International Journal of Robotics and Automation, Vol. 16, No. 1, 2001

UTILIZING THE TOPOLOGY OF CONFIGURATION SPACE IN REAL-TIME MULTIPLE MANIPULATOR PATH PLANNING

RECEIVED
APR 10 2001
By____

J.J. Fox* and A.A. Maciejewski*

Abstract

This article provides a set of algorithms that allow qualitative information regarding the connectivity of configuration space to be quickly established. A mechanism is presented that utilizes these results to determine the effects of the motions of one manipulator on the configuration space of the other. These algorithms are then used as a basis for a simple planner that is capable of rapidly computing collision-free paths for multiple SCARA manipulators operating within overlapping workspaces.

Key Words

Path planning, obstacle avoidance, multiple manipulators

1. Introduction

There has recently been growing recognition of the advantages achievable by placing more than one manipulator in a common workspace. Besides being able to perform tasks in parallel, the manipulators may be used cooperatively, thereby increasing the dexterity and load-carrying capabilities that may be brought to bear on a particular task. Unfortunately, these advantages come at a cost, including the problem of determining paths for each of the manipulators that will avoid striking obstacles in the environment while at the same time avoiding collisions with each other.

In the past several years there have been numerous approaches to this problem [1], including treating the manipulators as a redundant system [2] or using cellular decomposition techniques [3]. Among the numerous related algorithms that consider robots moving among moving obstacles are the spatial indexing of configuration space-time [4] and the use of the relative velocities of the objects and the robots to transform the problem into one of several static problems.

* Purdue University, 1285 Electrical Engineering Building, West Lafayette, Indiana 47907-1285 USA;
e-mail: maciejew@ecn.purdue.edu

(paper no. 206-2024)

One particularly popular approach [5] imposes priorities upon the manipulators and then plans the paths of one robot at a time, using the higher priority robots as obstacles in the configuration space-time representation of the lower priority robots. Another common approach to planning robot paths that must avoid moving obstacles is to decompose the problem into a two-phase approach, commonly referred to as path-velocity decomposition [6]. In this approach, the problem is simplified by solving for the motion among the static obstacles and subsequently planning the velocity along these paths so as to avoid the moving obstacles. Although this approach is both conceptually and computationally appealing, it suffers from being unable to find paths in situations in which a solution may be intuitively obvious. In particular, as illustrated in Fig. 1, there are cases in which the solution to the first

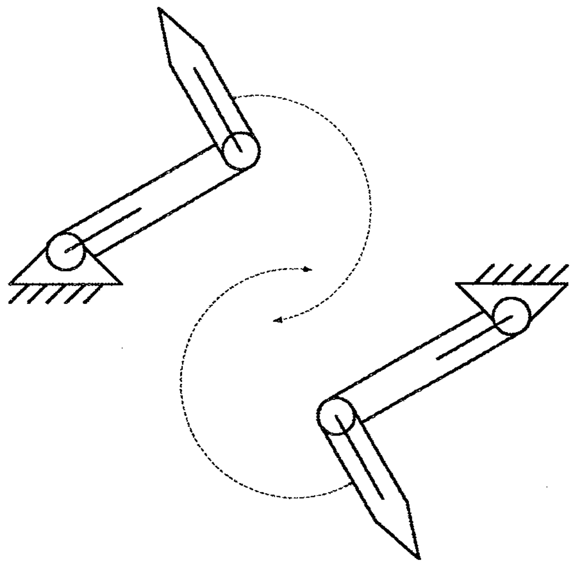


Figure 1. A typical situation in which the initial choice of paths keeps a path-velocity decomposition planner from finding a solution during velocity planning. The arrows indicate the volume that will be swept through by the links.

phase of planning results in paths along which no appropriate velocity profile exists. Often, if the path-planning phase had chosen some other path for either of the two manipulators, then a solution to the overall problem would have been found. This work focuses on this issue and attempts to find a solution in such a situation.

In particular, a set of algorithms has been developed for rapidly establishing the presence of intersections in configuration space between various classes of workspace obstacles. These algorithms are then used as a means of establishing the effects of one manipulator's motions on the connectivity of the other manipulator's free space. The net result is an approach for rapidly determining whether the motions of two manipulators will lead to a collision. This approach is then demonstrated by simulation on a SPARC workstation and average timing results are presented.

The remainder of this article is organized as follows: Section 2 reviews basic results from forward and inverse kinematics. Algorithms for establishing the presence of intersections between obstacles in $\mathcal{C}$ are developed in Section 3. Section 4 begins by presenting a mechanism for mapping connected regions in one manipulator's configuration space into the other manipulator's configuration space. It concludes by using the various algorithms that have been developed as the basis for a simple planner that computes collision-free motions for multiple manipulators. Section 5 illustrates the operation of this planner on a particularly simple example and goes on to provide typical timing results for randomly generated environments. Section 6 provides a discussion of the conclusions of this work and indicates some of its limitations.

1.1 Notation/Terminology

Throughout this article, the following notation will be employed.

W = the workspace
$\mathcal{C}$ = the configuration space
$\mathcal{B}_i$ = an obstacle in the workspace
$\mathcal{B}$ = all of the obstacles in $\mathcal{W}$
$\mathcal{CB}_i$ = the configuration space representation of the B_i
$\mathcal{CB}$ = all of the obstacles in $\mathcal{C}$
$\mathcal{C}_{free}$ = the manipulator's free space
$\theta_{12} = \theta_1 + \theta_2$
$c_i = \cos\theta_i$
$s_i = \sin\theta_i$

The following terms will also be used. A *region* will be considered a path-connected subspace of $\mathcal{C}$ that has the same pair of obstacles to its left and right in configuration space. A *channel* will be some sequence of regions, and a *path* will be a sequence of configurations in $\mathcal{C}_{free}$.

1.2 Assumptions

A number of simplifying assumptions were made in this work. The most obvious was to model obstacles in the workspace as points and the SCARA manipulators as line segments. The purpose of these initial simplifications was to focus the presentation on fundamental aspects of the algorithm. Section 3 develops a more general characterization of obstacles, and [7] discusses robot links that have been modelled as polygons.

2. Kinematics

The benefits of path planning in a robot's configuration space [8] have been well established in the literature [9, 10]. The underlying concept of this approach is the recognition that a robot may be represented as a point in configuration space travelling through a set of obstacles that are obtained as the result of a transformation on the real obstacles in the manipulator's workspace. The process of path planning is then heavily dependent on the relationship between the manipulator's configuration space, $\mathcal{C}$, and its workspace, $\mathcal{W}$. In this section, the nature of the relationship between these two spaces is presented at both the position level and the velocity level.

For the manipulator depicted in Fig. 2, the transformation that describes the relationship between a manipulator's configuration, (θ_1, θ_2), and the Cartesian position of the end-effector, (x_{eff}, y_{eff}), is easily calculated, using forward kinematics [11], as:

$$\begin{bmatrix} x_{eff} \\ y_{eff} \end{bmatrix} = \begin{bmatrix} L_1 c_1 + L_2 c_{12} \\ L_1 s_1 + L_2 s_{12} \end{bmatrix} \tag{1}$$

where L_1 and L_2 are the lengths of links 1 and 2.

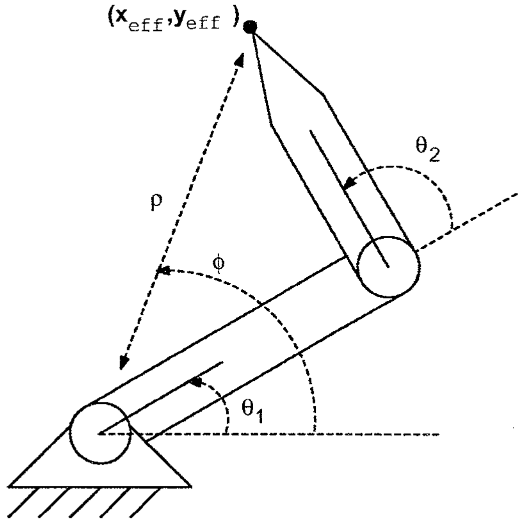


Figure 2.The geometry of a SCARA-type manipulator.

