Real-Time Path Planning Procedure for a Whole-Sensitive Robot Arm Manipulator

Edward Cheung and Vladimir Lumelsky

Report No. 8913

November 1989

Real-Time Path Planning Procedure for a Whole-Sensitive Robot Arm Manipulator*

Edward Cheung and Vladimir Lumelsky
Yale University, Department of Electrical Engineering
New Haven, Connecticut 06520

Abstract

We consider the problem of sensor-based motion planning for a three-dimensional robot arm manipulator operating among unknown obstacles. When every point of the robot body is subject to potential collision. The corresponding planning system must include these four basic components: sensor hardware; real-time signal/sensory data processing hardware/software; a local step planning subsystem that works at the basic sample rate of the arm; and finally, a subsystem for global planning. The arm sensor system developed at Yale University presents a proximity sensitive skin that covers the whole body of the arm and consists of an array of discrete active infrared sensors that detect obstacles by processing reflected light. The sensor data then undergoes low level processing via a step planning procedure, which converts sensor information into local normals at the contact points in the configuration space of the robot. This paper presents preliminary results on the fourth component, a real-time algorithm that realizes the upper, global level of planning. Based on the current collection of local normals, the algorithm generates preferable directions of motion around obstacles, so as to guarantee reaching the target position if it is reachable. Experimental results from testing the developed system are also discussed.

1. Introduction

We consider an extension of the problem of sensor-based motion planning to collision-free motion of three-dimensional (3d) robot arm manipulators operating in workspace with unknown obstacles. Uses for robot arms equipped with such capabilities include applications for space exploration [2], work in hazardous, hostile or unhealthy environments, and unstructured factory work cells.

The first obvious requirement in such systems is the capability to sense obstacles in the arm's environment. Assuming that, in general, every point of the arm body is subject to potential collisions, one approach is to cover the surface of the whole arm with an array of proximity sensors. Obstacles appearing from any direction can then be detected, and appropriate action taken to avoid them. Such a sensor system based on the infrared light sensitive skin, together with local sensor interpretation algorithms, has been developed at Yale University [1]. The last element to be added to complete the motion planning system is an

* Supported in part by the National Science Foundation Grants DMC-8712357 and IRC-8805943.

algorithm for transforming the preprocessed sensor data into global planning decisions that would guide the arm to its target position. A version of such an algorithm is presented in this paper.

The objective of the arm is to move from the starting (S) to the target configuration (T). If T is not reachable at all from S, the algorithm is to conclude so in finite time. No prior knowledge of the robot's environment is supplied, thus obstacles must be detected using sensors. The data supplied by the sensitive skin is not directly used by the motion planning algorithm but first undergoes low level processing. This local procedure calculates the local normals at the contact points and maps them into the configuration space *(C-space)* of the robot arm. The procedure generates local motion that amounts to the arm sliding about the contact point in the workspace. Unlike systems where the arm comes in contact with the environment [3,4], no contact with the obstacles occurs in our system.

Below, the robot arm and sensitive skin hardware are briefly discussed in Section 2, followed by the analysis of interaction between the arm and obstacles in Section 3. The developed procedure for global motion planning is described in Section 4. Section 5 discusses results of our experiments with the system.

2. Sensor skin and robot arm

A sketch of the arm and the sensor skin is shown in Figure 1. The motion planning system operates on the major linkage of the arm which includes the first three links, l_1, l_2, l_3 , and corresponding joints, j_1, j_2, j_3 . Since joints j_1 and j_2 coincide in the arm which was chosen for this work (General Electric Industrial Arm Manipulator, Model P5), the first link, l_1, is of length zero. The physical appearance of the sensitive skin, without sensors and before it is installed onto the arm, is shown in Figure 2. The arm manipulator with the fully installed skin is shown in Figure 3.

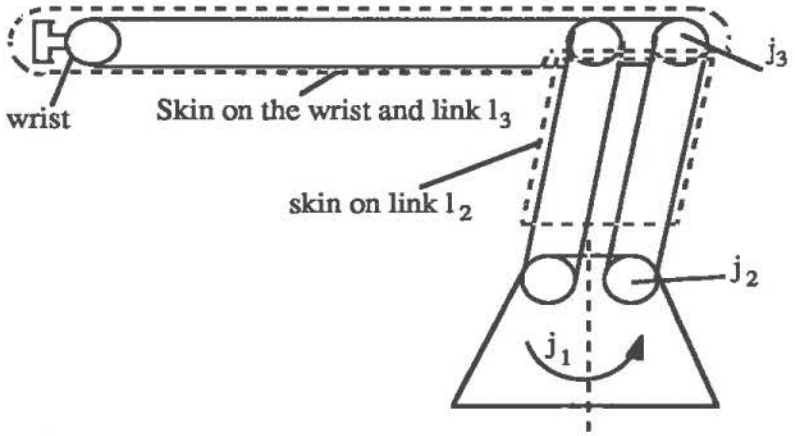


Figure 1. Sketch of the robot arm and the sensitive skin.

