'89 KACC 1989. 10. 27~28

A Nonlinear Programming Approach to Collision-Avoidance Trajectory Planning of Multiple Robots

Suk-Hwan Suh

Department of Industrial Engineering
POSTECH
P.O. Box 125, Pohang 790-600, Korea.

Myung-Soo Kim

Department of Computer Science
POSTECH
P.O. Box 125, Pohang 790-600, Korea.

Abstract

We formulated the multi-robot trajectory problem into a series of NLP problems, each of which is that of finding the optimal tip positions of the robots for the next time step. The NLP problem is composed of an objective function and three constraints, namely: a) Joint position limits, b) Joint velocity limits, and c) Collision-avoidance constraints. By solving a series of NLP problems, optimally coordinated trajectories can be determined without requiring any prior path information. This is a novel departure from the previous approach in which either all paths or at least one path is assumed to be given. Practical application of the developed method is for optimal synthesis of multiple robot trajectories in off-line. To test the validity and effectiveness of the method, numerical examples are illustrated.

1 Introduction

Recently there has been a great deal of interest shown to multi-robot systems as there are increasing number of applications requiring more than one robot. Also the multi-robot configuration is an attractive layout in the sense of resource utilization; e.g., tool sharing. For the multi-robot environment in which the work areas of two or more manipulators intersect, the collision-avoidance problem, the problem of determining paths for a set of robots so that they do not collide, is a critical issue. Strictly speaking, collision-avoidance path planning for multiple robots is not just a spatial planning problem for a single robot, but a trajectory planning problem as collision-avoidance is dependent on path execution timing. Thus, the previous path planning approach to single arms is not directly applicable to multi-robot systems.

Only a few research results have been reported on the trajectory planning for multi-robot systems. Freund *et al.* [1] proposed an approach to the findpath problem in multi-robot systems based on a hierachical system structure. They linearized the manipulator dynamics, and developed methods to detect and avoid collision on-line. Their collision-avoidance strategy is for a $(\theta - r)$ type of manipulator and mobile robot. In other words, the algorithm must be refined further for the general type of robots with six-degree-of-freedom. Toumassoud [2] presented a geometric approach for avoiding obstacles which may be applicable to two 3D robots. His method is a local method based on the extreme separating hyperplanes.

Lee *et al.* [3] considered trajectory planning of two robots. They adopted straight line paths for both robot hands which were simplified as wrist-centered spheres. A collision map displays the collision-time period in which the distance between two wrists is less than or equal to the sum of the radii of the two wrists. Although this method is simple and attractive in many cases, there are several potential problems. First, time delay and speed reduction schemes may not always solve the collision-avoidance problem as discussed in [3]. In such a case, the straight line paths have to be modified as well. Second, since the collision is detected only with reference to the wrist, the resultant trajectories may lead to collision at the lower links of manipulators.

Recently, *Chong et. al* [4] presented a collision-avoidance scheme for two robots. They took a leader-follower approach which has been commonly adopted for the control of two arms carrying a common object [5-7]. The leader's path is given by a straight line and the collision-aviodance scheme for the follower is determined by the linear programming. In this formulation, the physical meaning of the objective function is not clear.

Most of the previous collision-avoidance schemes require paths a priori for either one or two robots. The straight line path, typically used, is often hard to adopt for a general work environment in the presence of obstacles. For such a case, the problem of finding obstacle-free paths has to be addressed. Thus the trajectory planning is divided into two steps; i.e., obstacle-avoidance path planning followed by collision-avoidance path execution. In the paper [8], we developed a two-step minimum-time trajectory planning method for two robots. In this method the obstacle-avoidance paths and *interference region maps* are obtained by an interactive CAD program, namely MULTPATH, and the minimum-time collision-avoidance path execution scheme is determined by the dynamic programming.

In this paper, we present a new approach to finding both the obstacle-avoidance paths and collision-avoidance path execution scheme in a unified fashion. This method does not require any paths as they are determined in the algorithm. For a given set of starting and destination positions[1], the problem of finding optimally coordinated movements for each robot at each time increment is formulated into a non-linear programming (NLP). In this fashion, the whole trajectory can be determined by solving NLP problems sequentially.

NLP approach has been conventionally used for the optimization of static systems, but has not been used for the dynamic problems such as multi-robot trajectory planning problem that we are dealing with. In this paper, it has been our primary

[1] Henceforth, the term "position" or "point" is used to mean both the position and orientation of robot.

concern to formulate the trajectory problem into NLP and test the validity of this approach. Since constraints in NLP are conjunction of algebraic inequalities, previous approach to collision-avoidance requiring distance computation cannot be used. In other words, a new method has to be developed. In this paper, we use *configuration space* approach and derive non-intersecting conditions for two line segments and rectangles in 2D, and cylinders in 3D. Although our method is tested by using two robots modeled by line segments and rectangles in 2D, it can be extended to general robots with more degrees-of-freedom. Further, our method can be used for optimal synthesis of multiple robot trajectories as both path and velocity of robot without requiring any prior path information.

The remainder of this paper is organized as follows. Section 2 states the problem of multi-robot trajectory planning and introduces NLP solution approach. Section 3 derives algebraic expressions for objective function and constraints for collision-avoidance, and other constraints due to joint position and velocity limits. Numerical examples and simulation results are analyzed in Section 4, and this paper is concluded with a few remarks in Section 5.

