Algorithmica (1987) 2: 477–521

Algorithmica

On Multiple Moving Objects[1]

Michael Erdmann[2] and Tomás Lozano-Pérez[2]

Abstract. This paper explores the motion-planning problem for multiple moving objects. The approach taken consists of assigning priorities to the objects, then planning motions one object at a time. For each moving object, the planner constructs a configuration space-time that represents the time-varying constraints imposed on the moving object by the other moving and stationary objects. The planner represents this space-time approximately, using two-dimensional slices. The space-time is then searched for a collision-free path. The paper demonstrates this approach in two domains. One domain consists of translating planar objects; the other domain consists of two-link planar articulated arms.

Key Words. Robotics, Motion planning, Coordinated motion, Configuration space, Autonomous robots, Collision avoidance.

1. Introduction.

A planner solving complex manipulation problems should be able to synthesize motion strategies for multiple moving objects. The need for this capability is evident both in large assembly operations during which it is impractical to move only one part at a time, and in tasks whose solutions involve the cooperation of several robots.

1.1. Examples.

This paper considers the motion-planning problem for multiple moving objects. We have implemented planners for multiple moving objects in two domains. The first domain consists of translating planar objects, while the second domain consists of two-link planar articulated arms. A detailed discussion of these domains will be given later.

The approach taken consists of assigning priorities to the objects, then planning object motions one object at a time. The planner assumes that the prioritization is given, then plans object motions in the order determined by the prioritization. A given object's motion is planned taking into account the motions of all objects whose motions have already been planned, as well as all stationary obstacles.

[1] This report describes research performed at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Michael Erdmann is supported in part by a fellowship from General Motors Research Laboratories. Tomás Lozano-Pérez is supported by an NSF Presidential Young Investigator grant. Support for the Laboratory's Artificial Intelligence research is provided in part by the System Development Foundation, in part by the Office of Naval Research under Office of Naval Research Contract N00014-81-K-0494, and in part by the Advanced Research Projects Agency under Office of Naval Research Contracts N00014-80-C-0505 and N00014-82-K-0344.
[2] Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139, USA.

Received February, 1986; revised October 22, 1986, Communicated by Chee-Keng Yap.

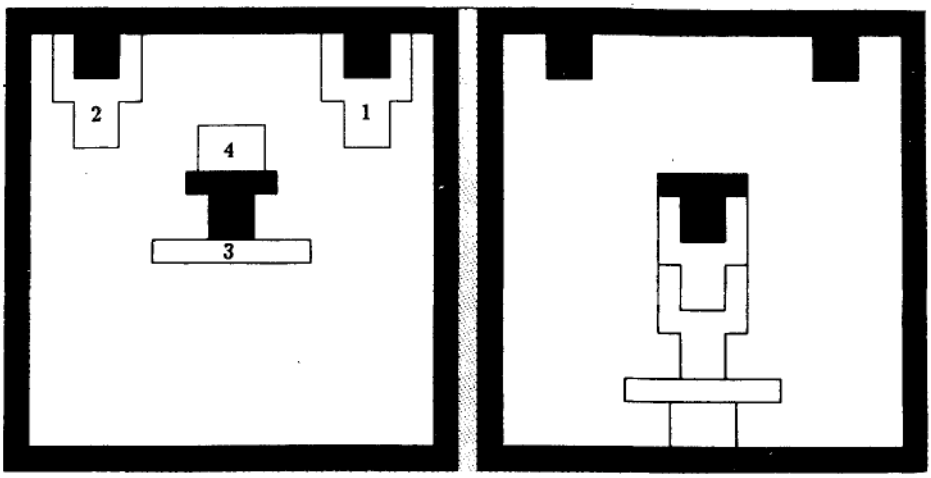

Fig. 1. The start and goal configurations of four translating planar objects. The numbers attached to the objects indicate the order in which object motions were planned.

The time-varying constraints imposed on a given object by other moving objects are represented in a configuration space-time. Details will be given later (see, for instance, Sections 3 and 4).

As an example, consider the objects of Figure 1. The figure displays both the start and goal positions of the objects. In this example, the objects are permitted to translate but not to rotate. The numbers attached to the objects indicate the order in which object motions were planned. Figure 2 displays a solution determined by the planner to be discussed in this paper. Running on a Lisp Machine, the time required by the planner to solve this problem was slightly under 7 minutes.

As another example, Figure 3 shows the start and goal configurations for three articulated arms. Again, the numbers attached to the arms indicate the order in which object motions were planned. A solution for this problem is shown in Figure 4. The planner required approximately 36 minutes to solve this problem.

In both examples the planner generated a series of collision-free motions taking the objects from their start configurations to their desired goal configurations. The objects generally move simultaneously, although the planner will also consider stopping an object to wait for other objects to pass, if doing so is advantageous.

1.2. Problem Statement. This paper concentrates on the motion-planning problem. There are, however, other important issues that a task planner should understand. In particular, the dynamics of object interactions, the effect of uncertainty on object motions, and the design of environments conducive to particular tasks, are problems that deserve attention. These issues are beyond the scope of this paper.

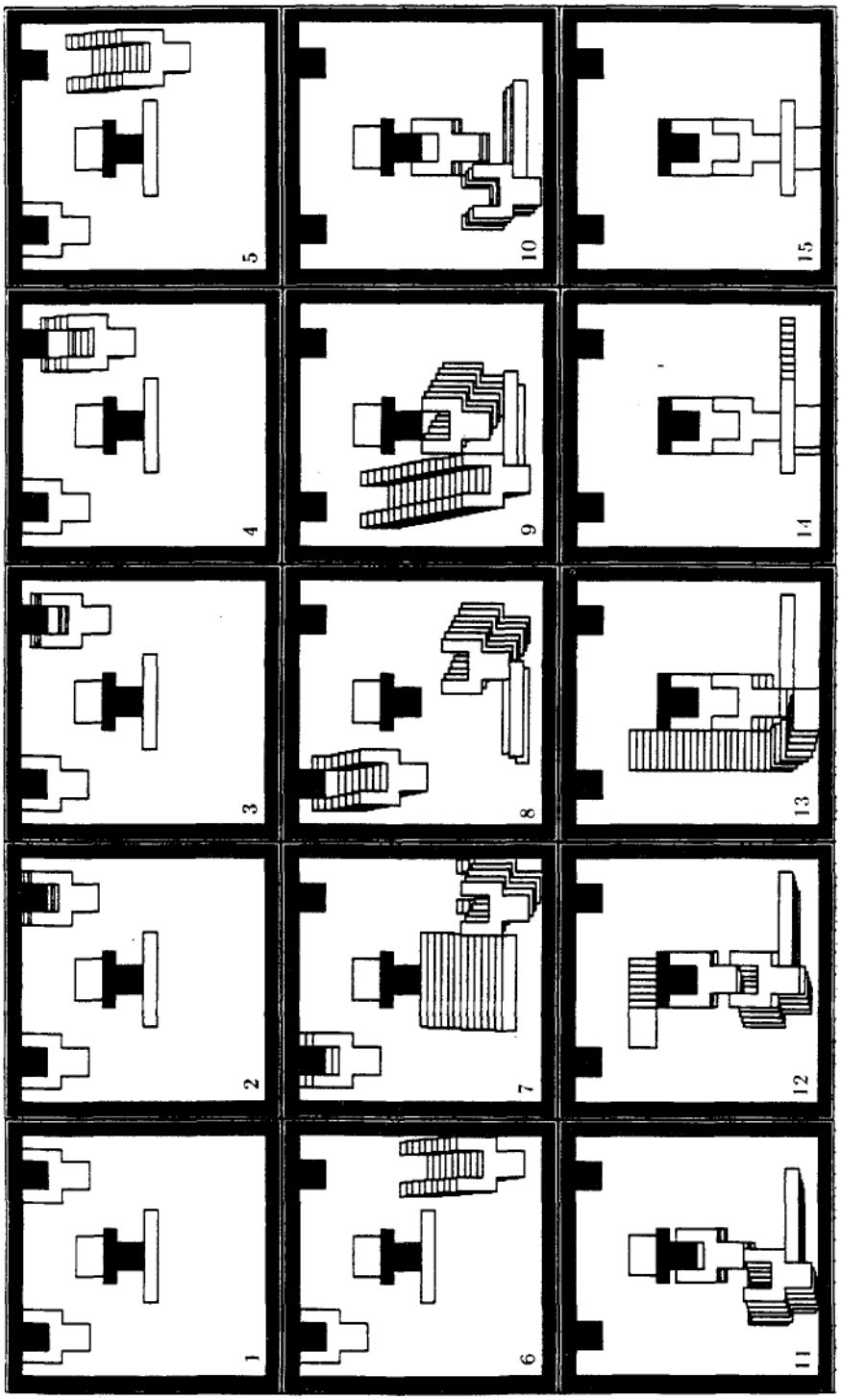

Fig. 2. This figure traces a solution determined by our planner for the moving objects of Figure 1.

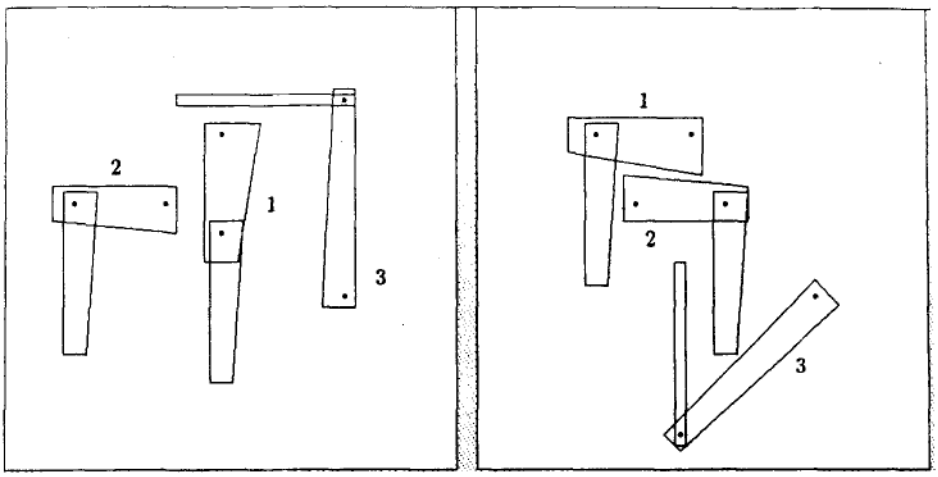

Fig. 3. Start and goal configurations for three articulated arms. The numbers attached to the arms indicate the order in which object motions were planned.

The assumptions of this paper are:

1. The environment consists of a set of stationary objects and a set of moving objects, modeled as polyhedra.
2. All objects perform rigid motions. In the case of articulated arms, this means that each link is rigid, although the links may rotate relative to each other.
3. Object interactions may be specified in geometrical terms. The physics of object interactions are not considered.
4. Planned motions should be correct to some resolution. Some of our implemented planners have resolution bounds.

The envisioned planner expects a specification of the moving objects' desired configurations at particular times. In the simplest case, this specification consists of a set of initial states and a set of goal states. Our implemented planner is of this form. In more complicated settings, a sequence of desired states could be specified. For instance, the sequence could be cyclical, representing the steps in some repetitive task. The planning process consists of determining a series of motions that satisfies the goal specifications, while avoiding object collisions. (By a collision we mean an overlap of object interiors; nonoverlapping touching contacts are permitted.)

1.3. Motivating Topics. The motivation for studying multiple moving objects stems from the complexity of manipulation tasks. The total degrees of freedom of all the objects in a task may be high. Traditional methods of robot planning are applicable primarily for a single moving object. (See, for example, Udupa [24], Lozano-Pérez and Wesley [15], Lozano-Pérez [13], [14], Brooks [3], and

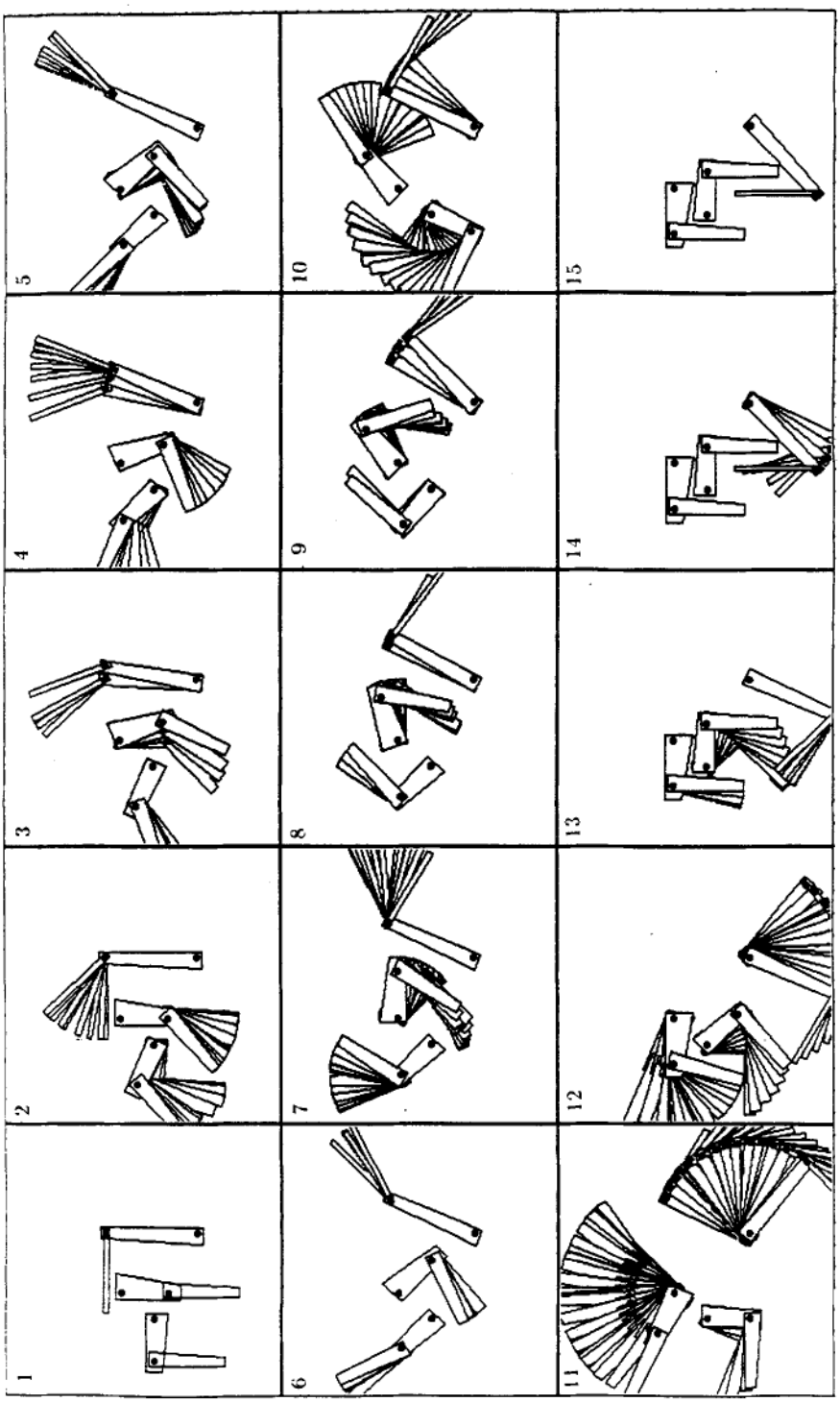

Fig. 4. This figure traces a solution determined by our planner for the three articulated arms of Figure 3.

