$$

Equation (12) is a quadratic equation in λ, which can be solved easily. There are three possible solution cases.

1) *No Real Roots:* Real roots of λ do not exist.
2) *Two Real Roots:* Two real roots $\lambda_1(k)$ and $\lambda_2(k)$ exist ($\lambda_1(k) < \lambda_2(k)$).
3) *One Double Real Root:* Only one real double root $\lambda_1(k)$ exists.

When no real root exists, we know that no collision occurs between robot 1 and robot 2 at time k. When two real roots exist, we know that the collision lengths range from $l_f \cdot \lambda_1(k)$ to $l_f \cdot \lambda_2(k)$. When only one real double root exists, it marks the beginning or ending times for the collision region. Since the trajectory information is based on the servo time interval, the collision region can be obtained by calculating the intersected lengths of the path A_2B_2 for all the servo time instants of collisions. It is worth noting that the same analysis applies if the path of robot 1 is a series of straight line segments.

The collision region is drawn as a shaded region in Fig. 5. The TLVSTC for robot 2 should not enter the collision region if collision is to be avoided. Since it is difficult to describe the geometric shape of the collision region exactly and analytically, a *bounding box* or *window* is established to approximate the collision region as shown in Fig. 5. From now on, the collision starting and ending time will be denoted as k_s and k_e, respectively, from the bounding box (collision box) approximation. Also, we denote the length from point A_2 to the collision starting point and ending point as l_s and l_e, respectively, and the total length of the robot 2 path as l_f. In a simple collision situation, such as we are presenting here, these numbers can be found analytically using (12) with calculus. However, in more complex collision situations, it is necessary to derive the entire collision region, and then numerically determine the bounding box.

C. Several Types of Collision Map

Various types of a collision map are identified here. It is assumed that we only focus on the situation of a single straight line path for each robot. A connected straight line path for the robot or other types of path segments may happen in practical applications. Even though there must be changes in the TLVSTC and the collision region, the basic approach will be the same.

Fig. 6 shows the collision maps which could happen in various path requirement situations. In Fig. 6(a) the collision occurs for most of the path of robot 2 for a long time interval. In Fig. 6(b) the collision occurs for most of the path of robot 2 for a short time interval. In Fig. 6(c) the collision occurs for a small portion of the robot 2 path for a long time interval. In Fig. 6(d) the collision occurs for a small portion of the path of robot 2 for a short time interval.

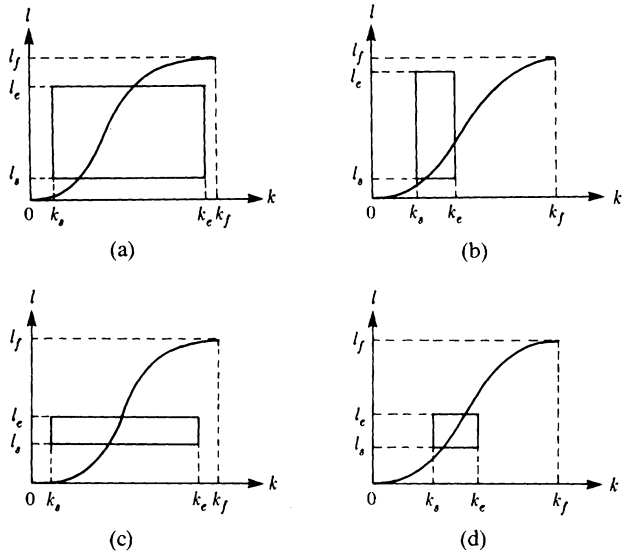


Fig. 6. Various types of collision map.

If $l_s = 0$ and the bounding box indicates a collision on the original TLVSTC, then this corresponds to a special case, where a different solution (category 2 or 3 in the Case 2 path requirement situation) must be obtained. This is caused by our assumption that we are already operating the robot at full acceleration and therefore only slow down or pause without resorting to a path modification. This results in shifting the original TLVSTC to the right (and never to the left). The physical interpretation is that the second robot cannot get out of the way fast enough along the original path without getting hit by the first robot.

The TLVSTC and the collision box are obtained from the path and trajectory information of the two robots. Thus the shape of the collision boxes varies depending on the collision situations. We assume only a simple path requirement situation in this section to illustrate the collision map. Multiple collision regions may occur in some collision situations. Obviously, a bounding box enclosing the whole scattered collision regions may result in rough approximation for a collision region. However, the discussions presented here can be applied to the case of multiple collision regions also.