2 Problems and Solution Approach

In a workcell consisting of many robots, we wish to determine robot trajectories minimizing a certain cost function. Let S_i and E_i be the initial and destination positions of the end effector of robot i, $i = 1, \ldots, n_R$, and $\{P_i(t),\ t \in [0, T_i]\}$ be its trajectory function specifying the end effector position at time t, where n_R is the number of robots and T_i is traveling time of robot i from S_i to E_i. Further let $\Upsilon_i(t)$ and Ω be the space occupied by the i-th robot, and obstacles, $\{\mathcal{O}_i,\ i = 1, \ldots, n_{\mathcal{O}}\}$, respectively.

Taking a general cost function associated with the traversal time of each robot,

$$\mathcal{C} = \mathcal{F}(T_1, \ldots, T_{n_R}), \tag{1}$$

the minimum-cost multi-robot trajectory (MCMRT) problem is to find an optimal set of trajectory functions, $\{P_i^*(t),\ t \in [0, T_i],\ i = 1, \ldots, n_R\}$, by minimizing the cost function (1) subject to the obstacle-avoidance constraint,

$$\Upsilon_i(t) \cap \Omega = \phi,\ \forall\ i,\ t, \tag{2}$$

the mutual collision-avoidance constraint,

$$\Upsilon_i(t) \cap \Upsilon_j(t) = \phi,\ \forall\ i,\ j,\ t,\ \text{where } i \neq j, \tag{3}$$

the joint position-limit constraint,

$$H^-[P_i(t)] \in [\Theta_i^-, \Theta_i^+],\ \forall\ i,\ t, \tag{4}$$

the joint velocity-limit constraint,

$$J^-[P_i'(t)] \in [\Theta_i'^-, \Theta_i'^+],\ \forall\ i,\ t, \tag{5}$$

and the boundary conditions,

$$P_i(0) = S_i,\ \ P_i(T_i) = E_i,\ \ i = 1, \ldots, n_R, \tag{6}$$

where H^- and J^- are inverse kinematics for position and velocity, respectively, and $X' = dX/dt$.

Although the expressions in (1)-(6) state the class of problem that we are dealing with, there are a number of difficulties to develop a solution method with the presented form. The main difficulty arises from the fact that Υ, i.e., the space occupied by the manipulator is hard to represent analytically. Thus, the obstacle/collision-avoidance (2)-(3) can only be checked by computing distances between objects in 3D space. Despite the existance of efficient methods for computing distances, e.g., [9], a numerical solution requiring pointwise distance of an object to the obstacle set[2] would impose considerable computation problem. Further, since the MCMRT problem is to determining multiple robot trajectories the complexity of the problem increases exponentially as the number of robots increases. This can be easily seen even in the dual robot case; i. e., $n_R = 2$. Suppose the total number of obstacle-free paths for robot i is N_i [3] then the optimal trajectories can be determined by evaluating $N_1 \times N_2$ paths in terms of cost. To evaluate a given pair of paths, one has to deal with velocity determination problem,[4] which imposes additional computation problem. Because of the above difficulties, we will seek an alternative approach to determine the multiple trajectories without counting on distance computation.

An alternative approach chosen in this paper is to discretize the problem and to apply discrete optimization technique. By discretization, the problem is converted to that of finding $P_i(k)$, where k is the time index for interval δ[5]. Thus, the trajectory planning problem is decomposed into a series of the following subproblem,

Subproblem 1: Find $P_i(k+1)$, for a *given* positions $P_i(k)$, where $i = 1, \ldots, n_R$, subject to the constraints (2)-(5) while optimizing a performance index (1).

It is worth mentioning that **Subproblem 1** is still a continuous problem as the constraints (2)-(5) express the conditions to satisfy for a time interval δ. If δ is small enough one may treat the constraint as the conditions at the final time only; i. e., $t = k\,\delta$. Note that larger value of δ will reduce the number of subproblems, and hence less computational cost. However, in this case, it is hard to treat the constraints, particularily (2) and (3), as final conditions, because sweeping volume of robots have to be considered in such a case. Taking "end-point concept", **Subproblem 1** can be formally stated as follows.

Subproblem 2: Find $P_i(k+1)$ optimizing a performance index (1) subject to the constraints:

$$\Upsilon_i(k+1) \cap \Omega = \phi,\ \forall\ i, \tag{7}$$

$$\Upsilon_i(k+1) \cap \Upsilon_j(k+1) = \phi\ \forall\ i,\ j,\ \text{where } i \neq j, \tag{8}$$

$$H^-[P_i(k+1)] \in [\Theta_i^-, \Theta_i^+],\ \forall\ i,\ t, \tag{9}$$

$$J^-[P_i'(k+1)] \in [\Theta_i'^-, \Theta_i'^+],\ \forall\ i,\ t. \tag{10}$$

[2] Viewing from one robot, rest of the robots are moving obstacles.

[3] say the paths are from a path planning scheme based on space discretization method, e.g., [10].

[4] as the cost (1) and collision-avoidance(3) are dependent on how the paths are executed; i.e., they are function of trajectories not just paths. In [5], we presented a velocity determination algorithm using dynamic programming technique, and can be used for such purpose.