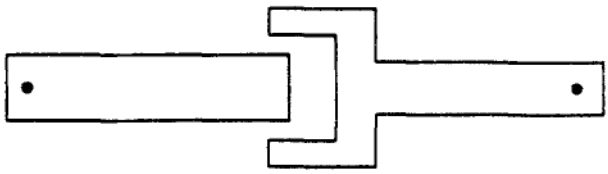

Fig. 5. Two interlocked links. In order to separate the two links, it is necessary to rotate both links simultaneously.

Brooks and Lozano-Pérez [4].) Thus all but one of the objects in a task must be held fixed during each major step of the task. This is seldom an efficient approach.

Furthermore, the approach of only moving one object at a time may actually limit the class of solvable tasks. For instance, consider the interlocked rotating links shown in Figure 5. In order to separate the two links, both links must be rotated simultaneously. The two links cannot be separated by moving only one link, since either link constrains the other link from moving.

Stepping outside the immediate problem boundaries considered by this paper, imagine an automated factory. The factory consists of numerous autonomous robots, all moving and interacting. Some of these robots are fixed, performing tasks at particular locations, others are feeders, serving to move objects through a gauntlet of operations, and still others are mobile, moving as independent vehicles throughout vast stretches of the factory. The aim here is to solve such motion tasks efficiently and safely.

2. Previous Work. We are aware of several major lines of previous work on multiple moving objects.

Campbell and Luh [5] construct a numerical optimization problem to find an optimal path of a manipulator between a sequence of edges in space. The positions of the edges may be time-varying. Thus, given a collection of moving objects, this approach may be used to compute the trajectory of an additional moving object.

A limitation of this algorithm is that it determines a shortest path between the specified sequence of edges without ensuring that the resulting path is collision-free. Effectively, Campbell and Luh's algorithm is used as a subroutine inside a shortest-path search of configuration space. In searching the space, some other mechanism must ensure that proposed paths between edges are collision-free. This may be done easily in an environment consisting solely of static obstacles. Unfortunately in an environment consisting of moving obstacles, a separate mechanism for proposing edge sequences must be developed.

A number of local techniques have been developed for computing collision-free motions of several moving objects. These techniques maintain collision-free trajectories over small periods of time. They are used in an on-line fashion to obtain collision-free trajectories between start and goal configurations over large time intervals. In particular, Freund and Hoyer [9] have devised an on-line control system for coordinating motions in a multirobot system. Along a different

direction, Tournassoud [23] has used separating hyperplanes to avoid locally collisions of multiple moving objects.

Kant and Zucker [12] decompose the multiple moving objects problem into two subproblems. The first subproblem consists of planning a path for each of the moving objects that avoids collisions with the static objects in the environment. The second subproblem consists of varying the velocities of the moving objects along their specified trajectories so as to avoid collisions with moving obstacles. Effectively, the algorithm plans the velocities of the moving objects one object at a time. Given that the velocities for some set of moving objects are known, the algorithm computes the velocities of another moving object so as to avoid collisions with any of the moving objects whose velocities have already been determined. The velocities may be so chosen as to ensure minimum traversal times.

The advantage of this approach lies in its explicit representation of time. In fact, the subproblem of determining velocities is formulated elegantly as a path-planning problem in a two-dimensional space-time. The algorithm guarantees that the resulting motions avoid any collisions of moving objects with stationary objects or other moving objects. The limitation of this approach is that it does not permit path alterations, only velocity alterations. This may prevent solutions in cases where some moving object forever remains in the path of some other moving object.

A number of researchers have focused on the special case of coordinating the motions of several circular bodies in two-dimensional regions bounded by collections of polygonal walls. (See the work by Schwartz and Sharir [20], Yap [26], and Ramanathan and Alagar [17].) These authors demonstrate various provably correct algorithms for solving the coordinated disk motion problem. The time complexities of these algorithms are shown to be polynomial in the number of walls and exponential in the number of disks.

Fortune *et al.* [8] use retraction and critical curve methods to solve the problem of coordinating the motions of two polar manipulators. These authors construct, for one of the manipulators, a graph that represents the manipulator's free space as a function of the second manipulator's configuration. They then partition the second manipulator's free space into regions and critical curves over which the topology of this graph is invariant. Thus they can represent the combined free space of the two manipulators as a series of graphs, which then can be searched for a collision-free coordinated motion.

Hopcroft *et al.* [10] have examined the complexity classification of the coordinated motion problem. In particular, they have shown that the two-dimensional problem of coordinating the motions of an arbitrary number of rectangles in a rectangular region is PSPACE-hard. Hopcroft and Wilfong [11] showed that the problem is in fact in PSPACE. Spirakis and Yap [22] considered the complexity of moving many disks, showing this problem to be NP-hard.

Reif and Sharir [18] have also considered the multiple moving objects problem. They showed that the problem of planning motions for a three-dimensional rigid body in an environment containing stationary and moving obstacles is PSPACE-hard given bounds on the moving object's velocity, and NP-hard without such velocity bounds. Furthermore, they exhibit a polynomial-time algorithm for the

two-dimensional version of the asteroid avoidance problem, assuming a bounded number of moving obstacles. They also exhibit a singly exponential-time algorithm for the three-dimensional version of the problem, assuming an unbounded number of moving obstacles.

3. General Problem Discussion. The structure of a multiple moving objects planner depends on the type of object interactions permitted. The relevant issues are centralization and cooperation.

3.1. Autonomous Planning. Perhaps the simplest case consists of planning for a single moving object in the presence of a number of other objects, some of which are stationary and some of which may be moving. The algorithms of Campbell and Luh [5] and Kant and Zucker [12] are formulated in these terms. Of course, given an algorithm for planning for a single moving object in the presence of other, possibly moving, objects, it is always possible to plan for a set of moving objects, by planning motions one object at a time. This approach, however, is not guaranteed to be complete.

The method of planning for a single moving object is also appropriate for an independent autonomous robot that is trying to navigate and work amidst a group of other independent entities. For small periods of time and in small neighborhoods about itself, the robot can observe the motions of other objects, predicting their immediate future behavior. The robot can then plan its motions using its algorithm for planning motions of a single moving object in the presence of other moving objects. Such an approach requires continuous real-time planning.

3.2. Centralized Planning. The previous discussion assumed decentralized control. Planning occurred in independent entities on the individual level. Another question to address is the centralized planning problem, in which the motions of several moving objects are being planned at once. For instance, the planning of complex assembly operations requires centralized planning for multiple objects and robots.

3.3. Configuration Space. Let us observe that the centralized multiple moving objects planning problem is conceptually no more difficult than the problem of planning motions for a single moving object in the presence of a collection of stationary objects. To see this, let us consider the motion-planning problem for a single moving object. One approach taken is to transform this problem into that of planning point motions in the object's configuration space. The configuration space [1], [13], [14], [19], [7], [6] of an object is the parameter space representing the degrees of freedom of the object. Obstacles in real space constitute constraints on the object's motion. These may be represented as hypersurfaces in the object's configuration space. The planning process consists of determining a path of a point in configuration space that does not violate any of these hypersurfaces. Figure 6 shows the configuration space of a translating triangle determined by two stationary obstacles.

Now observe that one can construct a configuration space for the multiple moving objects problem. The dimension of this space is the sum of the degrees

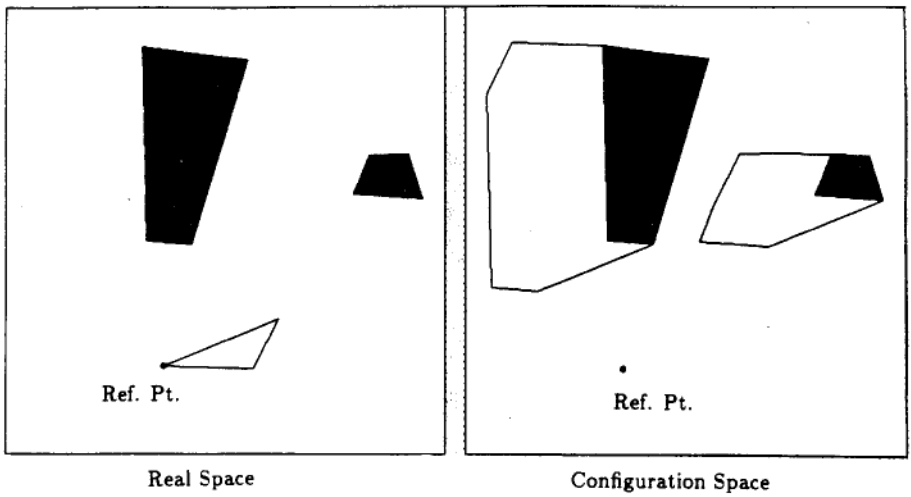

Fig. 6. The left figure consists of a translating triangle, and two stationary obstacles. The right figure displays the configuration space constraints imposed on the triangle by the obstacles. For comparison, the constraints are superimposed on the actual obstacles.

of freedom of all the moving objects. Again, constraints in real space may be represented as hypersurfaces in the configuration space. These surfaces correspond to configurations of the moving objects at which some moving object is touching another moving or stationary object. A point in this configuration space represents the configurations of all the moving objects, while a trajectory in the configuration space describes the motions of all the objects at once. The planning process, as for a single moving object, consists of determining a trajectory in configuration space that does not violate any of the hypersurfaces.

We see, therefore, that the approach for planning for a single moving object in the presence of purely stationary objects applies equally well to planning for multiple moving objects, except in higher dimensions. From this point of view, the multiple moving objects problem is solved conceptually. There are, however, some practical concerns. In particular, for tasks involving a large number of moving objects, the dimension of the configuration space may be very high. Thus, both the construction of the space and the search for a collision-free trajectory may be time consuming.

3.4. Prioritized Planning. We have already noted that methods for planning for a single moving object in the presence of other moving objects may be used for planning for several moving objects, by planning motions one object at a time. The appeal of this decomposition approach is that it reduces the problem from a single planning problem in a very high-dimensional space to a sequence of planning problems in low-dimensional spaces. Of course, in contrast to the configuration space approach, in general this decomposition approach need not be complete. By not considering all moving objects at once, the planner runs the risk of choosing a trajectory for an object early on that prevents finding a solution

for an object later in the planning sequence. In this section, we explore some cases that readily lend themselves to the decomposition approach.

In essence, any task in which a prioritization of motions may be assigned, may be approached using the decomposition approach. In the case of cooperating robots, a prioritization may be assigned in terms of master/slave relationships. In other words, one robot is actually performing the task, while the others are helping. Any robot helping another observes the other's motions, acting in a cooperating fashion.

In the case of assembly operations, in which many parts may be moving at once, the order in which parts fit together may determine a prioritization. For instance, if a robot is placing a part onto another part located on a conveyor belt, then the robot must cooperate with the conveyor. In turn, the motion of the conveyor belt and the motions of objects fed onto the conveyor are planned after observing the distribution of parts on the conveyor. If objects begin jamming up at some point, then the rate at which objects are fed onto the conveyor may have to be changed, or the conveyor may have to be stopped.

Notice that a prioritization does not imply that an object of lower priority must follow or assist an object of higher priority. Assistance is just one example in which priorities may be assigned naturally. In general, an object of low priority may be performing independent operations. A prioritization simply states that the burden of avoiding collisions between two objects falls on the object of lower priority. The high-priority object may move in any fashion that it desires. The low-priority object may also move in any fashion that it desires, so long as it does not collide with the high-priority object.

Finally, let us observe that the prioritization need not be constant. For instance, in the case of cooperating robots, the master/slave relationships may alternate during the course of performing a task. Furthermore, at times the particular prioritization chosen may be irrelevant. For instance, if during the course of a task two robots cannot possibly interfere with each other, then the order in which motions are planned does not matter.

These observations suggest that the prioritized decomposition scheme be used to plan subtasks inside of a larger planner that assigns priorities. The larger planner decomposes a task into smaller subtasks; it determines which objects could possibly interfere with each other; and it uses task constraints to decide on the order of motions within a subtask.

4. Outline of the Approach. We now outline a method for planning motions of several moving objects. It is assumed that the objects have been assigned priorities. Motions are planned one object at a time, according to the assigned priorities. Each object's motion is planned so as to avoid collisions with all stationary objects and all moving objects whose motions have already been determined. Situations in which this approach is suitable were discussed previously, in Section 3.

4.1. Incorporating Time. The constraints on a single moving object in an otherwise static environment are readily captured by the configuration space of the

object. Motions of the object are planned by planning motions of a representative point in the configuration space.

Now suppose that the environment is no longer static. In this case the constraints on the moving object whose motions are being planned vary with time. However, notice that it is still possible to construct a configuration space at any fixed point in time. The configuration space at a particular point in time geometrically captures the constraints on the object's degrees of freedom at that time. The configuration space is identical to a configuration space constructed from a static environment arranged as are all objects, both stationary and moving, at the given point in time. Considering all points in time, this construction produces a space-time configuration space that reflects the time-varying constraints on the object's possible motions. A particular slice of this space at any given time is just an ordinary configuration space. The constraints of stationary objects are constant as functions of time; the constraints arising from moving objects change as functions of time. Planning an object motion entails planning the motion of a point in the configuration space-time that does not violate any constraints.

Notice that it is indeed necessary to incorporate time in some fashion in order to accurately represent the time-varying constraints. This is not necessary if one is planning for all of the moving objects at once, for then it is possible to treat all the objects effectively as one composite abstract object, with a high number of degrees of freedom. By only planning motions for one object at a time, it is necessary to treat the environment as time-varying. This is because the planner must somehow consider the behavior of those objects whose motions have already been planned.

As a final comment, suppose that there are n objects, each with k degrees of freedom. The composite configuration space approach involves planning in a space of dimension nk. The prioritized decomposition scheme involves planning n motions in spaces of dimension $k+1$.

4.2. Issues. Configuration space-time correctly describes the problem of planning motions for a single object in a time-varying environment. It remains therefore to devise algorithms that efficiently consider the relevant portions of the configuration space-time while planning a motion. Some issues that arise while solving this problem include:

1. How to build the space-time configuration space.
2. How much of the space-time configuration space to build.
3. How to search the space for a collision-free trajectory.

We have explored these issues, and implemented algorithms, in two different domains. The first domain consists of polygonal objects in the plane. The objects are permitted to translate but not to rotate. The second domain consists of two-link planar arms with rotary joints. The arms are permitted to rotate at their joints, with their base points held fixed in the plane. The remainder of this paper is a description of our observations and results.

4.3. Searching in Time. Let us make a few general observations regarding searching in configuration space-time. First, unlike searches in general spaces,

it is always necessary to search forward in time. Objects are not permitted to move back in time and interact with themselves. In other words, configurations are assumed to be single-valued functions of time. As a simple extension, if object velocities have minimum or maximum bounds, then the angle of motion between slices is constrained to lie in some appropriate range. In other words, the space-time region reachable from some starting point is a cone whose edges are defined by the velocity bounds. The search algorithm must consider these restrictions when proposing paths.

