Robotica (1999) volume 17, pp. 131–141. Printed in the United Kingdom © 1999 Cambridge University Press

A configuration space approach to collision avoidance of a two-robot system

S.W. Lee,* B.H. Lee* and K.D. Lee†

(Received in Final Form: August 4, 1998)

SUMMARY

This paper suggests an efficient approach to collision avoidance of a practical two-robot system. The approach is based on the C-space of one robot, and we consider only two and three dimensional C-space in which nearly all industrial manipulators can be reasonably represented for collision avoidance problems. The C-space of one robot is discretized with the concentric circles or spheres centered at the goal configuration. We then introduce the concept of *free arc* which is a set of the candidate configuration points on the concentric circle or sphere at each sampling time to avoid collisions. It is represented simply with respect to the coordinate frame attached to the goal configuration. The free arc is used as a tool for collision avoidance of a two-robot system and sub-optimality can be considered in determining a collision-free path. The main contribution of this paper is that it provides a method to construct C-obstacles in a time-varying environment more efficiently than the existing methods (e.g. *slice projection* method[1]) by restricting the search space at the expense of sacrificing completeness. Thus, it enables us to implement a practical collision avoidance algorithm for a two-robot system with decreased computational cost. Simulation results for two 2-*d.o.f.* manipulators and two PUMA robots are presented to show the efficacy of the proposed method.

KEYWORDS: Two-robot system: Collision avoidance; C-space; Free arc.

1. INTRODUCTION

The collision-free motion planning problem for a robot has been studied extensively for almost three decades. Many researchers in various fields tackled the problem and proposed several approaches to cope with it. Most of the approaches may be categorized into several groups according to the type of a robot and its environment. As to a robot – point, polygon, mobile robot and a manipulator with low *d.o.f.*– in a stationary environment, it is conservative to say that the problem has been solved completely. A variety of complete and efficient algorithms have been reported and computational complexity of them has been analyzed in the literature.[2,3] Thus, finding a practical algorithm in a dynamic environment where only partial information is available about the obstacles[4–6] or dealing with a high *d.o.f.* manipulator[7–10] becomes an issue of principal concern nowadays.

Unfortunately, the problem in a time-varying environment has been much less researched, and the majority of the approaches to the problem adopted and modified the methods for stationary obstacle avoidance. In the configuration-time space (CT-space), the time-varying obstacle is converted to a stationary obstacle. Thus, motion planning for time-varying obstacle avoidance is reduced to path planning for stationary obstacle avoidance in the CT-space based methods.[11–13] Ko *et al*.[14] proposed an approach to the time-varying obstacle avoidance problem using the view-time concept. Although they formulated the problem mathematically in the C-space, they used an existing stationary obstacle avoidance scheme to construct C-obstacles and search for a collision-free path. Another powerful methodology is based on an artificial potential field.[15–18] However, only a few results have been reported in a time-varying environment due to the existence of local minima and its intractability to deal with arbitrarily shaped obstacles.

Especially, the case of a two-robot system is much more difficult to handle than the general time-varying obstacle avoidance, since we should consider the motion of two robots simultaneously and the obstacle composed of a manipulator changes not only its position but its shape with time. To overcome this difficulty, several approaches have been suggested under some assumptions or simplifications for the last decade. Freund and Hoyer[19] introduced the concept of hierarchical coordinator based on the nonlinear control approach and suggested a real-time collision-free motion planning method for multi-robot system. Although they provided a practical and systematic approach, the construction of the hierarchical coordinator is not so easy and optimality was not considered.

Lee and Lee[20] studied the collision avoidance problem of the dual robot. They modeled the robots as spheres and assumed that each robot should move along prespecified straight-line paths. By constructing a collision-map using the path and trajectory information of two robots, they proposed a time scheduling algorithm to modify the velocity profile of the secondary robot. This study provides an easy-to-use method for the collision-free motion planning of a multi-robot system. However, no collision-free trajectory may exist on the given path and there is no analytical method to construct a collision map. Chang *et al*.[21] improved the above results by constructing a more accurate geometric model of a robot and presenting a method to obtain the minimum delay time of one robot heuristically.

* School of Electrical Engineering, Seoul National University, San 56-1, Shinrim-dong, Kwanak-ku, Seoul 151-742 (Korea).
† Department of Computer Engineering, Yeungnam University, Kyungsan 712-749 (Korea).

https://doi.org/10.1017/S0263574799001113 Published online by Cambridge University Press

Although they still needed prespecified paths of two robots to avoid stationary obstacles, they relieved the computational burden of constructing a complete collision map.

Shin and Bien[22] developed the concept of coordination space which is constructed with the scalar variables defining the positions along prescribed paths. They also defined collision regions in the space. Bien and Lee[23] and Lee[24] proposed an analytic collision-free and time-optimal trajectory planning method for two robot manipulators in the coordination space. The robots are assumed to move along prescribed paths with limited actuator torques and velocities. These approaches in the coordination space are based on the path-velocity decomposition technique suggested by Kant and Zucker.[25] They planned the paths of two robots independently by taking only stationary obstacles into consideration at first, and then modified the velocity profile of two robots to avoid collision between them. In other words, they needed a stationary obstacle avoidance scheme in the path planning step, and they didn't consider moving obstacles except for the other robot.

As an alternative, we may consider the collision avoidance problem of a two-robot system in the C-space introduced by Lozano-Perez and Wesley.[26] C-space facilities handling of robot motion since the robot manipulator in the Cartesian space is reduced to a point in the C-space. Even though with this advantage, the global methods in the C-space have inherent shortcomings: (a) the computational complexity increases exponentially with the number of *d.o.f.* (the curse of dimensionality), and (b) it requires an expensive precomputation step to compute the C-obstacles (much more expensive in a time-varying environment).

To remedy the problems of the C-space based methods in a time-varying environment, we intend to present an efficient method to construct time-varying C-obstacles in this paper. Instead, we consider only two and three dimensional C-space in which nearly all industrial manipulators can be reasonably represented for a collision avoidance problem, in order to avoid the curse of dimensionality of C-space based methods. First, we discretize the C-space of one robot with the concentric circles or spheres centered at the goal configuration. We then introduce the concept of *free arc* which is a set of the candidate configuration points on the concentric circle or sphere at each sampling time to avoid collisions. It is represented simply with respect to the coordinate frame attached to the goal configuration. The free arc is used as a tool for collision avoidance of a two-robot system and sub-optimality can be considered in determining a collision-free path. The main contribution of this paper is that it provides a method to construct C-obstacles in a time-varying environment more efficiently than the existing methods (e.g. *slice projection* method[1]) by restricting the search space at the expense of sacrificing completeness. Thus, it enables us to implement a practical collision avoidance algorithm for a two-robot system with decreased computational cost.

This paper is organized as follows: In Section 2, we present a method to describe a time-varying environment of a robot manipulator in its C-space in the case of 2-dimensional C-space. Section 3 describes a method for transforming the collision avoidance problem of a robot into the SOCAP (Sub-Optimal Collision Avoidance Problem) to search for a collision-free path, and discusses the completeness of the proposed method. Computation time of the method is compared with that of an existing method - *slice projection*, and the result is discussed in Section 4. Subsequently, extensions to 3-dimensional C-space is outlined briefly in Section 5. In Section 6, the proposed method is applied to the collision-free motion planning of two 2-*d.o.f.* manipulators and two PUMA robots respectively, and the simulation results for the application are given. Finally, the characteristics of the proposed method and the future research direction are stated in Section 7.

2. C-OBSTACLE REPRESENTATION IN A TIME-VARYING ENVIRONMENT

The collision avoidance problem of a two-robot system is formulated as follows:

PROBLEM: Suppose that two robots have a task that they should move from their corresponding initial points to final points. Then, plan the collision-free path of one robot while the other moves along its prescribed trajectory. ■

To solve the problem, we build the C-space of one robot which should avoid collisions with the other. In the C-space, the other robot moving along its prescribed trajectory becomes a time-varying obstacle since its position and shape varies with time. The time-varying C-obstacle cannot be represented completely only in the C-space, instead CT-space should be built by adding an extra time dimension to the C-space. However, it increases the dimension of the problem and we should compute C-obstacles at every instance of time in the CT-space. To save computation time required to represent time-varying C-obstacles, we intend to present a simple and heuristic method in the C-space. For explanation use, the 2-dimensional C-space case is explained in this section and extensions to 3-dimensional C-space is presented in Section 5.