[5] It should be pointed out that the discretization is realistic as the robot control is made at every sampling instance. In such a case δ is the sampling interval of the robot control, Note however, that δ could be larger than the sampling interval in off-line synthesis of robot trajectories.

Notice that Subproblem 2 is a static optimization problem of following form:

$$\max z = C(x) \tag{11}$$

$$s.t.\ A(x) \leq b \tag{12}$$

where z, $x \in R^n$ are objective function, and decision variables, respectively, and Eq. (11) and (12) represent objective function and constraints consisting of m algebraic inequalities, respectively. Depending on functional type of C and A, the above problem is classified into linear programming (LP) or non-linear programming (NLP) problem. There are many solution packages for these problems, such as MINOS and GINO[6]. Therefore, one way to approach the multi-robot trajectory problem is to reformulate the problem into the above form and to apply existing solution packages. The remainder of this paper is mainly devoted to reformulate the problem; i. e., deriving algebraic expressions for cost function, collision-avoidance, and joint limits. It is worth pointing out that collision-avoidance strategy based on distance computation is hard to apply as the distance between objects in general can not be reprented by analytic form. We will show that collision-avoidance conditions can be represented by non-linear algebraic inequalities, and hence the trajectory problem is formulated into NLP form.

3 Derivation of algebraic conditions

In this Section, algebraic forms for the NLP formulation are derived. We first consider constraints for collision-avoidance and joint limits, followed by the selection of the performance index to form cost function. Then, the structure of the NLP solution procedure is summarized later in this Section.

3.1 Collision-Avoidance Strategy

Contemporary collision-avoidance scheme is to compute the distance between links of one robot and those of the other. Representing robot link by a spherical extension of line segment, collision-avoidance can be expressed as follows:

$$|e_1 - e_2| > (r_1 + r_2) \tag{13}$$

where $|e_1 - e_2|$ is distance between e_1 and e_2 and e_i is the spine of the $i-th$ link, and r_i is the radius of the $i-th$ link. This scheme is conceptually simple and has been widely used. With this method, however, one has to face a number of difficulties in solving the trajectory planning problem.

An alternative chosen in this paper is to express collision-avoidance condition by an *analytic form* in the parametric space so that it can be formulated into NLP. In what follows, we discuss algebraic collision-avoidance conditions (CAC) for a pair of links. First, we consider a simple case where the radii of two links are zero on plane, for which a closed-form expression is derived. Then, we treat a more complex and general case where the links are modeled as rectangles and cylinders.

3.1.1 CAC of linear links

Let $m_{i,j} = (x_{i,j}, y_{i,j})$ $(i,j = 1,2)$ be the two end points and $m_{i,0} = (x_{i,0}, y_{i,0}) = \frac{1}{2}(m_{i,1} + m_{i,2})$ be the center of the i-th link.

[6]MINOS is a NLP solver in GAMS [11] and GINO is a NLP solver [12].

Then the parametrized equation of edge i is

$$e_i(\lambda_i) = m_{i,1} + (\tfrac{1+\lambda_i}{2})(m_{i,2} - m_{i,1}) = \lambda_i(\tfrac{m_{i,2}-m_{i,1}}{2}) + m_{i,0}, \quad \lambda_i \in [-1,1], \quad i = 1,2. \tag{14}$$

Further, the two edges e_1 and e_2 have no intersection if and only if the two common solutions λ_1 and λ_2 satisfy the following condition:

$$|\lambda_1| > 1 \text{ or } |\lambda_2| > 1 \ \ (\text{equiv.}, \|(\lambda_1,\lambda_2)\|_\infty = \max(|\lambda_1|,|\lambda_2|) > 1) \tag{15}$$

As shown in Fig. 1, CAC in Eq. (15) is entire parametric space excluding a square with side length of 2. To express the feasible space (dotted region) in NLP, however, four "either-or type" constraints are necessary. In other words, to obtain the optimal solution, four NLP problems (each of which containing one of the four constraints) have to be solved, imposing computational complexity. To avoid the above problem, we replace Eq. (15) by the following single constraint:

$$\|(\lambda_1,\lambda_2)\|_n > 2^{\frac{1}{n}} \quad \text{or} \quad |\lambda_1|^n + |\lambda_2|^n > 2 \quad \text{for suff. large n}, \tag{16}$$

or

$$\begin{aligned} & 2^{n-1}\cdot[(y_{2,1}-y_{2,2})(x_{2,0}-x_{1,0}) + (y_{1,1}-y_{1,2})(y_{2,0}-y_{1,0})]^n \\ + \ & 2^{n-1}\cdot[(x_{2,2}-x_{2,1})(x_{2,0}-x_{1,0}) + (x_{1,2}-x_{1,1})(y_{2,0}-y_{1,0})]^n \\ > \ & [(x_{1,2}-x_{1,1})(y_{2,1}-y_{2,2}) - (x_{2,1}-x_{2,2})(y_{1,2}-y_{1,1})]^n \end{aligned} \tag{17}$$

where n is a positive integer. Note that the feasible space of Eq. (16) is an approximation of Eq. (15), and reduces the original feasible space a bit. For instance, with n of 2, wasted space is the area between the circle and the square. As n increases the wasted space decreases (See Fig. 2), in this case, however, numerical complexity increases.