As a side note, suppose that we placed no temporal restrictions on the search algorithm, but simply treated the space-time regions as one would treat purely spatial regions. The search algorithm might then propose a path from start to goal that would not be a single-valued function of time. Said differently, the graph of configuration versus time might contain vertical segments or it might curve back in time. In order to realize such a proposed path physically, the planner would at times have to slow down or even reverse the motions of the other objects whose motions were previously planned. Our implemented planners did not consider such paths.

Another observation concerns the safety of achieved goals. Suppose that a given moving object's goal is specified solely as a configuration, with no mention of time. Suppose that the planner achieves the goal at some time before the other moving objects have stopped moving. Then it is still possible for one of the other moving objects to collide with the given object. In order to avoid this, the planner must check that the achieved goal remains safe until all objects have stopped moving. A simple method for ensuring safety of attained goals is to specify goals as space-time configurations, where the spatial coordinate represents the actual goal and the temporal coordinate represents the earliest time at which the goal may be considered attained. Notice that no such check is explicitly necessary for cyclical tasks in which all objects continue to move after achieving their goals.

5. Translating Planar Objects. The first domain that we will explore consists of two-dimensional polygons. The environment is composed of both stationary objects and moving objects. The moving objects are allowed to translate but not to rotate. The objective is to plan a collision-free motion for a moving object in the presence of the stationary objects and those other moving objects whose motions have already been planned.

5.1. Constructing the Configuration Space-Time. Configuration space obstacles are shape-invariant under translations. Given some moving object and some stationary object, suppose we construct the resulting configuration space obstacle. Now suppose that we change the position of the stationary object. Then the shape of the resulting configuration space obstacle remains unchanged. Furthermore, the position of the configuration space obstacle translates exactly as does the real space obstacle. (See, for instance, Figure 7.)

This invariance greatly simplifies the computation of the configuration space-time. In order to compute the time-varying constraints imposed on a moving

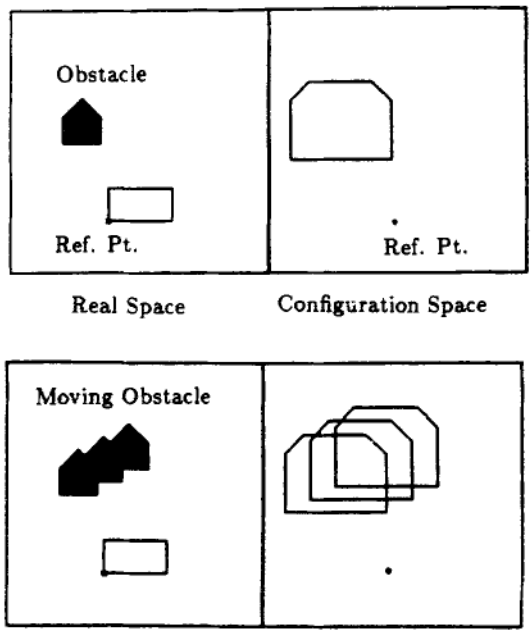

Fig. 7. The top two frames show a rectangle and a stationary obstacle, along with the constraints in the rectangle's configuration space that are determined by the obstacle. The bottom two frames show how the translation of the real space obstacle is reflected in configuration space by a translation of the configuration space obstacle.

object by some stationary objects and other moving objects, it is enough initially to treat the moving objects as stationary. Specifically, the planner computes a standard configuration space obstacle for each of the stationary and moving objects. The actual constraint imposed by a moving object at a particular time may then be determined simply by performing a polygonal translation of the associated configuration space obstacle. In short, for translation spaces, it is sufficient to compute the configuration space obstacles once. The time-varying constraints may be determined easily by performing translations of these configuration space obstacles.

5.2. Representing the Configuration Space-Time. Let us assume that all translations of moving objects in space are piecewise linear, with constant velocity over each segment. This assumption is reasonable in polyhedral environments. Then it is sufficient to represent the configuration space-time as a list of configuration space slices at particular points in time. The times are those at which some moving object whose motion has already been planned changes its velocity. This is because all object motions between such points in time are straight-line constant-velocity motions in space. In particular, the corresponding configuration space-time obstacles are simply swept volumes, determined by sweeping configuration space obstacles along straight lines through space-time.

Using this representation of configuration space-time, it is easy to decide whether a proposed path collides with any of the other stationary or moving objects. Specifically, the decision amounts to determining whether a moving point collides with a moving polygon. In turn, that computation may be reduced to deciding whether a stationary line segment intersects a stationary polygon (see Figure 8).

In summary, configuration space-time is represented as a list of configuration space slices at particular times. The times are those at which some moving object changes its velocity. A slice is computed from the slice at time zero by translating the configuration space obstacles that correspond to the moving obstacles in real space. Motions between slices are implicitly represented as straight-line translations of these configuration space obstacles.

5.3. Searching for a Collision-Free Path. Once the configuration space-time has

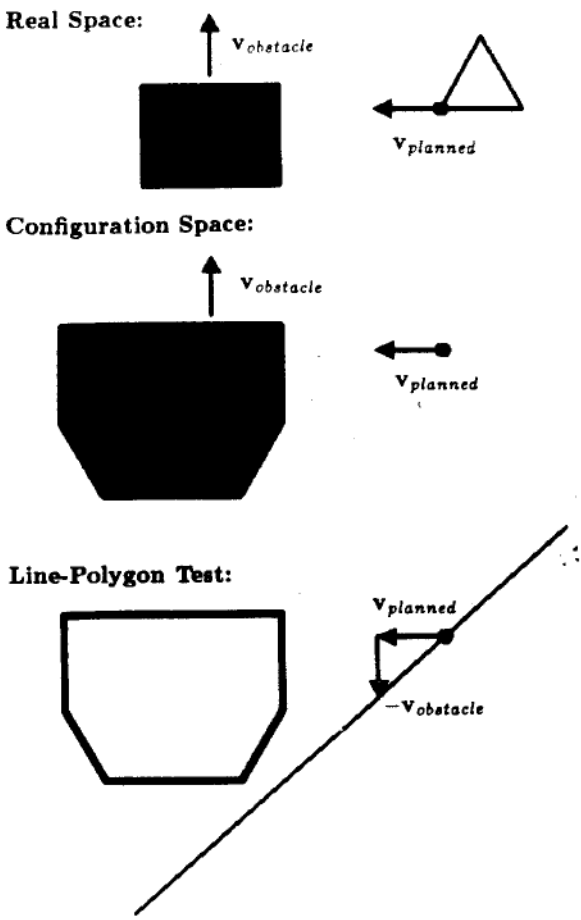

Fig. 8. The problem of deciding whether two moving objects collide is first transformed into the problem of deciding whether a moving point collides with a moving configuration space obstacle. This problem is then transformed into a line-polygon intersection test.

been constructed, a collision-free path may be determined by finding motions that do not intersect any of the constraints explicitly or implicitly represented in configuration space-time.

There are several methods for generating possible path segments between slices. For instance, one could consider all overlapping free space regions between slices. This approach has the advantage of not immediately choosing a particular trajectory, but instead implicitly representing all trajectories within a region.

The particular algorithm that we implemented considers all path segments between adjacent slices that terminate at vertices of obstacles. Any path segment that pierces a stationary configuration space obstacle or that intersects an implicitly represented moving obstacle is ignored. This algorithm is a variation of the V-graph algorithm used in Lozano-Pérez and Wesley [15] and Lozano-Pérez [14].

The difficulty with this approach is that the algorithm may not find a path because it generates too few path segments. The fundamental cause of this difficulty lies in the discrete representation of time. Time is only represented implicitly between slices. Thus there is no natural mechanism for performing motions over time intervals that are shorter than the interval between two adjacent slices. In order to alleviate this problem slightly, the planner does not use solely the slices arising from changes in object motions. Instead, the planner introduces a fixed number of extra configuration space slices between those that are already represented. This is equivalent to ignoring solutions that involve motions below some fixed time resolution. A complete algorithm would also need to consider the space-time between slices.

Finally, let us note that the search of configuration space-time may wish to take into account costs of particular paths. Both the distance traveled and the time to reach a goal configuration may be important. Standard search techniques apply.

5.4. Complexity and Completeness. For each slice, given a particular object whose motion is being planned, the configuration space constraints can be constructed in time that is linear in the number of edges in the environment. Thus, let $m = nr$, where n is the number of edges in the environment, and r is the number of slices constructed. Then the configuration space-time can be constructed in $O(m)$ time. Our planner uses a search that has time complexity $O(rn^3)$, although a faster version of the algorithm may be implemented, as we shall see. Note that the search must be able to decide whether a motion between two vertices in two adjacent slices lies in free space-time. This is done using the line-polygon intersection test described in Section 5.2. One such test is performed for each moving or stationary obstacle in the environment. *In toto*, the tests require time $O(n)$ per proposed motion. Thus, for a given vertex, the time required to find all vertices in adjacent slices that are reachable via collision-free straight-line motions through space-time is $O(n^2)$. The overall time for testing safety of all possible vertex-vertex transitions between adjacent slices is therefore $O(rn^3)$.

In fact, a slightly faster construction of the visibility graph is possible. To see this, consider a particular vertex w in a given slice. The objective is to find all

vertices in some adjacent slice that are reachable by a straight-line motion through space-time. Imagine constructing, for each configuration space obstacle, a certain visibility polygon. Specifically, this visibility polygon represents the set of points in the slice containing w that are visible from w along straight lines which do not intersect the obstacle. (Observe that a visibility polygon geometrically solves a line-polygon test for an entire set of lines.) Taken over all obstacles, the collection of these visibility polygons may be constructed using an algorithm of time complexity $O(n \log n)$. The total number of vertices in the polygons is $O(n)$. (See, for instance, Sharir and Schorr [21] and Asano *et al.* [2].) Now consider translating each visibility polygon into the adjacent slice. Each polygon should be translated with the same velocity as is used to translate its generating configuration space obstacle. A translated visibility polygon describes precisely all points in the adjacent slice that are reachable from w along straight lines which do not intersect the corresponding space-time obstacle. Consequently, the intersection of all the translated visibility polygons describes all points in the adjacent slice that are reachable from w along some straight-line motion that does not intersect any space-time obstacle. The vertices of this intersection polygon may be used to define the visibility graph. The intersection may be computed using an algorithm of time complexity $O((n+c) \log n)$, where c is the total number of edge-edge intersections that arise (see Nievergelt and Preparata [16]). Consequently, the entire visibility graph may be constructed in time $O(rn(n+c) \log n)$. We did not implement the search algorithm in this form.

The planner, as implemented, is complete only to the time resolution between slices. Observe, however, that the slice representation essentially represents the complete configuration space-time. In fact, between slices the space-time constraints are simply a collection of polyhedral swept volumes. Thus the space-time representation is complete, although the search algorithm is not. To be complete, a search algorithm would need to consider motions that could change direction between slices.

One approach toward making the algorithm complete, would be to introduce space-times slices at certain critical times. The objective is to introduce slices at those times at which the topology of free space changes. This is a function of the number of interactions between the other objects in the environment. A slice should be introduced whenever two moving or stationary configuration space obstacles touch or intersect. A conservative bound on the number of slices is given by $r = O(sn^2)$. Here s is the number of distinct time intervals over which each of the objects whose motions were previously planned performs a single straight-line constant-velocity motion. We did not implement the planner in this form.

5.5. Summary for Translating Planar Objects

1. The position of a configuration space obstacle at a particular time may be determined from its position at time zero by translation.
2. Configuration space-time is represented as a series of configuration space slices at fixed points in time.
3. Configuration space-time is searched using a visibility graph algorithm.

4. Collisions between proposed trajectories and moving objects are detected using line-polygon intersection tests.
5. The planner is complete only to the time resolution between slices, unless the free space-time between slices is also searched.

6. Linked Planar Arms with Rotary Joints. The second domain that we will explore consists of two-link articulated planar arms. Each arm consists of two links and two joints, about which the links may rotate. The base of each arm is fixed in the plane, so the arms do not translate. The links are modeled as convex polygons.

In addition to the arms, the environment contains stationary obstacles that are also modeled as convex polygons. The objective is to plan a collision-free path for an arm between specified start and goal configurations. The motions of the other arms are assumed to have been planned already. As explained previously, this approach may be used to plan motions for several arms, by assigning priorities to the arms, and planning motions one arm at a time.

6.1. Constructing the Configuration Space-Time. In the previous section we saw that for translational motions it is fairly easy to build the configuration space-time. One merely builds a standard configuration space at time zero, then translates the moving obstacles in configuration space in correspondence with the translations of the moving objects in real space. Unfortunately, there is no such simple technique available for building configuration space-time once rotations are permitted. The basic cause of the difficulty stems from the nonlinearity of the constraints imposed by obstacles in the environment on the rotational degrees of freedom of a moving object.

For rotating linked arms with fixed bases the basic motions performed are rotations of various polygons about various rotation centers. If a given arm's joints are allowed to rotate in unison, then several of these rotations may be superimposed. For convenience, therefore, let us assume that only one joint of any arm is allowed to move at a time. For further convenience let us also assume that each polygon is convex. This assumption of convexity is not necessary, but it simplifies the computation and complexity. Given these assumptions, the basic motions are indeed rotations of various convex polygons about various rotation centers. It is thus sufficient to concentrate on analyzing the interaction of two convex polygons, each rotating about its particular rotation center. The constraints resulting from the interaction of two arms may be built up from the constraints of several such pairs of polygons. In each pair, one of the two polygons is part of the arm whose motion has already been determined, while the other polygon is part of the arm whose motion is currently being planned.

6.2. Constraints Arising from Rotating Polygons. The task now is to derive the constraints imposed on one rotating polygon, the *planning object*, by the motion of another rotating polygon, the *obstacle polygon*. The situation is fairly analogous to the domain of translating planar polygons. The difference lies in the difficulty of computing nonlinear time-varying constraints.

One approach would be to reduce the problem further. For instance, the planner could compute the collection of half-space constraints arising from each pairing of an edge on one object with a vertex on the other object. By intersecting and unioning these constraints appropriately (see Lozano-Pérez [14], Donald [7], and Canny [6]), the planner could determine the effective constraints imposed on one object by the motion of the other object.

The difficulty with this approach lies in its complexity. By considering all pairs of edges and vertices, the planner would be expanding all possible constraints, even those subsumed by other constraints. As an example, the constraint imposed on a vertex by an edge on the near side of an obstacle subsumes the constraint imposed by an edge on the far side of the obstacle. In planning a path, the planner must test feasibility of the path against all possible constraints. This involves unnecessary work in the case that some constraints are subsumed by other constraints. Alternatively, the planner could first decide which constraints were active and which were subsumed by other constraints, and then only test path feasibility against the active constraints. Unfortunately, as the obstacle rotates, some active constraints become subsumed by other constraints, while some inactive constraints become active. Thus the planner would be forced to decide constantly which constraints were about to become active.