Solving (1) for θ_1 and θ_2 yields the equally well known inverse transformation describing the inverse kinematics for this manipulator:

$$\theta_1 = \tan^{-1}\frac{y_{eff}}{x_{eff}} \pm \cos^{-1}\frac{x_{eff}^2 + y_{eff}^2 + L_1^2 - L_2^2}{2L_1\sqrt{x_{eff}^2 + y_{eff}^2}} \tag{2}$$

and:

$$\theta_2 = \pm\cos^{-1}\frac{x_{eff}^2 + y_{eff}^2 - L_1^2 - L_2^2}{2L_1L_2} \tag{3}$$

The relationship between the end-effector velocities and the joint velocities is readily obtained by differentiating (1) to obtain:

$$\begin{bmatrix} \dot{x}_{eff} \\ \dot{y}_{eff} \end{bmatrix} = \begin{bmatrix} -L_1 s_1 - L_2 s_{12} & -L_2 s_{12} \\ L_1 c_1 + L_2 c_{12} & L_2 c_{12} \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix} \tag{4}$$

Recall that the ultimate goal of this development is to provide a mechanism for rapidly determining whether obstacles in $\mathcal{C}$ intersect. Computing this intersection using (2) and (3) would require the simultaneous solution of nonlinear equations, a task which is, in general, nontrivial. However, for many of the purposes of this work, it will suffice to establish the presence of an intersection without knowing precisely where it occurs. As will be shown in Section 3, a characterization of configuration space obstacles has been developed that is sufficient for establishing the presence of intersections between configuration space obstacles. The basis of this characterization is a representation of the obstacle by its tangents. As shown in [7], the tangent of an obstacle in $\mathcal{C}$ corresponding to a point in $\mathcal{W}$ is readily obtained as:

$$\frac{d\theta_2}{d\theta_1} = \frac{\ell_2 + L_1 c_2}{-\ell_2} \tag{5}$$

where ℓ_2 is the length along the second link at which the contact with the obstacle occurs.

3. The Topology of $\mathcal{C}_{free}$

The notion of utilizing topological properties in generating obstacle representations for planning has been studied extensively. Several researchers [12, 13] have generated representations of $\mathcal{C}$ by convolving a representation of a free-flying robot with a representation for an obstacle. Because the mechanism used for performing these convolutions tended to produce a small number of vertices, edges, and faces that were either redundant or otherwise nonrealizable, a second stage was employed that utilized topological information to cull out these extraneous pieces of information. Additional researchers [14] have chosen to characterize the topology of $\mathcal{W}$ rather than $\mathcal{C}$ and use an octree representation to evaluate the connectivity of the free workspace. Here, the topological properties of the obstacles are used as a filter for removing regions of $\mathcal{C}$ in which a collision-free path cannot be found.

Before we discuss the details of calculating topological features of $\mathcal{C}$, it is instructive to consider what type of global features can be used to improve the efficiency of most path planners. One important feature of free space for two-dimensional revolute manipulators that has been previously identified is the existence of "highways" [15]. Physically, a highway is a distinguished subspace of configuration space for which a collision-free path can be planned simply by using a line segment parallel to the θ_1 axis for some relatively large range of θ_2 values. Other global features of $\mathcal{C}_{free}$ that are not guaranteed to exist include a path from one highway to the other, referred to as an *isthmus*. If this feature does not exist, then this implies that the free space is further partitioned. Regions of $\mathcal{C}_{free}$ that are connected to only one of the two highways will be referred to as *peninsulas*. Additional details on these properties and their computation may be found in [7]; however, it should be apparent from their definitions that the ability to determine whether obstacles in $\mathcal{C}$ intersect will be critical to any algorithm that is used. The purpose of the remainder of this section is to address the question of how best to determine whether such an intersection exists.

3.1 Intersections Between Point Obstacles in $\mathcal{W}$

Consider the ways in which a SCARA manipulator may come into contact with a pair of point obstacles:

1. Both contacts may take place along the first link.
2. One contact may be along the first link and the other along the second link.
3. Both contacts may take place along the second link.

Testing for the first condition is trivial. If the two obstacles are represented in polar coordinates as (ρ_f, ϕ_f) and (ρ_g, ϕ_g), then the obstacles intersect if both are at a radius less than L_1 and $\phi_f = \phi_g$. The second case is only slightly more complicated, as one must only check to see whether the two values of θ_1 for the end-effector to be in contact with the one obstacle bracket the value of ϕ for the obstacle at $\rho \le L_1$. The remainder of this section will consider the final case.

Let $\mathcal{B}_f$ and $\mathcal{B}_g$ be point obstacles in $\mathcal{W}$ that have the Cartesian coordinates (x_f, y_f) and (x_g, y_g) with respect to the base of the manipulator. If the points are both assumed to be at a radius greater than L_1, then the following lemma provides a necessary condition for testing for an intersection between obstacles in configuration space.

Lemma 1: If $\mathcal{CB}_f \bigcap \mathcal{CB}_g \neq \emptyset$ then $\mathcal{CB}_f \bigcap \mathcal{CFL} \neq \emptyset$ and $\mathcal{CB}_g \bigcap \mathcal{CFL} \neq \emptyset$ where:

$$\mathcal{CFL} = \left\{ (\theta_1, \theta_2) \mid \theta_1 + \theta_2 = \tan^{-1}\left(\frac{y_f - y_g}{x_f - x_g}\right) \right\} \tag{6}$$

The proof to this can be readily seen by recognizing that Lemma 1 merely states that the second link of the manipulator must be parallel to the unique line passing through the two obstacles if it is to be in contact with both of them simultaneously. Unfortunately, because the inverse tangent function does not return a unique value, application of this lemma would require checking for the intersection of the configuration space obstacles with each of the lines, which have slope -1 and an appropriate intercept. However, by simply choosing a particular member of this set of functions, not only does this lemma become easier to apply, but it is also strengthened in such a way as to become sufficient. In particular, if the direction in polar coordinates of the vector from $\mathcal{B}_g$ to $\mathcal{B}_f$ is denoted by ϕ_Δ, then the intersection between $\mathcal{CB}_g$ and $\mathcal{CB}_f$ can be established by testing for intersection with the particular line:

$$\theta_1 + \theta_2 = \phi_\Delta \tag{7}$$

If the set of configurations that lie along this line are denoted $\mathcal{CL}$ where:

$$\mathcal{CL} = \{(\theta_1, \theta_2) \mid \theta_1 + \theta_2 = \phi_\Delta\} \tag{8}$$

then one can obtain the following lemma.

Lemma 2: Given the assumptions and definitions of the previous paragraphs, $\mathcal{CB}_f \bigcap \mathcal{CB}_g \neq \emptyset$ if and only if $\mathcal{CB}_f \bigcap \mathcal{CL} \neq \emptyset$ and $\mathcal{CB}_g \bigcap \mathcal{CL} \neq \emptyset$.

The necessity of this condition may be established fairly easily from Lemma 1. The sufficiency condition may also be established by using a simple geometric construction to examine the elements that are members of the intersections of $\mathcal{CL}$ with $\mathcal{CB}_f$ and $\mathcal{CB}_g$.

At first glance, it may appear that the benefits of having established this property are negligible, as the net effect appears to have been to eliminate the need to calculate the intersection between two nonlinear functions at the expense of now having to twice calculate the intersection between nonlinear functions with a straight line. Fortunately, however, the obstacles in configuration space have additional properties that prove particularly useful.