3.1.2 CAC of Planar Rectangular Links

Assume the i-th link is a planar rectangular link with radius $r_i > 0$. The collision-avoidance between the corresponding medial axes is a necessary but not sufficient condition for the collision-avoidance of the two rectangular links. We use the concept of *Configuration space* (*C-space*) obstacles and reduce the collision-avoidance problem to the collision-avoidance between the center of the second link and the *C-space* obstacle (See Fig. 3). Since the shape of exact *C-space* obstacle is rather complex, we simplify the *C-space* obstacle by slight over-estimations (See Figure 4).

Let $m'_{i,0} = \frac{1}{2}(m_{i,1} + m_{i,2})$ $(i = 1,2)$ be the center of link i, and $m'_{i,1}$ and $m'_{i,2}$ be the mid points of right and left sides of link i. That is:

$$m'_{i,2} = (\frac{r_i}{l_i})\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}(m_{i,2} - m_{i,0}) + m_{i,0} \tag{18}$$

$$m'_{i,1} = m_{i,0} - (m'_{i,2} - m_{i,0}) = 2m_{i,0} - m'_{i,2} \tag{19}$$

There is no collision if $m_{2,0}$ is outside the regions $R_0, R_1, \ldots, R_4$ simultaneously. The avoidance of $m_{2,0}$ from the region R_0 is equivalent to the collosion-avoidance between the two medial axes. We consider the avoidance from the region R_1. The avoidances from the other regions R_2, R_3, and R_4 can be derived in similar ways. The expanded rectangular region R_i has its end point $\hat{m}_{i,2}$ and its left side point $\hat{m}'_{i,2}$ as follows:

$$\hat{m}_{i,2} = m_{i,0} + (\frac{l_i + r_{3-i}}{l_i})(m_{i,2} - m_{i,0}) \quad (20)$$

$$\hat{m}'_{i,2} = m_{i,0} + (\frac{r_1 + r_2}{r_i})(m'_{i,2} - m_{i,0}) \quad (21)$$

To surround R_1 by a closed curve, we first translate R_1 so that its center is located at the origin and apply a linear transformation so that the rectangle is mapped into a standard square of length 2. This linear transformation can be represented by a matrix:

$$\begin{bmatrix} \hat{m}_{1,2} - m_{1,0} & \hat{m}'_{1,2} - m_{1,0} \end{bmatrix}^{-1} \quad (22)$$

$$= \begin{bmatrix} \hat{x}_{1,2} - x_{1,0} & \hat{x}'_{1,2} - x_{1,0} \\ \hat{y}_{1,2} - y_{1,0} & \hat{y}'_{1,2} - y_{1,0} \end{bmatrix}^{-1} \quad (23)$$

$$= \frac{1}{(\hat{x}_{1,2}-x_{1,0})(\hat{y}'_{1,2}-y_{1,0})-(\hat{x}'_{1,2}-x_{1,0})(\hat{y}_{1,2}-y_{1,0})} \cdot \begin{bmatrix} \hat{y}'_{1,2} - y_{1,0} & x_{1,0} - \hat{x}'_{1,2} \\ y_{1,0} - \hat{y}_{1,2} & \hat{x}_{1,2} - x_{1,0} \end{bmatrix} \quad (24)$$

Thus, $m_{2,0}$ translates to

$$m_{2,0} - (m_{1,0} + m_{2,2} - m_{2,0}) = 2m_{2,0} - m_{2,2} - m_{1,0} = m_{2,1} - m_{1,0} \quad (25)$$

and then linearly transforms into

$$\begin{bmatrix} \hat{m}_{1,2} - m_{1,0} & \hat{m}'_{1,2} - m_{1,0} \end{bmatrix}^{-1} (m_{2,1} - m_{1,0}) \quad (26)$$

The avoidance of R_1 is formulated by avoiding this transformed point from the closed curve enclosing the standard square as follows:

$$\begin{aligned} & [(x_{2,1} - x_{1,0})(\hat{y}'_{1,2} - y_{1,0}) + (y_{2,1} - y_{1,0})(x_{1,0} - \hat{x}'_{1,2})]^n \\ + \; & [(x_{2,1} - x_{1,0})(y_{1,0} - \hat{y}_{1,2}) + (y_{2,1} - y_{1,0})(\hat{x}_{1,2} - x_{1,0})]^n \\ > \; & 2 \cdot [(\hat{x}_{1,2} - x_{1,0})(\hat{y}'_{1,2} - y_{1,0}) - (\hat{x}'_{1,2} - x_{1,0})(\hat{y}_{1,2} - y_{1,0})]^n \end{aligned} \quad (27)$$

The avoidance of R_2, R_3, and R_4 can be formulated similarly.