Instead of expanding all the constraints and then deciding which are active, a slightly different approach is to consider only active constraints. As some of these expire and become subsumed by other constraints, the planner determines the newly activated constraints directly from the expiring constraints. This approach is based on the observation that for convex objects the conditions defining a constraint are purely local. In particular, the validity of a constraint depends only on the edges and vertices at the point of contact (see also Donald [7]). Furthermore, constraints expire and become subsumed by newly activated constraints precisely at configurations for which several constraints agree, that is, intersect as hypersurfaces in configuration space. Geometrically, these constraints represent the simultaneous contact of several vertices and edges, in the form of vertex-vertex contacts or edge-edge alignments. The conditions under which these events occur may be determined locally. Thus the planner can predict which constraints subsume previously active constraints directly from the previously active constraints. The remainder of this section considers the types of conditions under which constraints may change, while later sections analyze the constraints themselves in more detail.

In the current case we are dealing with the interaction of two polygons. Thus the type of constraints that the planner must consider are vertex-edge interactions. There are two types. One involves the interaction of an edge of the planning object with a vertex of an obstacle object. The other type involves the interaction of a vertex of the planning object with an edge of an obstacle object. (See Figure 9.)

Notice that any such constraint locally defines a rotational direction of forbidden angles. This direction is the direction along which a rotation of the planning object would result in an intersection with the obstacle polygon. The constraint itself represents the orientation at which the planning object just touches the

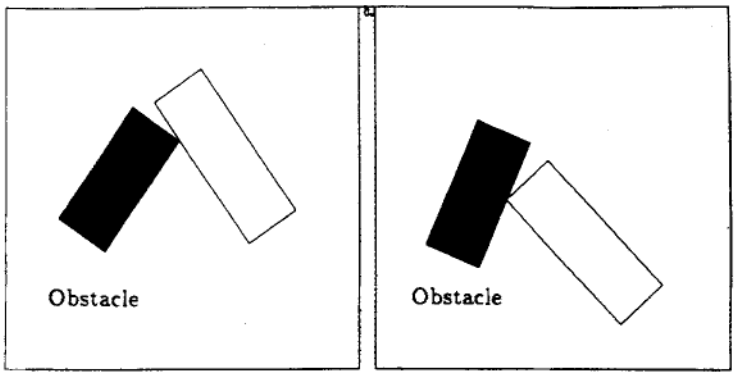

Fig. 9. There are two basic types of vertex-edge contacts.

obstacle polygon. The direction of forbidden angles is determined by those local orientations of the planning object at which the planning object and the obstacle polygon overlap.

In order to determine, for a particular configuration of the obstacle polygon, all orientations of the planning object that are forbidden, and all orientations that are valid, the planner considers all active constraints not subsumed by other constraints. For each such constraint, the planner locally determines the rotational direction of forbidden orientations. The planner then merges the constraints based on orientation, and pairs up adjacent constraints that have opposing directions of forbidden angles. For instance, in the example of Figure 10 there are two active constraints. When these are merged, the resulting forbidden range of angles is an arc of orientations, as shown. In general, even for convex objects, there may be more than two active constraints that define the range of forbidden orientations. Thus there may be more than one arc of legal orientations.

For a particular orientation of the obstacle polygon there are a finite number of orientations of the planning object at which the two polygons touch but do not overlap. As the obstacle polygon rotates about its rotation center, the orientations of the planning object at which these contacts occur change continuously. The basic strategy in constructing the configuration space-time entails tracing these touching orientations as the obstacle polygon rotates. The resulting constraint contours describe the boundaries of the forbidden regions in space-time.

Consider a specific constraint contour, arising from some vertex-edge or edge-vertex contact. As the obstacle polygon rotates, the point of contact between the vertex and the edge moves along the edge. A number of events can occur at which the constraint contour changes character:

1. The direction of travel of the contact along the edge may reverse sign.
2. The direction of rotation of the planning object required to maintain contact may reverse sign.
3. The contact may disappear, as when the obstacle rotates out of the reach of the planning object.

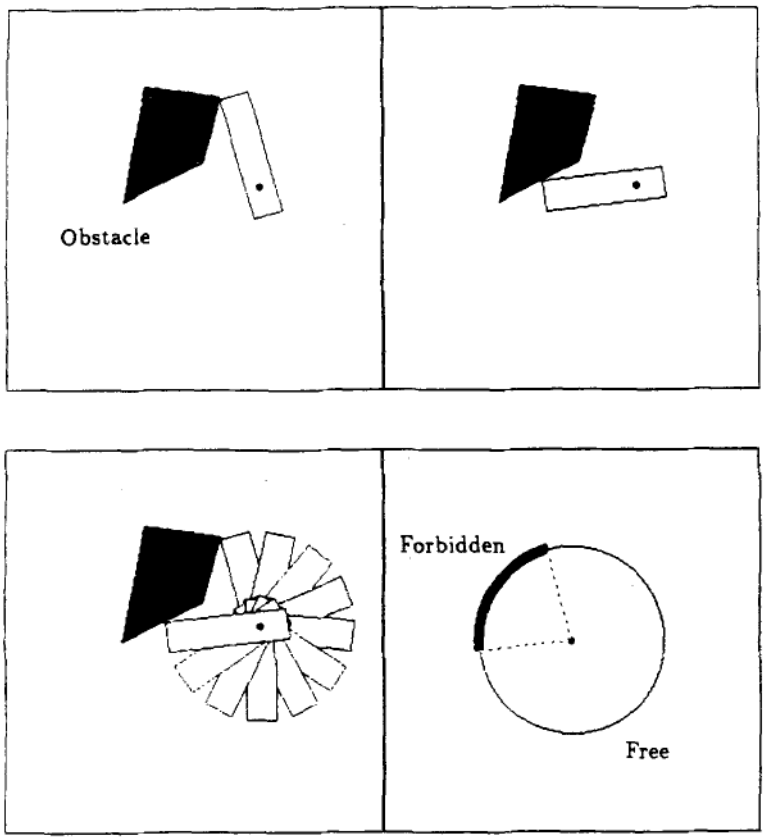

Fig. 10. This figure considers the constraints imposed by an obstacle polygon on the possible orientations of a rectangle rotating about a rotation center. The first two frames show the rectangle in two orientations that just touch the obstacle. The third frame shows the free range of motions possible. The last frame represents the range of forbidden and free orientations.

4. The contact may run off one end of the edge, that is, vertex-vertex contact may occur.
5. The edge defining the constraint may become aligned with one of the edges incident at the vertex defining the constraint, that is, edge-edge alignment may occur.

The planner analyzes the conditions under which these events occur. Of particular interest are the events in which contacts appear or disappear, and the events in which vertex-vertex contacts or edge-edge alignments occur. At these orientations the constraint contours fundamentally change character, either merging with other contours or splitting into several contours. In other words, some constraints may expire, perhaps becoming subsumed by newly activated constraints. Thus, analyzing the conditions under which the events listed above occur, and determining how the relevant constraints change during these events, directly solves the problem of constructing configuration space-time for rotating polygons.

To be more precise, in Section 6.5 we will derive the orientation constraint imposed on the planning object due to the interaction of an edge of the planning object with a vertex of the obstacle polygon, over the full range of orientations of the obstacle polygon. Similar constraints may be derived for all edge-vertex interactions. These constraints are derived independent of time. Each constraint simply describes the touching orientations of the planning object as a function of the orientation of the obstacle polygon. In other words, if α denotes the orientation of the obstacle polygon, and β the orientation of the planning object, then the constraint resulting from a particular edge-vertex interaction is some function $\beta(\alpha)$. Once a particular motion of the obstacle polygon is known, this constraint function may be thought of as a function of time. Specifically, if α varies as a function of time $\alpha(t)$, then the space-time constraint is $\beta(t) = \beta(\alpha(t))$.

We assume that all joint motions have piecewise constant velocities. Consequently, over each constant-velocity segment, a scaled subset of the graph of the constraint function $\beta(\alpha)$ describes the relevant space-time contour.

The planner augments each time-independent constraint function with those orientations of the obstacle polygon at which the constraint contour could change character, such as during vertex-vertex contact or edge-edge alignment. These conditions may also be determined as geometric conditions on α and β, independent of time. Once a particular motion of the obstacle polygon is known, the planner can then predict the points in time at which a given constraint expires or becomes subsumed by another constraint. In this fashion the planner handles the transitions between different kinds of contacts.

Summary of the Construction of Space-Time for Two Rotating Polygons. Let us summarize. The planner builds the boundary of the forbidden regions in space-time. In order to understand this construction, let us consider this boundary. The boundary describes those orientations at which the planning object is just touching the obstacle polygon. The boundary is the union of smaller segments, each segment representing the interaction of a particular vertex or edge on the planning object with a particular edge or vertex, respectively, on the obstacle polygon. The segments link up at points where the two objects have aligned edges or touching vertices. Each segment may be computed as a constraint function relating the orientations of the two objects at which the vertex-edge contact occurs. The structure of this constraint function is independent of time. However, its location in the space-time is determined by a particular motion of the obstacle polygon.

The planner thus computes the constraint functions for all edge-vertex interactions independent of time. Given a particular motion of the obstacle polygon, the planner describes the boundary of forbidden space-time by segments of the graphs of these constraint-functions. In order to decide which segments are relevant, the planner begins with the edge-vertex contacts in effect at the beginning of the motion. For any active contact constraint, the planner follows the constraint until a vertex-vertex or edge-edge alignment occurs, or the contact breaks. At this point, the constraint may become inactive, while other constraints may become active. The other constraints that might become active are precisely those determined by the other edges or vertices at the alignment. Whether or not these

constraints become active may be determined locally. The details are discussed in Sections 6.4 and 6.5.

The complete space-time boundary is constructed by repeatedly following constraint contours in this fashion.

6.3. Example. The first row in Figure 11 displays two rotating triangles along with their rotation centers. The next three rows in Figure 11 show the construction of the forbidden regions representing the constraints imposed on the smaller triangle by a rotation of the larger triangle. For simplicity we approximated the constraint contours using bounding rectangles. However, the equations of the exact boundary contours are given in Section 6.5. These could be used by a planner for exact descriptions of the forbidden space-time.

In order to explain Figure 11, consider the pair of frames labeled 1. In this pair of frames, the larger triangle performs some rotation. The smaller triangle is shown in two extreme orientations, at which it is just touching the larger triangle. The forbidden orientations of the smaller triangle during the motion of the larger triangle are all orientations between these two extreme orientations. They are represented in space-time by the rightmost rectangle constructed in the first frame. The horizontal portion of the rectangle represents the motion of the large triangle; the vertical portion represents the forbidden orientations of the small triangle.

The pair of frames labeled 2 depicts another rotation of the large triangle, along with the associated constraint rectangle.

Consider now the two pairs of frames labeled 3 and 4. The large triangle is shown performing the same motion in both frames. However, during this motion there are two distinct ranges of orientations that are forbidden for the small triangle. These two ranges are indicated by the two sets of extreme orientations shown in the frames, as well as the two rightmost constraint rectangles.

Finally, the pairs of frames labeled 5 and 6 show the construction of further constraint rectangles over the remaining motion of the large triangle.

6.4. Representing Constraints Arising from Edge-Vertex Interactions. In the next two sections we will consider the constraints arising from the interactions of pairs of edges and vertices. Notice that a vertex on an object describes a circle as the object rotates. Similarly, any edge of a rotating object defines a line rotating about some rotation center. The constraint defined by the interaction of a vertex and an edge may thus be represented by a circle and a rotating line. A particular contact of the vertex and the edge may be depicted by a specific point on the circle, and a specific orientation of the rotating line, for which the line passes through the point on the circle.

In order to analyze edge-vertex constraints, it is thus sufficient to consider points moving on circles and lines rotating about rotation centers in such a fashion that the lines pass through the points. This representation makes computation of the constraints easy. In particular, it allows a planner to determine how the rotation of the planning object must change in order that contact be maintained

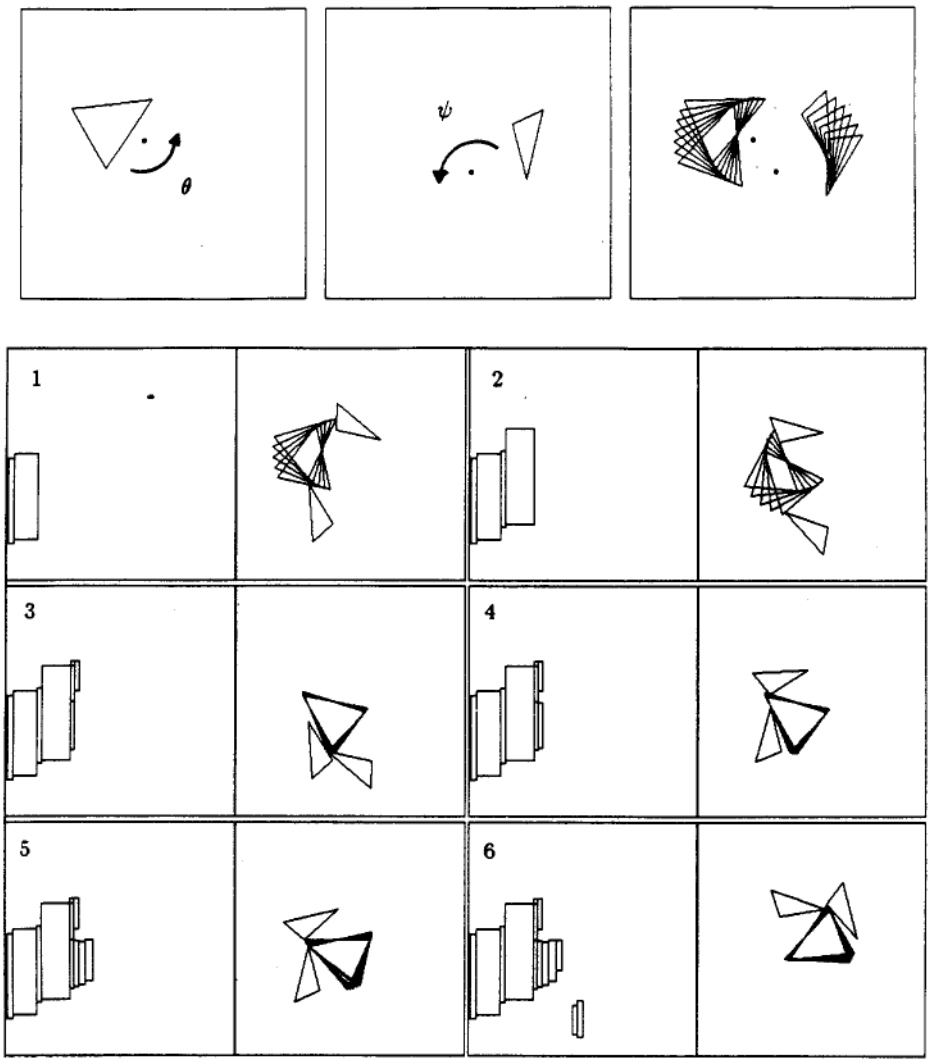

Fig. 11. Construction of the constraints imposed on the smaller triangle by a motion of the larger triangle. The larger triangle rotates by π about its rotation center. The constraints are approximated by rectangles. In an alternating fashion, the figures display the constraints constructed thus far, and the motion of the larger triangle over the most recently constructed constraint rectangle. The smaller triangle is displayed at the two extreme orientations of this constraint rectangle.

as the obstacle polygon rotates. This information defines the shape of the configuration space-time obstacle determined by the motion of the obstacle polygon. Local interior information at the edge determines which side of the obstacle contour in configuration space-time is forbidden. Additionally, by considering the orientations of the edges incident at the vertex involved in the edge-vertex interaction, the planner can predict the times at which edge-edge alignments occur. Similarly, the representation makes explicit the motion of the contact point along the line. By considering the length of the edge involved in the edge-vertex interaction, the planner can predict the times at which the contact moves off the end of the edge, that is the times at which vertex-vertex contacts occur. Finally, the representation makes explicit points on the circle and orientations of the line at which contact is impossible, thereby indicating orientations of the obstacle polygon at which contacts must vanish. Thus the circle-line representation, when augmented with local interior and incident edge information, provides the planner with a means for determining all the events at which constraints may change character.

