Proceedings of the 28th Conference
on Decision and Control
Tampa, Florida • December 1989

MINIMUM TIME TRAJECTORY PLANNING FOR DUAL ROBOT SYSTEMS[1]

Kang G. Shin and Qin Zheng
Real-Time Computing Laboratory
Department of Electrical Engineering and Computer Science
The University of Michigan
Ann Arbor, Michigan 48109-2122.

ABSTRACT

This paper addresses the problem of planning minimum time collision-free trajectories for a dual robot system. It is shown that this problem can be solved by first planning the minimum time trajectory for each robot without considering the existence of the other and then combining the two individual trajectories. This combination is accomplished by delaying one of the two trajectories in such a way that the time required for the coordinated motion of the two robots is minimized while avoiding collision between them. The optimality of the proposed method is rigorously proved and the practical aspects of its realization are discussed. An example is also presented to demonstrate our main idea.

1. INTRODUCTION

In a mult-robot system, robots' motion planning is often divided into two hierarchical levels: *task planning* and *spatial planning*. Given a specific task as input, the task planner divides each task into a series of actions which are then assigned to each robot. At the task-planning level, it is essential to ensure a correct sequence of robot actions so that the task will be completed successfully and efficiently. Many researchers addressed this problem [7,11,6], mainly based on the planning methods in artificial intelligence. Because of the lack of information about how long it will take to perform an action and how the action will be performed, the resulting plans are usually not optimal.

Once the actions are assigned to each robot, the spatial planner determines the movements of each robot to execute these actions. The key problem associated with a spatial planner is how collision-free, optimal (in some sense) paths and trajectories can be found. Note that robot-path planning is concerned with the determination of a geometric path in a three-dimensional workspace along which the robot will move, while robot-trajectory planning determines the robot's position and velocity along the geometric path as functions of time. At the spatial planning level, an optimal solution is often desirable for various reasons, such as productivity increase and quality assurance.

The minimum time path planning for a single robot subject to the constraints on accurator torques/forces has been a long-standing problem. The main difficulty of this problem is that the dynamics of a robot are highly nonlinear and coupled, and the classical optimal control theory cannot be applied directly. One way to overcome this difficulty is to decompose the problem into two sequential subproblems: geometric path planning first and then trajectory planning [1,9,10]. This path-trajectory decomposition has made the path planning problem tractable, although the resulting solution may not be overall optimal. One such optimal trajectory planning method, called the *Perturbation Trajectory Improvement Algorithm* (PTIA), was developed by Shin and Mckay [10] for robust trajectory planning, which will, due to its simplicity, be used in this paper for planning the trajectories of two robots working in the same workspace. It is worth noting that based on this path-trajectory decomposition, Sahar and Hollerbach [8] developed an overall optimal path planning method using joint-space tessellation and a graph search. However, their method suffers the problem of computational explosion when the grid resolution is high. Use of coarser grids for this method will degrade the optimality of their solution.

When more than one robot work in the same workspace, it is very difficult to use the path-trajectory decomposition (that applies to the single robot case), since collision avoidance must be taken into consideration for trajectory planning. This difficulty can be seen by the fact that intersection of two robots' paths may, or may not, lead to a collision, depending on when each robot passes through the intersecting point. (This difficulty does not exist for the single robot case). Various approaches to the problem of planning two robots' paths and trajectories have been proposed. One method is to assign priorities to robots and give the right-of-way to the robot with higher priority. There are several ways to realize this type of collision avoidance, including:

- Letting one robot move out of a common workspace before the other enters [7,11,6]. This idea is very simple, but may be inefficient and its applicability is limited (e.g. assembly tasks with two robots are not possible since the two robots must sometimes be in the same workspace).
- Searching for an optimal collision-free path of the second robot in the configuration space–time [5]. The application of this method is limited due to its computational complexity.
- Real-time collision-free pathfinding [2]. Although this kind of collision avoidance is highly desirable, the algorithm proposed in [2] can hardly be realized in practice because it is based on the assumption that the second robot can move in any direction as fast as necessary for collision avoidance, which may require very large, unrealizable actuator torques/forces.

Kant and Zucker [4] proposed an approach to solving the optimal collision-free trajectory planning problem by decomposing it into two subproblems: (1) planning a path to avoid collision with static obstacles, and (2) planning the velocity along the path to avoid collision with moving obstacles. Like in the single robot case, the results obtained may not be overall optimal, but this decomposition has reduced the computational complexity of the problem greatly. However, there are two main limitations to the implementation of this approach: (i) there are jumps in velocity, implying an infinite acceleration, and (ii) no general way of obtaining the forbidden regions in the position × time plane was given. Moreover, these regions were approximated as polygons, which may not be realistic.

[1]The work reported in this paper was supported in part by the National Science Foundation under Grant No. DMC-878721492. Any opinions, findings, and recommendations expressed in this publication are those of the authors and do not necessarily reflect the view of the NSF.

CH2642-7/89/0000-2506$1.00© 1989 IEEE

Combining the PTIA [10] with the idea of path–velocity decomposition, we develop in this paper a new algorithm to generate optimal collision-free trajectories for two robots working in the same workspace. The algorithm takes the actuator torque/force constraints into consideration and yields a very interesting result: two robots will reach their goal positions in minimum time if each moves along the minimum time trajectory determined by PTIA and one of them gets delayed by a certain amount of time, t_d, to avoid collision. The delay t_d can be easily determined without using the forbidden regions in the position × time plane, making the proposed algorithm very simple and useful.

This paper is organized as follows. In Section 2 we formulate the two-robot minimum time trajectory planning problem for which a solution is derived. The optimality of the solution is stated and proved in Section 3. Section 4 discusses how to determine the minimum delay t_d and detect a collision, and addresses some other relevant issues. A demonstrative example is given in Section 5, and the paper concludes with Section 6.