First, we discretize the C-space of the robot uniformly with N concentric circles centered at the goal configuration, $\boldsymbol{q}_N$, where N is the number of sampling during total travel time, so that the initial configuration, $\boldsymbol{q}_0$, may be located on the largest circle. In this discretized C-space, we assume that the robot configuration at each sampling time should lie on each concentric circle. This discretization of the C-space, where the number of sampling is 10, may be depicted as Fig. 1. Here, the equation of the concentric circle at k-th sampling time is described as

$$\|\boldsymbol{q}_k - \boldsymbol{q}_N\| = \|\boldsymbol{q}_0 - \boldsymbol{q}_N\| \frac{N-k}{N} \tag{1}$$

where $\|\boldsymbol{x}\|$ is *norm* or *length* of the vector $\boldsymbol{x}$, defined as

$$\|\boldsymbol{x}\| \equiv (\boldsymbol{x}\cdot\boldsymbol{x})^{\frac{1}{2}} = \left[\sum_{i=1}^{2} x_i^2\right]^{\frac{1}{2}}, \tag{2}$$

and $\|\boldsymbol{q}_0 - \boldsymbol{q}_N\| \frac{N-k}{N}$ is the radius of the circle.

https://doi.org/10.1017/S0263574799001113 Published online by Cambridge University Press

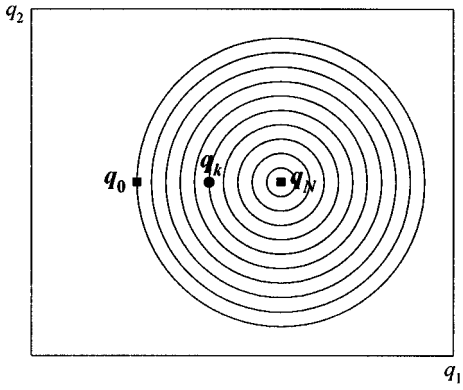


Fig. 1. Discretization of C-space ($N=10$).

We now attach the polar coordinate frame to the goal configuration in order to describe the circles effectively. Then, the configuration points, $\boldsymbol{q}_k=(q_{1,k}, q_{2,k})$, on the k-th concentric circle is described as follows:

$$\begin{aligned} q_{1,k} &= r_k \cos\theta_k + q_{1,N}, \\ q_{2,k} &= r_k \sin\theta_k + q_{2,N} \end{aligned} \tag{3}$$

where

$$r_k = \|\boldsymbol{q}_0 - \boldsymbol{q}_N\| \frac{N-k}{N}$$

and

$$\theta_k \in [0, 2\pi).$$

Figure 2 shows a configuration point represented with respect to the polar coordinate frame attached to the goal configuration. In this figure, it is notable that only θ_k determines the robot configuration since r_k is a fixed constant according to each sampling time. Therefore, a set of the candidate configuration points on each concentric circle to avoid collisions – we would call it as *free arc* – can be represented by the range of θ_k. In the subsequent subsections, we derive the procedure to obtain the closed form of free arc.

2.1 Initial arc included in the workspace

In most cases, a practical robot manipulator has its own workspace which restricts its joint motion to some region.

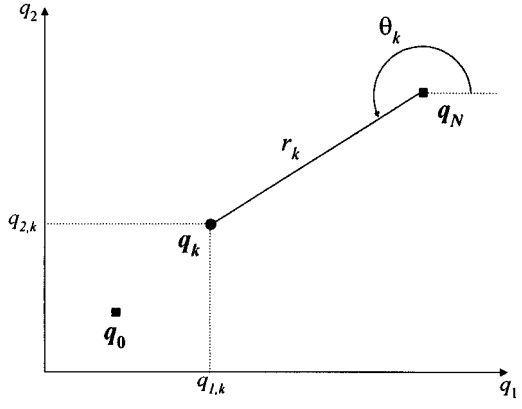


Fig. 2. The polar coordinate frame attached to the goal configuration.

Since a robot configuration outside its workspace is unrealizable, we should find initial arc included in the workspace, which is a pre-stage to construct the free arc.

If we denote the k-th concentric circle as C_k, C_k can be described as the following set of configuration points.

$$C_k = \{\boldsymbol{q}_k = (r_k \cos\theta_k + q_{1,N},\ r_k \sin\theta_k + q_{2,N}) \mid$$
$$r_k = \|\boldsymbol{q}_0 - \boldsymbol{q}_N\| \frac{N-k}{N} \text{ and } 0 \le \theta_k < 2\pi\}. \tag{4}$$

From now on, we abbreviate (4) to the following form as

$$C_k = \{\boldsymbol{q}_k \mid 0 \le \theta_k < 2\pi\} \tag{5}$$

since $\boldsymbol{q}_k$ is determined from the corresponding r_k and θ_k, where r_k is a fixed constant. This circle should be cut off with the boundary of the workspace given as

$$CS = \{\boldsymbol{q} \mid q_{i,min} \le q_i \le q_{i,max}, \quad i=1, 2\}. \tag{6}$$

Combining (4) and (6), the initial arc included in the workspace, IA_k, is derived as

$$\begin{aligned} IA_k &= CS \cap C_k \\ &= \{\boldsymbol{q}_k \mid \theta_{CS}^{min}(k) \le \theta_k \le \theta_{CS}^{max}(k)\}. \end{aligned} \tag{7}$$

Here, $\theta_{CS}^{min}(k)$ and $\theta_{CS}^{max}(k)$ are calculated by finding the range of θ_k that satisfies the following inequalities at the same time.

$$\begin{aligned} q_{1,min} &\le r_k \cos\theta_k + q_{1,N} \le q_{1,max}, \\ q_{2,min} &\le r_k \sin\theta_k + q_{2,N} \le q_{2,max}. \end{aligned} \tag{8}$$

The concept of the initial arc is depicted in Fig. 3.

2.2 Free arc avoiding collision with C-obstacles

By restricting the robot configuration at each sampling time to IA_k derived in the previous section, we can represent time-varying C-obstacles only through checking the collision of the robot along IA_k. If we select an appropriate resolution level of θ_k, we can find the set of configuration points occupying the C-obstacles at each sampling time as follows:

$$OS_k = \{\boldsymbol{q}_k \mid \theta_{OS}^{min}(k, i) \le \theta_k \le \theta_{OS}^{max}(k, i), \quad i=1, 2, \ldots, P(k)\} \tag{9}$$

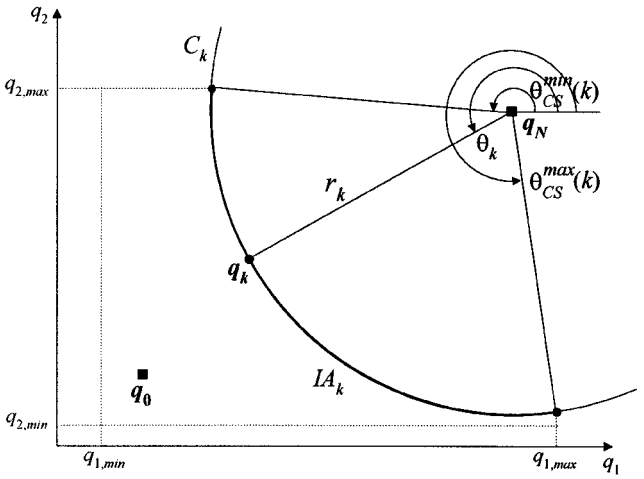


Fig. 3. Initial arc IA_k included in the workspace.