IV. Collision-Free Motion Planning

Using the collision map and the time scheduling concept, this section presents a method for collision-free motion planning of two moving robots. The method is considered in the Case 1 path requirement situation.

A. Various Approaches

In Case 1, one robot (robot 2) cannot change its original path but can change its arrival time k_f for collision avoidance with the other moving robot (robot 1). A collision map is utilized in obtaining a collision-free straight line trajectory by time scheduling.

From the collision map as shown in Fig. 7, the collision starts at time k_s. The time for robot 2 to travel the length l_s must be modified so that robot 2 reaches the location of length l_s at a different time. Thus to find a collision-free straight line trajectory by time scheduling, it is necessary to find a TLVSTC of robot 2 which does not touch or cross the collision box. It is assumed that the TLVSTC OM crosses the collision box, which indicates that a potential collision exists. Since the objective is to find a curve which avoids the collision box, it is necessary to make robot 2 slow down (OM′) or delay its motion.

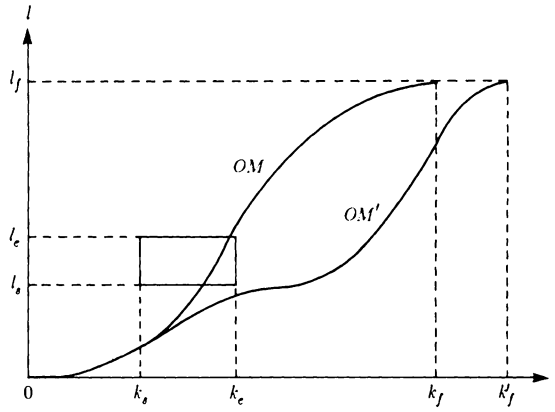


Fig. 7. Collision map with potential collision.

The arising problems in time scheduling of the original trajectory are then identified as follows.

- Is either a time delay or speed reduction of the robot 2 motion necessary for avoiding collisions?
- How much must the robot 2 motion be delayed and where should it occur?
- How much must the speed of robot 2 be reduced and where should it occur?
- What are the effects of time delay and speed reduction on the final arrival time?

These problems are addressed by using the collision map corresponding to the path requirement situation. Depending on the task scheduling requirements and the environment of path requirement situation, various approaches are discussed in obtaining a TLVSTC of robot 2 which can avoid the collision box. In the following discussion, we assume that the collision box is such that $l_s \neq 0$. If $l_s = 0$, then the only possible solution to avoid the collision box is to move robot 2 out of its original straight line path, as even standing still and waiting robot 2 will still get hit. Also, this case is contrary to our assumption for problem formulation. Again, this is assuming an existing collision along the original TLVSTC at maximum acceleration.

Approach 1: In Fig. 8(a), the curve OM_1 is a curve which avoids the collision box and can be obtained by purely delaying the starting time at the initial location. The time when robot 2 arrives at the location of length l_s is denoted as k_1 from the original trajectory information. Then the required time delay at the initial location will be $k_e - k_1$. The arrival time k_f^1 of robot 2 for the curve OM_1 will be $k_f + k_e - k_1$. However, if robot 2 has any constraints to leave the initial location as scheduled, then this curve OM_1 cannot be used for collision avoidance with

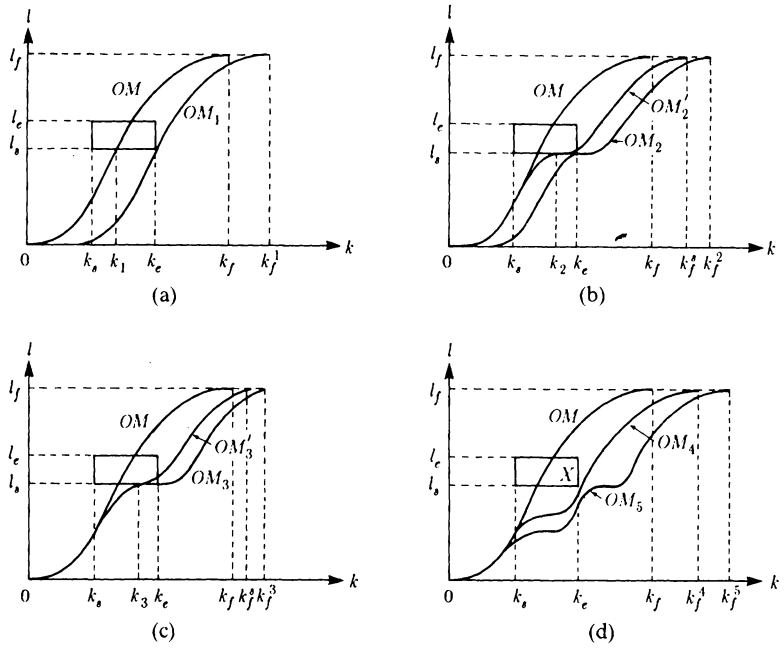


Fig. 8. Collision-free trajectory curves.