Recall from calculus that the local extrema in distance from a curve to a line is at the points along the curve at which the tangent matches the slope of the line. Setting the slope of the obstacle as described by (5) equal to the slope of the constraint equation (8) results in:

$$\frac{\ell_2 + L_1 c_2}{-\ell_2} = -1 \tag{9}$$

which has the solution $\theta_2 = \pm\pi/2$. If the obstacle does not extend past $\theta_2 = \pm\pi/2$, then the local extrema with respect to these lines will be at those points that correspond to the end-effector resting upon the obstacle. Hence, the local extrema of an obstacle with respect to the line along which an intersection must lie may be calculated via two applications of the inverse kinematics function. Thus, an algorithm to determine if the line defined by (8) intersects a configuration space obstacle would evaluate the curve describing the obstacle at only two points, those at $\theta_2 = \pm\pi/2$, and determine if they lie on opposite sides of this line. Similar results exist for obstacles that are not constrained to lie at a radius greater than L_1. An example of this test succeeding is illustrated in Fig. 3. The implication of this result is that by computing four inverse kinematic solutions and applying appropriate logic, one may determine if two configuration space obstacles intersect.

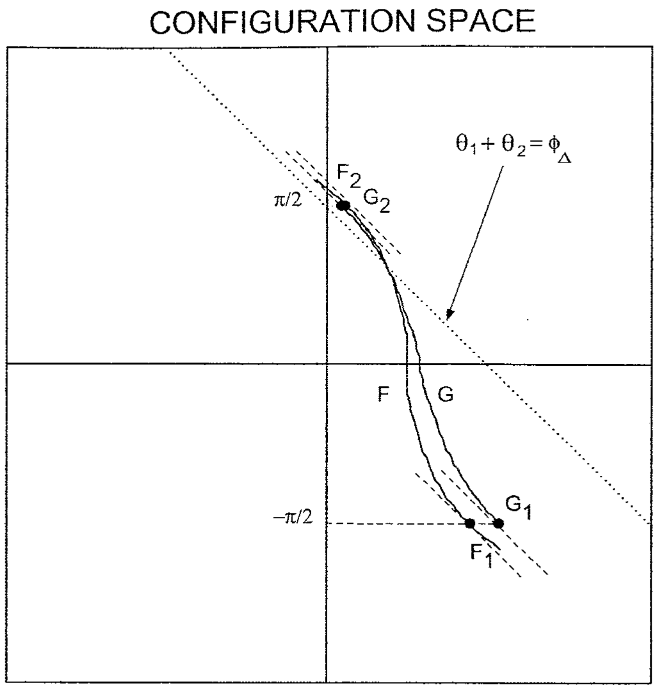


Figure 3. An example in which the intersection test between point obstacles F and G succeeds. F_i and G_i represent the point that must be computed in order to apply the test. The dotted like depicts the line along which a potential intersection must occur. The fact that these obstacles intersect is known by simply showing that the points G_1 and G_2 as well as the points F_1 and F_2 lie on opposite sides of this line.

3.2 Intersections between Points and Line Obstacles

In the previous section, the manner in which a manipulator may come into contact with multiple point obstacles was analyzed. This yielded an easily computed test for establishing the presence of intersections between the representation of point obstacles in $\mathcal{C}$. This section follows an analogous development for somewhat more general obstacles, namely line segments, with circular obstacles to be considered in the following subsection. The reason for the interest in these particular classes of obstacles will become apparent in Section 4, where the effects of motions through regions in one manipulator's configuration space are studied with regard to their effects on the connectivity of the other manipulator's configuration space.

As in the test developed above, the first step in developing this algorithm is to characterize the line segment with regard to its local extrema in $\mathcal{C}$ with respect to lines of slope -1. Clearly, these extrema must take place along the boundary of the configuration space obstacle. If one considers the manner in which the manipulator may be in contact with the obstacle, then it becomes clear that this boundary can be decomposed into simpler curves corresponding to cases in which either the second link of the manipulator slides along the end points of the line segment, or the end-effector of the manipulator traverses the interior of the line segment in a manner described by (4). The extrema of the obstacle may be computed by determining the extrema along each of the four portions of the boundary and applying appropriate logic.

Let $\mathcal{B}_S$ be a line segment in $\mathcal{W}$ with endpoints $\mathcal{B}_f = (x_f, y_f)$ and $\mathcal{B}_g = (x_g, y_g)$. A necessary first step in the characterization of $\mathcal{CB}_S$ is then to characterize $\mathcal{CB}_f$ and $\mathcal{CB}_g$ as in the previous section. The extrema of the remaining portions of $\mathcal{CB}_S$ may be computed by considering the obstacle at the velocity level. If it is assumed that $[\delta x, \delta y]$ is a vector along the line segment, then the tangent of those portions of the boundary due to the traversal of the end-effector along the interior of the line segment is obtained by solving (4) as:

$$\frac{d\theta_2}{d\theta_1} = \frac{-L_1c_1\delta x - L_2c_{12}\delta x - L_1s_1\delta y - L_2s_{12}\delta y}{L_2c_{12}\delta x + L_2s_{12}\delta y} \qquad (10)$$

Setting this slope to -1 and solving for θ_1 yields the condition that:

$$\theta_1 = \tan^{-1}\left(-\frac{\delta x}{\delta y}\right) \qquad (11)$$

which must be satisfied in order to be at a local extrema along the interior portions of $\mathcal{CB}_S$. Before continuing, it should be stressed that this result is not limited to line segments. In fact, the extrema of any obstacle with respect to lines in $\mathcal{C}$ of slope -1 that place the end-effector on the obstacle will always satisfy this condition, so long as the obstacle can be represented by a differentiable curve.

From (11), the specific configurations at which the extrema occur are given by:

$$\theta_1 = \tan^{-1}\left(-\frac{x_f - x_g}{y_f - y_g}\right) \qquad (12)$$

The corresponding value of θ_2 may be determined in a fairly straightforward manner.

Now, consider a point obstacle at (x_i, y_i). If $K_1 = \tan^{-1}\left(\frac{y_f - y_i}{x_f - x_i}\right)$ and $K_2 = \tan^{-1}\left(\frac{y_g - y_i}{x_g - x_i}\right)$, then the potential orientations of the second link that bring it simultaneously in contact with both the point obstacle and the line segment may be described by the family of lines:

$$\mathcal{CL}_{seg} = \{(\theta_1, \theta_2) \mid K_1 \le \theta_1 + \theta_2 \le K_2\} \qquad (13)$$

where it has been assumed, without loss of generality, that $K_1 \le K_2$. Given this information, and the characterization of $\mathcal{CB}_S$ illustrated in Fig. 4, an algorithm to test for intersections between the representations of points and line segments is readily obtained.

3.3 Intersections between Points and Circular Obstacles

Characterizing configuration space obstacles that represent circles or arcs may be done in a manner directly analogous to the method presented for characterizing line segments. First, consider the situation in which the end-effector is lying along a circle $x = x_0 + r\cos(\psi)$ and $y = y_0 + r\sin(\psi)$. Differentiating the equations describing such a circle and applying (11) yields a description of those configurations in which the end-effector is in contact with the circle and the corresponding obstacle in $\mathcal{C}$ is at a local extrema. This condition is described by:

$$\theta_1 = \psi + n\pi \qquad (14)$$

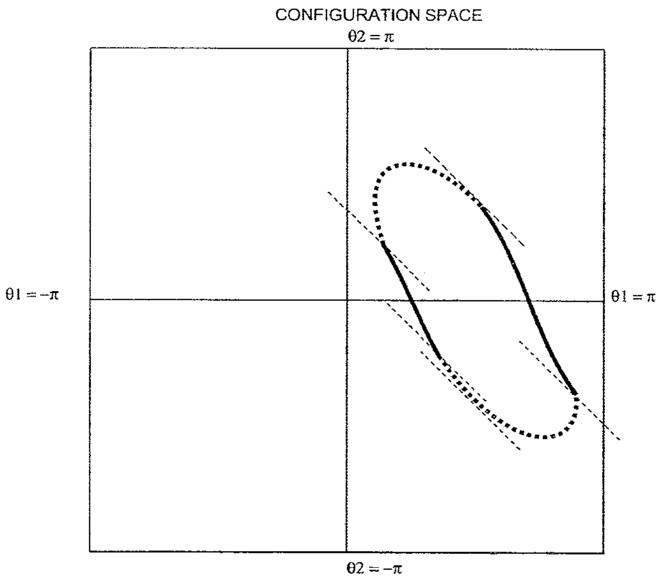


Figure 4. The characterization of an obstacle due to a line segment. The solid lines are those portions of the obstacle due to the manipulator sliding along the end points of the line segment. The bold dashed lines represent those portions of the obstacle due to the traversal of the end-effector along the interior of the line segment. The normal-weight dashed lines are the local extrema of the obstacle with respect to lines of slope -1.

or, in other words, the first link must be parallel to the line segment between the centre (x_0, y_0) and the point at which the end-effector touches the obstacle. Substituting this condition into the forward kinematic equations and solving for those configurations that place the end-effector on the circle yields those configurations as:

$$\theta_1 = \tan^{-1}\left(\frac{y_0}{x_0}\right) \pm \cos^{-1}\left(\frac{d^2 + (\pm L_1 - r)^2 - L_2^2}{2L_1 d}\right) \qquad (15)$$

and:

$$\theta_2 = \mp\cos^{-1}\left(\frac{d^2 - (\pm L_1 - r)^2 - L_2^2}{2(L_1 - r)L_2}\right) \qquad (16)$$

where $d = \sqrt{x_0^2 + y_0^2}$.