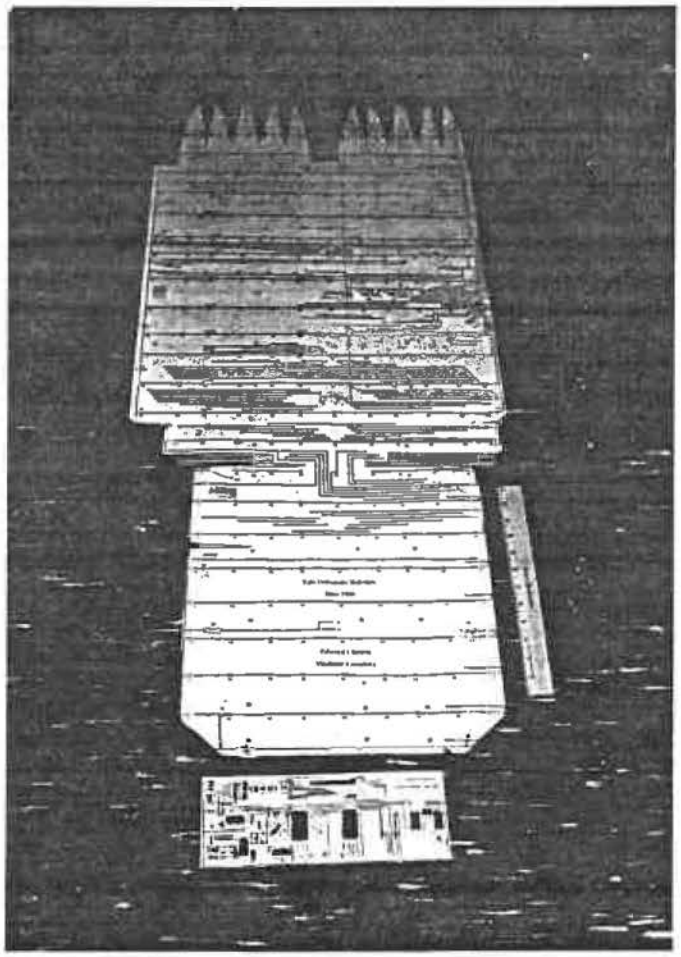


Figure 2. Sensitive skin before installation onto the robot arm (the picture has been taken at an angle about 30° to the surface of the table on which the skin lies). The circuit in the front is one of the sensor detection modules. The shape of the part of the skin furthest from the viewer reflects its future position around the wrist of the arm, see Figure 3.

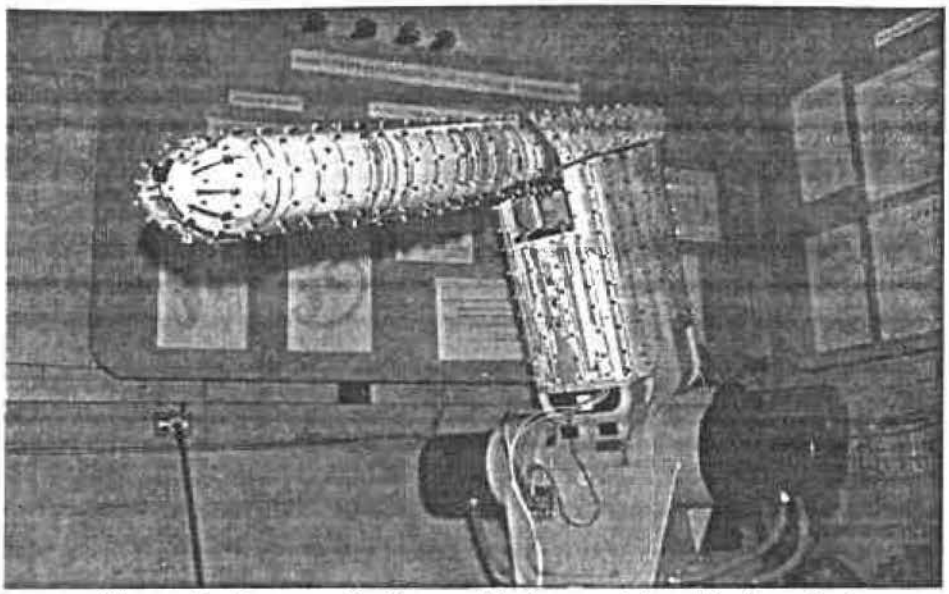


Figure 3. Arm manipulator with the sensitive skin installed.