robot 1 and a different method, which utilizes speed reduction of the robot 2 motion, will be required.

Approach 2: In Fig. 8(b), the curve OM_2 is a curve which avoids the collision box and can be obtained by slowing down the robot 2 motion and delaying the starting time at the initial location. The speed reduction of robot 2 is performed on the path of length l_s from the initial location. The time when robot 2 arrives at the location of length l_s after speed reduction is denoted as k_2. If we allow the robot to accelerate and decelerate again, the corresponding TLVSTC for the segment of length $l_f - l_s$ may touch or cross the collision box again. In this case, the time delay strategy can be realized at the initial location to avoid the collision box. The required time delay at the initial location will be $k_e - k_2$. The arrival time k_f^2 of robot 2 for the curve OM_2 will be $k_f^s + k_e - k_2$, where k_f^s is the arrival time of robot 2 for the curve OM_2' only after speed reduction. If $k_e < k_2$, then the speed reduction on the path of length l_s is sufficient to avoid the collision box.

Approach 3: In Fig. 8(c), the curve OM_3 is a curve which avoids the collision box and can be obtained by slowing down the robot 2 motion and stopping the movement temporarily at the location of length l_s to cause the required time delay. The time when robot 2 arrives at the location of length l_s after speed reduction is denoted as k_3. Then the required time delay at the location of length l_s will be $k_e - k_3$. The arrival time k_f^3 of robot 2 for the curve OM_3 will be $k_f^s + k_e - k_3$, where k_f^s is the arrival time of robot 2 for the curve OM_3' only after speed reduction. Again, if $k_e < k_3$, then the time delay at the location of length l_s is not necessary, which corresponds to the case that pure speed reduction is sufficient to avoid the collision box.

Approach 4: In Fig. 8(d), curve OM_4 is a curve which avoids the collision box and can be obtained by slowing down the robot 2 motion adequately. The speed of robot 2 can be reduced in a number of ways so that the curve OM_4 can avoid the collision box. Since the curve must avoid the corner point X, speed reduction will be applied to the robot 2 motion on the path of length l_s as an approach to the solution. The curve from the speed reduction of robot 2 motion can be obtained by applying the time scheduling technique. Recall that the TLVSTC OM is assumed to exist for the given straight line path. Thus the speed reduction can be realized on the acceleration portion of the straight line trajectory by selecting an appropriate break point or a segment. This corresponds to an appropriate selection of a point D or a segment $\overline{AE}$ in Fig. 2.

If one is only interested in pure speed reduction, the speed reduction can be performed initially on the segments of length l_s and $l_f - l_s$. The TLVSTC from the two segments may still touch or cross the collision box. Then further speed reduction must be applied to the segment of length l_s. This can be accomplished by dividing this segment of length l_s into two, three, or four, etc., segments until the combined TLVSTC does not touch or cross the collision box. As an example, the curve OM_5 is drawn from the combined trajectory information for three segments of length $l_s/2$, $l_s/2$, and $l_f - l_s$ in Fig. 8(d). It is understandable that if we use more segments in trajectory planning of a path, then the total traveling time will become longer. Thus the final arrival time of the curve OM_5 will be much longer than that of the curve OM.