Determining the configuration that places the contact somewhere within the interior of the second link is most readily accomplished by recognizing two critical facts. First, the necessary condition on the relationship between θ_1 and the polar coordinates of the contact given by (14) continues to hold, so $\theta_1 = \psi + n\pi$. Second, for the configuration to lie along the boundary of $\mathcal{C}_{circle}$, the second link of the manipulator must lie along a tangent of the circle; hence $\theta_1 + \theta_2 = \psi \pm \frac{\pi}{2}$. When these facts are combined, it is clear that $\theta_1 = \psi$ and $\theta_2 = \pm\frac{\pi}{2}$.

It is easy to show that the location along the link at which the contact takes place is given by:

$$\ell_2 = \sqrt{-(L_1 - r)^2 + d^2} \qquad (17)$$

If the evaluation of (17) results in $0 \leq \ell_2 \leq L_2$, then the configurations satisfying this condition are given by:

$$\theta_1 = \cos^{-1}\left(\frac{x_0(L_1 - d) \pm \ell_2 y_0}{\ell_2^2 + (L_1 - d)^2}\right)$$

$$\theta_2 = \mp\frac{\pi}{2} \tag{18}$$

An approach similar to the one described for line segments would then result in an intersection test.

4. A Simple Planner

The key to our approach to planning collision-free paths is a mechanism for meshing channels in different configuration spaces. The method for accomplishing this is based upon mapping regions in one configuration space into the configuration spaces of the other manipulators as an obstacle. Clearly, this approach is heavily motivated by the work of [5], in that the motions of one manipulator are modelled as an obstacle in the configuration space-time of the other manipulators. The difference in this work is that the choice of a specific path within the region is not considered until after the global planning stage has been completed. Instead, the set of all possible paths through a region are, in effect, considered when generating the configuration space-time obstacle. The remainder of this section deals first with the mechanism chosen for transforming the set of possible motions of one manipulator into obstacles in the other manipulators' configuration spaces, and then describes the planner that has been implemented.

As mentioned above, the principal result of this section is an ability to study the sets of possible motions of one manipulator with regard to their effects on the topology of another manipulator's configuration space. More specifically, a representation is built that approximates the set of all possible postures of the robot when it is in any configuration within a region. The approach used to perform this operation relies heavily on the computation of those portions in the workspace called *shadows* [16], which describe the regions through which the link of the manipulator will sweep while it stays in contact with the obstacle. When an approximation is built that encloses the shadows of all of those points obtained by interpolating between the two obstacles, the resulting area is a conservative approximation of the set of all postures in which the manipulator may find itself when it is at any configuration within the region. This is illustrated in Fig. 5. The resulting region may then be mapped into the configuration space of the other robot and treated as though it were a static obstacle in this manipulator's workspace. Finally, the effects of this artificial obstacle on the topology of the second manipulator's free space are determined using the tests of the previous section.

First, consider the portion of $\mathcal{W}$ in which the manipulator may lie when it is in any configuration that brings it into contact with a point obstacle. Let $\mathcal{B}_i$ denote such a point obstacle, which is at the polar coordinates (ρ_i, ϕ_i) with respect to the base of the manipulator. The curve

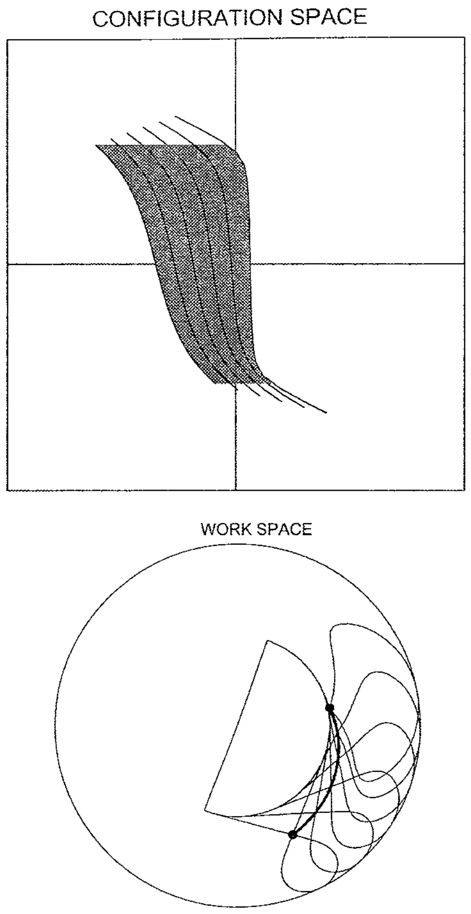


Figure 5. The process used for determining the potential postures of a manipulator when it is in any configuration in a region. The region in $\mathcal{C}$ filled with grey is being mapped into its corresponding manipulator postures. Also depicted is the c-space representation of the artificial obstacle obtained by interpolating in polar coordinates between the actual obstacles. The corresponding workspace is also depicted. The bold, filled circles represent the actual obstacles. The bold line between them depicts the artificial obstacle. The normal-weight curves represent the shadows of some of the points along the artificial obstacle.

drawn with a solid line in Fig. 6 illustrates the path followed by the end-effector as the manipulator moves under such constraints. The equation describing this curve may be obtained in polar coordinates as:

$$\Theta(r) = \cos^{-1}\left[\frac{\rho^2 + (L_2 - r)^2 - L_1^2}{2\rho(L_2 - r)}\right] \tag{19}$$

where Θ is measured with respect to the line passing between the origin of the manipulator's base coordinate system and the point obstacle, and r is measured with respect to the obstacle. Note that the relationship between this curve and the robot's configuration as it tracks this curve is given by $\Theta = \theta_1 + \theta_2 - \phi$.

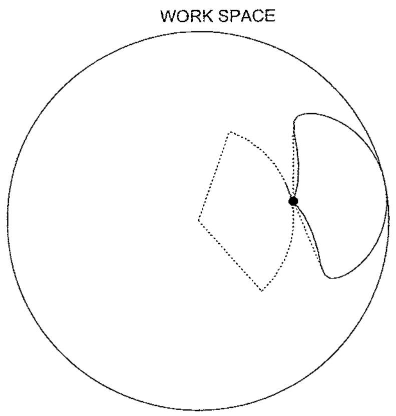


Figure 6. Generating the shadow of a bounding obstacle. The solid line in the depiction of the workspace represents the path followed by the end-effector as it slides along the point obstacle. The dashed line illustrates the portions of $\mathcal{W}$ that must be added to this area to obtain the shadow of the obstacle.

The set of points swept out by the entire link as it slides along $\mathcal{B}_i$ may be determined by expanding the area enclosed by (19) to include all those points that may be reached by projecting a line segment of length L_2 from each of the points along (19) through $\mathcal{B}_i$. The line segments that form the boundaries of this region are quickly obtained by recognizing that they must represent the positions of the second link when it is tangent to the curve of (19) and its end point is at an extrema in Θ. Furthermore, the robot configurations that place the second link in such a position are given by noting that these extrema in Θ occur when $\theta_2 = \pm\frac{\pi}{2}$.