The sensitive skin consists of an array of discrete infrared sensor pairs mounted on a flexible printed circuit board that is wrapped around the robot arm. The sensor skin covers the entire arm, including the joints, so that obstacles approaching from any direction will be detected. The sensors' output voltage is proportional to the amount of reflected light. This analog signal is used for extracting the proximity data, and is converted to local normal information by the step planning procedure described in [1]. The procedure calculates the local normals at the contact point to the obstacle in the C-space. The main purpose of the procedure is to determine which incremental local motions of the arm can be considered safe in the vicinity of the obstacles that are obstructing the arm.

This local normal information allows the robot to move away from an approaching obstacle by traveling in the direction of its local normal, or can be used for contour following of the obstacle by moving in a direction locally perpendicular to a chosen normal. After the completion of a small unit step motion that is executed within the basic sample rate of the arm (20-30 steps per second), a new set of local normals is calculated, and the procedure is repeated. The number of considered local normals is determined by the number of points of simultaneous contact between the arm and obstacles.

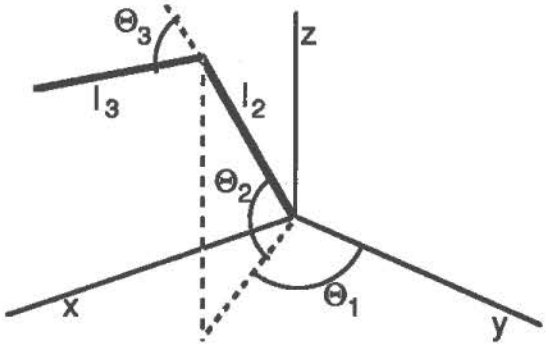


Figure 4. Sequence of arm joints and links.

Equations for the local normals depend on the type of the arm, and are different for different kinematic configurations. The expressions for the local normal presented in [1] were evaluated for an open kinematic chain consisting of three links (l_1, l_2, l_3) and three joints (j_1, j_2, j_3.) of revolute type (RRR type arm); a sketch of the arm configuration is shown in Figure 4, with the joint variables denoted (Θ_1,Θ_2,Θ_3). The same kinematic configuration is considered in this paper. Since only the first three degrees of freedom of the robot are used for active control, the motion of the robot can be represented by a point automaton that operates in the three-dimensional C-space formed by its joint variables.

3. Analysis of arm - obstacle interaction

If the point automaton were operating in a planar (2d) C-space, a number of maze searching algorithms could be used to solve the motion planning problem. These algorithms rely on the fact that there are only two directions for the point automaton to move around an obstacle - clockwise or counterclockwise.

The 3d algorithm described below makes use of a procedure for moving a point automaton in the plane, called *Bug2* [5]. Briefly stated, under this algorithm the point automaton first moves along the straight line, called *M-line* (for "main line"), which connects the start and target positions. If an obstacle is encountered, a *Hit Point* is defined at the encounter point, the point automaton turns in the prespecified *local direction*, left or right, and begins following the contour of the obstacle. This continues until the M-line is again met at a distance closer to the target than the lastly defined Hit Point; here, distance is defined as the Euclidean distance. This encounter point is then defined as a *Leave Point* and the point automaton continues its motion towards the target. The algorithm will reach the target if a path exists, or conclude that no such path exists if this is the case. This algorithm was successfully used as the overall motion planning algorithm in the 2d experimental system described in [6].

The difficulty with extending such algorithms to the three-dimensional case is that, in general, there is an infinite number of ways for the point automaton to maneuver around a 3d obstacle. The technique presented below is similar to that used in [7] in that it exploits the natural constraints imposed by the kinematics of the arm in order to narrow down the available choices during the arm navigation of the point automaton.

3.1. Obstacle types

Depending on their location with respect to the arm links, obstacles can be grouped into three major categories easily recognized by the sensor system: *Type I*, *Type II*, and *Type III*. Referring to Figure 5, obstacles of Type II obstruct link l_2, and obstacles of Type III obstruct link l_3. Since in the considered arm the first two joints coincide and so link l_1 is absent, Type I obstacles never occur. Because the location of every sensor is known to the sensor controlling computer, the type of the obstacles obstructing the arm can be easily identified through the use of a look-up table. Note that in principle the same obstacle can be of different types depending on the link it interacts with. Also, the arm has no way of distinguishing between two obstacles of the same type, and so, from the arm standpoint, it never interacts with more than two obstacles -- one of Type II and the other of Type III.