B. Final Arrival Time Due to Time Scheduling

The effects on the final arrival time due to the time scheduling of a trajectory are considered in more detail. Hereafter, discussions are based on the assumption that the servo time interval is very small and the effects from

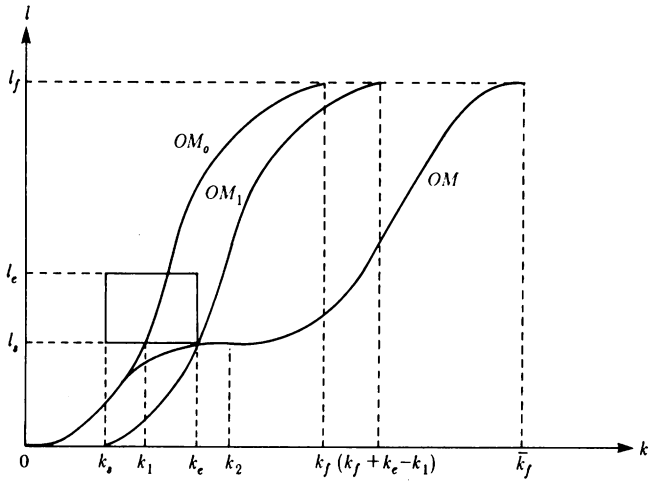


Fig. 9. Effect of time scheduling on final arrival time.

discretization are negligible. Later, the results will be verified through an illustrated example.

We consider a path requirement situation in Fig. 9, where the curve OM_o is a TLVSTC from the original trajectory information of robot 2. The curve OM is a TLVSTC with the final arrival time $\bar{k}_f$, which is obtainable by reducing the robot 2 speed on the path of length l_s. The time k_1 is the time when the original curve OM_o reaches the location of length l_s. The time k_2 is the time when the curve OM reaches the location of length l_s after speed reduction.

If the motion of robot 2 is delayed for the purpose of collision avoidance, then the required time delay will be $k_e - k_1$ at the initial location as evident from Fig. 9. The original curve OM_o is shifted right for the time delay to avoid the collision box, which corresponds to the curve OM_1. From the speed reduction on the acceleration portion, we should have $k_1 \leqslant k_2$ for the purpose of collision avoidance. Thus the following inequality from speed reduction must be satisfied for the collision avoidance with robot 1

$$k_1 \leqslant k_e \leqslant k_2. \tag{13}$$

It is notable that the traveling time $k_f - k_1$ is the lower bound for all those motions going from the location of length l_s to the final location, where the speed of OM at l_s is lower than the speed of OM_o at l_s. Thus considering the traveling time on the path of length $l_f - l_s$ when the robot follows the curve OM (where the velocity at time k_2 is almost equal to zero), we have

$$k_f - k_1 < \bar{k}_f - k_2 \tag{14}$$

where time $\bar{k}_f$ is the final arrival time of the curve OM.

Since $k_e - k_1 \leqslant k_2 - k_1$, we have

$$\bar{k}_f > k_f + k_2 - k_1 \geqslant k_f + k_e - k_1. \tag{15}$$

Therefore,

$$\bar{k}_f > k_f + k_e - k_1. \tag{16}$$

The time $k_e - k_1$ is the required time delay for the curve to avoid the collision box (the curve OM_1 in Fig. 9). The above equation indicates that the final arrival time from the speed reduction on the path of length l_s is greater than that from the time delay. However, if speed reduction occurs at an arbitrary location of the straight line path, the final arrival time from time delay may not always give rise to shorter traveling time than that from speed reduction of the robot motion. These aspects will be discussed with an example in Section V.

C. Time Scheduling Procedure

A procedure for the time scheduling of a straight line trajectory is outlined. From the discussion of Section III-A, specifying the break points or the segments is necessary for speed reduction. In this section, a segmentation procedure is developed for the path of length l_s. The procedure utilizes a user-designated integer δ_1 for specifying the maximum number of the segments for the path of length l_s. If a collision still occurs even after the path is segmented by δ_1, then time delay of the robot 2 motion will be used for the purpose of collision avoidance.

Procedure TS (Time Scheduling): This procedure addresses the iterative steps in obtaining a collision-free straight line trajectory by speed reduction.

TS1. [Initialize the index i.] Set $i = 1$.

TS2. [Construct a collision map from the path requirement situation.] Obtain a collision map from the path and trajectory information of robots 1 and 2.

TS3. [Obtain the TLVSTC.] Identify the collision starting length l_s. Then apply the search algorithms [15] to the segments of length l_s and $l_f - l_s$, and draw the combined TLVSTC from the resultant trajectory information in the collision map of step TS2.

TS4. [Check the collision map and divide the segment of length l_s if necessary.]

IF the curve in step TS3 still touches the collision box,
THEN increment i by 1, and divide the segment of length l_s by i, and go to step TS5;
OTHERWISE, stop. (Planning of collision-free path by time scheduling of the trajectory is successful.)