Finally, the construction is completed by considering those points that come into contact with the first link. This is accomplished by considering the sector of radius L_1 that subtends the angle formed by the end points of the two line segments computed in the previous step and the base of the manipulator. The area that results from this construction is illustrated with dashed lines in Fig. 6. For the sake of notation, the portion of this region that lies at a radius greater than ρ_i will be referred to as the *outer shadow* of $\mathcal{B}_i$. The remainder of this region will be denoted the *inner shadow* of obstacle $\mathcal{B}_i$.

An approximation for the area in $\mathcal{W}$ in which the manipulator may lay when it is in any configuration for an entire region is illustrated in Fig. 7. This area is constructed by first establishing the shadows of each of the two obstacles forming the boundaries of the region. These shadows are then enclosed by a pie-shaped wedge, which uses the obstacle as its vertex and is of sufficient radius to enclose the shadow. Then these two approximations are connected by an arc centred at the base of the manipulator and of sufficient radius to enclose the two outer shadows. The angular extent of this arc is determined by the points along the shadows that are at maximum distance from the base of the manipulator. This approximation is then completed by including the inner shadows of the two obstacles along with the area that lies between them.

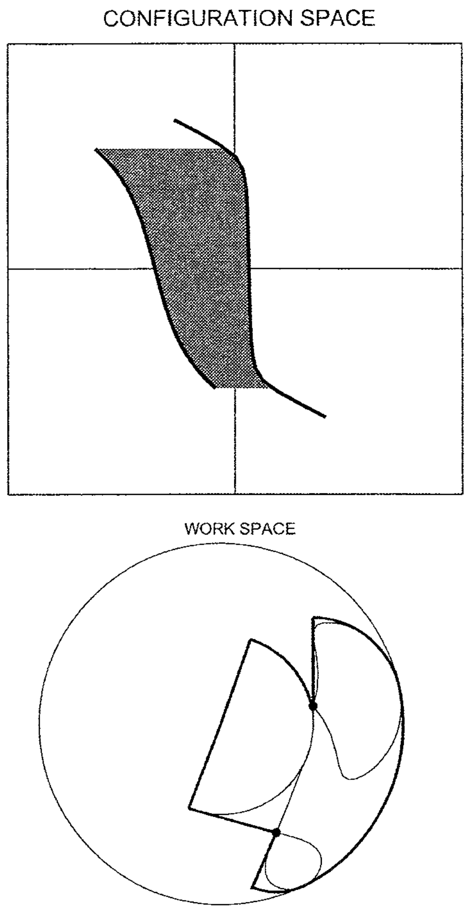


Figure 7. The bold lines in the workspace approximate the set of all possible positions for the manipulator when it is in any configuration in the c-space region illustrated in grey. The figure also depicts the shadows of the two bounding obstacles.

At this point, a conservative approximation has been developed that represents the entire area in $\mathcal{W}$ in which a manipulator may be when it is at any configuration within the region. The utility of this information becomes evident when determining the effects of choosing any path through a region in one configuration space on the topology of the other manipulator's configuration space. Because the approximation that has been developed is composed entirely of line segments and arcs, considering this approximation to be an obstacle in the other robot's workspace permits the tests of Section 3 to be utilized in establishing the effects of one robot's motions on the topology of the other robot's free space without knowing *a priori* which particular path will be chosen.

Given these results, a simple planning algorithm can be readily described. The basis of the work being presented, as well as that of our earlier work [7], is that the planning process may be broken into a two-phase approach, during which the free space is first searched for a channel using qualitative information on its topology and then fitted with a specific path using local geometric information. The process of searching for the channels is simplified by limiting the search to those that use the highways as intermediate goals, not unlike the approach employed in [15]. Furthermore, the search is guided by using the heuristic that the channel be the one most likely to yield the shortest path. These channels are then tested for potential conflicts by using the results of the previous sections. If a conflict is found via the tests of the previous section, then a velocity planner is invoked to modify the rate at which the channel will be traversed. If the velocity planning does not yield a pair of conflict-free channels, then the planner continues by iterating through pairs of channels of increasing length until it finds a solution.

More specifically, the planner is initialized by evaluating the topology of each manipulator's free space, a process that involves computing the sets of isthmuses and to which highways, if any, the initial and goal configurations are connected. Having done this, the planner proceeds by choosing the pair of channels that have not yet been examined and which are most likely to yield the shortest paths. By using its extent in θ_2 as an approximation for the time required to pass through a region, the two channels are then temporally synchronized. Once this is accomplished, the results of Sections 3 and 4 are used to determine whether the traversal of a particular region will affect the connectivity of the other manipulator's configuration space in such a way as to indicate a potential collision. If so, the planner generates a new pair of channels and estimates the amount of time that would be required to traverse these new channels. If the time required to traverse the new pair exceeds the time to traverse the pair that has just failed, then the planner attempts to modify the manipulators' velocities along the older pair of channels in an attempt to avoid collisions, as in path-velocity decomposition. If the result of this attempt at velocity planning is a set of trajectories that require no more time than the estimated time to traverse the new set of channels, then the planner returns these trajectories as the result. Finally, if the planner iterates through all of the possible channels without finding a pair that does not result in a collision, then it returns with a failure.

5. Simulation Results and Examples

The path planner described above has been implemented in the *C* language on a SPARC-II work station containing a 28.5 MIPS RISC architecture that results in a relatively modest 4.2 Mflops performance. In this section, a pair of