2. PROBLEM FORMULATION AND SOLUTION

Suppose there are two robots, R_1 and R_2, working in the same workspace. The path for each robot to follow is given in parametric form: $\mathbf{q}_i = \mathbf{g}_i(s_i)$, $0 \le s_i \le 1$, $i = 1,2$, where $\mathbf{q}_i$ is the R_i's joint position vector, and s_i is the parameter representing the robot's position. For example, s_i could be the normalized arc length along the path. As described in [9], the trajectory planning problem for robot R_i is to determine a function f_i that relates s_i to time t by optimizing some given objective function. The beauty of using a parametric representation is the reduction of the dimension of state space from $2n$ to 2 for an n–jointed robot. The interested readers are referred to [9].

Our trajectory planning problem for a dual robot system is to determine a pair of feasible collision-free trajectories $\pi : s_i = f_i(t)$, $i = 1,2$ such that $f_i(0) = 0$, $f_i(t_{ie}) = 1$, $\frac{df_i(0)}{dt} = \frac{df_i(t_{ie})}{dt} = 0$, and the maximum finish time of the two robots, $J(\pi) = max(t_{1e}, t_{2e})$, is minimized, where t_{ie} is the (unknown) time for R_i to complete the motion. A trajectory is said to be *feasible* if the above terminal conditions are satisfied and it does not require torques beyond the robots' capabilities. Clearly, a necessary condition for $s_i = f_i(t)$ to be a feasible trajectory is that $f_i(t)$ is C^1, or continuously differentiable. A pair of trajectories $\pi : s_i = f_i(t)$, $i = 1,2$, is said to be *collision-free* if $V(S_1(f_1(t)) \cap S_2(f_2(t))) = 0$, $0 \le t \le max(t_{1e}, t_{2e})$, where $S_i(f_i(t))$ denotes the physical space occupied by robot R_i at position $s_i = f_i(t)$ and V(S) denotes the volume of space S. Note that by this definition, R_1 does not collide with R_2 if R_1 slides on the surface of R_2. This can be easily met by "growing" robots by an amount determined based on the required clearance for safety.

We propose the following solution algorithm for the above trajectory planning problem for a dual robot system, providing a pair of optimal feasible collision-free trajectories π^*.

ALGORITHM 1:

Step 1: Apply the PTIA in [10] to derive two minimal time trajectories $s_1 = f_1^*(t)$ and $s_2 = f_2^*(t)$ for R_1 and R_2, respectively. Let $f_i^*(t) = 0$, $\forall\, t < 0$ and $f_i^*(t) = 1$, $\forall\, t > t_{ie}$, $i = 1,2$.

Step 2: Let $T_{d1} = \{t_{d1} : t_{d1} \ge 0,\ s_1 = f_1^*(t - t_{d1})$ and $s_2 = f_2^*(t)$ are collision-free trajectories$\}$ and $T_{d2} = \{t_{d2} : t_{d2} \ge 0, s_1 = f_1^*(t)$ and $s_2 = f_2^*(t - t_{d2})$ are collision-free trajectories$\}$. Let t_{di}^* be the infimum of T_{di}, $i = 1,2$.

Step 3: Let $\pi_1 : s_1 = f_{1d}^*(t) = f_1^*(t - t_{d1}^*), s_2 = f_2^*(t)$, and $\pi_2 : s_1 = f_1^*(t), s_2 = f_{2d}^*(t) = f_2^*(t - t_{d2}^*)$. Then, $\pi^* = \pi_1$ if $J(\pi_1) \le J(\pi_2)$, and $\pi^* = \pi_2$ otherwise.

The ideas of the above algorithm are very simple. Two individual trajectories for R_1 and R_2 were first created by using PITA. Then, coordination of two robots to prevent collision was achieved by simply delaying one robot a certain amount of time at its starting position. Obviously, there could be many other ways to avoid collision. For example, one robot could first move to the intersection point, wait there till the other robot passes by, and then move again. Or both robots' velocities could be changed during their motion to avoid collision. So arises a natural question: what is the best way to avoid collision for Algorithm 1? In the next section, we shall show that under certain conditions, the trajectories obtained from Algorithm 1 are optimal among all feasible collision-free trajectories if the maximum finish time is used as the optimazation criterion.

3. OPTIMALITY OF ALGORITHM 1

The optimality of the trajectories π^* obtained from Algorithm 1 can be proved by investigating some graphs on $s_i \times t$ planes. For the convenience of our discussion, several definitions and notations are introduced to represent these graphs and their relations.

Definition 1: Let $s_i = f_i(t)$, $0 \le t \le t_{ie}$ be a feasible trajectory of R_i, then the *locus* of $s_i = f_i(t)$ on the $s_i \times t$ plane is defined as $\mathcal{L}(f_i) = \{(s_i, t) : s_i = f_i(t),\ \ 0 \le t \le t_{ie}\}$.

Definition 2: Let D be a set on the $s \times t$ plane. Define the *projection* of D on s axis as $P_s(D) = \{u : \exists v$ such that $(u,v) \in D\}$. The *upper boundary* of D is defined as

$$bd_{upper}(D) = \{(u,v) : u \in P_s(D),\ v = sup\{t : (u,t) \in D\}\},$$

and the *lower boundary* of D is defined as

$$bd_{lower}(D) = \{(u,v) : u \in P_s(D),\ v = inf\{t : (u,t) \in D\}\}.$$

Definition 3: Let D_1, D_2 be two sets on $s \times t$ plane. D_1 is said to *lie above* D_2, denoted by $D_1 \ge D_2$, (or D_2 lies under D_1, denoted by $D_2 \le D_1$) if

$$\forall s \in P_s(D_1) \cap P_s(D_2),\ inf\ \{t : (s,t) \in D_1\} \ge sup\ \{t : (s,t) \in D_2\}.$$

D_1 is said to *lie left to* D_2 (or D_2 lies right to D_1) if

$$\forall t \in P_t(D_1) \cap P_t(D_2),\ inf\ \{s : (s,t) \in D_2\} \ge sup\ \{s : (s,t) \in D_1\}.$$

where $P_t(D_1)$ and $P_t(D_2)$ are projections of D_1 and D_2 on t axis, respectively.

Definition 4: Let $f_{i\ (t_0,s_0)}(t)$ denote a feasible trajectory of R_i that passes through the point (t_0, s_0), i.e., $f_{i\ (t_0,s_0)}(t_0) = s_0$, and let $f_{i\ (t_0,s_0)}^*(t)$ denote the minimum time feasible trajectory of R_i obtained by delaying a certain period of time such that $f_{i\ (t_0,s_0)}^*(t_0) = s_0$.