TS5. [Obtain the combined TLVSTC.] Apply the search algorithms to each of the segments, draw the combined TLVSTC in the collision map, and go to step TS6.

TS6. [Check the stopping variable.]

IF $i \geqslant \delta_1$, where δ_1 is a designated integer that specifies the maximum allowable number of segments on the path of length l_s,
THEN time delay of the robot 2 motion will be used for the collision avoidance purpose;
OTHERWISE, go to step TS4.

V. An Example of the Case 1 Path Requirement Situation

The time scheduling procedure is applied to the Case 1 path requirement situation. The trajectory information from [14] has been used to construct the TLVSTC as

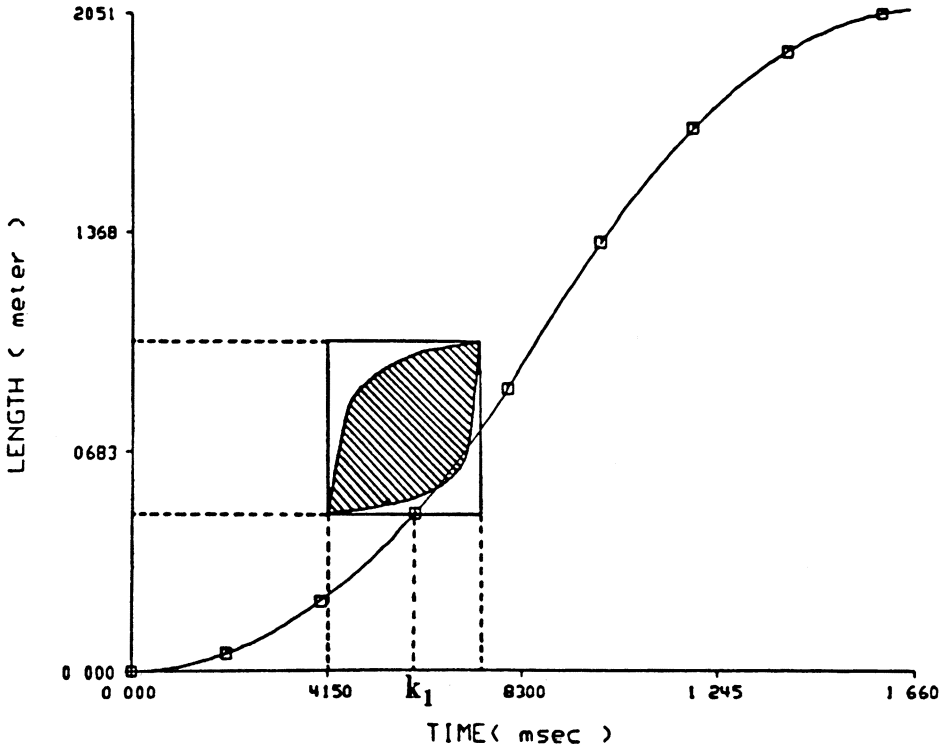


Fig. 10. Collision map for example.