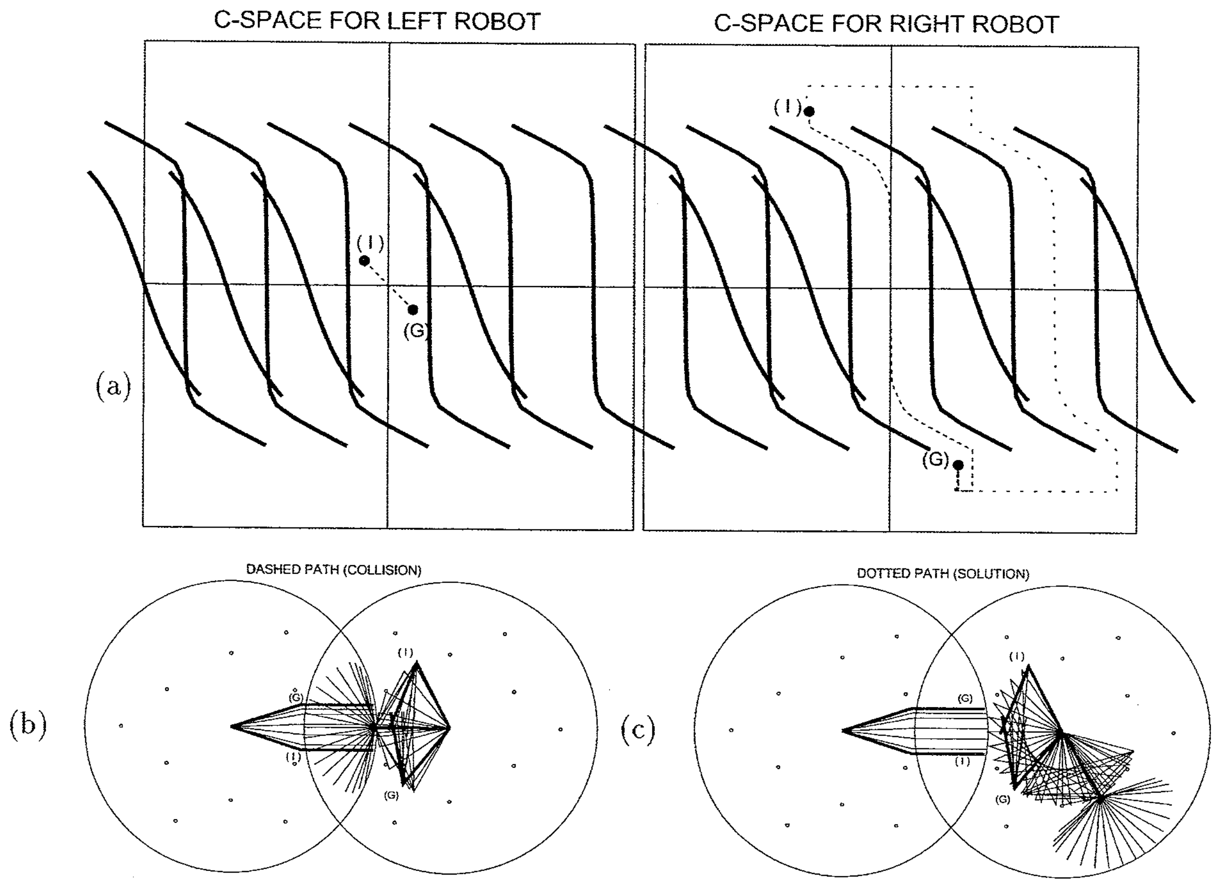


Figure 8. An example of a situation in which a path-velocity decomposition will not yield a solution although one exists. The configuration spaces in (a) depict the paths that represent the most straightforward solution for each robot independent of the other. The path for the manipulator to the right, depicted with a dashed line, yields a collision that cannot be avoided via velocity planning, as shown by the corresponding robot motions in (b). Once the planner determines that this path is not possible, it computes the alternative path shown in (c), which is collision free. This solution required approximately 130 ms, on a SPARC-II work station.

simple examples are discussed to provide insight into the operation of the planner. Following this, timing statistics for randomly generated examples are shown so that the behaviour of the planner may be assessed under a wide variety of problems. The section concludes with a discussion of these results.

Consider the problem of planning a path between the configurations labeled (I) and (G) in the contrived example illustrated in Fig. 8(a). In this example, the planner first attempts to plan a path that leads the robot on the left along the path illustrated, while having the robot on the right traverse the isthmus closest to the initial configuration. Both solutions would be reasonable if considered individually, as they would be close to being the shortest paths in $\mathcal{C}$ for each manipulator. If, however, the robots were to traverse these channels they would sweep through the regions in $\mathcal{W}$ illustrated in Fig. 8(b), and it is clear that a collision would occur regardless of the velocity profiles chosen along the paths. Hence, if a path-velocity decomposition were employed in this situation, it is not unreasonable to believe that it would not be able to find a solution. However, by utilizing the algorithms described above, the planner is capable of quickly recognizing that the two proposed paths are unacceptable. It then chooses an alternate channel for one of the robots and tests this pair of channels for collisions. The resulting motions of the manipulators are shown in Fig. 8(c). Computing this solution required approximately 130 ms of CPU time. In

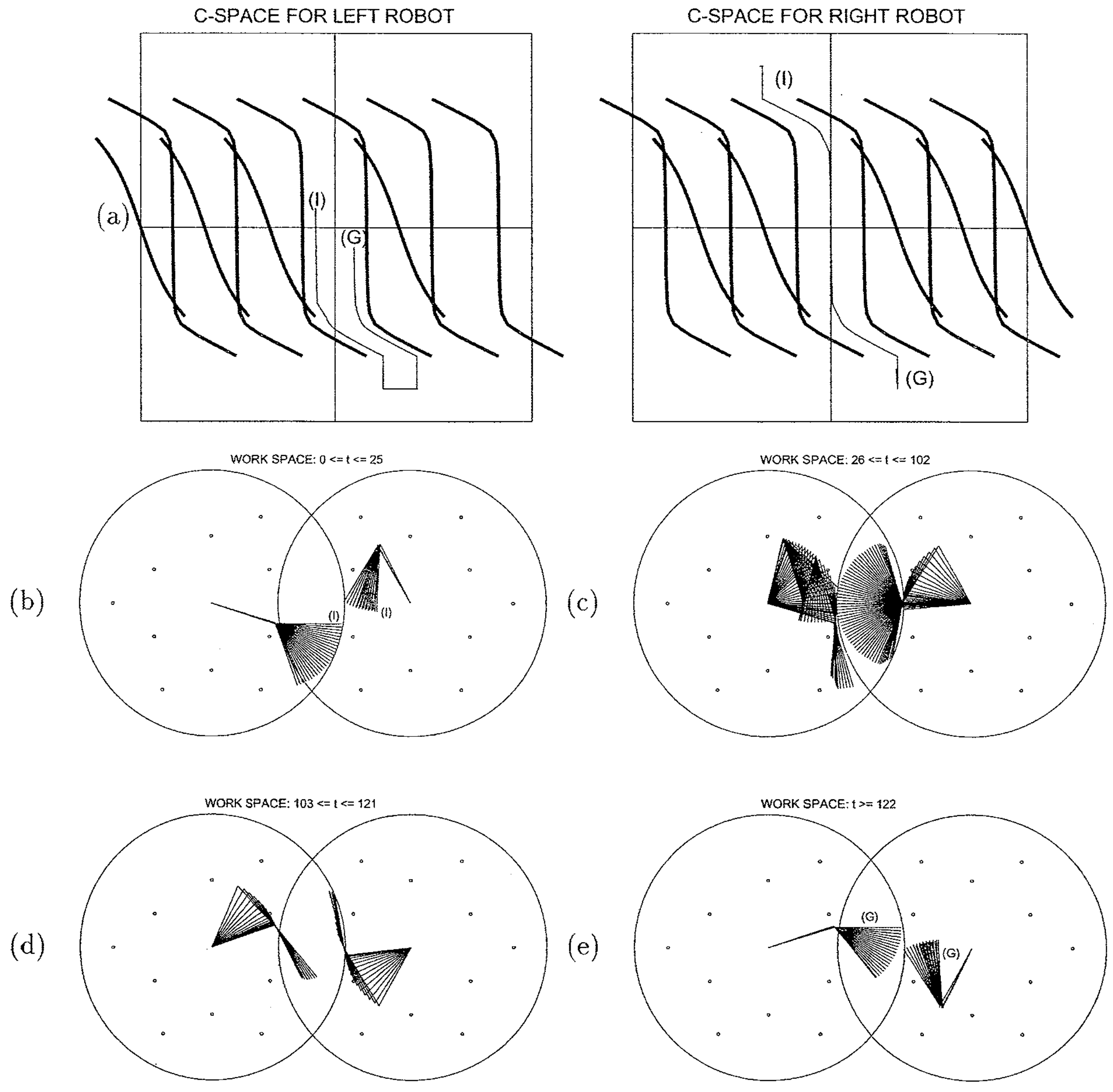


Figure 9. An example of a solution that requires the application of velocity planning. This workspace is identical to that of Fig. 8 except that the isthmus that was required for avoiding collisions has been closed by introducing an additional obstacle. This solution required approximately 220 ms. on a SPARC-II work station. The configuration spaces in (a) illustrate the paths taken by each manipulator. Note that the left manipulator must now move to a highway to let the right manipulator pass. Figures (b)–(e) illustrate the motions of the manipulators as they traverse these paths.

this case, the new channels do not bring the manipulators into potentially dangerous situations and, as a result, it is not necessary to perform velocity planning, although such planning is possible.

Fig. 9 illustrates a solution that follows a similar line of reasoning in attempting to find a solution. In this case, however, the velocity-planning capabilities of the algorithm have been utilized in building a solution trajectory. This particular example was generated by adding an obstacle to the world used in the previous example so that the isthmus used in the solution path for the right robot is no longer available. Determining a solution in this case initially follows the same steps as those described in the previous paragraph. Here, however, there is no alternative available to the robot on the right, so the planner attempts to find a solution by choosing the shortest alternate channel available to the manipulator on the left. In this case, this channel goes from the initial configuration through the lower highway and then back up the same isthmus to the goal configuration. The planner then applies the algorithms previously described for "meshing" the channels and discovers that, once again, there is a conflict. This causes the invocation of the velocity-planning module, which recognizes that by inserting delays along the path, all potential collisions may be avoided. The resulting solution required approximately 290 ms. of computation time and is illustrated in Fig. 9.