A final comment is in order. Suppose that edge-edge alignment or vertex-vertex contact occurs. At that point in time, there are up to four pairs of edge-vertex contacts in effect. The planner must decide which contacts, if any, remain active as the obstacle polygon continues to rotate. The decision for a particular contact pair depends on two derivatives. One derivative is the derivative of the motion of the contact point along the edge. The other derivative is essentially the instantaneous relative motion of the planning object to the motion of the obstacle polygon. The distance derivative allows the planner to decide whether a contact cannot possibly exist because the contact would have to occur outside the boundaries of the edge. The relative motion derivative allows the planner to decide whether a contact cannot possibly exist because it would force two object edges to pass through each other. (See Section 6.5 for details.)

In fact, the exact values of the derivatives are not important. All that is required of the distance derivative is its sign. All that is required of the relative motion derivative is whether it is larger or smaller than unity. This information is readily available while computing the circle-line representation, and may be incorporated directly into the representation. In the next section we shall compute such a representation.

6.5. Analyzing Constraints Arising from Edge–Vertex Interactions. As we have already noted, there are two types of edge-vertex interactions, determined by whether the vertex is part of the obstacle polygon or part of the planning object. In this section we will analyze the circle-line representation for one of these two types, namely, the case in which the vertex is part of the obstacle polygon and the edge is part of the planning object. The symmetric case, in which the vertex is part of the planning object and the edge is part of the obstacle polygon may be analyzed in a similar fashion.

For the case we have chosen, the circle of the circle-line representation corresponds to the vertex of the obstacle polygon, while the line corresponds to the edge of the planning object whose motion is being planned. The canonical picture

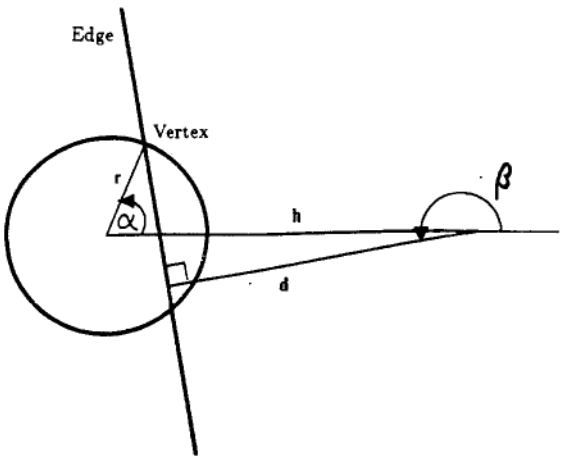

Fig. 12. Canonical circle-line picture representing the interaction of a vertex and an edge. In this case the vertex belongs to the obstacle polygon, and the edge belongs to the object whose motion is being planned. The distance of the vertex from its rotation center is given by r. The distance of the edge from its rotation center is given by d. The separation of the two rotation centers is given by h. The orientation of the obstacle is measured in terms of α, while the orientation of the planning object is measured in terms of β.

for this circle-line representation is given by Figure 12. The two rotation centers are separated by a distance h, which we will assume is greater than zero. For convenience we will depict the two rotation centers on a horizontal line with the obstacle's rotation center to the left of the planning object's rotation center. The distance from the obstacle's rotation center to the vertex is r, while the normal distance from the planning object's rotation center to the line representing the edge is d. We will measure the orientation of the obstacle by the angle between the horizontal and the line from the obstacle's rotation center to the vertex. Denote this angle by α. Similarly, we will measure the orientation of the planning object by the angle between the horizontal and the edge normal pointing away from the planning object's rotation center. Denote this angle by β.

We are interested in determining the constraint imposed on the planning object by the obstacle polygon over its range of possible orientations. It is important to note that we will derive this constraint independent of time, simply as a geometric relation between α and β. Specifically, we will determine the orientation of the planning object at which its edge just touches the obstacle polygon's vertex, as a function of the obstacle polygon's orientation. Denote this function by $\beta(\alpha)$. This function describes the geometric dependence of β on α. This dependence does not involve time. Once a particular motion of the obstacle polygon is known, say as a function $\alpha(t)$ of time, the forbidden orientation β of the planning object may also be determined as a function of time, simply by noting that

$$\beta(t) = \beta(\alpha(t)).$$

Over any interval of time, the planner then uses the function $\beta(t)$ as a description

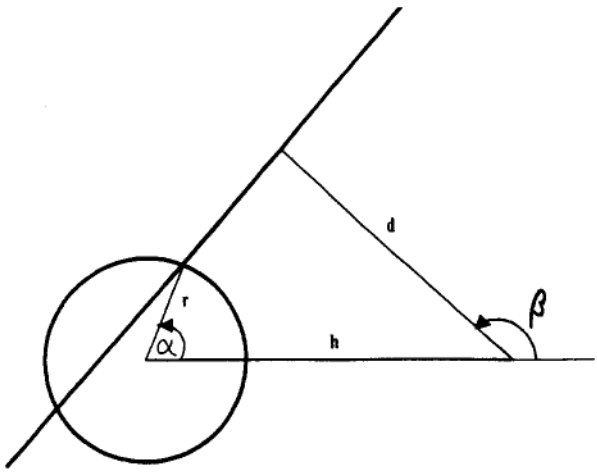

Fig. 13. In general, for a given orientation α, there are two values of β for which the line determined by β intersects the point on the circle determined by α. One such orientation of β was shown in Figure 12. The other orientation is shown in this figure.

of the constraint contour arising from the interaction of the particular vertex on the obstacle polygon and the particular edge on the planning object over the given time interval. This contour is combined with constraint contours arising from other vertex-edge contacts in the manner described in Section 6.2.

Returning now to our analysis of the particular vertex-edge contact of Figure 12, notice that for each value of α there may be a corresponding value of β for which the line representing the edge passes through the point representing the vertex. In general, of course, there are two values of β (see Figure 13), and in some cases no values. In any event, the problem of finding the function $\beta(\alpha)$ reduces to determining how β varies as α varies from 0 to 2π. In general, the behavior depends on the relative values of h, r, and d. We will consider the case for which $h > r$ and $h + r > d > h - r$. The other cases are similar.

Since $d > h - r$ there are values of α for which no value of β produces a contact between the vertex and the line. Specifically, consider the two points on the circle that are exactly distance d away from the planning object's rotation center. One of the two arcs on the circle that connect these two points can never intersect the rotating line, as all points on this arc are less than distance d away from the planning object's rotation center. Let

$$\alpha_0 = \cos^{-1}\left(\frac{r^2 + h^2 - d^2}{2rh}\right),$$

where we take the value of $\cos^{-1}$ to lie between 0 and π. The two points on the circle are thus given by α_0 and $2\pi - \alpha_0$. The forbidden arc is the arc between these two points that includes the point $\alpha = 0$. For values of α in this range there is no solution of β in terms of α. (See Figure 14.)

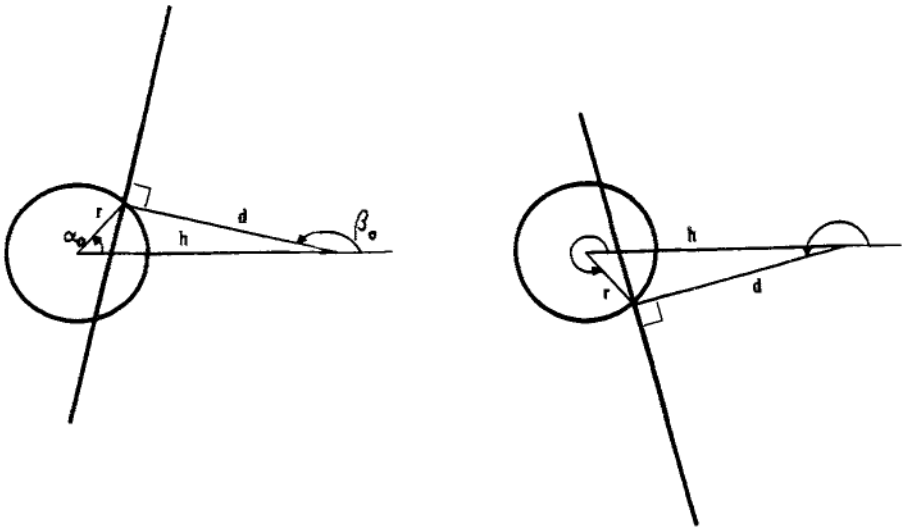

Fig. 14. This figure displays the orientations of α at which the vertex represented by α is exactly distance d away from the right rotation center. If the vertex lies on the arc through the α-origin that connects these two orientations, then there is no orientation β for which the edge represented by β intersects the vertex.

Beginning at $\alpha = \alpha_0$ there is a single solution of β in terms of α that has multiplicity two. This root splits into two distinct roots as α increases from α_0 to $2\pi - \alpha_0$, merging again at $\alpha = 2\pi - \alpha_0$. Let us trace the behavior of one of these two roots. The behavior of the other root is similar. As α increases from α_0, for the root we are tracing, initially β increases as well.

Let us measure the motion of the contact point along the line representing the contact edge in terms of the signed distance s between the contact point and the point on the line that is closest to the planning object's rotation center, as shown in Figure 15. As indicated in Section 6.4, the sign of the derivative of s with respect to α allows the planner to decide whether a motion causes the contact point to move beyond the endpoints of the edge.

Let us denote by D the relative rotation rates of β to α, that is $D = d\beta/d\alpha$. The planner uses the sign of $D-1$ to decide whether edges might try to pass through each other after an edge-edge alignment has occurred. This is because the sign of $D-1$ determines which of the two objects is rotating more quickly, as α increases.

Before tracing through the details of one of these roots let us write down expressions for β and s in terms of α. The picture to keep in mind is Figure 16.

If we write

$$l = \sqrt{r^2 + h^2 - 2rh\cos\alpha},$$

then

$$\beta = \cos^{-1}\left(\frac{d}{l}\right) + \arctan(r\sin\alpha, r\cos\alpha - h),$$

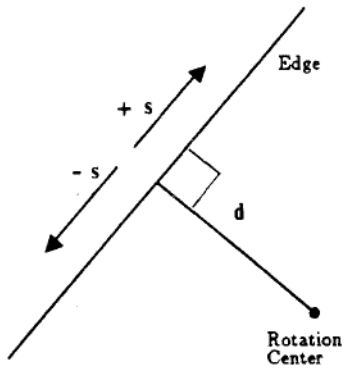

Fig. 15. This figure shows how to measure the distance s of a point on the line representing the edge of contact. Distance is measured as the signed distance from the point on the line that is closest to the line's rotation center.

assuming that $l > d$. Furthermore, we have that

$$s = r \sin(\beta - \alpha) - h \sin \beta.$$

Observe that

$$l^2 = r^2 + h^2 - 2rh \cos \alpha.$$

Thus, implicitly differentiating with respect to α yields

$$\frac{dl}{d\alpha} = \frac{rh}{l} \sin \alpha.$$

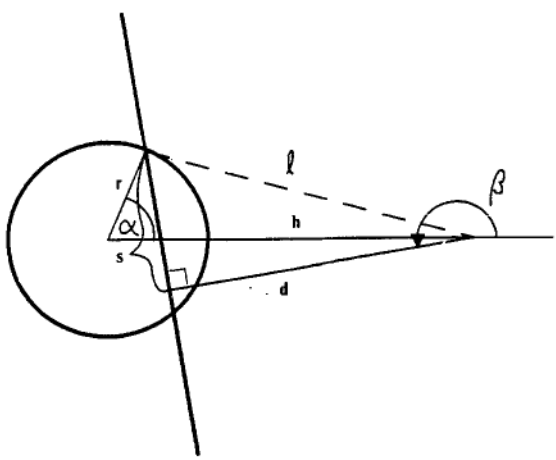

Fig. 16. The canonical circle-line representation augmented with further notation.

From this we see that $dl/d\alpha$ changes sign exactly when $\alpha = 0$ or $\alpha = \pi$. Implicitly differentiating

$$s^2 + d^2 = l^2,$$

we see that

$$s\frac{ds}{d\alpha} = l\frac{dl}{d\alpha}.$$

Since $l > 0$ and since $s \geq 0$ for the root we are considering, it follows that $ds/d\alpha$ and $dl/d\alpha$ change sign simultaneously. Thus we have determined the conditions under which the contact point changes its direction of motion along the edge. This event occurs whenever contact is possible and the vertex participating in the contact lies on the line passing through the two rotation centers.

We must also decide how to compute $D = d\beta/d\alpha$. Observe that

$$r\cos(\beta - \alpha) = h\cos\beta + d.$$

Hence, implicitly differentiating with respect to α,

$$r\left(\frac{d\beta}{d\alpha} - 1\right)\sin(\beta - \alpha) = h\frac{d\beta}{d\alpha}\sin\beta.$$

Thus $D = 1$ exactly when $\beta = 0$ or $\beta = \pi$, that is, exactly when the edge participating in the contact is perpendicular to the line passing through the two rotation centers. At these points the relative rotational motion of the two objects changes sign. When $D > 1$ the planning object is rotating faster than the obstacle object. When $D < 1$, it is rotating more slowly.

Finally, notice also that

$$(*) \qquad \frac{d\beta}{d\alpha} = \frac{r\sin(\beta - \alpha)}{r\sin(\beta - \alpha) - h\sin\beta}.$$

Thus $d\beta/d\alpha$ vanishes when $\beta = \alpha$ or $\beta = \alpha + \pi$. In other words, the derivative vanishes whenever the line representing the contact edge is tangent to the circle at the point of contact. At these tangency points the motion of β changes direction. Said differently, at these tangency points local minima or maxima occur in the function $\beta = \beta(\alpha)$ which represents the value of β at which contact occurs as a function of α. We will refer to the graph of this function as an α-β contour henceforth.

This tangency information is useful for approximating the α-β constraint contours. For instance, one might wish to approximate the contours conservatively using rectangles that completely enclose the forbidden regions in configuration space-time. Over a region in which β varies monotonically with α, an enclosing rectangle may be constructed directly from the start and end points of the α-β contours (see Figure 17). Knowing the points at which $d\beta/d\alpha$ vanishes, allows the planner to split the contours into such monotonic segments.