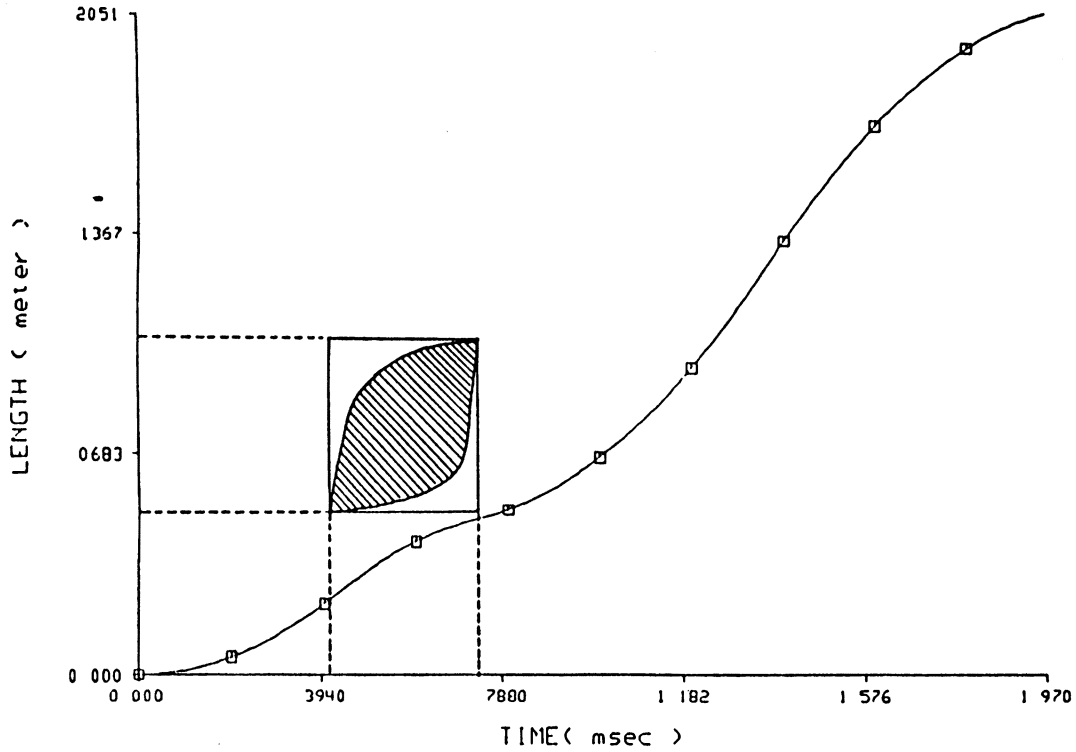


Fig. 11. Collision map after time scheduling.

shown in Fig. 10, where the collision box is shown to cover the length from 4.9 cm to 10.8 cm during the time interval from 0.42 to 0.74 s. That is, the collision starting length l_s is assumed to be 4.9 cm, and the collision ending length l_e is assumed to be 10.8 cm.

From the collision map, we have two straight line segments: one ranges from the initial point to the collision starting point, while the other ranges from the collision starting point to the desired final point. The first segment was planned to constitute an acceleration portion and a deceleration portion and was found able to avoid the collision box up to the location of length l_s. The second segment was also planned by applying the search algorithms [14], [15]. In Fig. 11, it is shown that the

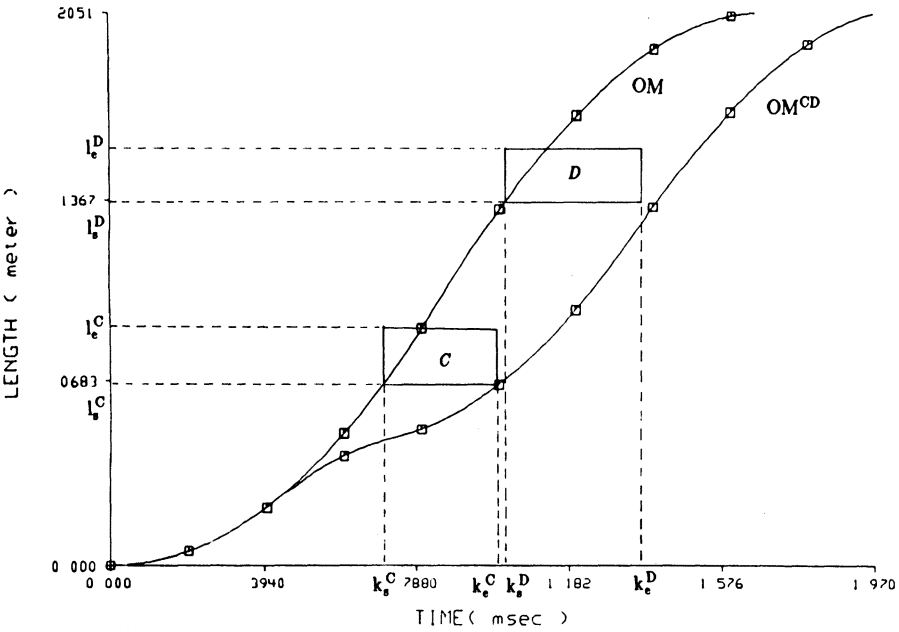


Fig. 12. Comparison of final arrival times.

TLVSTC avoids the collision box. The total traveling time before time scheduling was found to be 1.66 s from the trajectory planner, while the total traveling time after time scheduling was found to be 1.97 s. We now investigate the effect of time scheduling on the final arrival time.

When the speed reduction is performed on the segment of length l_s, the traveling time from speed reduction on the segment of length l_s must be equal to or greater than that from the original trajectory on the same segment plus the required time delay for avoiding the collision box. If we assume that the robot motion can be delayed at the initial location, then the required time delay can be computed from the collision map and the trajectory information. In Fig. 10, the time k_1 was found to be 0.60 s from the original trajectory information, thus the required time delay at the initial location is $(0.74 - k_1)$ s or 0.14 s. Then the final arrival time is 1.80 s which is shorter than that from speed reduction (1.97 s).