Obviously, $\mathcal{L}(f_{i\ (t_0,s_0)}^*(t))$ lies above and left to $\mathcal{L}(f_{i\ (t_0,s_0)}(t))$ for $0 \le s \le s_0$ and $\mathcal{L}(f_{i\ (t_0,s_0)}^*(t))$ lies under and right to $\mathcal{L}(f_{i\ (t_0,s_0)}(t))$ for $s_0 \le s \le 1$. Further, $T_e(f_{i\ (t_0,s_0)}^*(t)) \le T_e(f_{i\ (t_0,s_0)}(t))$, where $T_e(f_i) = inf\ \{\ t : f_i(t) = 1\}$.

Definition 5: For a given R_2's trajectory $s_2 = f_2(t)$, define a region on the $s_1 \times t$ plane as:

$$\bar{D}(f_2) = \{(s_1,t) : S_1(s_1) \cap S_2(f_2(t)) \ne \emptyset,\ 0 \le s_1 \le 1,\ t \ge 0\}.$$

Then, the *collision region* on the $s_1 \times t$ plane corresponding to $f_2(t)$ is defined as:

$$D(f_2) = \bar{D}(f_2) - bd(\bar{D}(f_2)),$$

where $bd(D)$ denotes the boundary of D. Note that $bd(\bar{D}(f_2))$ corresponds to the situations in which R_1 and R_2 slide on each other. The collision region on the $s_2 \times t$ plane corresponding to $s_1 = f_1(t)$ can be defined similarly as:

$$D(f_1) = \bar{D}(f_1) - bd(\bar{D}(f_1))$$

where $\bar{D}(f_1) = \{(s_2,t) : S_2(s_2) \cap S_1(f_1(t)) \ne \emptyset,\ 0 \le s_2 \le 1,\ t \ge 0\}$.

The collision-free trajectory planning problem can now be stated as: find trajectories $s_1 = f_1(t)$ and $s_2 = f_2(t)$ such that $\mathcal{L}(f_1) \cap D(f_2) = \emptyset$ (or $\mathcal{L}(f_2) \cap D(f_1) = \emptyset$).

Definition 6: Let $\bar{D}_c = \{(s_1,s_2) :\ S_1(s_1) \cap S_2(s_2) \ne \emptyset,\ 0 \le s_i \le 1,\ i = 1,2\}$. Then, the collision region on the $s_1 \times s_2$ plane is defined

as $D_c = \bar{D}_c - bd(D_c)$. It is important to note that D_c is independent of $f_i(t)$.

Definition 7: D_c is said to be *strongly connected* if $\forall\ 0 \le s_1^1 \le s_1^2 \le 1,\ 0 \le s_2^1 \le s_2^2 \le 1,\ D_c \cap \{(s_1,s_2):\ s_1^1 \le s_1 \le s_1^2,\ s_2^1 \le s_2 \le s_2^2\}$ is either connected or an empty set. It is easy to see that this condition is stronger than that of connectedness, but weaker than that of convexity of D_c.

Definition 8: Let $s_i^{min} = \inf\ P_{s_i}(D_c)$, $s_i^{max} = \sup\ P_{s_i}(D_c)$, $i = 1,2$. Define the *collision zone* on $s_i \times t$ plane as follows: $D_{z_i} = \{(s_i,t):\ s_i^{min} \le s_i \le s_i^{max}\}$. Note that if one robot is outside the collision zone, no collision will occur.

Based on the above definitions and notations, we establish a set of lemmas which will be used to prove the optimality of Algorithm 1.

Lemma 1:

(1) Let $s_1 = f_1(t)$ be a feasible trajectory of R_1. Then, $D(f_1)$ is connected if (i) D_c is strongly connected, and (ii) $f_1(t)$ is monotone outside the collision zone D_{z_1}.

(2) Let $s_2 = f_2(t)$ be a feasible trajectory of R_2. Then, $D(f_2)$ is connected if (i) D_c is strongly connected, and (ii) $f_2(t)$ is monotone outside the collision zone D_{z_2}.

Proof: We first prove part (1) of Lemma 1. To prove $D(f_1)$ is connected, we need to prove that for any two points $p_s = (s_2^s, t_s)$, $p_e = (s_2^e, t_e) \in D_1(f_1)$, $\exists$ a continuous curve C connecting p_s and p_e such that $C \subset D(f_1)$. Without loss of generality, we can assume $t_s \le t_e$.

Since $p_s, p_e \in D(f_1)$, the corresponding points on $s_1 \times t$ plane $p_s' = (f_1(t_s), t_s)$, $p_e' = (f_1(t_e), t_e)$ belong to D_{z_1}. Further, since $f_1(t)$ is monotone outside D_{z_1}, the trajectory $s_1 = f_1(t)$ can enter and leave D_{z_1} at most once, implying that the trajectory connecting all points in $C' = \{(s_1,t):\ s_1 = f_1(t), t_s \le t \le t_e\}$ lies in D_{z_1}.

Since $f_1(t)$ is continuously differentiable over $[t_s, t_e]$, there exist $t_s = t_0 < t_1 < ... < t_{n-1} < t_n = t_e$, $n \le \infty$ such that $f_1(t)$ is either strictly monotone or a constant over $[t_{i-1}, t_i]$, $i = 1,2,...,n$. Let $q_0' = (f_1(t_0), t_0) = p_s'$, $q_1' = (f_1(t_1), t_1)$, ... , $q_n' = (f_1(t_n), t_n) = p_e'$. Clearly, these points lie in D_{z_1} since $C' \subset D_{z_1}$. This fact, together with the fact that D_c is a connected region, implies that there exist s_2^i ($s_2^0 = s_2^s, s_2^n = s_2^e$) such that $q_i'' = (f_1(t_i), s_2^i) \in D_c$, $i = 1,...,n$. Then, the corresponding points on $s_2 \times t$ plane: $q_0 = (s_2^0, t_0) = p_s$, $q_1 = (s_2^1, t_1)$, ... , $q_n = (s_2^n, t_n) = p_e$ lie in $D(f_1)$. We now prove that there exist continuous curves C_i connecting q_{i-1}, q_i such that $C_i \in D(f_1)$.