To fully evaluate the performance of the algorithm, it has been tested by computing paths for sets of randomly generated environments. The execution times required for computing paths in these environments were collected by the UNIX execution profiling utility *gprof*, and are summarized in Table 1. In each case, the positions of the obstacles were determined by sampling a uniform distribution within the reachable Cartesian workspace until the predetermined number of obstacles per robot was achieved. The initial and final configurations were chosen by sampling uniform distributions over the manipulators' configuration spaces. Of these sets of configurations, 21% resulted in one of the manipulators either beginning or ending in the area corresponding to the intersections of the two workspaces. Those trials that could not result in a solution because the initial and final configurations were in disjoint subspaces of C were discarded, as they yielded substantially lower execution times. On the other hand, six trials that did not yield a solution, despite trying all combinations of channels along with velocity planning, were included because they were representative of the worst-case scenarios. Each entry in the table represents data collected over 50 trials.

Table 1
Timing Data for the Path-Planning Algorithm as a Function of the Number of Obstacles per Robot

Obs/ Robot	Init. Avg.	Plan Avg.	Plan Min	Plan Max	Plan Std. Dev.	# Highway
10	47	215	44	1042	192	1.240
20	102	262	24	2550	365	1.180
30	182	203	22	540	133	1.064
40	290	314	42	3342	466	1.340
50	417	242	48	1830	308	1.043

Note. All times are in milliseconds on a SPARC-II.

The entries of the table provide information on the average time required for initial preprocessing of the environment; the average time required for the actual planning; and the minimum, maximum, and standard deviation of the data accumulated to compute the planning time. Furthermore, the table attempts to provide some insight into the relative difficulty of the various environments by providing the average number of highway traversals required by the resulting path.

The most easily explained of the results indicated in this table appears in the column reflecting the average time to perform the initialization step, during which the data structures that will be used for the search are built. During this stage, each of the obstacles undergoes a constant amount of processing required to evaluate those points needed for using the intersection tests described earlier. This is then followed by the pairwise testing of each of the obstacles for possible intersections. As the computations required to characterize the obstacles consist mainly of the application of the inverse kinematic equations (2) and (3), whereas the actual process of performing the tests involves an interval intersection and an interval membership test, this entire stage has complexity $O(c_1 n + c_2 n(n-1))$, where $c_1 \gg c_2$ and n is the number of obstacles. This is reflected in the timing data of Table 1, in which the average time required to perform the initialization step is dominated by a linear component but also exhibits a comparatively small n^2 component.

Next, consider the statistics describing the amount of time required to perform the actual planning. In general, the overall trend of the data seems to indicate that the most difficult environment for this planner is one that consists of 40 obstacles. The increase in planning time for environments with fewer than 40 obstacles seems to correspond to the fact that the number of isthmuses that may need to be explored tends to rise as the number of obstacles increases. The drop-off beyond this point seems to support the notion that as the number of obstacles increases, pairs of channels will be less likely to have some sort of conflict.

Cases that require the minimum amount of time are those in which no search is required and in which the first attempt at planning produces a collision-free solution. Of these minimum-time solutions, the variation between solution times of approximately 20 ms and 40 ms reflects to some degree the complexity of the resulting solution. In those cases that required the shorter times, the initial and final configurations were situated so that a simple straight line could serve as a valid solution path. In the latter case, the solutions required that at least one of the robots traverse a channel of the form peninsula-highway-peninsula. The resulting variations in timing are then a reflection of the time required to perform local path planning along a longer channel. Conversely, the cases that require the maximum amount of time are generally those in which the planner must exhaustively pair up each of the potential channels, recognize that there is a potential conflict, and then perform velocity planning on each of the pairs of channels.

To provide the reader with a better feel for the distribution of these results, we provide a histogram of the times required to perform the planning for those cases that constituted the 20 obstacles per robot test set (Fig. 10). For clarity, the plot has been truncated at 1.25 seconds, and thus does not show the outlier at 2.55 seconds. The data shown do not include the time needed for initializing the planner, which is more or less constant. This plot exhibits the characteristics that seem to be most typical of the distributions achieved in performing these tests, namely, there appear to be two relatively large clusters of results centred at approximately 50 ms and 225 ms and then additional smaller clusters at quantized intervals. Those cases that required times that fell into the cluster at 50 ms were ones in which the solution took the form of either a direct line or a peninsula-highway-peninsula and did not require any velocity planning. As additional reasoning was required, whether to perform velocity planning or to test a path that included an isthmus traversal, the time required for planning seemed to grow by fairly regular intervals of approximately 150 ms.

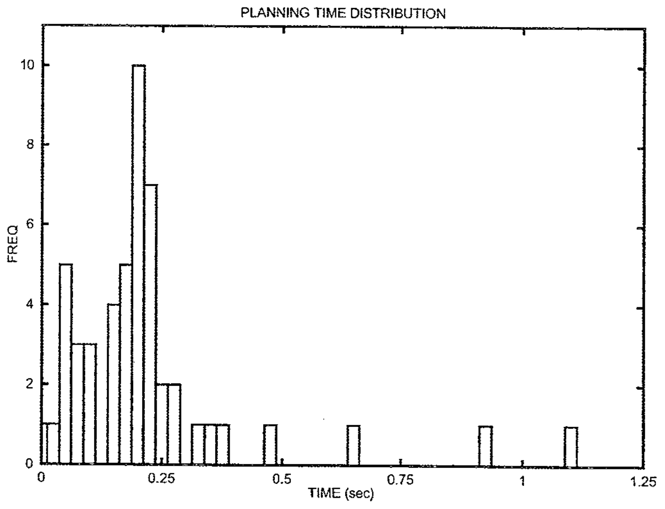


Figure 10. The distribution of the time in seconds on a SPARC-II required to perform the planning on those cases that constituted the 20 obstacles per robot test set. The time axis has been truncated for clarity and does not show the data point at 2.55 seconds.

An example of a solution returned by the planner on one of these randomly generated workspaces is provided in Fig. 11. In this example, each of the workspaces contains 50 randomly generated point obstacles. Of particular interest are those portions of the path depicted in (b) and (e), which illustrate the delays introduced into the path of the manipulator on the left. Computing this solution required approximately 750 ms of planning following an initialization of 390 ms.

It should be noted that in running these simulations, the velocity planner was limited in the amount of time it could add to any particular path. Specifically, once the two paths were chosen, velocity planning could take place only on the shorter of the two paths, and the total amount of time added to this path could not exceed the difference in time between the two original paths. This mechanism for choosing allowable delays was used to avoid situations in which the planner chose a path in which one of the robots folded in on itself, waited for the other manipulator to traverse its entire path, and then proceeded on to its goal. Although allowing these types of trajectories would represent a practical solution to cases in which the total time to traverse the path was not an issue or in which more stringent requirements precluded a planner from finding a solution, they were not included in this study. However, if the constraint on the delay times is lifted, then those cases in which the planner failed to find a solution may be solved.

6. Conclusion

This article has described an approach to the problem of planning collision-free motions for multiple SCARA manipulators operating within overlapping workspaces. The primary results have been the development of two fundamental concepts:

1. the ability to quickly establish the presence of certain topological features in C_{free}, and
2. the ability to quickly compute an approximation of the effects of one robot's motion on the topology of the other robot without *a priori* knowledge of the particular path that will be chosen through the region.

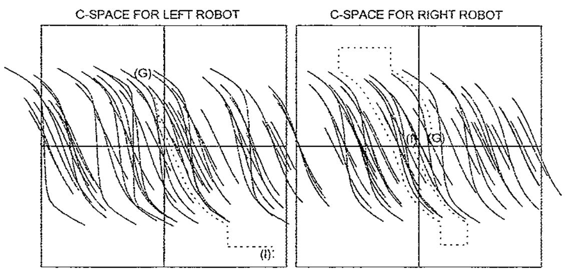
These concepts have been illustrated by the development of a simple planning system for multiple SCARA manipulators that finds solutions that form a superset of those found through a straightforward implementation of path-velocity decomposition. Furthermore, the low computational costs of generating candidate paths and testing them for interactions tends to offset the combinatoric nature of the search process, yielding a relatively quick algorithm. The major drawbacks of the presented planner are that it is not complete and some of the paths that result may be considerably less than optimal.