When the speed reduction is performed on an arbitrary segment of the straight line path, it is not always true that time delay gives rise to a shorter traveling time than speed reduction of the robot motion. (See the collision map in Fig. 12 which is the same as Fig. 11 except for the collision box.) The TLVSTC OM^{CD} was obtained by speed reduction on the path of length l_s (4.9 cm) as before. From the trajectory information of the curves OM and OM^{CD}, two collision boxes C and D are assumed in the collision map, which have a collision with the TLVSTC OM and are avoidable by the TLVSTC OM^{CD}. The values of k_s^C, k_e^C, k_s^D, and k_e^D are 0.70, 0.97, 0.99, and 1.36 s, respectively. Also, the values of l_s^C, l_e^C, l_s^D, and l_e^D are 6.69, 9.11, 13.29, and 15.32 cm, respectively. The final arrival time of the TLVSTC OM^{CD} is the same as before (1.97 s). However, if time delays are considered at the initial location for avoiding the collision boxes C and D, then the total final arrival time is 1.93 s for avoiding the collision box C, and 2.03 s for avoiding the collision box D. Clearly, we find that 1.93 s is shorter than the final arrival time of the TLVSTC OM^{CD}, and 2.03 s is longer than the final arrival time of the TLVSTC OM^{CD}.

VI. Conclusion

A collision-free motion planning problem of two robots was investigated systematically by using a sphere model for the wrist of the robot, straight line trajectory planning, and the notions of a collision map and time scheduling. Several path requirement situations were identified and classified for obtaining collision-free paths of two moving robots. In time scheduling of a trajectory, four different approaches were discussed using speed reduction and/or time delay of the robot motion. Finally, an example was presented to illustrate the concept of time scheduling and the collision map for achieving collision-free motion planning of two robots. These achievements and new sets of ideas would be the basis in solving the collision avoidance problem for the multiple robots.

Acknowledgment

The authors would like to thank reviewers of this paper for their helpful suggestions.

References

[1] N. Ahuja *et al.*, "Interference detection and collision avoidance among three dimensional objects," in *Proc. 1st Annu. Nat. Conf. Artificial Intelligence*, pp. 44–48, Aug. 1980.

[2] A. K. Bejczy, "Robot arm dynamics and control," Jet Propulsion Lab., Pasadena, CA, Tech. Memo. 33-669, Feb. 1974.

[3] Beyer, *Standard Math Tables*. CRC Press, 1978, pp. 283–316.

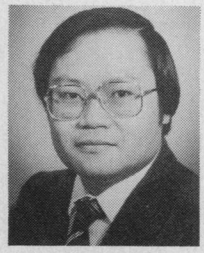
[4] R. A. Brooks, "Solving the find-path problem by good representation of free space," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-13, pp. 190–197, Mar. 1983.