https://doi.org/10.1017/S0263574799001113 Published online by Cambridge University Press

where $P(k)$ denotes the number of collision regions at k-th sampling time. Then, the free arc avoiding collision with C-obstacles is derived as

$$FA_{C,k}=\{\boldsymbol{q}_k|\boldsymbol{q}_k\in IA_k-OS_k\}. \tag{10}$$

Combining (7) and (9), $FA_{C,k}$ is arranged as

$$FA_{C,k}=\{\boldsymbol{q}_k|\ \theta_{CS}^{min}(k)\le\theta_k<\theta_{OS}^{min}(k,1),\theta_{OS}^{max}(k,1)\le\theta_k<\theta_{OS}^{min}(k,2),\ \ldots,\ \theta_{OS}^{max}(k,P(k))<\theta_k\le\theta_{CS}^{max}(k)\}. \tag{11}$$

Note that the following equations are always satisfied since $OS_k\subset IA_k$ for $\forall k$.

$$\begin{aligned}&\theta_{CS}^{min}(k)\le\theta_{OS}^{min}(k,i),\ \text{for } i=1,2,\ldots,P(k)\\ &\qquad\text{and } k=0,1,\ldots,N-1,\\ &\theta_{OS}^{max}(k,i)\le\theta_{CS}^{max}(k),\ \text{for } i=1,2,\ldots,P(k)\\ &\qquad\text{and } k=0,1,\ldots,N-1,\end{aligned} \tag{12}$$

To guarantee a safe motion of the robot more completely, we reduce the range of free arcs by introducing marginal degree, θ_{margin}. Then, (11) is rearranged into the following form.

$$FA_{C,k}=\{\boldsymbol{q}_k|\ \theta_{CAC}^{min}(k,i)<\theta_k<\theta_{CAC}^{max}(k,i),\ i=1,2,\ldots,P(k)+1\} \tag{13}$$

where

$$\theta_{CAC}^{min}(k,1)=\theta_{CS}^{min}(k),$$

$$\theta_{CAC}^{min}(k,i)=\theta_{OS}^{max}(k,i-1)+\theta_{margin}\ \text{for } i=2,3,\ldots,P(k)+1,$$

and

$$\theta_{CAC}^{max}(k,P(k)+1)=\theta_{CS}^{max}(k),$$

$$\theta_{CAC}^{max}(k,i)=\theta_{OS}^{min}(k,i)-\theta_{margin}\ \text{for } i=1,2,\ldots,P(k).$$

When there are two collision regions at t_k, the free arc $FA_{C,k}$ with $\theta_{margin}=0$ is depicted as Fig. 4.

2.3 Consideration of reachable constraint

Because a robot has its velocity constraint, its motion during each sampling period is restricted physically. In our method, it is difficult to consider the velocity constraint in the entire path since the robot configuration is restricted to free arc at each sampling time. Instead, we impose the reachable constraint from a reference point on robot motion at each sampling time and so it is possible to satisfy the velocity constraint only if the velocity bound is assigned to an acceptably low value.

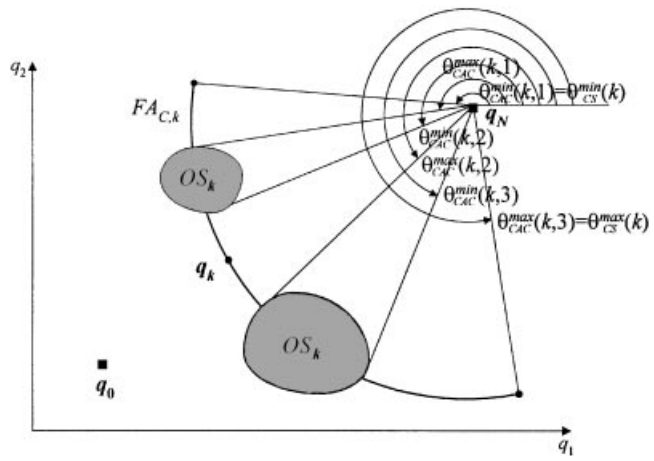


Fig. 4. Free arc $FA_{C,k}$ avoiding collision with C-obstacles.

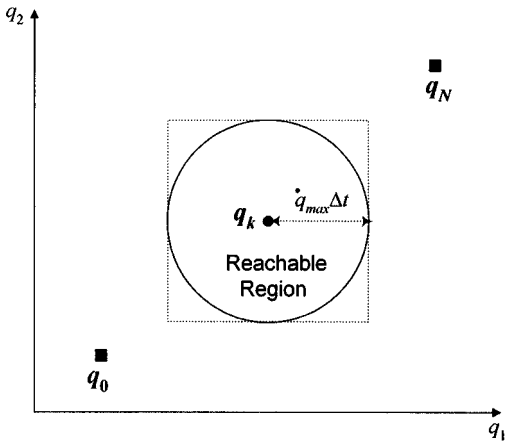


Fig. 5. The region in which the robot can move during Δt: Reachable region.

Let us denote the velocity bound of each joint as $\dot{q}_{max}$, then the robot manipulator can travel from $\boldsymbol{q}_k$ within the following range in the C-space during each sampling period, Δt.

$$|q_{i,k}-q_{i,k+1}|\le\dot{q}_{max}\Delta t,\ i=1,2. \tag{14}$$

The reachable region is drawn as a rectangle that has the center point at $\boldsymbol{q}_k$, and of which edge length is $2\dot{q}_{max}\Delta t$. This rectangle can be approximated by a circle determined from (15) and Fig. 5 shows this approximation.

$$\|\boldsymbol{q}_k-\boldsymbol{q}_{k+1}\|\le\dot{q}_{max}\Delta t. \tag{15}$$

From the above inequality, the reachable region that the robot manipulator can reach at $t=t_k$ from $\boldsymbol{q}_{k-1}$ is the inner-circle of radius of $\dot{q}_{max}\Delta t$, centered at q_{k-1}. The region is described as the following set, $R_{reach,k}$.

$$R_{reach,k}=\{\boldsymbol{q}_k|\|\boldsymbol{q}_{k-1}-\boldsymbol{q}_k\|\le\dot{q}_{max}\Delta t\}. \tag{16}$$

Thus, RA_k denoting the arc satisfying the reachable constraint is derived as

$$RA_k=C_k\cap R_{reach,k}. \tag{17}$$

If we analyze RA_k geometrically in the C-space as Fig. 6, we

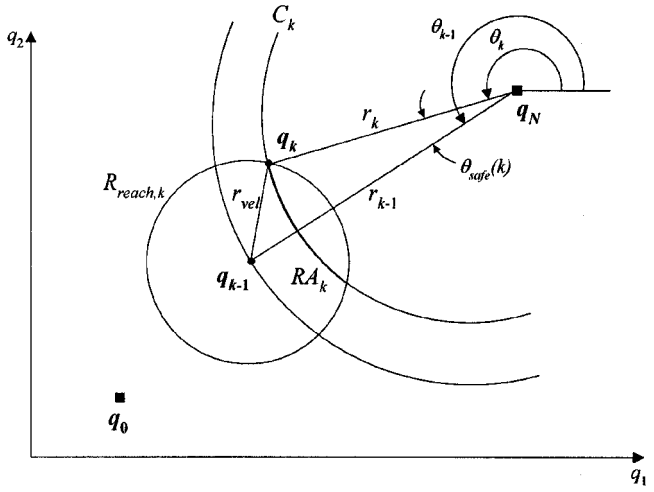


Fig. 6. Reachable arc RA_k satisfying reachable constraint.

https://doi.org/10.1017/S0263574799001113 Published online by Cambridge University Press

find that it is transformed to the linear constraint on θ_{k-1} and θ_k as follows.

$$RA_k = \{\boldsymbol{q}_k \| \theta_{k-1} - \theta_k | \le \theta_{safe}(k)\}. \quad (18)$$

Here, $\theta_{safe}(k)$ is calculated using *the law* of *cosines*. If we denote the radius of the reachable region as r_{vel}, then

$$r_{vel}^2 = r_{k-1}^2 + r_k^2 - 2r_{k-1}r_k \cos\ \theta_{safe}(k). \quad (19)$$