Without loss of generality, we can assume $f_1(t_{i-1}) \le f_1(t_i)$. Since D_c is strongly connected, the region $D_c^i = D_c \cap \{(s_1,s_2): f_1(t_{i-1}) \le s_1 \le f_1(t_i)\}$ is also connected. So, there exists a continuous curve $\bar{C}_i$ connecting q_{i-1}'', q_i'' and belonging to D_c^i.

If $f_1(t)$ is strictly monotone over $[t_{i-1}, t_i]$, its inverse function $f_1^{-1}(s_1)$ exists and is continuous over $[f_1(t_{i-1}), f_1(t_i)]$. Thus, $C_i = \{(t,s_2): t = f_1^{-1}(s_1),\ (s_1,s_2) \in \bar{C}_i\}$ is a continuous curve connecting q_{i-1} and q_i, and $C_i \subset D(f_1)$.

If $f_1(t)$ is a constant over $[t_{i-1}, t_i]$, then $C_i = \{(t,s_2):\ t_{i-1} \le t \le t_i, s_2 = s_2^i\}$ is a continuous curve connecting q_{i-1}, q_i and $C_i \subset D(f_1)$.

Thus, $C = \cup_{i=1}^n C_i$ is a continuous curve connecting p_s, p_e, implying that $D(f_1)$ is connected.

Part (2) of Lemma 1 can be proved similarly. □

Lemma 2: Let Π be the set of all pairs of feasible collision-free trajectories, and let Π_m be the set of all pairs of *strictly monotone* feasible collision-free trajectories. If the collision region D_c is strongly connected, then $\exists \pi^m \in \Pi_m$ such that $J(\pi^m) \le J(\pi)$, $\forall \pi \in \Pi$.

Proof: Let $\pi:\ s_1 = f_1(t), s_2 = f_2(t)$ be a pair of feasible collision-free trajectories. We first find a strictly monotone function $\bar{f}_1(t)$ and an $\bar{f}_2(t)$ which is monotone outside the collision zone D_{z_2} such that $\bar{\pi}:\ s_1 = \bar{f}_1(t),\ s_2 = \bar{f}_2(t)$ is a pair of feasible collision-free trajectories and $J(\bar{\pi}) \le J(\pi)$.

Let $t_2 = \min\{t:\ f_2(t) = s_2^{max}\}$, $t_1 = \max\{t:\ f_2(t) = s_2^{min}, t < t_2\}$. Apply the PTIA to derive two minimal time trajectories $s_2 = h_1(t)$ such that $h_1(t_1) = s_2^{min}$, $\frac{dh_1(t)}{dt}|_{t=t_1} = \frac{df_2(t)}{dt}|_{t=t_1}$, $\frac{dh_1(t)}{dt}|_{t=h_1^{-1}(0)} = 0$, and $s_2 = h_2(t)$ such that $h_2(t_2) = s_2^{max}$, $\frac{dh_2(t)}{dt}|_{t=t_2} = \frac{df_2(t)}{dt}|_{t=t_2}$, $\frac{dh_2(t)}{dt}|_{t=h_2^{-1}(0)} = 0$. Let

$$\bar{f}_2(t) = \begin{cases} 0 & \text{if } 0 \le t < h_1^{-1}(0) \\ h_1(t) & \text{if } h_1^{-1}(0) \le t < t_1 \\ f_2(t) & \text{if } t_1 \le t < t_2 \\ h_2(t) & \text{if } t_2 \le t < h_2^{-1}(1). \end{cases}$$

Clearly, $\bar{f}_2(t)$ is monotone outside D_{z_2}, $\pi':\ s_1 = f_1(t), s_2 = \bar{f}_2(t)$ is a pair of collision free trajectories, and $J(\pi') \le J(\pi)$.

Since $s_2 = \bar{f}_2(t)$ is monotone outside D_{z_2}, by Lemma 1 $D(\bar{f}_2)$ is a connected region. Hence, the trajectory $s_1 = f_1(t)$ lies either left to $D(\bar{f}_2)$ or right to $D(\bar{f}_2)$. Let

$$\bar{f}_1(t) = \begin{cases} f_{1\,(T_s(f_1),1)}^*(t) & \text{if } s_1 = f_1(t) \text{ lies left to } D(\bar{f}_2) \\ f_{1\,(0,0)}^*(t) & \text{if } s_1 = f_1(t) \text{ lies right to } D(\bar{f}_2). \end{cases}$$

Clearly, $\bar{f}_1(t)$ is strictly monotone , $\bar{\pi}:\ s_1 = \bar{f}_1(t), s_2 = \bar{f}_2(t)$ is a pair of collision-free trajectories, and $J(\bar{\pi}) \le J(\pi')$, implying that $J(\bar{\pi}) \le J(\pi)$.

Following the above procedure, one can further find an $\bar{f}_2(t)$ such that $\bar{f}_2(t)$ is a strictly monotone function, $\pi^m:\ s_1 = \bar{f}_1(t), s_2 = \bar{f}_2(t)$ is a pair of feasible collision-free trajectories, and $J(\pi^m) \le J(\bar{\pi})$. Clearly, $\pi^m \in \Pi_m$ and $J(\pi^m) \le J(\bar{\pi}) \le J(\pi)$. □

Lemma 2 plays a key role in solving our two–robot trajectory planning problem because it allows us to consider only strictly monotone trajectories.

Lemma 3:

(1) Let $s_1 = f_1(t)$ and $s_1 = f_2(t)$ be two strictly monotone feasible trajectories of R_1 such that $\mathcal{L}(f_1)$ lies under $\mathcal{L}(f_2)$, then $bd_{upper}(\bar{D}(f_1))$ lies under $bd_{upper}(\bar{D}(f_2))$.

(2) Let $s_2 = f_1(t)$ and $s_2 = f_2(t)$ be two strictly monotone feasible trajectories of R_2 such that $\mathcal{L}(f_1)$ lies under $\mathcal{L}(f_2)$, then $bd_{upper}(\bar{D}(f_1))$ lies under $bd_{upper}(\bar{D}(f_2))$.