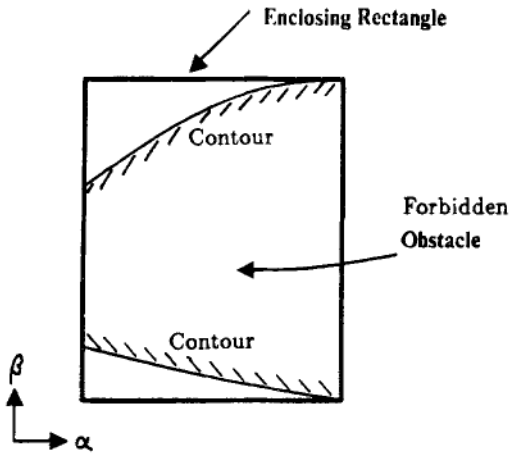

Fig. 17. Over regions in which the α-β contours are monotonic, the planner can easily construct a bounding rectangle that conservatively approximates the forbidden regions defined by the α-β contours. The planner uses this approximation for simplicity. The constraints are, in fact, analytically representable.

As an aside, notice that the denominator of the right-hand side of equation (*) is just the signed distance s. Thus this denominator vanishes precisely at the orientations of α for which s is zero, namely, at $\alpha = \alpha_0$ and $\alpha = 2\pi - \alpha_0$. This makes sense, for it says that the derivative $d\beta/d\alpha$ becomes unbounded precisely at those orientations of the obstacle polygon for which the planning object must break contact.

Let us now trace through the contact constraint as α varies.

1. The contact constraint first appears at $\alpha = \alpha_0$. The orientation of the edge is given by $\beta = \beta_0$, where

$$\beta_0 = \pi - \cos^{-1}\left(\frac{d^2 + h^2 - r^2}{2dh}\right).$$

   The value of s at this point is, by construction, zero.
2. For the root we are tracing, as α increases from α_0, both β and s increase as well. Furthermore, D is bigger than unity. The first relevant event occurs when $\beta = \pi$. (See Figure 18.) At this point $\alpha = \cos^{-1}((h-d)/r)$ and $s = \sqrt{r^2 - (h-d)^2}$. The event that occurs is that $D-1$ changes from being positive to negative, that is, at the point we have $D = 1$.
3. The next event that occurs is that the motion of the contact point along the line changes direction. In other words, s, which was increasing, starts to decrease. This event occurs when $\alpha = \pi$, $\beta = \pi + \cos^{-1}(d/(h+r))$, and $s = \sqrt{(h+r)^2 - d^2}$. (See Figure 19.)
4. The third event that occurs is that D changes sign, which happens when the line representing the contact edge is tangent to the circle at the point of contact.

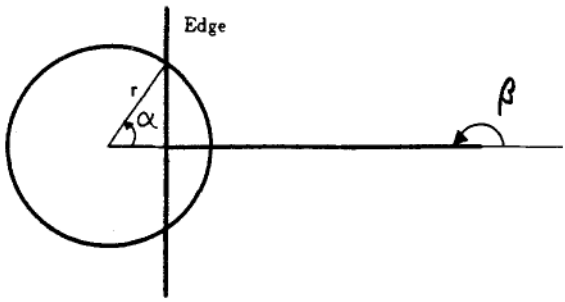

Fig. 18. The relative rotation rates of the two objects portrayed by α and β are identical whenever β is 0 or π. This is the case whenever the edge defining the vertex-edge constraint is perpendicular to the line joining the two rotation centers. To either side of one of these orientations one object is rotating faster than the other object. Furthermore, the faster object on one side is the slower object on the other side.

(See Figure 20.) The significance of this event is that β changes direction. In other words, whereas β was increasing with α previously, it now decreases as α increases. The event occurs for $\alpha = \beta = \pi + \cos^{-1}((d-r)/h)$ and $s = \sqrt{h^2-(d-r)^2}$.

5. The last event that occurs is that the two roots remerge. This occurs when $\alpha = 2\pi - \alpha_0$, $\beta = 2\pi - \beta_0$, and $s = 0$. As α increases beyond this point, the contact constraint must disappear. This is because there is no orientation β for which the line representing the contact edge can make contact with the point on the circle representing the contact vertex at orientation α.

6.6. Representing Multiple Joints. The circle-line representation provides a simple method for describing the constraints arising from the interaction of two

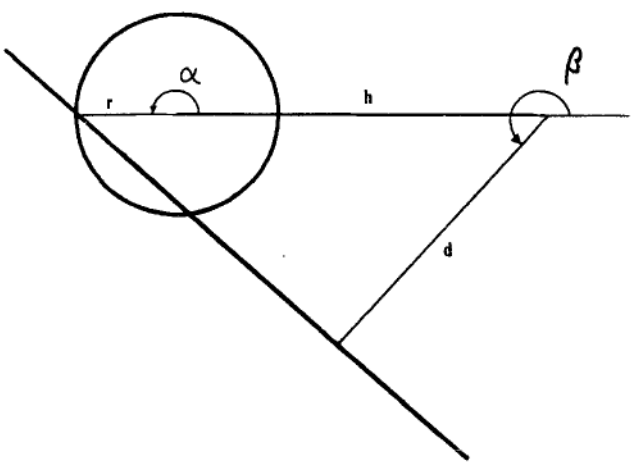

Fig. 19. The direction of travel of the contact point along the edge changes direction whenever α is 0 or π. This is the case whenever the contact point lies on the line joining the two rotation centers.

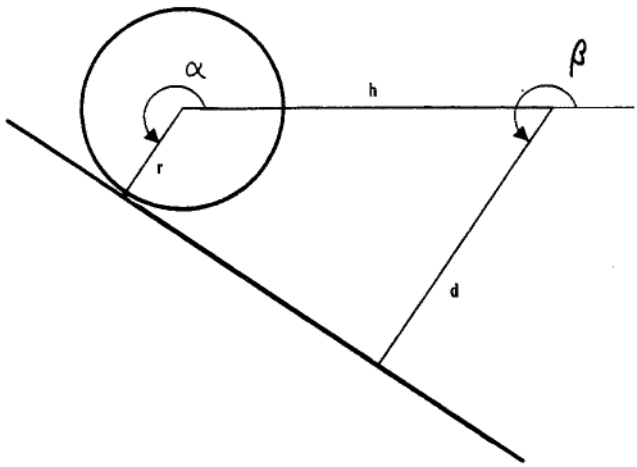

Fig. 20. The direction of rotation of β required to maintain contact as α increases changes direction whenever the edge defining the vertex-edge constraint is tangent to the circle at the point of contact.

rotating polygons. For nonconvex polygons the superposition of several such representations determined from a convex decomposition of the polygons provides a two-dimensional space-time description of the relevant constraints. As before, the constraints may be approximated by bounding rectangles.

For multilink arms, it is necessary to extend the dimension of the configuration space-time by the additional number of joints. For example, for a two-link arm in the presence of other moving arms, the configuration space-time is three-dimensional. Our objective now is to construct the relevant constraints imposed on an arm by the motions of other arms and by stationary obstacles.

The basic approach consists of reducing the dimensionality of the problem by computing constraints as two-dimensional slices. Consider Figure 21, which portrays a typical two-link arm. For each fixed orientation of Link 1, the planner can construct a two-dimensional space-time for Link 2 that represents the constraints imposed on Link 2 by the other rotating arms and by the stationary

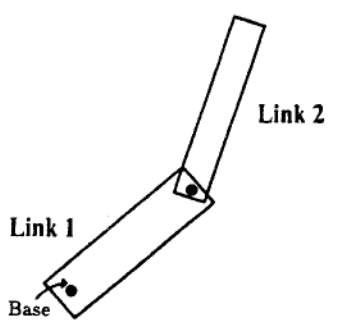

Fig. 21. A two-link arm. The arm has two rotational degrees of freedom. The arm's base is fixed in the plane.

obstacles. During each such computation the base point about which Link 2 rotates is held fixed at the orientation given by Link 1. Recall, we assumed that for each arm in the environment, only one joint is actually rotating at any time. Thus the approach previously outlined may be applied to compute this two-dimensional space-time. Staggering these two-dimensional slices produces an approximate representation of the complete three-dimensional configuration space-time.

A few issues deserve comment. First, the planner discretizes the orientations of Link 1, computing slices for the resulting finite number of orientations. The representation of configuration space-time is thus only approximate, limited in resolution to the angular separation between slices.

Second, it is possible to reduce the number of orientations of Link 1 for which it is necessary to compute space-time slices. Certainly, any orientation of Link 1 for which there is a collision between Link 1 and any other arm or object in the environment, is an invalid configuration at the time of collision. All orientations of Link 2 are automatically forbidden at that time. Thus it is only necessary to compute portions of space-time slices over time intervals and at orientations of Link 1 that are collision-free. In order to determine these orientations it is sufficient to compute a two-dimensional configuration space-time for Link 1 that represents the constraints imposed on Link 1 by the other rotating arms and by the stationary obstacles. This space may be computed in the manner outlined previously. Slices are then computed for Link 2 for all the discretized orientations of Link 1 that lie outside Link 1's constraint contours.

Further optimizations are possible. For instance, one can determine orientations of Link 1 for which all possible orientations of Link 2 are guaranteed to be collision-free. This may be done in a conservative manner by approximating the swept volumes of rotating links using sectors of circles.

6.7. Searching the Configuration Space-Time. The planner represents the free regions of configuration space-time as a collection of rectangles in each of the slices. These free space rectangles are simply the complements of the constraint rectangles described earlier. Thus the free space rectangles are conservative representations of free space, in the sense that they are subsets of the actual free space. Searching for a collision-free path for a given two-link manipulator consists of determining a sequence of motions within and across slices that remains in free space and leads from the start to the goal configuration. Motions within a slice represent rotations of Joint 2 alone, while motions across slices represent rotations of Joint 1 alone.

Most of the issues that arise in searching for a collision-free path in configuration space-time have already been alluded to. In particular, the path should never lead backward in time. Additionally, if there exist maximum velocity constraints on the joints then the path may have to maintain certain slope constraints as a spatial function of time. Finally, the search may wish to take into account the temporal or spatial cost of proposed motions.

A statement about the representation of paths. Whereas for the planar case of translating objects we chose only to consider particular paths, namely, those that

connected vertices in the environment, in this case we shall choose to represent all paths implicitly. This is accomplished by regarding regions as descriptive of paths. A path is possible between two regions only if the regions connect. While searching, the planner does not construct explicit paths, but merely considers the connectivity between regions. The output of the search phase is thus an ordered list of regions in which any two regions that are adjacent in the list are connected in the configuration space-time. A particular path may then be determined by choosing any path that passes through the regions in the order specified and across the regions of connectivity.

Deciding whether two regions are connected involves both a spatial and a temporal decision. The distinction is necessary because of the prohibition on moving backward in time. Let us first consider the case of two regions in the same space-time slice. Consider Figure 22. In order that two regions be connected it is necessary that they are connected in the spatial dimension. In both Part A and Part B of Figure 22, R_1 and R_2, and R_2 and R_3 are connected spatially, since each of the regions in each pair shares a common point in space-time. However, there is a path from region R_1 to region R_3 only for the arrangement of Part A, not for that of Part B. This is because any such path for Part B would require motion backward in time.

In general, the temporal decision regarding connectivity of two regions depends on the particular partial sequence of regions in which the two regions find themselves during the search. Specifically, it is necessary to associate a minimum time with any partial sequence. The minimum time represents that value of time below which no space-time path may venture lest it be moving backward in time.

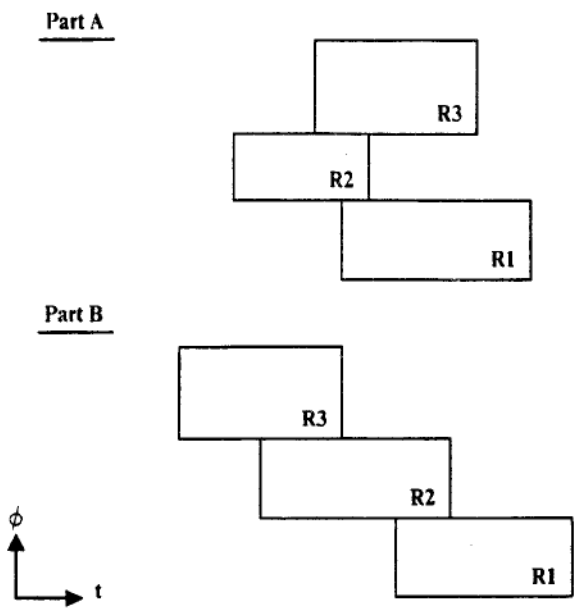

Fig. 22. There is a path from region R_1 to region R_3 only for the example of Part A. Any path from R_1 to R_3 in the example of Part B would require moving backward in time.

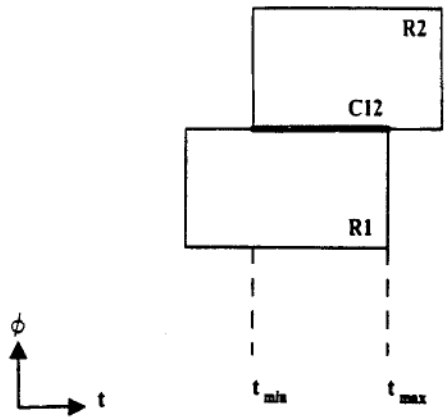

Fig. 23. Regions R_1 and R_2, along with their intersection C_{12}. The minimum and maximum time coordinates of points in C_{12} are given by t_{min} and t_{max}.

Initially this value is set to zero. As the search phase expands a partial sequence it updates this value.

Updating the minimum time value of a partial sequence is fairly straightforward. Suppose that region R_1 is the current last region on a partial sequence, and suppose that the planner is considering adding region R_2 to the partial sequence. (See Figure 23.) Let $C_{12} = R_1 \cap R_2$. The planner first checks that C_{12} is nonvoid. This is effectively the spatial decision of connectivity. Now let t_{min} be the minimum time coordinate of any point in C_{12}, and let t_{max} be the maximum time coordinate of any point in C_{12}. Then the new value of the minimum time associated with the new partial sequence is simply the maximum of the old minimum time value and t_{min}. Furthermore, the new partial path is valid only if this new value of the minimum time is no greater than t_{max}.

The case of transitions across space-time slices, that is, the case of rotations in Joint 1 alone, is handled similarly. The only real difference is that the intersection of two regions will in general not be a line segment, but some two-dimensional region. In our implemented planner, these regions are always rectangles.

Let us briefly comment on the case in which the joints have maximum velocity bounds. In this case, the intersection region C_{12} must be shrunk to account for points that are not reachable from the previous region of intersection. (See Figure 24.) In some cases this may cause the region C_{12} to become empty. One option is for the planner to discard the current partial sequence. Alternatively, the planner could slow down the other arms, thereby effectively dilating time and possibly enlarging the intersection region. Our implemented planner does not consider slowing down the other arms.