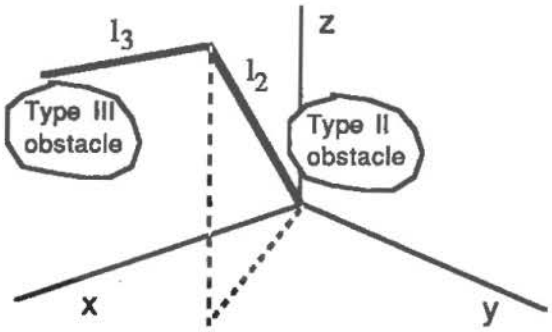


Figure 5. Obstacle types

3.2. Properties of C-space obstacles

Referring to Figure 5, observe that a Type II obstacle will, by definition, never obstruct link l_3, thus allowing joint j_3 to move freely. In terms of the robot three-dimensional C-space, if the point automaton is obstructed by a Type II obstacle, it will still be able to move freely in the Θ_3 direction. This means that Type II obstacles in the C-space have a shape of a *generalized cylinder* whose axis is parallel to the Θ_3 axis, refer to Figure 7. A generalized cylinder has the property that the crossection in the plane perpendicular to its axis is a simple closed curve that does not change along the axis. Therefore, when the arm is obstructed by a Type II obstacle, the motion of the point automaton in the Θ_3 direction is of no use as far as obstacle avoidance is concerned. The topology of a Type III obstacle exhibits no such distinct characteristics, and in the worst case the arm may be forced to carry out an exhaustive search of the obstacle surface in order to find a path to T.

3.3. Effect of joint range limits

Because in real arm manipulators the range of motion of each joint is typically limited, it is desirable to incorporate the joint limits into the obstacle model. It is easy to show [7] that in terms of their effect on motion planning and obstacle avoidance the joint limits produce in the C-space "obstacles" fully equivalent to images of physical obstacles. For example, the upper limit of joint Θ_3 produces a horizontal plane parallel to the plane of axes Θ_1 and Θ_2. This plane will behave like an obstacle limiting the C-space from above. Based on this observation, the joint limits are simply treated by the motion planning algorithm as artificial "obstacles" of the respective types -- that is, of Type II in case of joint j_1 and joint j_2, and Type III in case of joint j_3.

4. Path planning algorithm

We intend to use the basic idea of a planar algorithm -- clockwise or counterclockwise maneuvering around obstacles -- for 3d motion planning, and construct the 3d trajectory of the point automaton in 3d C-space as a combination of path segments along 2d surfaces. We start by confining the motion of the point automaton in free space - that is, when no interaction with obstacles takes place - to the *M-line* (for "main line") which is the straight line connecting start and target points S and T. Then, if the motion along the M-line becomes impossible because of interfering obstacles, the automaton will attempt to maneuver around the obstacles while confining its motion to an appropriately chosen plane. The choice of the plane has to satisfy two requirements: it has to guarantee convergence and result in reasonable (from the standpoint of the length of generated paths) trajectories.

This plane, called *P-plane* (for "preferred"), should obviously contain the M-line. Otherwise, in principle the plane can be chosen in a variety of ways. A consideration that guided our choice was the desire to minimize, on the average, motion in the Θ_3 direction. The reason for this is a special topology of C-space obstacles, as indicated above: if the point automaton encounters a Type II obstacle, motion in the Θ_3 direction will be of no use as far as obstacle avoidance is concerned. The plane in C-space that satisfies these requirements is defined by the M-line and the line t that is perpendicular to both the Θ_3 axis and the M-line. This plane always exists and is unique, see Figure 6, except in the degenerate cases when the M-line intersects or coincides with the Θ_3 axis. In the former case, we can still find the direction of the line t by taking the cross product of the Θ_3 axis and M-line; in the latter case, the choice of t is arbitrary as long as t does not coincide with the M-line.

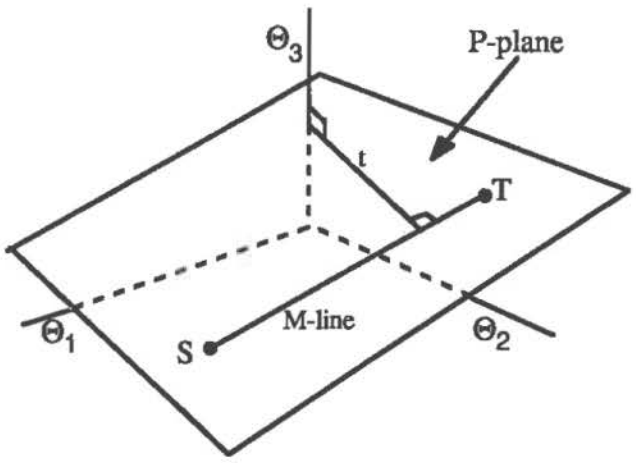


Figure 6. The P-plane

Similar to the Bug2 algorithm, a *local direction* (left or right) is defined for the P-plane. Given the fact that the notion of left or right can change depending on the direction from which one looks at the P-plane, any definition is appropriate as long as it defines the local direction in a unique way. Distances are defined as Euclidean distances along the M-line.