3.1.3 CAC of Cylindrical Links in 3D

(1) When the medial axes are parallel, we consider the plane Π determined by the two parallel axes and the projections of the two cylindrical links onto Π. The cylindrical links collide if and only if the projected rectangular links collide on Π. (2) When these axes are not parallel, there are two planes Π_1 and Π_2 containg each medial axis and parallel to the other axis. Further, the cross product of the two medial axis vectors is the common normal vector of Π_1 and Π_2. If the distance between Π_1 and Π_2 is larger than $r_1 + r_2$, the links are collision-free. This condition can be formulated as follows:

$$\begin{aligned} & (A(x_{1,1} - x_{2,1}) + B(y_{1,1} - y_{2,1}) + C(z_{1,1} - z_{2,1}))^2 \\ > \; & (r_1 + r_2)^2 \cdot (A^2 + B^2 + C^2) \end{aligned} \quad (28)$$

where $A = (y_{1,2} - y_{1,1})(z_{2,2} - z_{2,1}) - (y_{2,2} - y_{2,1})(z_{1,2} - z_{1,1})$, $B = (x_{2,2} - x_{2,1})(z_{1,2} - z_{1,1}) - (x_{1,2} - x_{1,1})(z_{2,2} - z_{2,1})$, and $C = (x_{1,2} - x_{1,1})(y_{2,2} - y_{2,1}) - (x_{2,2} - x_{2,1})(y_{1,2} - y_{1,1})$.
Otherwise, we consider the projection of the second link onto the plane Π_1 containing the first link. The collision-avoidance between these two projected rectangular links, a necessary but not sufficient condition for the collision-avoidance between the two cylindrical links, is considered in the following.

We can linearly transform the space so that Π_1 is mapped onto the xy-plane and the first medial axis onto the x-axis. Let

$$\begin{aligned} \tilde{m}_1 &= m_{1,2} - m_{1,0} \\ \tilde{m}_2 &= m_{2,2} - m_{2,0} - \frac{(m_{1,2}-m_{1,0})\cdot(m_{2,2}-m_{2,0})}{\|m_{1,2}-m_{1,0}\|}(m_{1,2} - m_{1,0}) \\ \tilde{m}_3 &= (m_{1,2} - m_{1,0}) \times (m_{2,2} - m_{2,0}) \end{aligned} \quad (29)$$

Then, $\tilde{M} = [\tilde{m}_1 \tilde{m}_2 \tilde{m}_3] = [\tilde{m}_1 \tilde{m}_2 \tilde{m}_3]^{-1}$ is such a transformation.

Let T be the translation by $-m_{1,0}$ and P_2 be the projection onto the xy-plane such that $P_2(x, y, z) = (x, y)$. Further, let $\tilde{m}_{i,j} = P_2 \tilde{M} T m_{i,j}$ ($i = 1, 2$ and $j = 0, 1, 2$) and $\tilde{m}'_{i,j} = P_2 \tilde{M} T m'_{i,j}$ ($i = 1, 2$ and $j = 1, 2$). Then the collision avoidance between two projected rectangular links can be formulated in a similar way as in the planar case.

3.2 Joint position/velocity-limit constraints

Using the differential relationship between Cartesian and joint space, i.e.,

$$(P_i(k+1) - P_i(k)) = J\left[H^-[P_i(k)]\right] dq, \quad (30)$$

the joint position-limit constraint is

$$\begin{aligned} & J^{-1}\left[H^-[P_i(k)]\right](P_i(k+1) - P_i(k)) \\ \in \; & [\Theta^- - H^-[P_i(k)], \Theta^+ - H^-[P_i(k)]]. \end{aligned} \quad (31)$$

Note that Eq. (20) can be represented by $2\sum_{i=1}^{n_R} n_i$ linear inequalities for n_R robots as there are $2n_i$ linear inequalities for the i-th robot, where n_i denotes the number of degree-of-freedom of the i-th robot.

Similiarily, using the differential relationship (19) the joint velocity-limit constraint is

$$J^-[P_i(k)]\,(P_i(k+1) - P_i(k)) \in \delta\left[\Theta_i'^-, \Theta_i'^+\right], \quad (32)$$

which can be represented by $2\sum_{i=1}^{n_R} n_i$ linear inequalities.

3.3 Selection of the performance index

Selection of the performance index is important as it is the key factor in determining the robot trajectories. Since the global information is not available at the time of next positions are determined, the performance index should be selected such that the available information is fully utilized in determining the next positions. We address three issues in selecting the performance index. First, we want each robot to move along the most efficient direction toward the destination. Ideal case is that every robot moves along the shortest path from the starting position to the destination. Second, we want to coordinate the robot motions based on the urgentness of each robot. In other words, higher priority should be given to the robot which is most likely late for the scheduled time. Third, the robots should converge to their destinations without blocking with each other. Note that the third issue is important in the navigation in which only local information is available.

Suppose robot i is currently at $P_i(k)$, and $P_i(k+1)$ is the next position. Then the effective movement of the robot can be defined as the projection of the incremental vector, $P_i(k+1) - P_i(k)$, onto the reference vector, $(E_i - P_i(k))/|E_i - P_i(k)|$, i.e.,

$$ED_i(k) = (P_i(k+1) - P_i(k)) \cdot \left(\frac{E_i - P_i(k)}{|E_i - P_i(k)|}\right). \quad (33)$$

Note that the reference vector is unit vector, and $|\cdot|$ is Euclidean

norm. In the sense of the projected distance, the closer to the reference vector the superior the position is. For instance, position B (Fig. 5) is superior to position A, although the distance is the same (Fig. 5(a)), or shorter (Fig. 5(b)).