Proof: We first prove part (1) of Lemma 3. For $i = 1,2$, $bd_{upper}(\bar{D}(f_i))$ are two curves on the $s_2 \times t$ plane, which are denoted by $t = h_i(s_2)$, $s_2 \in P_s(\bar{D}(f_i))$. To prove that $bd_{upper}(\bar{D}(f_1))$ lies under $bd_{upper}(\bar{D}(f_2))$, we only need to prove $h_1(s_2) \le h_2(s_2), \forall s_2 \in P_s(\bar{D}(f_1)) \cap P_s(\bar{D}(f_2))$. For any given $s_2 \in P_s(\bar{D}(f_1)) \cap P_s(\bar{D}(f_2))$, define $s_1' = \sup\ \{s_1 : (s_1,s_2) \in \bar{D}_c\}$. Since $f_i(t)$ is strictly monotone, $h_i(s_2) = f_i^{-1}(s_1')$, $i = 1,2$. Since $\mathcal{L}(f_2)$ lies above $\mathcal{L}(f_1)$, $f_1^{-1}(s_1') \le f_2^{-1}(s_1')$. Thus, $h_1(s_2) \le h_2(s_2)$.

Part (2) of Lemma 3 can be proved similarly. □

Lemma 3 means that if a trajectory moves up along t axis on $s_1 \times t$ plane, then its corresponding collision region also moves up along t axis on $s_2 \times t$ plane.

Using the above results, we can now formally state and prove the optimality of Algorithm 1.

Theorem: Let $s_1 = f_1^*(t)$ and $s_2 = f_2^*(t)$ be the minimum time trajectories obtained in Step 1 for R_1 and R_2, respectively, and π^* be the trajectories obtained in Step 3. If (1) no collision will occur when at least one robot is at its starting position (s=0) or ending position (s=1), and (2) D_c is strongly connected, then $J(\pi^*) \le J(\pi)$ for any feasible collision-free trajectories $\pi: s_i = f_i(t)$, $i = 1,2$.

Proof: We first prove that π^* is well-defined.

Let $t_{d1} = T_s(f_2^*)$. Assumption (1) of the theorem implies that $s_1 = f_1^*(t - t_{d1}), s_2 = f_2^*(t)$ is a pair of collision-free trajectories. Thus, $T_{d1} \ne \emptyset$ because at least it contains t_{d_1}. Further, T_{d1} is lower bounded (≥ 0), so $t_{d1}^* = inf\ T_{d1}$ always exists. This proves that π_1 is well-defined. Symmetrically to the above, one can prove that π_2 is well-defined, and thus, π^* is well-defined.

It is also clear that the π^* constructed in Step 3 is a pair of feasible collision-free trajectories. Now, we want to prove the optimality of π^*, or $J(\pi^*) \le J(\pi)$.

Based on Lemma 2, we can assume that both $f_1(t)$ and $f_2(t)$ are strictly monotone. Then, $D(f_2)$ is a connected region. Since π is collision-free (i.e., $\mathcal{L}(f_1) \cap D(f_2) = \emptyset$), $f_1(t)$ is strictly monotone, and $D(f_2)$ is connected, it is easy to see that either $\mathcal{L}(f_1) \geq D(f_2)$ or $D(f_2) \geq \mathcal{L}(f_1)$. Let us first prove that $J(\pi_1) \leq J(\pi)$ when $\mathcal{L}(f_1) \geq D(f_2)$. (Recall that π_1 is defined as $s_1 = f_{1d}^*(t) = f_1^*(t - t_{d1}^*)$, $s_2 = f_2^*(t)$ in Step 3 of Algorithm 1 and $\pi: s_1 = f_1(t)$, $s_2 = f_2(t)$ is any pair of feasible strictly monotone collision-free trajectories.)

If $t_{d1}^* = 0$, then $T_e(f_{1d}^*) = T_e(f_1^*) \leq T_e(f_1)$ and $T_e(f_2^*) \leq T_e(f_2)$. Thus, $J(\pi_1) = max(T_e(f_{1d}^*), T_e(f_2^*)) \leq max(T_e(f_1), T_e(f_2)) = J(\pi)$.

If $t_{d1}^* > 0$, define $\pi': s_1 = f_1(t)$, $s_2 = f_2^*(t)$. Since f_2^* is the minimum time trajectory, $\mathcal{L}(f_2^*)$ lies under $\mathcal{L}(f_2)$. ¿From Lemma 3, we have $bd_{upper}(\bar{D}(f_2)) \geq bd_{upper}(\bar{D}(f_2^*))$. Thus,

$$\mathcal{L}(f_1) \geq bd_{upper}(\bar{D}(f_2)) \geq bd_{upper}(\bar{D}(f_2^*)) \geq D(f_2^*),$$

or $\mathcal{L}(f_1)$ lies above $D(f_2^*)$.

Further, from $T_e(f_2) \geq T_e(f_2^*)$ (i.e., f_2^* is the R_2's minimum time trajectory) and $T_e(f_1) = T_e(f_1)$, we have $J(\pi) \geq J(\pi')$.

Now we want to prove $J(\pi_1) \leq J(\pi')$. ¿From the definition of π_1 and π', we only need to prove $T_e(f_{1d}^*) \leq T_e(f_1)$ since the trajectories of R_2 are the same in π_1 and π'. Let $(s_0, t_0) \in \mathcal{L}(f_{1d}^*) \cap bd_{upper}(\bar{D}(f_2^*))$. From the definition of f_{1d}^* and $t_{d1}^* > 0$, such a point always exists. Since $\mathcal{L}(f_1)$ lies above $D(f_2^*)$, $\delta t = f_1^{-1}(s_0) - t_0 \geq 0$. Let $\bar{f}_1(t) = f_1(t + \delta t)$. Then, $T_e(\bar{f}_1) = T_e(f_1) - \delta t \leq T_e(f_1)$. Since $\bar{f}_1$ and f_{1d}^* intersect at (s_0, t_0), $T_e(f_{1d}^*) \leq T_e(\bar{f}_1)$. then, $T_e(f_{1d}^*) \leq T_e(\bar{f}_1) \leq T_e(f_1)$, i.e., $J(\pi_1) \leq J(\pi')$. Thus, it can be concluded that $J(\pi_1) \leq J(\pi') \leq J(\pi)$.