In a more complex case when no path to the target can found in the P-plane, the point automaton will be forced to move off the P-plane until it either reaches the target, concludes that the target cannot be reached, or finds a previously unexplored region of the P-plane. One consequence of having only two types of obstacles is that their intersection curves form closed curves with no branches. This fortunate property guarantees that the automaton will easily recognize some characteristic points (e.g., intersections of the said intersections with the P-plane) and will never fall into an infinite loop. The intersection curves thus lie in (2d) surfaces of C-space obstacles and are in general non-planar; they form potential path segments that may connect regions of the P-plane that cannot be reached one from another within the P-plane. If, while moving along the P-plane, the point automation encounters an intersection curve between a Type II and Type III obstacle, it may simply follow this intersection curve until it spots another region of the P-plane.

The path planning algorithm consists of three phases. In *Phase 1*, the point automaton moves within the P-plane. If Phase 1 terminates and the algorithm cannot conclude whether T is reachable, *Phase 2* or *Phase 3* is used to continue the search. The purpose of Phases 2 and 3 is to guide the automaton off the P-plane. During Phase 2, an intersection curve between two obstacles found during the Phase 1 is followed in the search of T or an unexplored region of the P-plane. If such a region is found, Phase 1 is used again and the process repeats. Otherwise, Phase 3 is invoked to carry out an exhaustive search of the surface of the current obstacle. This process can result in reaching T or in finding another unexplored region of the P-plane, and the process repeats. Depending on the task and environment, the order of the actual phase execution may change. Furthermore, a given phase may or may not appear during the motion. Only Phases 1 and 2 are discussed in detail in this paper.

When, while moving along the M-line, the automaton encounters an obstacle, the point of encounter is defined as a *Hit point*. When, while maneuvering around obstacles, the automaton encounters the M-line at a point that is closer to T than the lastly defined Hit point, a *Leave point* is defined.

While following obstacles, whenever the automaton makes a transition from a Type II obstacle to a Type III obstacle, or reverse, the location where this occurs is noted. These locations, together with points S, T, and Hit and Leave points, form *nodes* of a continuously growing *connectivity graph*. The *edges* of the graph are segments of the M-line, segments of the intersection curves between the P-plane and obstacles, and segments of intersection curves between obstacles of different types.

Note that each of the nodes S and T has one adjacent edge; each node formed by a Hit or

Leave point has three adjacent nodes -- a segment of the M-line and two adjoining segments of the obstacle/P-plane intersection curve; and each of the remaining nodes has four adjacent edges formed by the obstacle/P-plane intersection curve, on the one hand, and the intersection curve between two obstacles, on the other hand. Since the size of the graph is proportional to the number of obstacles in the workspace, it is relatively small -- in an experimental setup, for example, it would be rather difficult to generate more than 10-15 nodes. The part of the graph that has been already explored is stored and used during Phase 2 of the algorithm if Phase 1 fails to complete the task. Unless the target T is spotted, the whole graph may have to be explored during the search.

The next subsections contain a more detailed description of Phases 1 and 2, followed by a summary of the path planning algorithm in Section 4.4.

4.1 Phase 1 of path planning

During Phase 1, the point automaton moves along the P-plane. This includes two types of motion - from S towards T along the M-line, or along the intersection curve between an obstacle and the P-plane. The motion planning procedure first completes processing of the P-plane region that is being currently explored. Only after it becomes clear that the task cannot be completed within this region, the automaton will leave the P-plane by invoking another phase. Algorithmically, the procedure for planning a Phase 1 motion is analogous to the Bug2 algorithm mentioned before.

If during Phase 1 the lastly defined Hit Point is encountered again, before the next Leave Point is defined, the Phase 1 procedure concludes that no path to T can be found within the current P-plane region (for proof, see [5]). This means that the obstacle the automaton is following may be dividing the P-plane into disconnected regions, and so the only option left is to look for another connected region of P-plane. This can be done only by leaving the P-plane, via another phase of the procedure.

If the task has not been completed during the exploration of the current P-plane region and all the encountered obstacles were found to be of Type II, we can already conclude that the target T is unreachable. This is because Type II obstacles are generalized cylinders whose axes are parallel to the Θ_3 axis, forming unsurmountable walls in the Θ_3 direction. Therefore, if the task cannot be completed within a P-plane region, formed by Type II obstacles, no other regions of the P-plane can be reached; see Figure 7.

If during the exploration of the current P-plane region the automaton identifies a Type III obstacle, the point of encounter, besides becoming a new node of the connectivity graph, becomes the starting point for the next path segment which lies outside of the P-plane. To follow this direction, Phase 2 of the procedure is invoked.