To maximize the projected distance, each robot will tend to move far away from the current position along the best direction. To coordinate the robot motions effectively, we introduce a *coordination factor*, which can be thought of as a relative importance of *unit* effective distance. Suppose motion finishing times of robots are given by task scheduler, and let T_i be that of robot i. Then, at the time of k, some robot may have long way to move to its destination, while some robot is close to its destination. In the notion of urgency, more weight is given to the robot which possesses higher potential of being late for its scheduled time. Considering the velocity limit of each robot, the weighting factor, $W_i(k)$, can be defined as the ratio of an estimated velocity to reach the destination from $P_i(k)$ over the Cartesian velocity limit, V_i, i.e.,

$$W_i(k) = \frac{|E_i - P_i(k)|}{(T_i - k\delta)\, V_i}. \tag{34}$$

Finally, we want the robots to converge to their destinations without blocking in the middle. Suppose the configuration of robots are currently as shown in Fig. 6. In this case, the robots will try to move to their destinations. However, because of the collision-avoidance constraints in Section 3.1, they may have to wander around the current positions and fall into deadlock. In such a case, we want the robots retract to prevent deadlock in advance. The degree of retraction should be function of how close they are; i. e., the current distance between two robots. Besides, if they are far enough to ignore deadlock possibility, the weight must be given by 0. In other words, we want to maximize the following term:

$$\nu \ |e_1^{k+1} - e_2^{k+1}|, \tag{35}$$

where ν is the blocking weigh defined as

$$\nu = exp(-c\,|e_1^k - e_2^k|). \tag{36}$$

Note that c is constant determining the blocking weight, and ν approaches 0 (1) as current distance between two robots increases (decreases).

In this way the weight of blocking term can be represented algebraically. However, since Eq. (35) contains distance between two links at the next time increment which can not be represented algebraically, we approximate the distance by the two points defining the distance of current links; i.e.,

$$|e_1^{k+1} - e_2^{k+1}| \approx |e_1^k(\lambda_1) - e_2^k(\lambda_2)|, \tag{37}$$

where $|e_1^k - e_2^k| = |e_1^k(\lambda_1) - e_2^k(\lambda_2)|$.

Considering all the aspects discussed above, the performance index to maximize is

$$PI = (1-\nu)W_1 ED_1 + (1-\nu)W_2 ED_2 + \nu|e_1^k(\lambda_1) - e_2^k(\lambda_2)|. \tag{38}$$

It is worth mentioning that W_i can be defined differently based on the task characteristics and the scheduling. For instance, if the motion finishing times are not given but to be determined, then $W_i = 1,\ \forall\, i$, may be taken. Note also that W_i can be viewed as a coordination rule for the central controller, or it can be viewed as a design factor for the robot trajectory planner.

3.4 NLP Solution Procedure

Based on the discussion so far, the problem of finding the optimal positions for time $k+1$, i.e., $P_i(k+1)$, $i = 1, \ldots, n_R$, for given positions of $P_i(k)$, $i = 1, \ldots, n_R$ can be formulated into nonlinear programming (NLP). The NLP is a maximization problem with the objective function (38) subject to the collision/obstacle-avoidance constraints ((17), or (27)) joint position-position limit constraints (31), and joint velocity-limit constraints (32). This problem can be readily solved by using NLP solver; e.g., MINOS [11], GINO [12]. Since the NLP algorithm itself is beyond the scope of this paper, we only present solution procedure to apply it for optimal trajectory planning. The solution procedure consists of three steps:

- *Step 1*: Initialization.
  1. Read in S_i, E_i, $i = 1, \ldots, n_R$, and the obstacle geometry and robot kinematic data.
  2. Set $k := 0$.
  3. For $i = 1, \ldots, n_R$
     $P_i(k) = S_i$, $g(i) = 1$ (indicating robot i has not reached the destination)
- *Step 2*: Termination check.
  1. For i such that $g(i) = 1$
     If $|P_i(k) - E_i| \approx 0$, then $g(i) = 0$ (indicating robot i has reached the destination)
  2. If $g(i) = 0$, $\forall i$, stop.
     Otherwise, go to *Step 3*.
- *Step 3*: Finding $P_i^*(k+1)$, $\forall\, i \in g(i) = 1$.
  1. Solve the NLP problem by applying a NLP algorithm. (Note: Treat the *reached* robots, i.e., $g(i) = 0$, as fixed obstacles in the NLP formulation.)
  2. Set $P_i(k) = P_i^*(k+1)$, and $k := k+1$.
     Go to *Step 2*.

4 Numerical Examples

To test the developed algorithm, we implemented it to an IBM-PC. Since the objective of our research was not to develop a NLP solver and many commercial packages are available, we only implemented the application algorithm described in Section 3. The NLP solver we used is MINOS in GAMS developed by Murtage and P. Gill [11]. However, the package we used was a "Demonstration Version" in which number of variables and constraints were limited, full 3D robotic model could not be implememted requiring more variables and equations than it accommodate.[7] The test was performed for two two-d.o.f. Scara type robots whose kinematic data including the starting and destination points are shown in Table 1.

With $r_i = 0$, $i = 1, 2$ (i. e., the linear links), the optimal paths at each time step ($\delta = 0.1sec$) is shown in Fig. 7. The

[7] Note, however, that professional versions can easily accommodate these requirments.

results show that robot 1 and 2 reaches their destinations in 1.6 and 1.0 seconds, respectively. The optimal paths are close to straight lines except for the midway where the two arms are close to each other around t = 0.5 second. At that time robot 1 moves down so that robot 2 proceeds toward the destination since robot 2 has the way of right due to the scheduled time (T_1 = 1.5 sec, T_2 = 1.0 sec). Note, however, that they do not always reach destinations in the time T_i, and in some cases the arrival time may be far from being specified. This is because the purpose of T_i is in essence to provide traffic rule in the jammed area.