If $D(f_2) \geq \mathcal{L}(f_1)$, then $\mathcal{L}(f_2) \geq D(f_1)$. Similarly to the above, one can prove that $J(\pi_2) \leq J(\pi)$, making $J(\pi^*) = min(J(\pi_1), J(\pi_2)) \leq J(\pi)$. □

4. REMARKS

3.1 Determination of Minimal Delays

Existence of an effective method to determine the minimum delays t_{di}^* for $i = 1, 2$ is crucial to the applicability of Algorithm 1. Theoretically, t_{d1}^* (t_{d2}^*) can be obtained by moving $\mathcal{L}(f_1^*)$ ($\mathcal{L}(f_2^*)$) upward along t-axis until it no longer intersects the collision region $D(f_2^*)$ ($D(f_1^*)$). However, determination of the collision region $D(f_i^*)$ is difficult in practice. So, it is necessary to develop an alternative way of determining the minimum delays.

For practical applications, we do not need the exact values of t_{di}^*. Often the approximate minimum delays t'_{di} such that $0 \leq t'_{di} - t_{di}^* \leq \tau$ is good enough, where τ is the desired tolerance. Let $s_1 = f_1^*(t)$ and $s_2 = f_2^*(t)$ be the minimum time trajectories of R_1 and R_2 obtained in Step 1 of Algorithm 1, and let $T = max(T_e(f_1^*), T_e(f_2^*))$. Then the approximate delay t'_{d1} can be obtained with the following algorithm (t'_{d2} can be obtained similarly).

ALGORITHM 2:

Step 1: Set $t_1 = 0$, $t_2 = T + \tau$, $t'_{d1} = 0$

Step 2: Let R_1 and R_2 move along the trajectories $\mathbf{q}_1 = \mathbf{g}_1(f_1^*(t - t'_{d1}))$, $\mathbf{q}_2 = \mathbf{g}_2(f_2^*(t))$, respectively. Check if a collision occurs.

Step 3: If no collision occurs and $t'_{d1} = 0$, STOP.

Step 4: If no collision occurs, $t'_{d1} \neq 0$ and $t'_{d1} - t_1 > \tau$, then let $t_2 = t'_{d1}$, $t'_{d1} = (t_1 + t_2)/2$. Goto Step 2.

Step 5: If no collision occurs, then $t'_{d1} \neq 0$, $t'_{d1} - t_1 \leq \tau$, and STOP.

Step 6: If collision occurs, then let $t_1 = t'_{d1}$, $t'_{d1} = (t_1 + t_2)/2$. Goto Step 2.

Assumption 1 in the theorem implies that $t'_{d1} \leq T$. Thus, the above algorithm will terminate within at most $1 + log_2(T/\tau)$ iterations. Further, the connectivity of $D(f_2^*)$ (since f_2^* is strictly monotone and D_c is strongly connected) ensures that $0 \leq t'_{d1} - t_{d1}^* \leq \tau$, i.e., the t'_{d1} is an acceptable approximate minimum delay.

Note that for Step 2 of the above algorithm, no one would perform real experiments just to check if two robots collide with each other, because it could result in collisions between them. So, we need to devise a means of collision detection, the subject of the next subsection.

3.2 Collision Detection

Suppose two robots, R_1 and R_2, move along trajectories $\mathbf{q}_1 = \mathbf{q}_1(t)$ and $\mathbf{q}_2 = \mathbf{q}_2(t)$, $0 \leq t \leq T$, respectively. Define a collision detection function $C(\mathbf{q}_1, \mathbf{q}_2)$ as follows: $C(\mathbf{q}_1, \mathbf{q}_2) = 1$ if R_1 at position $\mathbf{q}_1$ is to collide with R_2 at posiotion $\mathbf{q}_2$. Otherwise, $C(\mathbf{q}_1, \mathbf{q}_2) = 0$. Then, the following algorithm can be used to detect a collision, i.e., the algorithm returns COLLISION=1 if a collision is to occur and COLLISION=0 if no collision is to occur.

ALGORITHM 3:

Step 1: Set $t = -\tau$, COLLISION=0.

Step 2: Let $t = t + \tau$. If $C(\mathbf{q}_1(t), \mathbf{q}_2(t)) = 0$ goto Step 3. Otherwise, set COLLISION=1. STOP.

Step 3: If $t < T$, goto Step 2. Otherwise, STOP.

In the above algorithm, we assume that $\mathbf{q}_1 = \mathbf{q}_1(t)$ and $\mathbf{q}_2 = \mathbf{q}_2(t)$ are smooth trajectories so checking the discrete points $t = 0, \tau, 2\tau, ..., T$ is enough to detect a collision. The determination of $C(\mathbf{q}_1, \mathbf{q}_2)$ depends on the geometric relation between the two robots. Here, we use a simple example to show how to determine $C(\mathbf{q}_1, \mathbf{q}_2)$.

Consider the configurations of two robots shown in Fig. 1. Both robots move in the same plane with two degrees of freedom (dofs) which are aligned with a polar coordinate system: angle β and and length r. Both robots are modeled as straight line segments, ignoring the thickness of their links.

For this example, the collision detection function $C(r_1, \beta_1, r_2, \beta_2)$ can be determined as follows. $C(r_1, \beta_1, r_2, \beta_2) = 1$ if one of the following three conditions occurs: (1) $\beta_1 = \beta_2 = 0$ and $r_1 + r_2 > a$; (2) $\beta_1 = 0, \beta_2 \neq 0$ and $r_1 > a$ (or $\beta_2 = 0$, $\beta_1 \neq 0$ and $r_2 > a$); (3) $\beta_1 \beta_2 > 0$ and $r_1 > a * sin(\beta_2)/sin(\beta_1 + \beta_2)$, $r_2 > a * sin(\beta_1)/sin(\beta_1 + \beta_2)$. Otherwise, $C(r_1, \beta_1, r_2, \beta_2) = 0$.