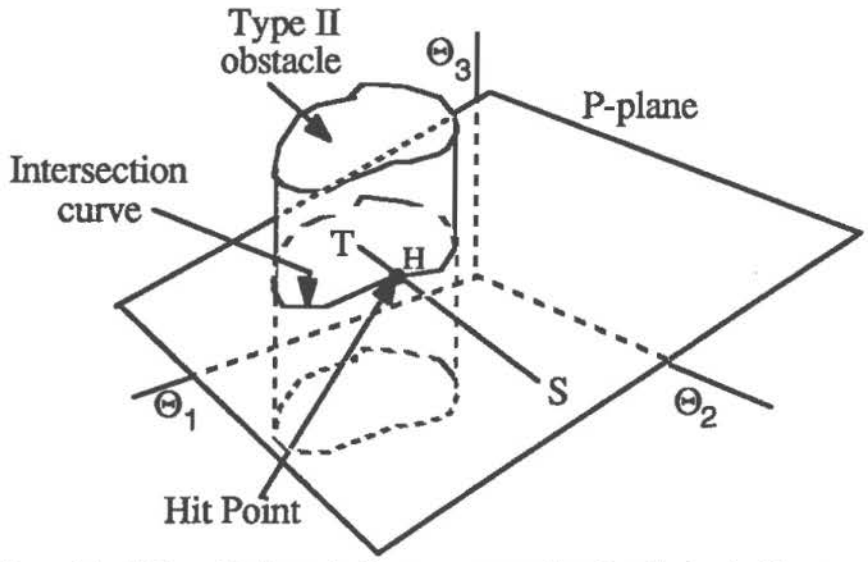


Figure 7. A Type II obstacle forms a generalized cylinder in C-space.

4.2. Phase 2 of path planning

Phase 2 concentrates on generating those motions outside the P-plane that are short of an exhaustive search. As mentioned above, the intersection curve between two obstacles, one of Type II and the other of Type III, can easily be distinguished by the arm's sensors. In general, this curve presents a closed curve with no branches. Unless the Type III obstacle is locally tangential to the P-plane, the seam will have an even number of intersections with the P-plane. If followed by the point automaton, the seam may lead to a yet unexplored region of the P-plane. To find out whether or not the newly encountered region of the P-plane has already been explored, the procedure simply checks whether the intersection point between the seam and the P-plane region is already in the graph.

Every node in the graph is connected to two edges that lie in the P-plane, and two edges that belong to the intersection curve between a Type II and Type III obstacle. The two former edges are known, since they are part of the path that the point automaton traveled during Phase 1 when the node was found. The two latter edges are explored during Phase 2.

At the start of Phase 2 the automaton is at the lastly defined Hit Point. If the current node has unexplored edges they are explored first. Otherwise, the search then proceeds to any known node with an unexplored edge that is also on the same connected region of the P-plane as the point automaton. Failing this, the search expands to any other known node in the graph that has an unexplored edge. If there are no nodes with unexplored edges, Phase 2 terminates. Once a node is found with an unexplored edge, a path to that node using the already explored

edges is found using a standard depth-first graph search technique [8]. Phase 2 terminates as soon as T is found, or after all the identified nodes on the graph have been explored.

4.3 Motion Planning algorithm: Summary

Initiation: Define the M-line and P-plane. Go to Phase 1.

Phase 1 (Motion along the P-plane).

Step 1. Move toward T, checking for these conditions:

1. Arrive at T. The procedure terminates.
2. Meet an obstacle. Define a Hit Point; turn in the local direction; go to Step 2.

Step 2. A. Follow the obstacle boundary in P-plane, checking for one of these conditions:

1. Arrive at T. The procedure terminates.
2. Encounter the M-line. If the distance between the current position and T is shorter than that between the lastly defined Hit Point and T, define a Leave Point and go to Step 1; otherwise, continue - go to A.
3. Encounter an intersection curve between a Type II and a Type III obstacle. Mark this location as a node in the graph; continue - go to A.
4. Encounter the lastly defined Hit Point before the next Leave Point is defined. If the current obstacle is of Type II, no path to T exists; the procedure terminates. Otherwise, go to Phase 2.

Phase 2 (Motion along the intersection curve between a Type II and Type III obstacles).

Step 1. Identify a graph node with a yet unexplored edge in the current P-plane region or, if none are found, in another P-plane region. If none are found, the procedure terminates. Otherwise, move to the node; go to Step 2.

Step 2. Move along the unexplored edge until P-plane is encountered. If this point is already in the connectivity graph, go to Step 1. Otherwise, enter the point as a node into the graph; go to Phase 1.

5. Experimental results

The purpose of these experiments was to establish real-time feasibility of the developed approach and to assess the quality of the arm motion, especially during its maneuvering around obstacles. Two experiments are described; in both, the work space contains a single obstacle of irregular shape unknown to the arm manipulator and such that the whole motion, though involving all three joints of the arm major linkage, takes place in the P-plane. No additional information about the obstacles was given to the arm beforehand, and no information collected in one experiment was used in the other experiment. The generated paths were documented by photographing some of the arm positions during the motion, and also via the C-space graphics presentation of the paths on a MicroVax work station which was included in the system for the experiment monitoring and documenting purposes.