Finally, in assigning temporal costs to transitions the planner should keep track of both the minimum and maximum times required to pass from one region to another. Again, given a partial sequence and the previous and current intersection regions C_{01} and C_{12}, the planner can determine the minimum and maximum times required to move from C_{01} to C_{12}. Any actual path will have a time cost

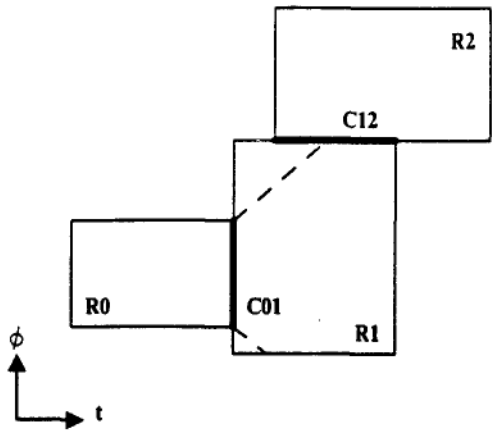

Fig. 24. The intersection region C_{12} may be only partially reachable by motions from the previous intersection region C_{01} if joint velocities are bounded from above.

that lies within this range. Since the planner does not consider explicit paths, but only paths implicit in regions, it is not possible to assign cost more accurately. The planner must select an actual sequence of regions by comparing both the minimum and maximum costs of different proposed sequences. Spatial costs are handled identically.

In fact, our planner does not first compute intersections regions that are too large, such as C_{12}, and then explicitly shrink them. Instead, the planner computes the shrunken version of C_{12} directly, as the set of all points that lie on the boundary of the region R_2 and are reachable from the previous intersection region C_{01}. Thus every point in C_{12} is reachable from some point in C_{01}. C_{01} is computed similarly, as are all regions in any partial sequence of regions from the start configuration. It follows that the minimum and maximum bounds on the time required for any trajectory to reach the region C_{12} from the start configuration are given precisely by the minimum and maximum time coordinates of points in C_{12}. In other words, since motions never move back in time, temporal costs alone may be determined directly from the time coordinates of points in a region. Spatial costs, however, must be computed as the sum of transition costs between pairs of regions.

Our implementation considers only temporal costs, in an attempt to find the minimum time solution. The algorithm employs a best-first search (see Winston [25]) based on the minimum time required to reach a given region plus the estimated minimum time to reach the goal from the given region. Once a sequence of regions from the start to the goal has been found, an actual path is computed backward from the goal. By construction, every point in any region in this sequence is reachable from some point in the predecessor region. Thus the method is guaranteed to select a valid path from the sequence of regions found by the planner.

6.8. Example. The beginning of the paper contained a sample problem involving

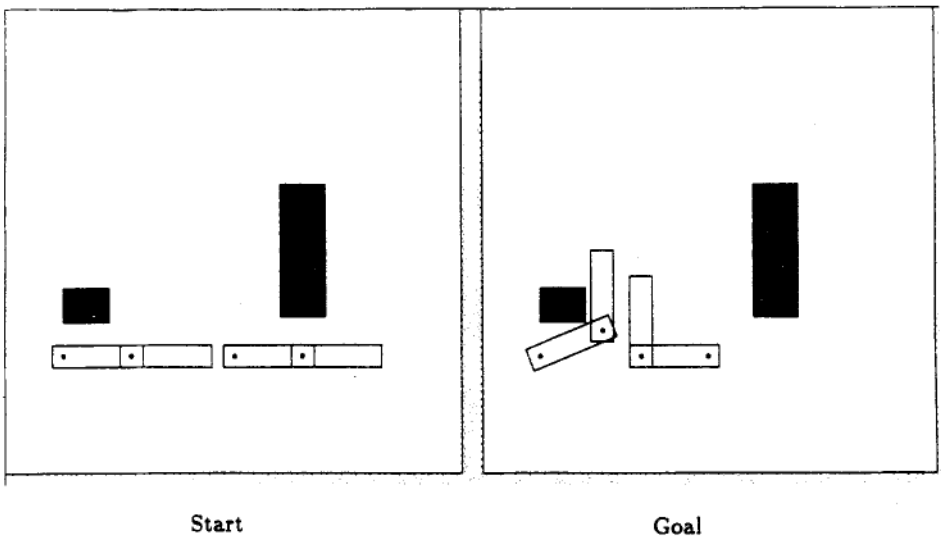

Fig. 25. Start and goal configurations for two arms. The motion of the left arm is planned first. The motion of the right arm is planned given the motion of the left arm.

three arms. In this section we consider a simpler problem involving two arms. The point of this example is to display the configuration space-time representing the constraints imposed on one of the two arms by the motion of the other arm and by the stationary obstacles in the environment.

Figure 25 shows the start and goal configurations of the two arms. The planner first planned the motion of the left arm, then that of the right arm. While planning the motion of the left arm, the planner considered the stationary obstacles, but ignored the right arm. Once the motion of the left arm had been determined, the planner computed the configuration space-time of the right arm, given the stationary obstacles, and the motion of the left arm. The planner then determined a path for the right arm that safely avoided the obstacles and the left arm. The time required by the planner was 76 seconds.

Figure 26 shows some snapshots of the resulting motions. Figure 27 displays the construction of the space-time slice representing the constraints imposed on Link 2 of the right arm at a fixed orientation of Link 1. The constraints were defined by the motion of the left arm and by the stationary obstacles. Notice that each frame contains a space-time obstacle that is constant over time. This obstacle represents the constraint imposed by the stationary rectangle in real space. The remaining space-time obstacle varies with time, reflecting the time-varying constraint imposed by the motion of the left arm.

Finally, Figure 28 displays several slices of the configuration space-time of the right arm. Each slice captures the constraints imposed on Link 2 of the right arm at a fixed orientation of Link 1. There are several observations worth noting in Figure 28. Consider the three slices corresponding to Link 1 orientations of 0, $\pi/4$, and $\pi/2$ radians. In the slice with $\theta_1 = 0$ there appears a single space-time obstacle that is constant over time. This obstacle represents the time-invariant

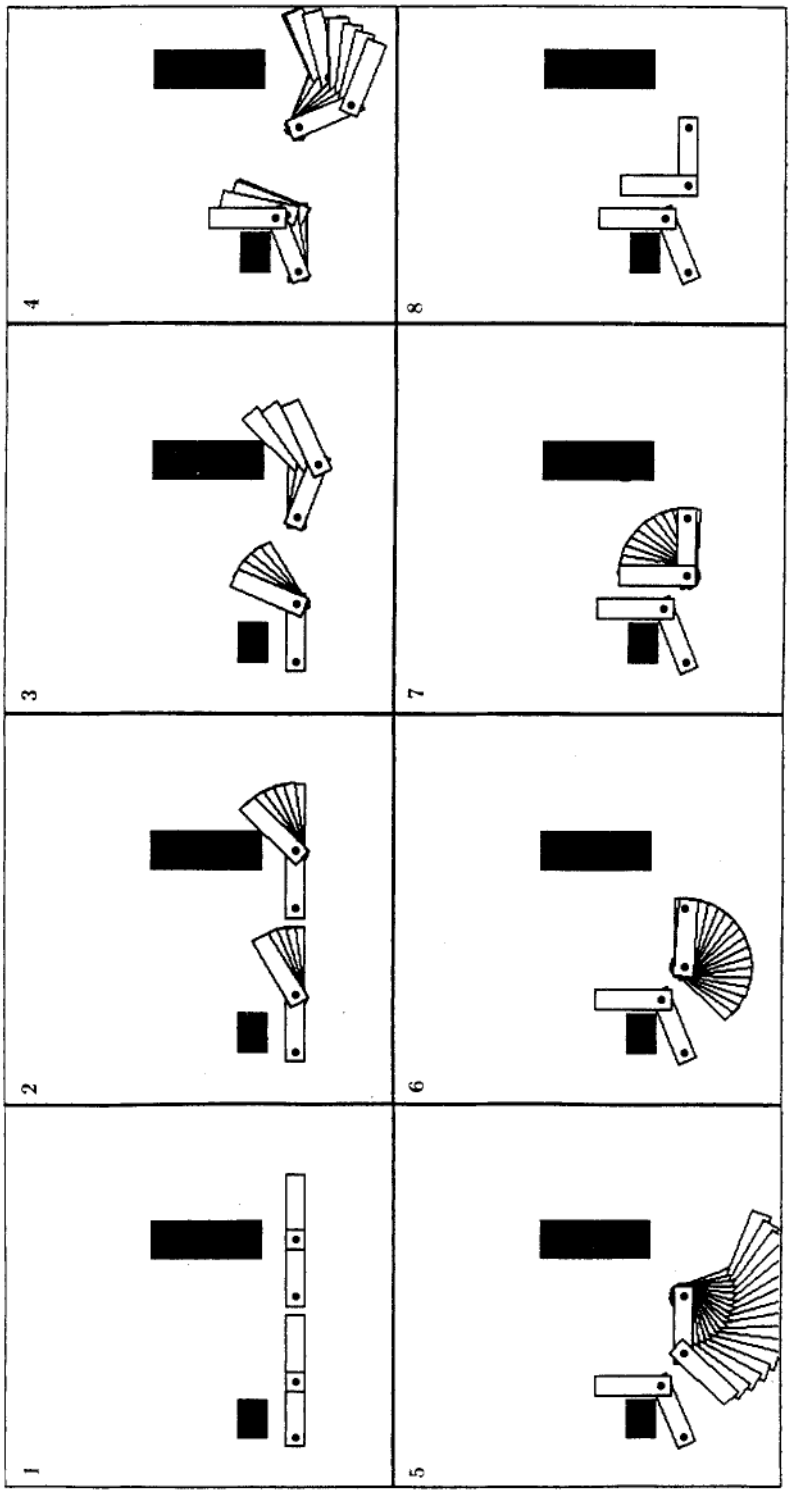

Fig. 26. Some sample snapshots of the motions determined by the planner while solving the problem of Figure 25.

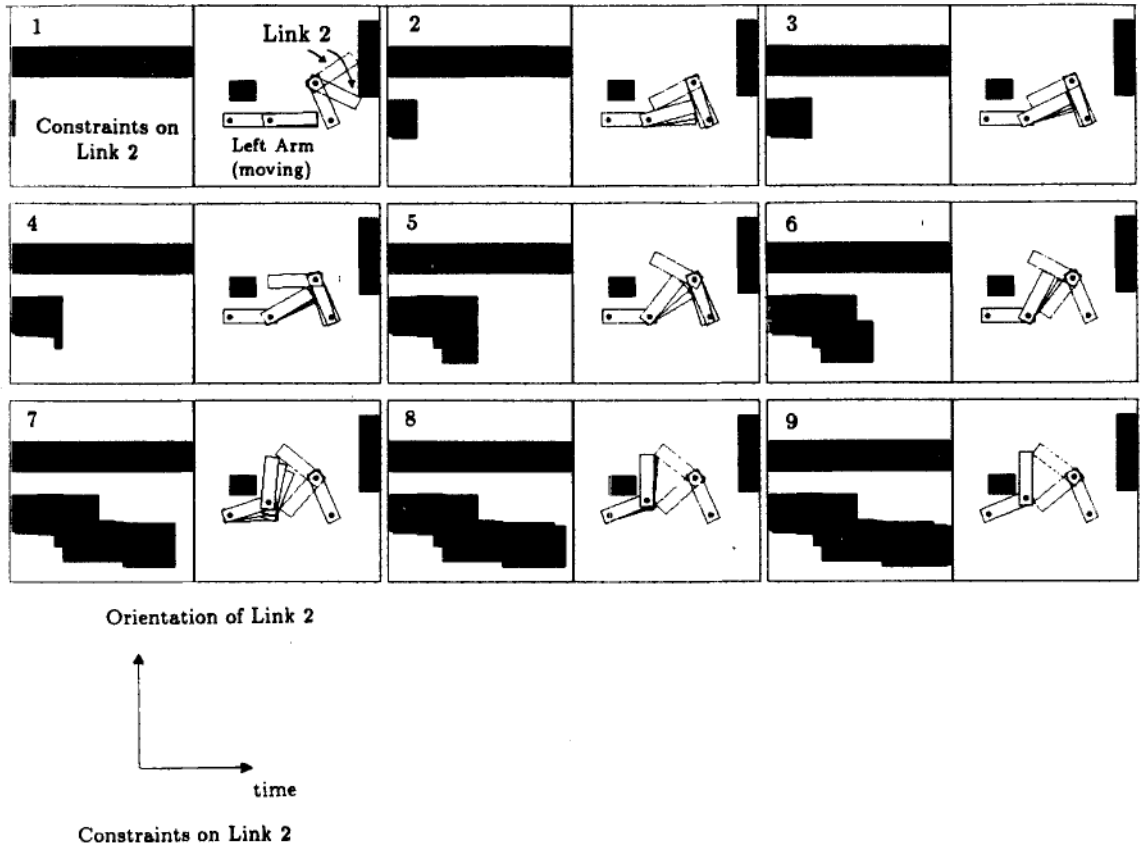

Fig. 27. This figure displays the construction of the constraints on Link 2 of the right arm for a fixed orientation of Link 1. In an alternating fashion, the figures display the constraints constructed thus far, and the motion of the left arm over the most recently constructed constraint rectangle. This right arm is displayed at the two extreme orientations of this constraint rectangle.

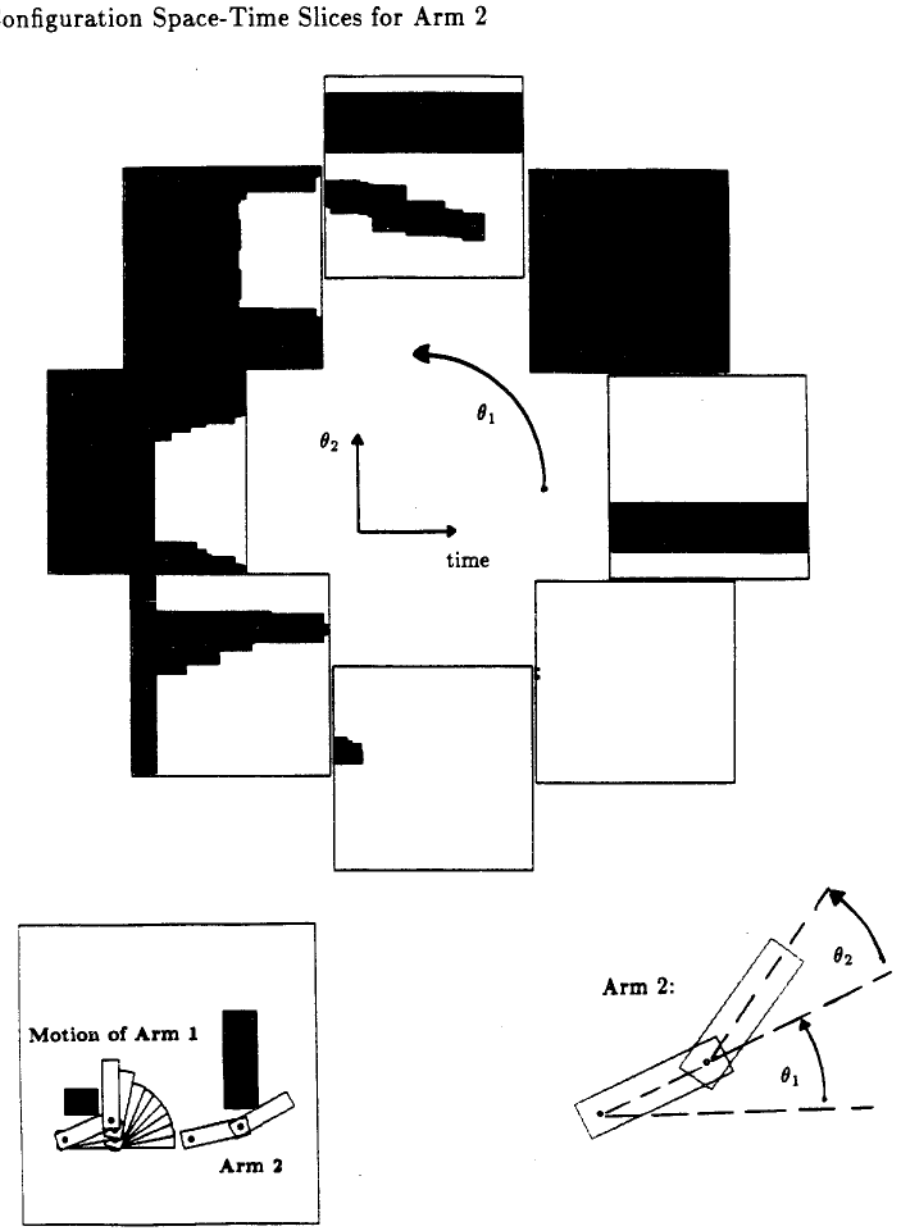

Fig. 28. Several slices of the configuration space-time representing the constraints imposed on the right arm by the motion of the left arm and by the stationary obstacles in the environment. Each slice depicts the constraints imposed on Link 2 for a fixed orientation of Link 1.

constraint imposed on Link 2 by the large stationary rectangular obstacle in real space. The motion of the left arm (Arm 1) imposes no constraint on Link 2 of the right arm at this orientation of Link 1.