The method described above can be generalized to detect collisions of robots with more than 2 dofs. Suppose each robot has an arbitrary number of links. Then, the collision detection function can be derived by examining each pair of links, one from R_1 and the other from R_2, which are located in the same plane: $C(\mathbf{q}_1, \mathbf{q}_2) = 1$ if at least one pair of links collide with each other; $C(\mathbf{q}_1, \mathbf{q}_2) = 0$ otherwise. Note that two links in different planes will not collide with each other.

A more general method of determining the collision detection function $C(\mathbf{q}_1, \mathbf{q}_2)$ is to model each link as a combination of several polytopes (or spherical extension of polytopes) and compute the distance between each pair of polytopes, one from R_1 and the other from R_2 as was done in [3].

3.3 Assumptions of Theorem

Two assumptions have been used to prove the optimality of Algorithm 1: (A1) no collision will occur when at least one robot is at its starting position (s=0) or ending position (s=1), and (A2) D_c is strongly connected. More specifically, A1 consists of four conditions:

1. no collision will occur when R_1 is at its starting position.
2. no collision will occur when R_1 is at its ending position.
3. no collision will occur when R_2 is at its starting position.
4. no collision will occur when R_2 is at its ending position.

From the proof of the theorem, it is easy to see that satisfaction of all these four conditions will ensure the existence of π^* in Algorithm 1, satisfaction of Conditions 1 and 4 will ensure the existence of π_1, and satisfaction of Conditions 2 and 3 will ensure the existence of π_2. Using Algorithm 3, these four conditions can be easily justified in practice according to the robots' geometric shapes and predetermined paths.

Satisfaction of A2 ensures the solutions obtained from Algorithm 1 to be optimal. A natural approximation is to tessellate the area

$\{(s_1, s_2) : 0 \leq s_1 \leq 1, 0 \leq s_2 \leq 1\}$ into fine grids and compute the values of the collision detection function for all grids. Then, the strong connectivity of D_c can be easily determined from these values.

In the rest of this subsection, we will discuss the conclusions to be drawn after checking the assumptions of the theorem.

Case 1: Both assumptions are satisfied.

Algorithm 1 will give a pair of optimal feasible collision-free trajectories among all feasible collision-free trajectories.

Case 2: A2 and Conditions 1 and 4 of A1 are satisfied.

Algorithm 1 will give a pair of feasible collision-free trajectories with a total finish time shorter than, or equal to, that of all feasible collision-free trajectories for which priority is given to R_2.

Case 3: A2 and Conditions 2 and 3 of A1 are satisfied.

Algorithm 1 will give a pair of feasible collision-free trajectories with a total finish time shorter than, or equal to, that of all feasible collision-free trajectories for which priority is given to R_1.

Case 4: A1 is satisfied, but A2 is not.

Algorithm 1 will give a pair of feasible collision-free trajectories π^*. π^* may, or may not, be optimal.

Case 5: A1 is not satisfied.

Algorithm 1 may, or may not, give a pair of feasible collision-free trajectories.

Since the spatial planner can usually guarantee the satisfaction of A1, Algorithm 1 can produce a pair of feasible collision-free trajectories.

5. AN EXAMPLE

A Fortran program, called the PLANNER, has been developed to implement Algorithms 1—3 and the PTIA of [10]. The program takes dynamic equations, torque constraints, preassumed paths and geometric relations of robots as input, and outputs a pair of optimal feasible collision-free trajectories. In what follows we describe an example of using this program.

Consider the two robot configuration of Fig. 1. Let $a = 2\ m$ and suppose R_1 is to move from position $r_1 = 1\ m$, $\beta_1 = \pi/2$ to position $r_1 = 2\ m$, $\beta_1 = -\pi/2$ and R_2 is to move from position $r_2 = 1\ m$, $\beta_2 = -\pi/2$ to position $r_2 = 2\ m$, $\beta_2 = \pi/2$. Let the preassumed paths be: $r_1 = 1 + s_1$, $\beta_1 = (1 - 2s_1)\pi/2$ and $r_2 = 1 + s_2$, $\beta_2 = (2s_2 - 1)\pi/2$, $0 \leq s_1 \leq 1$, $0 \leq s_2 \leq 1$. Clearly, two paths intersect and collisions may occur between the two robots moving along these two paths. For simplicity, the torque constraints are assumed to be $|d^2r_1/dt^2| \leq 1(m^2/s)$, $|d^2\beta_1/dt^2| \leq 3$, $|d^2r_2/dt^2| \leq 1(m^2/s)$, $|d^2\beta_2/dt^2| \leq 2$.

The outputs of PLANNER are plotted in Figs. 2 and 3. R_1 was delayed by 0.81 sec. and the resulting total finish time was 2.86 sec. To see how the two robots avoid collision more clearly, their movements along the optimal collision-free trajectories are plotted in Fig. 4 and their movements along the minimum time trajectories obtained in Step 1 of Algorithm 1 (i.e., without delaying R_1) are plotted in Fig. 5. Note that a collision occurred in the latter case.

Clearly, both assumptions of the theorem are satisfied for this example. Thus, from the optimality of Algorithm 1, the trajectories obtained will ensure that two robots reach their ending positions along the preassumed paths in minimum time. However, because of the path–velocity decomposition used in the algorithm, the trajectories may not be overall optimal. For example, the total finish time was reduced to 2.77 seconds when we chose the preassumed paths to be $r_1 = 1 + s_1$, $\beta_1 = (1 - 2s_1)\pi/2$ and $r_2 = 1 + s_2 * s_2$, $\beta_2 = (2s_2 - 1)\pi/2$, $0 \leq s_1 \leq 1, 0 \leq s_2 \leq 1$. This fact reveals the need of an algorithm for combined path and trajectory planning. However, it is very difficult to find such an algorithm because many other factors, like collision avoidance with static obstacles and the limitation of robots' reachable spaces, should be considered all together. In most practical applications, this non-overall optimality can be reduced to some extent by inputing a set of feasible paths (instead of one) to the PLANNER and choosing the best among them.