So,

$$\theta_{safe}(k) = \cos^{-1}\left(\frac{r_{k-1}^2 + r_k^2 - r_{vel}^2}{2r_{k-1}r_k}\right) \quad (20)$$

where

$$r_{vel} = \dot{q}_{max}\Delta t. \quad (21)$$

3. CONSTRUCTION OF SOCAP AND COMPLETENESS ANALYSIS

The composite free arc FA_k at k-th sampling time is derived from the results in the previous section as follows.

$$\begin{aligned} FA_k &= FA_{C,k} \cap RA_k \\ &= \{\boldsymbol{q}_k | \theta_{CAC}^{min}(k, i) < \theta_k < \theta_{CAC}^{max}(k, i),\ i = 1, 2, \ldots, P(k)+1 \\ &\qquad \text{and } | \theta_{k-1} - \theta_k | \le \theta_{safe}(k)\}. \end{aligned} \quad (22)$$

Since the free arc is a set of candidate configuration points to avoid collisions, the motion planning problem for obstacle avoidance is reduced to finding the sequence of points on the free arcs $\{FA_0, \ldots, FA_k, \ldots, FA_N\}$. If only the sequence of free arcs is obtained, the sequence of $\{\theta_0, \ldots, \theta_k, \ldots, \theta_N\}$ can be determined appropriately from it. Subsequently, the collision-free path $\{\boldsymbol{q}_0, \ldots, \boldsymbol{q}_k, \ldots, \boldsymbol{q}_N\}$ in the C-space is easily formed from the sequence $\{\theta_0, \ldots, \theta_k, \ldots, \theta_N\}$.

By the proper choice of performance index as a function of $\theta_0, \ldots, \theta_k, \ldots, \theta_N$, sub-optimality is considered when the sequence $\{\theta_0, \ldots, \theta_k, \ldots, \theta_N\}$ is determined from the sequence of free arcs $\{FA_0, \ldots, FA_k, \ldots, FA_N\}$. Thus, the Sub-Optimal Collision Avoidance Problem (SOCAP) is constructed as follows.

$$\begin{aligned} \textit{Find} \quad & \{\theta_0, \ldots, \theta_k, \ldots, \theta_N\}\ s.t. \\ \textit{minimize} \quad & G(\theta_0, \ldots, \theta_k, \ldots, \theta_N) \\ \textit{subject to} \quad & \theta_{CAC}^{min}(k, i) < \theta_k < \theta_{CAC}^{max}(K, i) \\ & | \theta_{k-1} - \theta_k | \le \theta_{safe}(k)\ for\ i = 1,2, \ldots, P(k)+1. \end{aligned}$$

The index i on linear inequality constraint suggests the possibility of splitting the above optimal problem into several independent problems. With this aim, we define *path tree*, PT_i, as a sequence of free arcs having only one arc segment at each sampling time as follows:

$$PT_i = \{FA_{C,0}(i_0), \ldots, FA_{C,k}(i_k), \ldots, FA_{C,N-1}(i_{N-1})\} \quad (23)$$

where

$$FA_{C,k}(i) = \{\boldsymbol{q}_k | \theta_{CAC}^{min}(k, i) < \theta_k < \theta_{CAC}^{max}(k, i)\} \quad (24)$$

and satisfying (1) $\boldsymbol{q}_0 \in FA_{C,0}(i_0)$, and (2) common range of θ_{j-1} in $FA_{C,j-1}(i_{j-1})$ and θ_j in $FA_{C,j}(i_j)$ should exist for $j = 1, \ldots, N-1$. The first condition means that only the free arc segment of $FA_{C,0}$ including the initial configuration is valid, and the second condition keeps a robot from moving between adjacent free arc segments of which common range of θ_k is null, since transition between such free arc segments may result in collision. Then, SOCAP is formulated for each PT_i as an optimal problem with linear inequality constraint on state variable θ_k. We compare the solutions from the SOCAPs for each PT_i with one another and select the final solution $\{\theta_0, \ldots, \theta_k, \ldots, \theta_N\}$ with the minimal performance index among them. The collision-free path $\{\boldsymbol{q}_0, \ldots, \boldsymbol{q}_k, \ldots, \boldsymbol{q}_N\}$ is calculated from the final solution $\{\theta_0, \ldots, \theta_k, \ldots, \theta_N\}$.

Unfortunately, a question arises when the solution of the SOCAPs does not exist. Can we say definitely that there is no collision-free path only if we cannot find the solution of the SOCAPs? This is a serious problem about the completeness of the proposed approach. Indeed, our approach cannot guarantee completeness since discretization of the C-space is involved and furthermore it restricts the search space to the free arcs instead of the whole C-space. The former reason forms a general problem that other algorithms involving any discretization of the C-space cannot avoid. Like in the other algorithms, we can say that our approach is complete up to the resolution of discretization, i.e. resolution-complete in theory, if the search space is not restricted. However, restriction of search space may cause a very serious bottleneck to the completeness of our approach, i.e. our approach may not find a collision-free path even if such a path exists. To address this problem as much as possible, we introduce backward motion of a robot (*backward* indicates the direction of becoming far from the goal configuration in the C-space) through examining several cases in detail.

Figure 7(a) shows a situation that we cannot find a solution of SOCAP, even if path tree does exist, because the reachable constraint cannot be satisfied. This situation may happen when a large obstacle exists in the vicinity of initial or goal configuration. Without the reachable constraint, a collision-free path is obtained by solving the SOCAP. However, consideration of the reachable constraint forces us to miss a legal collision-free path. To correct this type of misjudgement, we allow the robot to move backward from the initial configuration. Adding a new free arc, of which radius is larger than the initial free arc, enables this backward motion. A new free arc is added repeatedly until we can find a solution of SOCAP, i.e. the robot retreats enough to go around the large obstacle. Inevitably, adding one free arc results in increasing the travel time of the robot by two sampling periods. One may argue that this backward motion is inefficient since it is possible to move along a shorter path only if the robot motion along the same free arc is allowed. We should remind him of the major assumption of our approach – the robot configuration at each sampling time should lie on each free arc – to represent time-varying C-obstacles efficiently.

The second case in Fig. 7(b) indicates that there exists a trap surrounding the initial configuration. The difference with the first case is that even the initial path tree does not exist. This trap can be escaped by backward motion – the

https://doi.org/10.1017/S0263574799001113 Published online by Cambridge University Press

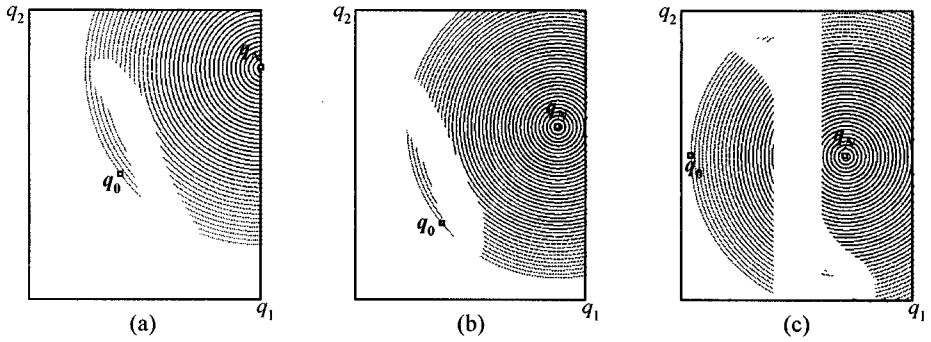


Fig. 7. Example cases for introducing backward motion.

same procedure as the first case. Searching algorithm for valid path tree is repeated at each step until the path tree is built successfully. If the SOCAP for the found path tree is not solvable, additional backward motion continues again. The final situation we should consider is that a large obstacle barrier is located between the initial and the goal configuration. This situation is depicted in Fig. 7(c). In this case, we can know that no collision-free path exists, by applying the backward motion repeatedly until there remains no more free arc to add in the workspace. The result returns failure of the approach since legal path tree cannot be built to the end, which means that there is no collision-free path. Generally, the backward motion algorithm to search for a collision-free path to the end is described as follows:

Procedure: This procedure addresses the iterative steps in checking the success of the proposed approach.