[5] ——, "Planning collision-free motions for pick-and-place operations," *Int. J. Robotics Res.*, vol. 2, No. 4, pp. 19–44, Winter 1983.
[6] H. Bruhm *et al.*, "Cartesian path planning by general polynomial interpolation," in *Proc. 2nd IASTED Int. Symp. Robotics and Automation*, 1983, pp. 156–159.
[7] R. T. Chien *et al.*, "Planning collision-free paths for robotic arm among obstacles," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-6, pp. 91–96, Jan. 1984.
[8] E. Freund, "Hierarchical control of guided collision avoidance for robots in assembly automation," in *Proc. 4th Int. Conf. Assembly Automation*, 1983, pp. 91–103.
[9] ——, "On the design of multi-robot systems," in *Proc. IEEE Int. Conf. Robotics and Automation*, 1984, pp. 477–490.
[10] E. G. Gilbert and D. W. Johnson, "Distance functions and their application to robot path planning in the presence of obstacles," *IEEE J. Robot. Automat.*, vol. RA-1, pp. 21–30, Mar. 1985.
[11] L. Gouzenes, "Collision avoidance for robots in an experimental flexible assembly cell," in *Proc. IEEE Int. Conf. Robotics and Automation*, 1984, pp. 474–476.
[12] J. M. Hollerbach, "Dynamic scaling of manipulator trajectories," *Trans. ASME, J. Dynamic Syst., Meas., Contr.*, vol. 106, pp. 102–106, Mar. 1984.
[13] O. Khatib, "Dynamic control of manipulators in operational space," presented at the 6th IFTOMM Congress on Theory of Machines and Mechanisms, New Delhi, India, Dec. 15–20, 1983.
[14] B. H. Lee, "An approach to motion planning and motion control of two robots in a common workspace," Ph.D. dissertation, Computer, Information, and Control Eng., Univ. of Michigan, Ann Arbor, 1985.
[15] ——, "Algorithmic approach to straight line trajectory planning for mechanical manipulators," in *Proc. 1986 American Control Conf.*, pp. 121–126.
[16] T. Lozano-Peres, "Spatial planning: A configuration space approach," *IEEE Trans. Comput.*, vol. C-32, pp. 108–120, Feb. 1983.
[17] T. Lozano-Peres, "Automatic planning of manipulator transfer movements," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-11, pp. 681–698, Oct. 1981.
[18] J. Y. S. Luh and C. E. Campbell, "Collision-free path planning for industrial robots," in *Proc. 20th CDC*, 1982, pp. 84–88.
[19] ——, "Minimum distance collision-free path planning for industrial robots with a prismatic joint," *IEEE Trans. Automat. Contr.*, vol. AC-29, pp. 675–680, Aug. 1984.
[20] J. K. Myers *et al.*, "A supervisory collision-avoidance system for robot controllers," in *Proc. Winter Annu. Meeting Amer. Soc. Mechanical Engineers*, 1982, pp. 225–232.
[21] R. P. Paul, *Robot Manipulators: Mathematics, Programming, and Control*. Cambridge, MA: MIT Press, 1981.
[22] A. de Pennington, *et al.*, "Geometric modeling: A contribution towards intelligent robots," in *Proc. 13th ISIR/Robot 7 Conf.*, pp. 7.35–7.54.
[23] A. A. Petrov and I. M. Sirota, "Obstacle avoidance by a robot manipulator under limited information about the environment," *Avtomatika i Telemekhanika*, no. 4, pp. 29–40, Apr. 1983.
[24] R. C. Smith and D. Nitzan, "A moduler programmable assembly station," in *Proc. 13th ISIR/Robot 7*, 1983.
[25] S. M. Udupa, "Collision detection and avoidance in computer controlled manipulators," in *Proc. 5th Int. Joint Conf. Artificial Intell.*, 1977, pp. 737–748.

B. H. Lee (S'84–M'85) was born in Seoul, Korea, on May 29, 1955. He received the B.S.E.E. and M.S.E.E. degrees in electronics engineering from Seoul National University, Seoul, Korea, in 1978 and 1980, respectively, and the Ph.D. degree in computer, information, and control engineering from the University of Michigan, Ann Arbor, in 1985.

From 1978 to 1980, he was a Departmental Assistant in the Department of Electronics Engineering at Seoul National University, and from 1980 to 1981 he was employed as an instructor at Joong-Ang University in Korea. From 1985 to 1986 he was with the School of Electrical Engineering at Purdue University, West Lafayette, IN, as an Assistant Professor. He is now with the Department of Control and Instrumentation Engineering, Seoul National University as an Assistant Professor. His research interests include path/trajectory planning, kinematics and dynamics, obstacle avoidance of mechanical manipulators, multirobot operations, computer integrated manufacturing, artificial intelligence, and control system analysis and design.

Dr. Lee is the Indiana Chapter President of Korean Scientists and Engineers in America.



C. S. G. Lee (S'71–S'78–M'78–SM'86) received the B.S. and M.S. degrees in electrical engineering from Washington State University, Pullman, in 1973 and 1974, respectively, and the Ph.D. degree from Purdue University, West Lafayette, IN, in 1978.

In 1978–1985, he taught at Purdue University and the University of Michigan, Ann Arbor. He is currently an Associate Professor of Electrical Engineering, Purdue University. His current research interests include computational algorithms and structures for manipulators and intelligent manufacturing systems. He is a coeditor of *Tutorial on Robotics* (second edition, IEEE Computer Society Press), and a coauthor of *Robotics: Control, Sensing, Vision and Intelligence* (McGraw-Hill).

Dr. Lee was an IEEE Computer Society Distinguished Visitor in 1983–1985. He is a Technical Area Editor of the *IEEE Journal of Robotics and Automation*. He is a member of Sigma Xi, Tau Beta Pi, and the SME/RI.