Through out the navigation, the linear links does not get close to much. This can be explained by n value of CAC and blocking weight ν. The value of n used is 2 (i. e., circle in parametric space), which in turn over-constrained the feasible space a bit. Consequently, it prohibits two linear links getting close too much. Alternatively, the robot could not get close too much because the blocking weight ν increases as they get close to each other. This can be more clearly seen in an example consisting of two rectangular links.

The example shown in Fig 8 is the same as that of Table 1, except for the radii of links are 5. The robot 1 and 2 reached the destinations in 1.7 sec and 1.1 sec, respectively. The early part of the navigation is about the same as that of the linear links. Since $t = 0.6sec$ the robot 1 and 2 are getting close to each other, and at $t = 0.7$ they give up proceeding to their best directions, and try to get apart instead. Robot 1 moves back to the direction whose effective distance is negative, while robot 2 moves to the direction whose effective distance is positive small value. Consequently, robot 1 takes more time to recover (to getting out of the jam) than robot 2.

5 Concluding remarks

The problem of determining multiple robot trajectories without priori path information is addressed in this paper. We formulated the problem into a static optimization problem and solved by using NLP solver. To apply NLP, we derived algebraic expressions for objective function, collision-avoidance and joint limit constraints. Derivation of joints limit constraints are straightforward, while collision-avoidance constraints calls for special attentions as conventional strategy requiring distance computation is hard to apply. Thus, we derived collision-avoidance conditions in a *configuration space* for a pair of linear, rectangular, and cylindrical links. Also, selection of performance index is important as it is a key factor to determines robot trajectories. We addressed path direction and motion priority by which all the robots can reach their destinations without blocking. The developed algorithm has been implemented in PC and the validity of our method has been proved through the numerical examples. The test results convinced us the strength of the NLP approach such that it can be extended to multi-link robotic models, and hence it will provide a means for optimal synthesis of multiple robot trajectories.

References

[1] E. Freund, and H. Hoyer, "Pathfinding in multi-robot systems:Solution and applications," in *Proc. Int. Conf. on Robot. Automat.* (San Francisco, CA, Apr. 1986), pp. 103-111.

[2] P. Tournassoud, "A strategy for obstacle avoidance and its application to multi-robot systems," *Proc. Int. Conf. on Robot. Automat.* (San Francisco, CA, Apr. 1986), pp. 1224-1229.

[3] B. H. Lee and C. S. G. Lee, "Collision-Free motion planning of two robots," *IEEE Trans. on Syst., Man, and Cybern.*, vol. SMC-17, no. 1, pp. 21-32, Jan./Feb. 1987.

[4] N. Y. Chong, D. H. Choi, and I. H. Suh, "Collision-Free trajectory planning for dual robot arms," *Proc. Conf. on Automat. Contr.* (Seoul, Korea, Oct. 1988), pp. 951-957.

[5] T. J. Tarn, A. K. Bejczy, and X. Yun, "Design of dynamic control of two cooperating robot arms: Closed chain formulation," *Proc. Int. Conf. on Robot. Automat.* (Raleigh, NC, Mar. 1987), pp. 7-13.

[6] Y. F. Zheng and J. Y. S. Luh, "Joint torques for control of two coordinated moving robots," *Proc. Int. Conf. on Robot. Automat.* (San Francisco, CA, Apr. 1986), pp. 1375-1380.

[7] J. H. Lim and D. H. Chyung, "On a control scheme for two coorperating robot arms," *Proc. Int. Conf. on Robot. Automat.* (St. Louis, MO, Mar. 1985), pp. 334-337.

[8] S. H. Suh, "A study on the trajectory planning for multi-robot systems," *Proc. Joint Conf. of Korean Management Science and Korean Industrial Engineering* (Pohang, Korea, Mar. 1989), pp. 18 - 25.

[9] E. G. Gilbert, D. T. Johnson, and S. S. Keerthi, "A fast procedure for computing the distance between complex objects in three-dimensional Spac," *IEEE Trans. on Robot. Automat.*, vol. 4, no. 2, pp. 193-203, Apr. 1988.

[10] C. E. Thrope, "Path relaxation: Path planning for a mobile robot," Carnegie Mellon Univ., Pittsburgh, PA, Tech. Rep. CMU-RI-TR-84-5, 1984.

[11] A. Brooke, D. Kendrick, and A. Meeraus, *GAMS: A user's Guide*, Redwood City, CA: The Scientific Press, 1988.

[12] J. Liebman, L. Lasdon, L. Schrage, and A. Waren, *Modeling and optimization with GINO*, Palo Alto, CA: The Scientific Press, 1986.