6. CONCLUSIONS

We have proposed an optimal collision-free trajectory planning algorithm for dual robot systems. The optimality of the algorithm has been formally stated and rigorously proved. The most important property of the algorithm is that it includes the dynamics of the two robots and their actuator constraints in the collision-free trajectory planning. This feature, together with its simplicity, has made the algorithm very attractive for practical use. Using the results of this paper, it may be possible to develop overall optimal algorithms (similar to that in [8]) for systems with more than two robots. Our future research will address the problem of developing optimal trajectory planning algorithms for multi-robot systems with disconnected collision regions.

References

[1] J. E. Bobrow, S. Dubowsky, and P. Gibson. On the optimal control of robotic manipulators with actuator constraints. In *Proc. 1983 Automat. Contro. Conf.*, pages 782–787, June 1983.

[2] E. Freund and H. Hoyer. Real-time pathfinding in multirobot systems including obtacle avoidance. *The International Journal of Robotics Research*, 7(1):42–70, February 1988.

[3] E. G. Gilbert, D. W. Johnson, and S. S. Keerthi. A fast procedure for computing the distance between complex objects in three-dimensional space. *IEEE Journal of Robotics and Automation*, 4(2):193–203, April 1988.

[4] K. Kant and S. W. Zucker. Toward efficient trajectory planning: The path-velocity decomposition. *The International Journal of Robotics Research*, 5(3):72–89, Fall 1986.

[5] Lozano-Perez. Spatial planning: A configuration space approach. *IEEE Trans. on Computers*, C–32(2):108–120, December 1983.

[6] O. Z. Maimon and S. Y. Nof. Coordination of robots sharing assembly tasks. *Journal of Dynamic Systems, Measurement, and Control*, 107:299–307, December 1985.

[7] J. W. Roach and M. N. Boaz. Coordinating the motions of robot arms in a common workspace. *IEEE Journal of Robotics and Automation*, RA-3(5):437–444, Octorber 1987.

[8] G. Sahar and J. M. Hollerbach. Planning of minimum-time trajectories for robot arms. *The International Journal of Robotics Research*, 5(3):90–100, Fall 1986.

[9] K. G. Shin and N. D. Mckay. Minimum-time control of a robotic manipulator with geometric path constraints. *IEEE Trans. on Automatic Control*, AC-30(6):531–541, June 1985.

[10] K. G. Shin and N. D. Mckay. Robust trajectory planning for robotic manipulators under payload uncertaineies. *IEEE Trans. on Automatic Control*, AC-32(12):1044–1054, December 1987.

[11] K. Honda T. Nagata and Y. Teramoto. Multirobot plan generation in a continuous domain: Planning by use of plan graph and avoiding collisions among robots. *IEEE Journal of Robotics and Automation*, 4(1):2–13, Feberuary 1988.

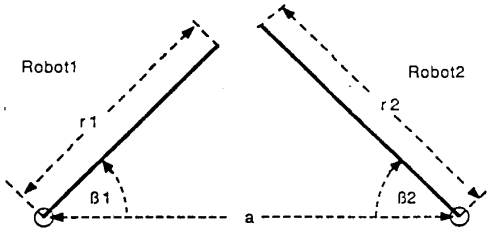

Fig. 1 Configurations of a Dual Robot System

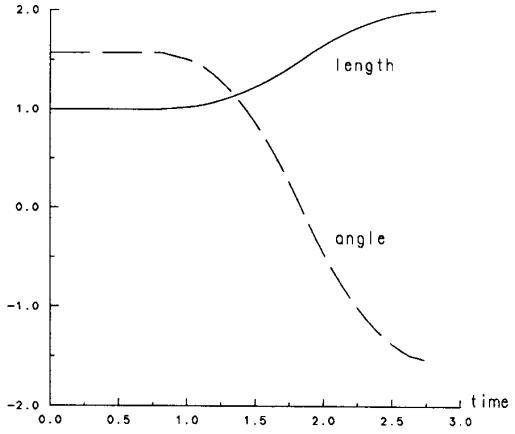

Fig. 2 Trajectory of robot 1

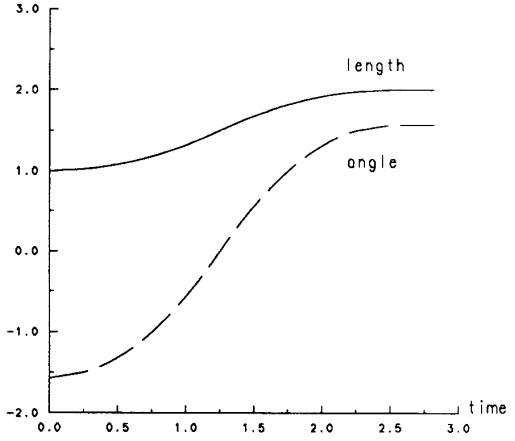

Fig. 3 Trajectroy of robot 2

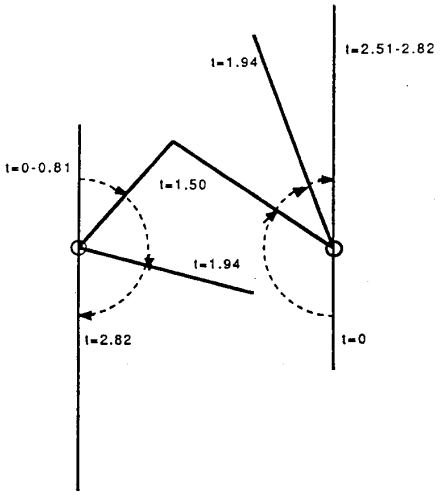

Fig. 4 Collision Free Trajectories

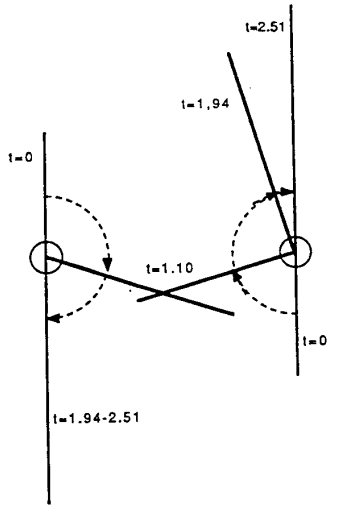

Fig. 5 Trajectories With Collision