On the total, the generated motion has been quite acceptable. No collisions with the obstacle took place; the minimum distance between the arm and the obstacle varied within the range 1 to 3 inches. The arm was able to maintain a good velocity typical for such industrial manipulators, in spite of tight time constraints; polling of all 500 sensors of the skin is done within the normal sample rate of the arm; also, at some positions the planning system had to base its decisions on up to 10 local normals -- that is, up to 10 sensors along the skin simultaneously sensed the obstacle.

In the first experiment, the arm is requested to move from its starting position, S, to a given target position, T. In the second experiment, the arm is requested to move from position T back to position S. Taken together, the experiments allow one to assess the shape of the obstacle image in C-space and the quality of the decision-making and motion execution during the arm maneuvering around the obstacle and around the perceived "obstacles" generated by the arm joint limits.

Experiment 1. The sequence of photographs in Figure 9 present some of the arm positions along its path. Figure 10 presents the screen dump of the corresponding images of the motion in C-space generated at the MicroVax work station. The three trajectories in Figure 10 show respectively the actual trajectory in the P-plane and the projections of this trajectory onto the planes (Θ_1, Θ_2) and (Θ_2, Θ_3); the arm positions a,b,c,... refer to the corresponding photos of Figure 9. The robot encounters the obstacle and defines the Hit Point at the position shown in Figure 9c, then maneuvers around the obstacle (positions 9c to 9g), meets the M-line and defines the Leave Point at position 9h, and then proceeds to the target position T.