Given travel time T_f, sampling period Δt;
Determine the number of sampling $N = T_f/\Delta t$.
Discretize the C-space with N concentric circles.

repeat: Construct the free arcs $\{FA_k, k=0, 1, \ldots, N-1\}$.
Build the path tree set $\{PT_i\}$;
if $\{PT_i\} = \emptyset$, goto *backward*.
else solve the SOCAPs for each PT_i;
if any feasible solution does not exist, goto *backward*.
else return *success*.

backward: if there remains no more free arc to add in the workspace,
then return *failure*.
else one-step backward motion by adding one free arc, which results in increasing the travel time by $2\Delta t$, i.e. $N \leftarrow N+2$. Goto *repeat*.

Of course, this backward motion cannot enable us to say that our approach is complete since (1) the shape of time-varying C-obstacles varies at each backward motion, and (2) traversing along the same free arc is not allowed. However, we can check the success of our approach to the end with the aid of backward motion. We believe that the case where our approach misses legal paths even with backward motion seldom happens in most ordinary situations, except for the very difficult problem such as bottleneck.

4. COMPARISON OF COMPUTATION TIME WITH SLICE PROJECTION METHOD

In this section, we compare computation time of our approach to represent time-varying C-obstacles with that of an existing method – *slice projection* – in widespread use. Lozano-Perez[1] presented *slice projection* – the idea of quantizing joint angles to describe the collision-free space. He approximated the C-obstacles for an *m-d.o.f.* manipulator by sets of $m-1$ dimensional slices, recursively built up from one-dimensional slices. This method can approximate the collision-free space efficiently, however, the completeness is not guaranteed like most of other algorithms involving any discretization of the C-space.

First, we derive computation time of each method to represent C-obstacles in a time-varying environment analytically. Let us consider a 2-*d.o.f.* manipulator for simplicity. To use the *slice projection* method, it is required to determine the resolution level of each joint. Let Δq_1 and Δq_2 be the resolution level of the first and the second joint, respectively. If the number of sampling is N during total travel time and the average run time of a collision checking routine is T_c, then the required time to compute time-varying C-obstacles is bounded as

$$\frac{Q_1}{\Delta q_1} \cdot (N-1) \cdot T_c \le t \le \frac{Q_1}{\Delta q_1} \cdot \left(1 + \frac{Q_2}{\Delta q_2}\right) \cdot (N-1) \cdot T_c \quad (25)$$

where Q_i means the interval of workspace, that is, $Q_i = q_{i,max} - q_{i,min}$. It takes the minimum time to compute C-obstacles when there is no collision-free space (unrealistic case). If the first link does not collide with any obstacle, the maximum time is required. Moreover, an additional approximation of the computed C-obstacle by a convex polygon may be necessary according to the algorithm to search for a collision-free path.

On the other hand, only the resolution level of free arc should be determined in our approach. If we denote this resolution level as $\Delta\theta$, computation time required to represent time-varying C-obstacles is bounded as

$$\frac{90^\circ}{\Delta\theta} \cdot (N-1) \cdot T_c \le t \le \frac{360^\circ}{\Delta\theta} \cdot (N-1) \cdot T_c \quad (26)$$

The minimum time case happens when the goal configuration is located at the corner of the workspace, so that it is sufficient to examine only the quarter portion of all the

https://doi.org/10.1017/S0263574799001113 Published online by Cambridge University Press

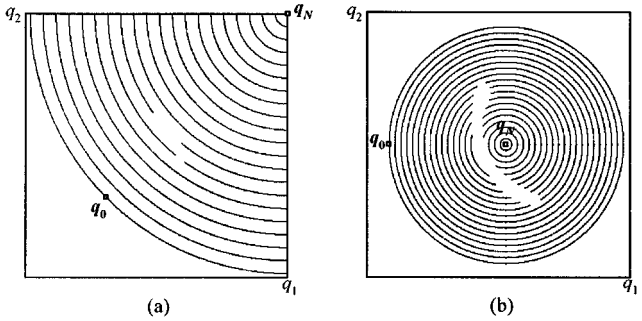


Fig. 8. (a) The minimum time case and (b) the maximum time case to represent time-varying C-obstacles in the proposed method ($N=20$).

concentric circles. On the contrary, the maximum time case indicates that all the concentric circles centered at the goal configuration are included completely in the workspace. Figure 8 shows an example of two cases.

Comparing (25) and (26), it is obvious that computation time of our approach is much shorter than that of *slice projection* in most cases only if the workspace and the resolution level of free arc are not extremely small. To ascertain this analysis, four different cases of a two-robot system composed of two 2-*d.o.f.* planar robots are examined. Figure 9 shows the example cases. In these examples, *robot1* should avoid collisions with *robot2*. The two robots have the same link lengths, $l_1=l_2=350(mm)$. In Case I and Case II, the distance between two robots is 1,200(*mm*), so that the first link of *robot1* does not collide with *robot2* at all. In Case III and Case IV, even the first link of *robot1* may fall into potential collisions with *robot2* since the distance between two robots is reduced to 900(*mm*), and therefore it must be difficult to avoid collision with each other. The initial and goal configurations of *robot2* are fixed as ($-90°$, 60°) and (60°, $-30°$), respectively, while those of *robot1* are changed at each case in order to produce the similar cases to those shown in Fig. 8. The resolution level of each joint in *slice projection* and the resolution level of free arc are set to the same value as 0.5°. The common data for the

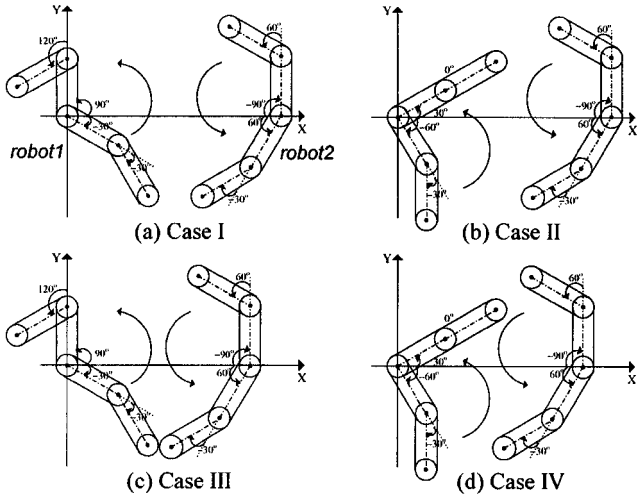


Fig. 9. Four example cases.

example system is outlined in Table I.

The C-obstacles constructed by the proposed method are depicted for each case in Fig. 10 and Table II shows the comparison result between the two methods. Here, T_{FA} is the time required to compute the C-obstacles using our approach and T_{SP} is the time using *slice projection*. From the result, we can confirm that the proposed method saves much more computation time than *slice projection* for most cases. As expected, computation time of our approach does not depend on the distance between two robots, i.e. the size of potential collision region. Instead, it depends on the location of initial and goal configuration. It is faster to compute the C-obstacles of Case I and III – similar cases to the minimum time case in Fig. 8(a) – than Case II and IV. On the other hand, more time is needed in the *slice projection* method when the size of potential collision region is smaller. T_{FA} can be reduced if more powerful hardware and efficient coding are adopted. In conclusion, our approach provides a much more efficient method to represent time-varying C-obstacles than *slice projection*.