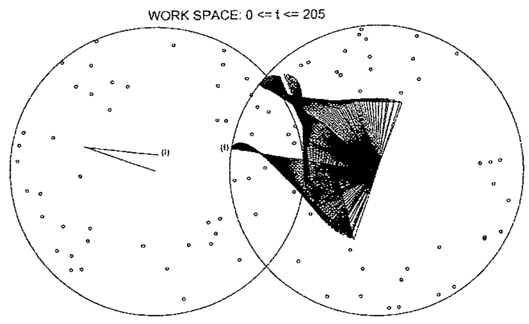
Acknowledgements

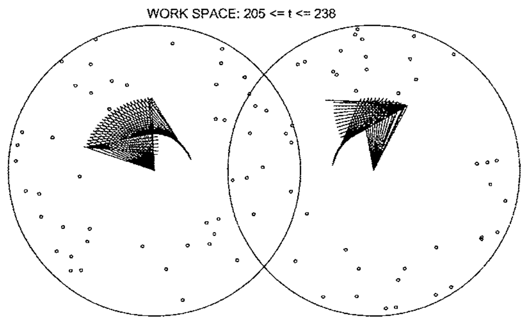
This work was supported by the National Science Foundation under grant CDR 8803017 to the Engineering Research Center for Intelligent Manufacturing Systems and by the NEC Corporation. An earlier version of this paper was presented at IROS '94, Munich, Germany, Sept. 12–16, 1994.

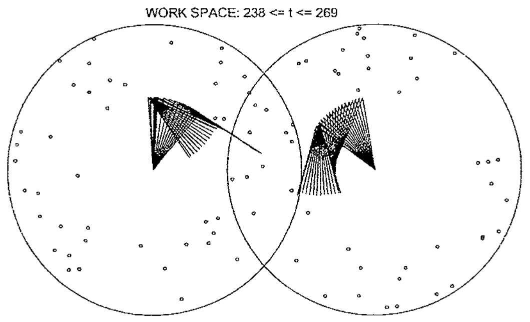
References

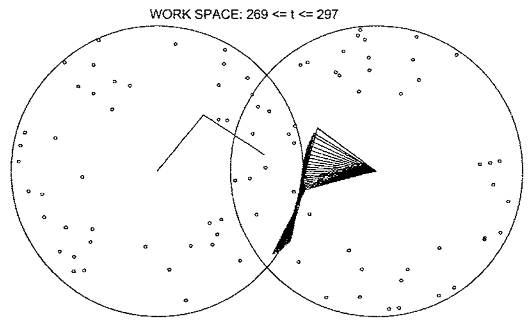
[1] J.C. Latombe, Motion planning in the presence of moving obstacles, in O. Khatib, J.J. Craig, & T. Lozano-Pérez (Eds.), *Robotics Review 2* (Boston, MA: MIT Press, 1992).

(a)


(b)


(c)


(d)


(e)


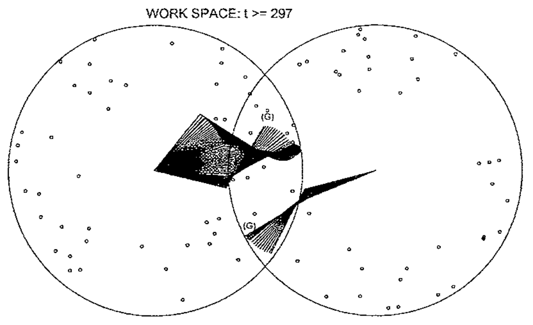
(f)


Figure 11. An example of a randomly generated workspace with 50 obstacles per manipulator and with a solution that requires the application of velocity planning. The configuration spaces in (a) illustrate the paths taken by each manipulator. (b)-(f) illustrate the motions of the manipulators as they traverse these paths. Of particular interest are those portions of the path depicted in (b) and (e), which illustrate the delays introduced into the path of the manipulator on the left. Computing this solution required approximately 750 ms of planning following an initialization of 390 ms.

[2] C. Shih, J. Sadler, & W. Gruver, Collision avoidance for two SCARA robots, *Proc. 1991 IEEE Int. Conf. Robotics Automat.*, Sacramento, CA, April 9–11, 1991, 674–679.

[3] S. Fortune, G. Wilfong, & C. Yap, Coordinated motion of two robot arms, *Proc. 1986 IEEE Int. Conf. Robotics Automat.*, San Francisco, CA, April 7–10, 1986, 1216–1223.

[4] K. Fujimura & H. Samet, A hierarchical strategy for path planning amongst moving obstacles, *IEEE Trans. Robotics Automat.*, *5*(1), 61–70, 1989.

[5] M. Erdmann & T. Lozano-Pérez, On multiple moving objects, *Algorithmica*, 2, 477–521, 1987.

[6] K. Kant & S.W. Zucker, Toward efficient trajectory planning: The path-velocity decomposition, *Int. J. Robotics Res.*, *5*(3), 72–89, 1986.

[7] A.A. Maciejewski & J.J. Fox, Path planning and the topology of configuration space, *IEEE Trans. Robotics Automat.*, *9*(4), 444–456, 1993.

[8] T. Lozano-Pérez, Automatic planning of manipulator transfer movements, *IEEE Trans. on Syst., Man, and Cybern., SMC-11*(10), 681–698, 1981.

[9] J. Latombe, *Robot motion planning* (Boston, MA: Kluwer, 1991).

[10] Y. Hwang & N. Ahuja, Gross motion planning—A survey, *ACM Comp. Surv.*, *24*(3), 219–292, 1992.

[11] K.S. Fu, R.C. Gonzalez, & C.S.G. Lee, *Robotics: Control, sensing, vision, and intelligence* (New York: McGraw-Hill, 1987).

[12] C. Bajaj & M.S. Kim, Generation of configuration space obstacles: The case of a moving sphere, *IEEE J. Robotics Automat.*, *4*(1), 94–99, 1988.

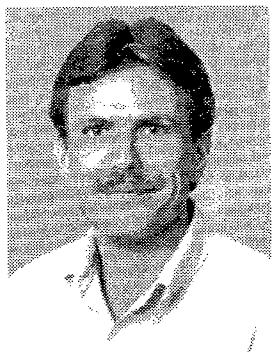
[13] R.C. Brost, Computing metric and topological properties of configuration-space obstacles, *Proc. 1989 IEEE Int. Conf. Robotics Automat.*, Scottsdale, AZ, May 14–18, 1989, 170–176.

[14] P. Wenger & P. Chedmail, On the connectivity of manipulator free workspace, *J. of Robotic Systems*, *8*(6), 767–799, 1991.

[15] W. Meyer & P. Benedict, Path planning and the geometry of joint space obstacles, *Proc. 1988 IEEE Int. Conf. Robotics Automat.*, Philadelphia, PA, April 24–29, 1988, 215–219.

[16] V. Lumelsky, Effect of kinematics on motion planning for planar robot arms moving amidst unknown obstacles, *IEEE J. Robotics Automat.*, *RA-3*(3), 207–223, 1987.

Biographies



Anthony A. Maciejewski received his B.S.E.E., M.S., and Ph.D. degrees in electrical engineering from Ohio State University, Columbus, OH, USA, in 1982, 1984, and 1987, respectively. In 1985–86 he was an American Electronics Association Japan Research Fellow at the Hitachi Central Research Laboratory in Tokyo, Japan. He is currently a professor of electrical and computer engineering at Purdue University, West Lafayette, IN. His primary research interests centre on the simulation and control of kinematically redundant and failure-tolerant robotic systems.

John J. Fox received his B.Sc. degree in electrical engineering from Carnegie-Mellon University, Pittsburgh, PA, in 1986, an M.Sc. from Pennsylvania State University, University Park, PA, in 1988, and his Ph.D. from Purdue University, West Lafayette, IN, in 1994. He is currently employed at Alphatech, Inc., in Burlington, MA.