The slice corresponding to $\theta_1 = \pi/4$ is completely darkened, meaning that all orientations of Link 2 are forbidden. This is because Link 1 of the right arm, oriented at $\theta_1 = \pi/4$, actually intersects the large rectangle. Thus there is no orientation of Link 2 for which the arm lies in free space.

Finally, in the slice for $\theta_1 = \pi/2$, there appears both a time-invariant space-time obstacle and a time-varying obstacle. The time-invariant obstacle arises, as before, as the constraint imposed by the large rectangle on Link 2, while the time-varying constraint arises as the constraint imposed by the motion of the left arm.

6.9. Complexity. Let n be the number of edges in the environment, let r be the number of space-time slices constructed, and let s be the number of time divisions over which other arms in the environment perform distinct motions. By this we mean that the time axis has been split into s intervals. The intervals are so chosen that over each interval each of the arms whose motions have already been determined performs at most a single uniform motion of one joint. Our planner constructs the configuration space-time constraints for a pair of interacting convex polygons using an algorithm of time complexity $O(rsn^3)$. A simple variant of the algorithm would require time complexity $O(rsn^2 \log n)$.

To see this, note that at most $O(n^2)$ constraints can arise from vertex-edge interactions in any given slice over any given time interval. *A priori* this suggests that there could be $O(n^4)$ constraint regions. However, while determining the forbidden regions of space-time, the planner only considers active constraints. Specifically, the planner considers sets of active constraints, one such set for each obstacle polygon in the environment. The planner sweeps a data-structure across space-time, sweeping in the time direction, while maintaining active orientation constraints in the data-structure, ordered by orientation. Whenever a constraint changes character, the planner updates this data-structure, removing expiring constraints while adding newly activated constraints. This amounts to tracing along the constraint boundaries, as indicated at the end of Section 6.2.

For any vertex-edge constraint, consider the conditions under which the constraint expires or becomes activated as a result of interacting with other edges or vertices on the two objects defining the constraint. By convexity, these conditions are determined by the two edges abutting the vertex and the two vertices bounding the edge. There are thus four possible conditions, namely, the two vertex-vertex contacts and the two edge-edge alignments, under which a constraint may expire or become activated as a consequence of object geometry.

Furthermore, for any constraint there are only a constant number of conditions under which the constraint changes character purely by virtue of the circle-line equations, such as reaching a local extremum in the α-β contour, or expiring because the edge or vertex rotates out of reach. (See Section 6.5.) The constant depends on the exact type of interaction, but is never more than six. This says that for each constraint the planner needs to consider no more than ten events at which the constraint changes character. Thus the planner constructs at most

$O(n^2)$ constraint regions per slice per time interval. Ordering in time the conditions under which active constraints change character requires time $O(n^2 \log n)$ per slice per time interval. Furthermore, by convexity of the objects, the sweep structure contains at most $O(n)$ constraints at any given time. Updating the sweep data-structure can thus require $O(\log n)$ time whenever an update must be made. Since there are r slices and s time intervals per slice, this gives the overall time complexity of $O(rsn^2 \log n)$.

In fact, our actual implementation requires $O(n)$ time to perform an update of the sweep structure, creating $O(n)$ constraint regions each time. Thus the time complexity is $O(rsn^3)$, and the number of regions constructed is also $O(rsn^3)$. Notice that the $O(n)$ constraint regions constructed at any update of the sweep data-structure all have the same endpoints along the time axis.

In order to search the configuration space-time, our planner requires a representation of free space-time. However, the output of the space-time construction phase is a collection of rectangles representing forbidden regions of space-time. In fact, there are $O(sn^3)$ rectangles per slice, arranged in $O(sn^2)$ columns of $O(n)$ rectangles each. The rectangles in a column all have the same temporal endpoints. By taking complements of the forbidden rectangles, the planner constructs $O(n)$ rectangles per column that represent free space-time. This construction has time complexity $O(n \log n)$. Thus the complete free space-time representation for a pair of interacting convex polygons consists of $O(rsn^3)$ rectangles, constructed in time $O(rsn^3 \log n)$.

The search of configuration space-time that we have implemented, assuming maximum velocity bounds, requires exponential time in the worst case. It is, however, possible to implement a polynomial-time search. To see this, observe that our planner's worst-case exponential search time is a direct consequence of the reachability conditions arising from maximum velocity bounds, as discussed in Section 6.7. Assuming maximum velocity bounds, the connectivity between free space regions depends on the order in which regions are traversed. This was the gist of Section 6.7. Only part of a free space-time region may be reachable from a previous region. This partial reachability introduces a branching factor in our search that results in an exponential algorithm.

As an alternative, suppose that the arms can perform motions requiring infinite velocities. In this case, the planner can represent the configuration space-time as a graph. The nodes of the graph correspond to the free space-time regions; the edges correspond to the connectivity between regions. If the graph contains v nodes and e edges, then the time complexity required to search the graph for some path from the start to the goal is $O(\max(v, e))$.

We could also search for the shortest path in the graph. Generally, a graph containing v nodes and e edges can be searched for the shortest path between two nodes using an algorithm of time complexity $O((v+e) \log v)$. In the case of dense graphs it is appropriate to use Dijkstra's algorithm, which has time complexity $O(v^2)$. In our case, v and e are themselves both $O(rskn^3)$, where k is the total number of convex polygons.

This same polynomial-time search algorithm may be used even if the arm has maximum velocity bounds. During the search, the planner assumes no velocity

bounds. Once the search has found a path from the start to the goal, the planner may have to slow down the motions of previous arms. Doing so scales time, thereby allowing the planner to satisfy the velocity constraints of the arm whose motions were just planned. Slowing down the other arms may also be done in polynomial time. We did not implement the planner in this form.

6.10. Summary for Two-Link Articulated Planar Arms

1. The constraints imposed on one rotating polygon by another rotating polygon are determined by tracing the orientations required to maintain contact between the two polygons.
2. The constraint contours change character at critical orientations. These include vertex-vertex contacts and edge-edge alignments.
3. Configuration space-time is represented as a series of space-time slices. Each slice represents the time-varying constraints imposed on Link 2 of the arm at a particular orientation of Link 1.
4. Configuration space-time is searched via connecting free space regions.

7. Summary.

This paper has explored the motion-planning problem for multiple moving objects. Two domains were considered. The first domain consisted of translating planar objects. The second domain consisted of rotating two-link planar articulated arms. The approach taken consisted of assigning priorities to each of the moving objects. Motions were planned for the objects in sequence as determined by the prioritization. Thus the problem was reduced to several versions of the problem of planning for a single moving object in the presence of other moving objects and stationary obstacles.

The problem of planning for a single moving object in the presence of other moving and stationary objects was solved by constructing a configuration space-time. The configuration space-time captured the constraints imposed on a moving object by its time-varying environment. A motion for the object was then found by searching for a path from the start to the goal configurations through the configuration space-time.

In the case of translating planar objects, it was noted that the configuration space-time could be constructed by initially treating all the objects as stationary and constructing a stationary configuration space. The complete configuration space-time could then be obtained by translating the configuration space obstacles in correspondence with the translations of the real space obstacles.

In the case of rotating arms, the configuration space-time was constructed as the union of the constraints resulting from the interactions of pairs of convex polygons rotating about various rotation centers. The constraints imposed on the orientation of one polygon by the motion of another rotating polygon were determined by explicitly examining the interactions of vertices and edges. For each possible type of vertex-edge interaction, the conditions under which contact could occur, and the conditions under which contact types could change, were analyzed. This analysis led to a representation that made explicit the constraint

contours in configuration space-time. Effectively, while constructing the configuration space-time, the planner would trace a particular vertex-edge contact until that contact encountered a contact change, such as a break or a vertex-vertex contact or edge-edge alignment. At the contact change, the representation allowed the planner to determine the new contacts that were possible. The planner would then examine these contacts, tracing the new constraint contours.

In the case of translating planar objects, the configuration space-time was represented as a series of two-dimensional spatial slices, corresponding to the constraints imposed on the moving object at different values of time. In the case of rotating planar arms, the configuration space-time was represented as a series of two-dimensional space-time slices, corresponding to the constraints imposed on the arm's second link at different orientations of the arm's first link.

Having constructed the configuration space-time, the planner would then search for a path from the start configuration to the goal. It was noted that such a search could never move backward in time. Furthermore, given maximal bounds on the objects' velocities, the planner was further required to observe slope restrictions on proposed paths.

Acknowledgments. We would like to thank Rod Brooks, John Canny, Bruce Donald, and Eric Grimson for reading drafts of this paper, and providing valuable comments.

References

[1] V. I. Arnold, *Mathematical Methods of Classical Mechanics*, Springer-Verlag, New York, 1978.

[2] T. Asano, T. Asano, L. Guibas, J. Hershberger, and H. Imai, Visibility of disjoint polygons, *Algorithmica.* **1** (1986), 49-63.

[3] R. A. Brooks, Solving the find-path problem by good representation of free space, *IEEE Trans. Systems Man Cybernet.*, **13** (1983), 190-197.

[4] R. A. Brooks and T. Lozano-Pérez, A subdivision algorithm in configuration space for findpath with rotation, *IEEE Trans. Systems Man Cybernet.*, **15** (1985), 224-233.

[5] C. E. Campbell and J. Y. S. Luh, A preliminary study on path planning of collision avoidance for mechanical manipulators, Technical Report-EE 80-48, School of Electrical Engineering, Purdue University, Lafayette, 1980.

[6] J. F. Canny, Collision detection for moving polyhedra, *IEEE Trans. Pattern Anal. Machine Intel.*, **8** (1986), 200-209.

[7] B. R. Donald, Motion planning with six degrees of freedom, AI-TR-791, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA, 1984.

[8] S. Fortune, G. Wilfong, and C. Yap, Coordinated motion of two robot arms, *Proceedings of the 1986 IEEE International Conference on Robotics and Automation*, San Francisco, 1986, pp. 1216-1223.

[9] E. Freund and H. Hoyer, On the on-line solution of the findpath problem in multi-robot systems, in *Robotics Research. The Third International Symposium* (O. Faugeras and G. Giralt, eds.), MIT Press, Cambridge, MA, 1986, pp. 253-262.

[10] J. E. Hopcroft, J. T. Schwartz, and M. Sharir, On the complexity of motion planning for multiple independent objects; *PSPACE*-hardness of the "warehouseman's problem," *Internat. J. Robotics Res.*, **3**(4) (1984), 76-88.

[11] J. E. Hopcroft, and G. T. Wilfong, Reducing multiple object motion planning to graph searching, Technical Report No. 84-616, Computer Science Department, Cornell University, Ithaca, NY.

[12] K. Kant and S. W. Zucker, Toward efficient trajectory planning: the path-velocity decomposition, *Internat. J. Robotics Res.*, **5** (1986), 72–89.

[13] T. Lozano-Pérez, Automatic planning of manipulator transfer movements, *IEEE Trans. Systems Man Cybernet.*, **11** (1981), 681–698. Reprinted in *Robot Motion* (M. Brady, *et al.*, eds.) MIT Press, Cambridge, MA, 1982.

[14] T. Lozano-Pérez, Spatial planning: A configuration space approach, *IEEE Trans. Comput.*, **32** (1983), 108–120.

[15] T. Lozano-Pérez and M. Wesley, An algorithm for planning collision-free paths among polyhedral obstacles, *Comm. ACM*, **22** (1979), 560–570.

[16] J. Nievergelt, and F. Preparata, Plane-sweep algorithms for intersecting geometric figures, *Comm. ACM*, **25** (1982), 739–747.

[17] G. Ramanathan and V. S. Alagar, Algorithmic motion planning in robotics: coordinated motion of several disks amidst polygonal obstacles, *Proceedings of the 1985 IEEE International Conference on Robotics and Automation*, St. Louis, 1985, pp. 514–522.

[18] J. Reif and M. Sharir, Motion planning in the presence of moving obstacles, *Proceedings of the 26th IEEE Symposium on the Foundations of Computer Science*, Portland, OR, 1985, pp. 144–154.

[19] J. T. Schwartz and M. Sharir, On the piano movers' problem: II. General techniques for computing topological properties of real algebraic manifolds, Technical Report No. 41, Computer Science Department, Courant Institute of Mathematical Sciences, New York University, New York, 1982.

[20] J. T. Schwartz and M. Sharir, On the piano movers' problem: III. Coordinating the motion of several independent bodies: the special case of circular bodies amidst polygonal barriers, *Internat. J. Robotics Res.*, **2** (1983), 46–75.

[21] M. Sharir and A. Schorr, On shortest paths in polyhedral spaces, *Proceedings of the 16th Annual ACM Symposium on Theory of Computing*, Washington, 1984, pp. 144–153.

[22] P. Spirakis and C. Yap, Strong NP-hardness of moving many discs, *Inform. Process. Lett.*, **19**, (1984), 55–59.

[23] P. Tournassoud, A strategy for obstacle avoidance and its application to multi-robot systems, *Proceedings of the 1986 IEEE International Conference on Robotics and Automation*, San Francisco, 1986, pp. 1224–1229.

[24] S. M. Udupa, Collision detection and avoidance in computer controlled manipulators, Ph.D. Thesis, Department of Electrical Engineering, California Institute of Technology, 1977.

[25] P. H. Winston, *Artificial Intelligence*, 2nd edn., Addison-Wesley, Reading, MA, 1984.

[26] C. K. Yap, Coordinating the motion of several discs, Technical Report No. 105, Computer Science Department, Courant Institute of Mathematical Sciences, New York University, New York, 1984.