Table I. Data for the example system

Symbol	Description	Value	
		robot1	*robot2*
$\boldsymbol{q}_0$	Initial configuration	Case I-IV	$(-90, 60)[deg.]$
$\boldsymbol{q}_N$	Goal configuration	Case I-IV	$(60, -30)[deg.]$
$[q_{1,min}, q_{1,max}]$	Workspace of joint 1	$[-120, 120][deg.]$	
$[q_{2,min}, q_{2,max}]$	Workspace of joint 2	$[-150, 150][deg.]$	
$\dot{q}_{max}$	Joint velocity bound	$120[deg./sec.]$	
l_1	The length of link 1	$350[mm]$	
l_2	The length of link 2	$350[mm]$	
D	Distance between two robots	Case I-IV	
$\Delta\theta$	Resolution level of free arc	$0.5[deg.]$	
Δq_i	Resolution level of each joint	$0.5[deg.]$	
θ_{margin}	Marginal degree for safe motion	$5.0[deg.]$	
N	Total number of sampling	40	
Δt	Sampling period	$0.1[sec.]$	

https://doi.org/10.1017/S0263574799001113 Published online by Cambridge University Press

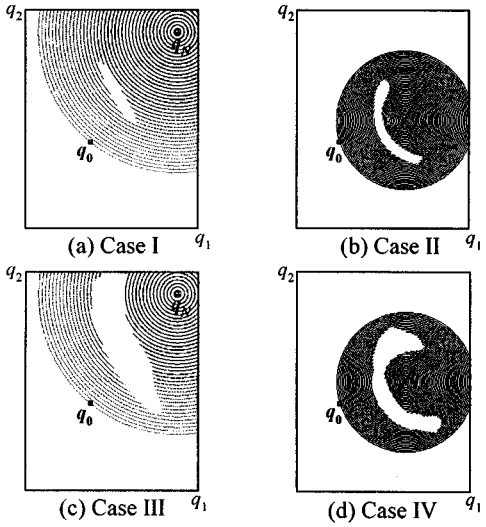


Fig. 10. C-obstacles constructed by the proposed method.

5. EXTENSIONS TO 3-D C-SPACE

Extensions of the proposed approach to the case of 3-dimensional C-space is outlined in this section. The method to find the free arc in 3-dimensional C-space is similar to that in the case of 2-dimensional C-space except that the spherical coordinate frame is attached to the goal configuration, instead of the polar coordinate frame. In the case of 3-dimensional C-space, we discretize the C-space of one robot uniformly with N concentric spheres centered at the goal configuration. In this discretized C-space, we assume that the robot configuration at each sampling time should lie on each concentric sphere.

In order to describe the spheres effectively, the spherical coordinate frame is attached to the goal configuration. Then, the configuration points, $\boldsymbol{q}_k=(q_{1,k},\ q_{2,k},\ q_{3,k})$, on the k-th concentric sphere is described as follows:

$$\begin{aligned} q_{1,k} &= r_k \cos\theta_k \sin\varphi_k + q_{1,N}, \\ q_{2,k} &= r_k \sin\theta_k \sin\varphi_k + q_{2,N}, \\ q_{3,k} &= r_k \cos\varphi_k + q_{3,N}, \end{aligned} \tag{27}$$

where

$$r_k = \|\boldsymbol{q}_0 - \boldsymbol{q}_N\| \frac{N-k}{N}$$

and

$$\theta_k \in [0,\ 2\pi), \quad \varphi_k \in [0,\ \pi).$$

Table II. Comparison of computation time between the proposed method and the *slice projection*

Case	$\boldsymbol{q}_0[deg.]$	$\boldsymbol{q}_N[deg.]$	$D[mm]$	$T_{FA}[sec.]$	$T_{SP}[sec.]$
I	$(-30,\ -30)$	(90, 120)	1,200	0.97	284
II	$(-60,\ -30)$	(30, 0)	1,200	2.37	285
III	$(-30,\ -30)$	(90, 120)	900	0.95	249
IV	$(-60,\ -30)$	(30, 0)	900	2.34	248

Note: Times are in seconds on a Pentium-PC (133MHz).

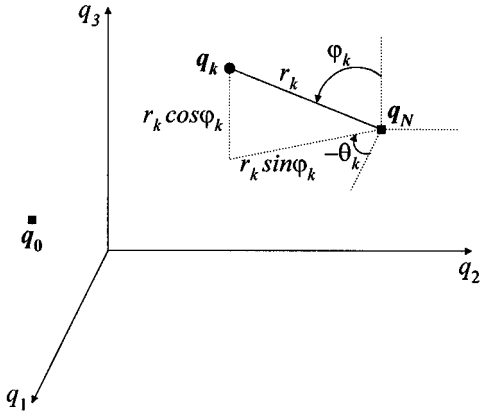


Fig. 11. The spherical coordinate frame attached to the goal configuration.

Figure 11 shows a configuration point represented with respect to the spherical coordinate frame attached to the goal configuration.

In the same way as 2.1, the initial arc included in the workspace is derived as

$$IA_k = CS \cap C_k \tag{28}$$

where

$$CS = \{\boldsymbol{q} \mid q_{i,min} \le q_i \le q_{i,max}, \quad i = 1,\ 2,\ 3\} \tag{29}$$

and

$$C_k = \{\boldsymbol{q}_k \mid 0 \le \theta_k < 2\pi,\ 0 \le \varphi_k < \pi\}. \tag{30}$$

CS is the workspace of the robot and C_k is the concentric sphere on which the configuration points at k-th sampling time should lie. IA_k is constructed by finding the range of $[\theta_k\ \varphi_k]$ that satisfies the following inequalities at the same time.

$$\begin{aligned} q_{1,min} &\le r_k \cos\theta_k \sin\varphi_k + q_{1,N} \le q_{1,max}, \\ q_{2,min} &\le r_k \sin\theta_k \sin\varphi_k + q_{2,N} \le q_{2,max}, \\ q_{3,min} &\le r_k \cos\varphi_k + q_{3,N} \le q_{3,max}. \end{aligned} \tag{31}$$

Here, we find the range of φ_k at first, and sample the range of φ_k at a specified resolution, $\Delta\varphi$. For each sampled value of φ_k, we find the range of θ_k satisfying the above inequalities. Then, we can build the free arc avoiding collision with C-obstacles by checking the collision of the robot for each sampled value of θ_k at a specified resolution, $\Delta\theta$.

In 3-dimensional C-space, the reachable region of a robot is described as a sphere of radius of $\dot{q}_{max}\Delta t$, instead of a circle. However, if the reachable region and the two concentric spheres, C_{k-1} and C_k, are projected onto q_1q_2 and q_2q_3 plane respectively, we can transform the reachable constraint into two linear constraints on $(\theta_{k-1},\ \theta_k)$ and $(\varphi_{k-1},\ \varphi_k)$ in the same way as 2.3. Therefore,

$$RA_k = \{\boldsymbol{q}_k \mid |\theta_{k-1} - \theta_k| \le \theta_{safe}(k) \text{ and } |\varphi_{k-1} - \varphi_k| \le \varphi_{safe}(k)\} \tag{32}$$

where

$$\theta_{safe}(k) = \cos^{-1}\left(\frac{r_{k-1}^2 + r_k^2 - r_{vel}^2}{2r_{k-1}r_k}\right) \tag{33}$$

https://doi.org/10.1017/S0263574799001113 Published online by Cambridge University Press

and

$$\varphi_{safe}(k) = \cos^{-1}\left(\frac{r_{k-1}^2 + r_k^2 - r_{vel}^2}{2 r_{k-1} r_k} \right). \quad (34)$$

Here, r_{k-1} and r_k are radii of the concentric spheres, C_{k-1} and C_k, respectively, and r_{vel} is the radius of the reachable region.