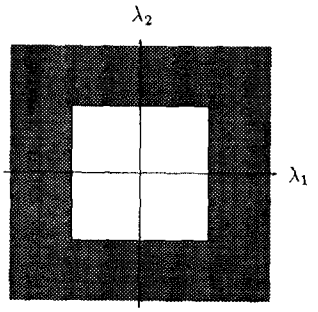


Figure 1: Feasible Space

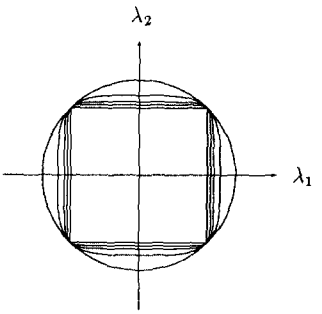


Figure 2: $x^n + y^n = 2$ for $n = 2, 4, 8, 16$

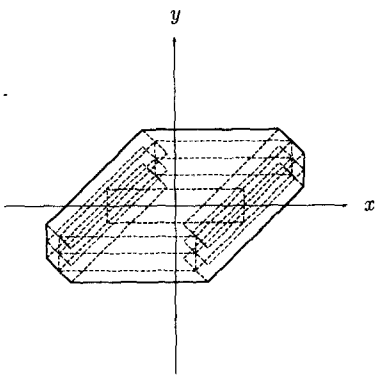


Figure 3: C-space obstacle

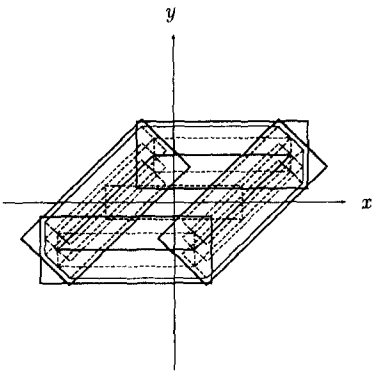
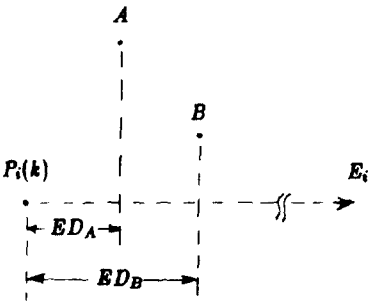
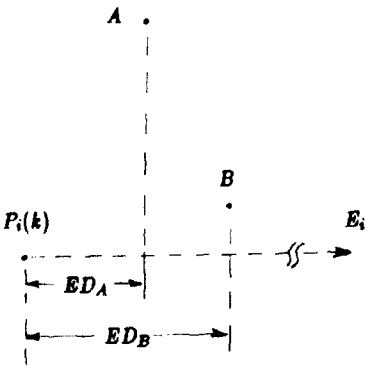


Figure 4: Approximated C-space obstacle



(a) Position B is superior to A, although the distance is the same.



(b) Position B is superior to A, although the distance is shorter.

Figure 5: Comparison of Various Positions.

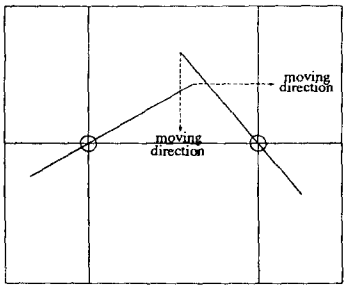


Figure 6: Blocking Configuration.

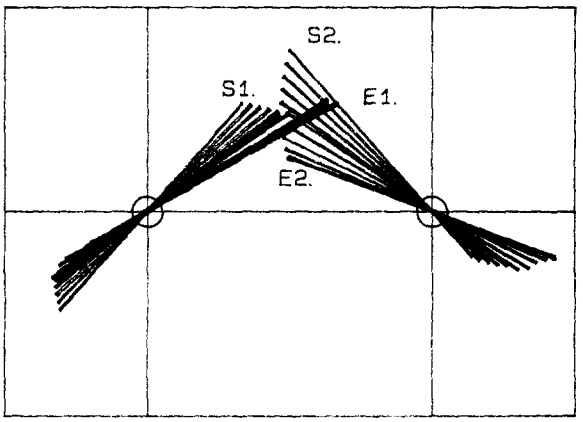


Figure 7: Optimal Paths of Linear Links.

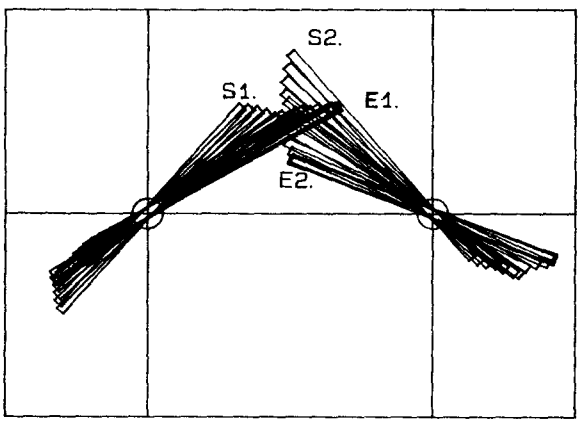


Figure 8: Optimal Paths of Rectangular Links.

Table 1. Robot kinematic data including starting and destination positions.

Items	Value	Unit
Arm length (L)*	1.5	meter
Joint position limit*	Joint 1: [0.3, 1.4] Joint 2: [0, 360]	meter degree
Joint velocity limit*	Joint 1: [-0.3, 0.3] Joint 2: [-30, 30]	meter/sec degree/sec
Starting positions	Robot 1: [0.5, 0.5] Robot 2: [0.75, 0.75]	meter meter
Ending positions	Robot 1: [1.0, 0.5] Robot 2: [0.75, 0.25]	meter meter

* Robot 1 and 2 have the same values.