6. SIMULATION RESULTS

To show the efficacy of our approach, we simulated four situations treated in Section 4 and a two-robot system composed of two PUMA robots. In Case I and II, two path trees are organized and two SOCAPs are formulated for each path tree. We should decide the performance index $G(\theta_0, \ldots, \theta_k, \ldots, \theta_N)$ which we want to minimize while *robot1* moves to its goal configuration avoiding collision with *robot2*. Here, we adopted two performance measures – *the measure of manipulability* (*MOM*) suggested by Yoshikawa[27] and minimum moving distance. *MOM* is the quantitative measure of manipulating ability of robot arms in positioning and orienting the end-effectors. For a 2-*d.o.f.* planar robot manipulator, *MOM* is computed as

$$MOM = l_1 l_2 |\sin q_2| \quad (35)$$

where l_1, l_2 are link lengths and q_2 is the position of the second joint. See the reference[27] for a detailed derivation. Since l_1, l_2 are constants and we intend to maximize the summation of *MOM* at each sampling time, the SOCAP has the following form:

$$\begin{aligned} &\textit{Find} && \{\theta_0, \ldots, \theta_k, \ldots, \theta_N\}\ s.t. \\ &\textit{minimize} && -\Sigma_{k=1}^{N} |\sin\{r_k \sin\theta_k + q_{2,N}\}| \\ &\textit{subject to} && \theta_{CAC}^{min}(k) < \theta_k < \theta_{CAC}^{max}(k) \\ & && |\theta_{k-1} - \theta_k| \le \theta_{safe}(k)\ \ for\ \ k = 1, \ldots, N. \end{aligned}$$

To minimize moving distance of *robot1*, $G(\theta_0, \ldots, \theta_k, \ldots, \theta_N)$ should be replaced with $\Sigma_{k=1}^{N}(\theta_{k-1} - \theta_k)^2$ in the above SOCAP.

We solved the formulated SOCAPs using *Sequential Quadratic Programming* (*SQP*) method[28] and the results are shown in Figs. 12 and 13. Comparing two performance

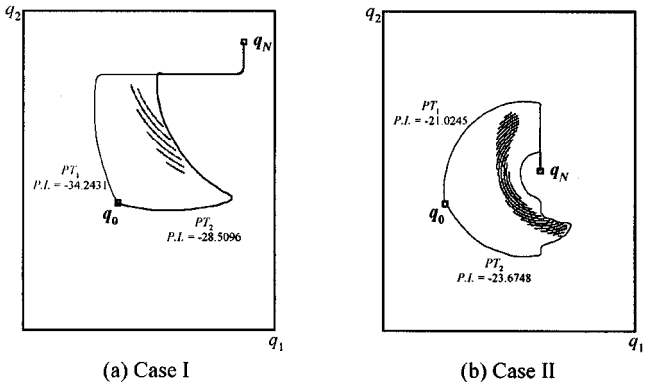


Fig. 12. Sub-optimal collision-free path maximizing *MOM* for Case I and II.

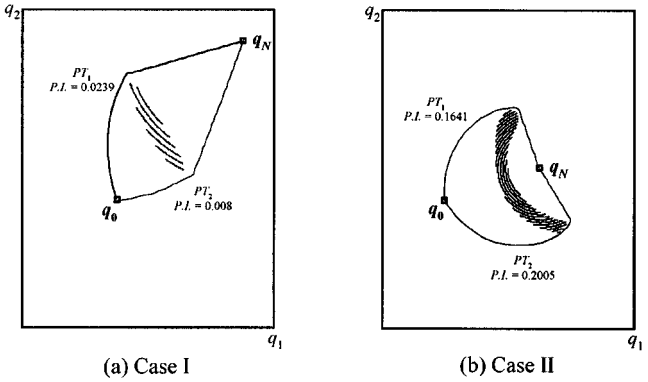


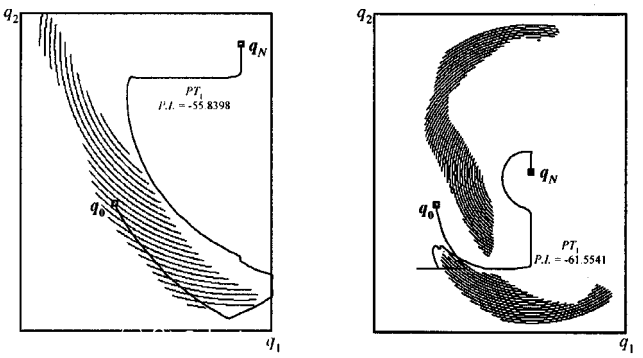
Fig. 13. Sub-optimal collision-free path minimizing moving distance for Case I and II.

indices from each SOCAP, the solution path from PT_1 in Case I and from PT_2 in Case II is selected as the final result path respectively when *MOM* is used as a performance index. On the other hand, the solution path from PT_2 in Case I and from PT_1 in Case II is selected as the final result path respectively when minimum moving distance is used as a performance index. It can be noted that the shape of solution paths is changed according to the performance index. In Fig. 12, the robot moves along the longer path than the path minimizing moving distance in Fig. 13 in order to maximize the summation of *MOM*. Of course, the above solution paths are not real optimal but only sub-optimal since the robot configuration is restricted to free arcs at each sampling time. However, it may be useful for some situations, e.g. the most important objective is to avoid collision with time-varying obstacles efficiently, and optimizing certain criteria is desired simultaneously.

In Case III and IV, two path trees and the SOCAPs for each path tree are constructed successfully. But, any feasible solution does not exist in the SOCAPs since the reachable constraint is not satisfied. This means that the two cases are very difficult situations to avoid collision because the distance between two robots is relatively short compared with link length. Thus, initial departure delay should be given to *robot1* if *robot1* path is prespecified. In our approach, backward motion explained in Section 3 allows us to find a collision-free path in these situations. After 13 step backward motion in Case III and 19 step in Case IV, we can find a feasible solution for each SOCAP. The solution paths are depicted in Fig. 14 and Fig. 15. In these figures, note that backward motion of the robot is collision-free even though the C-obstacles during backward and forward motion are overlapped simultaneously. These results show that the proposed approach can solve an easy problem for collision avoidance of a two-robot system efficiently, and in addition can cope with a very difficult problem with the aid of backward motion, although total travel time increases.

Finally, the proposed approach is applied to a two-robot system composed of two PUMA robots to generalize its efficacy to a 6-*d.o.f.* industrial robot. Although a PUMA robot has 6-*d.o.f.*, it can be reasonably approximated as a 3-*d.o.f.* robot for collision avoidance problem since the last three links composing end-effector can be modeled as a part of link 3. In this simulation, real link size and workspace

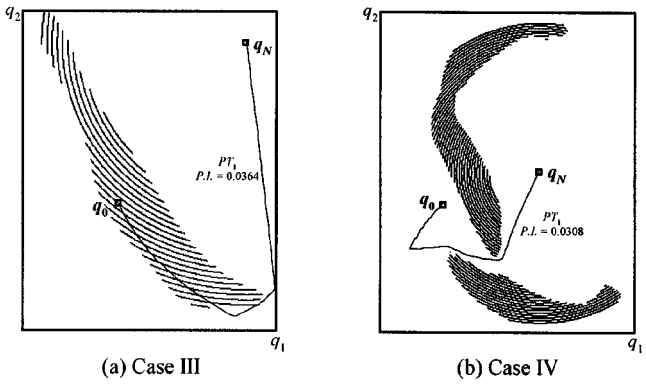
https://doi.org/10.1017/S0263574799001113 Published online by Cambridge University Press



(a) Case III (b) Case IV

Fig. 14. Sub-optimal collision-free path maximizing *MOM* for Case III and IV.

area are used in modeling the robot, and distance between two PUMA robots is set to 1,500(*mm*). As explained in the previous section, we constructed C-obstacle and solved two SOCAPs in order to minimize the moving distance of *robot1*. Two solution paths are depicted in Fig. 16 and the solution path from PT_2 is selected as the final result path. Computation time to represent the time-varying C-obstacles is 165(*sec*.) when 3° is used as the resolution level of free



(a) Case III (b) Case IV

Fig. 15. Sub-optimal collision-free path minimizing moving distance for Case III and IV.

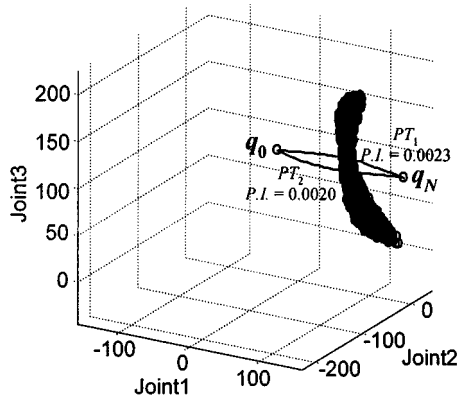


Fig. 16. Sub-optimal collision-free path minimizing moving distance of a PUMA robot.

arc. This computation time is even shorter than the case of 2-dimensional C-space using *slice projection*.

7. CONCLUDING REMARKS

This paper presents an efficient approach to the collision avoidance problem of a two-robot system. Unlike most algorithms reported so far, it is not required to decompose the problem into a path planning problem for stationary obstacle avoidance and a velocity planning problem for time-varying obstacle avoidance. Thus, we can find a collision-free path to avoid both the stationary and the time-varying obstacles (including the other robot) in the same stage. Instead of sacrificing completeness, our approach gains more efficiency than an existing method by providing an effective way to represent time-varying C-obstacles without building CT-space. Moreover, it has an advantage that computation time to represent obstacles and search for a collision-free path is not affected by the complexity of an obstacle model. To cope with the *completeness* problem, we introduced backward motion and verified its ability to check the success of our approach to the end.

Although our approach is primarily developed for a two-robot system, it is applicable to various collision-free motion planning problems of a single robot manipulator, point robots, and mobile robots with stationary and time-varying obstacles. However, there remains a few subjects that require further investigation. Generally, efficiency of a path planning algorithm depends on both the representation of C-space and the path searching strategy. Even though we provided an efficient way to represent time-varying C-obstacles by decreasing the number of collision detection computations, we should use an existing method – *Sequential Quadratic Programming* – to search for a collision-free path. We believe that a less time-consuming search algorithm can be devised if sub-optimality is not needed. Another problem to be mentioned is the curse of dimensionality. In this paper, we considered only two and three dimensional C-space, maintaining that nearly all industrial manipulators can be approximated in two or three dimensional C-space for a collision avoidance problem, and we verified our assertion through a simulation for a two-PUMA robot system. But, our approach should be refined and generalized in order to apply it to the high dimensional C-space, since 3-dimensional C-space is not enough to cover various robot types. Indeed, it is certainly not an easy task to construct free arc in high dimensional C-space, since the geometrical coordinate frame attached to the goal configuration does not exist. Hence, our future research is mainly directed to the generalization of our approach to the high dimensional C-space.

References

1. T. Lozano-Perez, “A simple motion-planning algorithm for general robot manipulators” *IEEE Trans. Robotics and Automation* **RA-3**, No. 3, 224–238 (June, 1987).
2. J.-C. Latombe, *Robot Motion Planning* (Kluwer Academic Publishers, Boston, 1991).
3. Y.K. Hwang and N. Ahuja, “Gross motion planning-A survey” *ACM Computing Surveys* **24**, No. 3, 219–291 (Sept., 1992).
4. N.S.V. Rao, “Robot navigation in unknown generalized polygonal terrains using vision sensors” *IEEE Trans. Syst.*,

https://doi.org/10.1017/S0263574799001113 Published online by Cambridge University Press

Man, *Cybern*. **25**, No. 6, 947–962 (June, 1995).
5. H.R. Beom and H.S. Cho, "A sensor-based navigation for a mobile robot using fuzzy logic and reinforcement learning" *IEEE Trans*. *Syst*., *Man*, *Cybern*. **25**, No. 3, 464–477 (Mar., 1995).
6. K. Sun and V. Lumelsky, "Path planning among unknown obstacles: The case of a three-dimensional cartesian arm" *IEEE Trans. Robotics and Automation* **8**, No. 6, 776–786 (Dec., 1992).
7. D. Challou, D. Boley, M. Gini and V. Kumar, "A parallel formulation of informed randomized search for robot motion planning problems" *IEEE Int. Conf. on Robotics and Automation* (1995) pp. 709–714.
8. L.E. Kavraki, P. Svestka, J.-C. Latombe and M.H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces" *IEEE Trans. Robotics and Automation* **12**, No. 4, 566–580 (Aug., 1996).
9. K. Gupta and X. Zhu, "Practical motion planning for many degrees of freedom: A novel approach within sequential framework" *IEEE Int. Conf. on Robotics and Automation* (1994) pp. 2038–2043.
10. J. Barraquand, B. Langlois and J.-C. Latombe, "Numerical potential field techniques for robot path planning" *IEEE Trans*. *Syst*., *Man*, *Cybern*. **22**, No. 2, 224–241 (Mar./Apr., 1992).
11. M. Erdmann and T. Lozano-Perez, "On multiple moving objects" *IEEE Int. Conf. on Robotics and Automation* (1986) pp. 1419–1424.
12. K. Fujimura and H. Samet, "A hierarchical strategy for path planning among moving obstacles" *IEEE Trans*. *Robotics and Automation* **5**, No. 1, 61–69 (Feb., 1989).
13. C.L. Shih, T.-T. Lee and W. A. Gruver, "A unified approach for robot motion planning with moving polyhedral obstacles" *IEEE Trans*. *Syst*., *Man*, *Cybern*, **20**, No. 4, 903–915 (July/Aug., 1990)
14. N.Y. Ko, B.H. Lee and M.S. Ko, "An approach to robot motion planning for time-varying obstacle avoidance using the view-time concept" *Robotica* **11**, Part 4, 315–327 (1993).
15. O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots" *Int. J. Robotics Research* **5**, No. 1, 90–98 (Spring, 1986).
16. R. Volpe and P. Khosla, "Manipulator control with superquadric artificial potential functions: Theory and experiments" *IEEE Trans*. *Syst*., *Man*, *Cybern*. **20**, No. 6, 1423–1436 (Nov./Dec., 1990).
17. J.-O. Kim and P. Khosla, "Real-time obstacle avoidance using harmonic potential functions" *IEEE Trans. Robotics and Automation* **8**, No. 3, 338–349 (June, 1992).
18. J. Guldner, V.I. Utkin, H. Hashimoto and F. Harashima, "Obstacle avoidance in R^n based on artificial harmonic potential fields" *IEEE Int. Conf. on Robotics and Automation* (1995) pp. 3051–3056.
19. E. Freund and H. Hoyer, "Real-time pathfinding in multirobot systems including obstacle avoidance" *Int. J. Robotics Research* **7**, No. 1, 42–70 (Feb., 1988).
20. B.H. Lee and C.S.G. Lee, "Collision-free motion planning of two robots" *IEEE Trans*. *Syst*., *Man*, *Cybern*. **SMC-17**, No. 1, 21–32 (Jan./Feb., 1987).
21. C. Chang, M.J. Chung and B.H. Lee, "Collision avoidance of two general robot manipulators by minimum delay time" *IEEE Trans*. *Syst*., *Man*, *Cybern*. **24**, No. 3, 517–522 (Mar., 1994).
22. Y. Shin and Z. Bien, "Collision-free trajectory planning for two robot arms" *Robotica* **7**, Part 3, 205–212 (1989).
23. Z. Bien and J. Lee, "A minimum-time trajectory planning method for two robots" *IEEE Trans. Robotics and Automation* **8**, No. 3, 414–418 (June, 1992).
24. J. Lee, "A dynamic programming approach to near minimum-time trajectory planning for two robots" *IEEE Trans*. *Robotics and Automation* **11**, No. 1, 160–164 (Feb., 1995).
25. K. Kant and S.W. Zucker, "Toward efficient trajectory planning: The path-velocity decomposition" *Int. J. Robotics Research* **5**, No. 3, 72–89 (Fall, 1986).
26. T. Lozano-Perez and M.A. Wesley, "An algorithm for planning collision-free paths among polyhedral obstacles" *Commun*. *ACM* **22**, No. 10, 560–570 (Oct., 1979).
27. T. Yoshikawa, "Manipulability of robotic mechanisms" *Int. J. Robotics Research* **4**, No. 2, 3–9 (Summer, 1985).
28. P.E. Gill, W. Murray and M.H. Wright, *Practical Optimization* (Academic Press, London, 1981).

https://doi.org/10.1017/S0263574799001113 Published online by Cambridge University Press