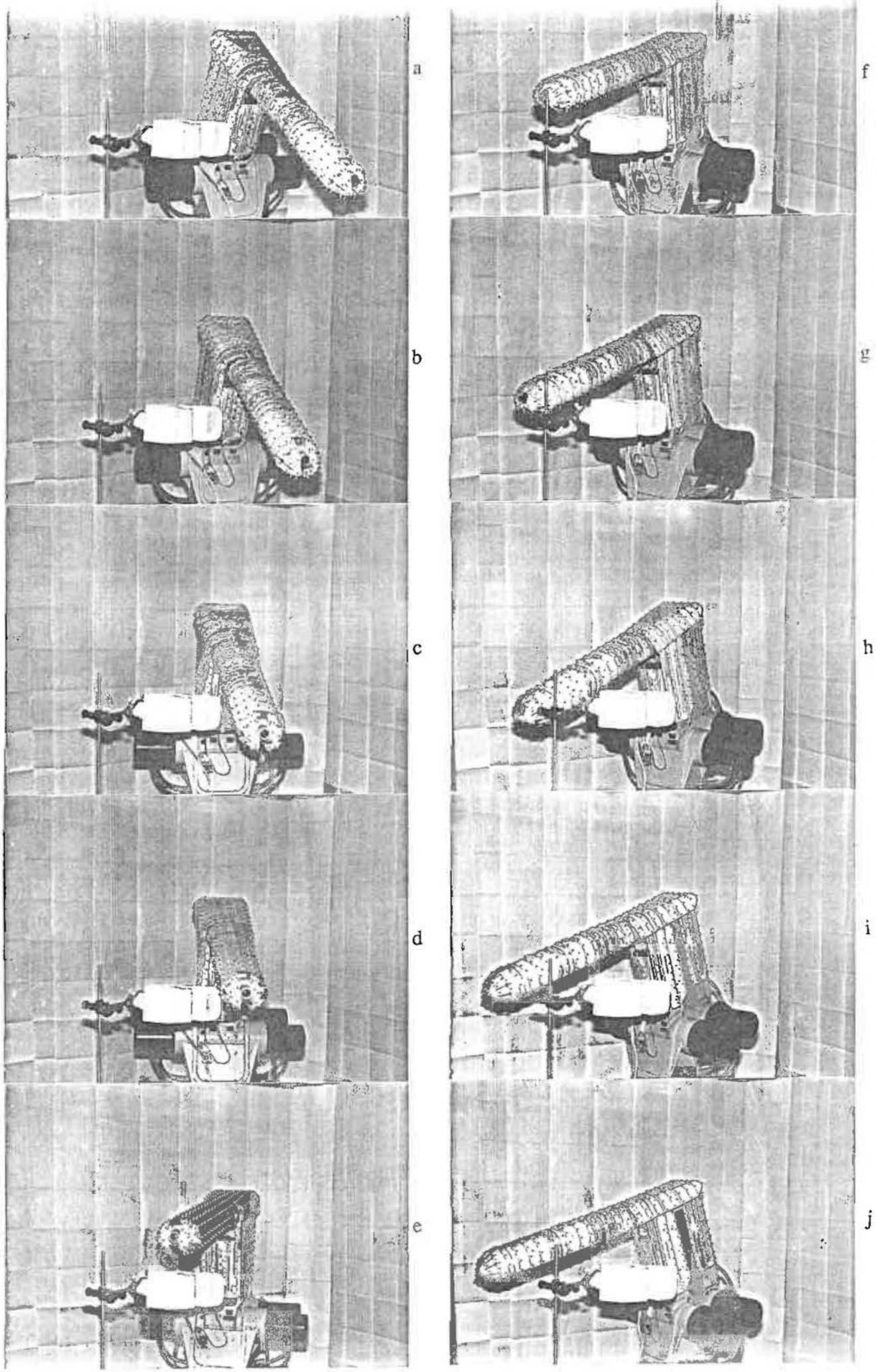


Figure 9.
Experiment 1.

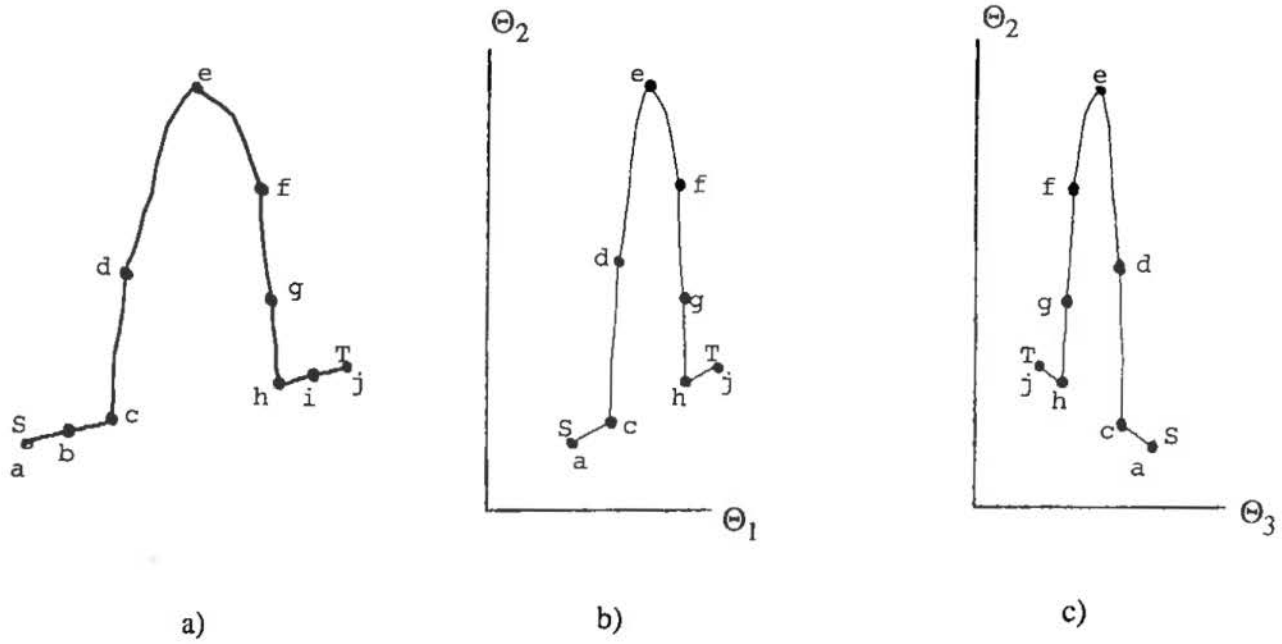


Figure 10: Experiment 1. C-space image of the arm path shown in Figure 9: (a) the path in P-plane; (b) projection of the path onto the reference plane (Θ_1, Θ_2); (c) projection of the path onto the reference plane (Θ_2, Θ_3). The segment (c,d,...,h) corresponds to the arm maneuvering around the obstacle.

Experiment 2. The motion here is from T towards S. The set of photos in Figure 11 present some of the arm positions along the path. Figure 12 presents the MicroVax screen dump of the corresponding images in C-space. The first contact with the obstacle takes place at position 11c, where a Hit Point is defined. Then, at positions 11c to 11e the arm maneuvers around the obstacle by moving along the intersection between P-plane and the obstacle; note that, unlike the Experiment 1, the arm attempts here to pass the obstacle from below. During this attempt, at position 11e the arm hits one of the arm joint limits and, following the algorithm, takes a long journey (positions 11e to 11m). At 11m, the arm encounters again the obstacle and starts moving along the intersection curve between P-plane and the obstacle (positions 11m to 11o). At 11o, the M-line is met again, a Leave Point is defined, the arm proceeds along the M-line toward T (positions 11o and 11p), and completes the task.

Figure 13 shows the combination of both paths, from Experiments 1 and 2, in P-plane -- note that the sum of the two gives the complete representation of the intersection of C-space with P-plane.

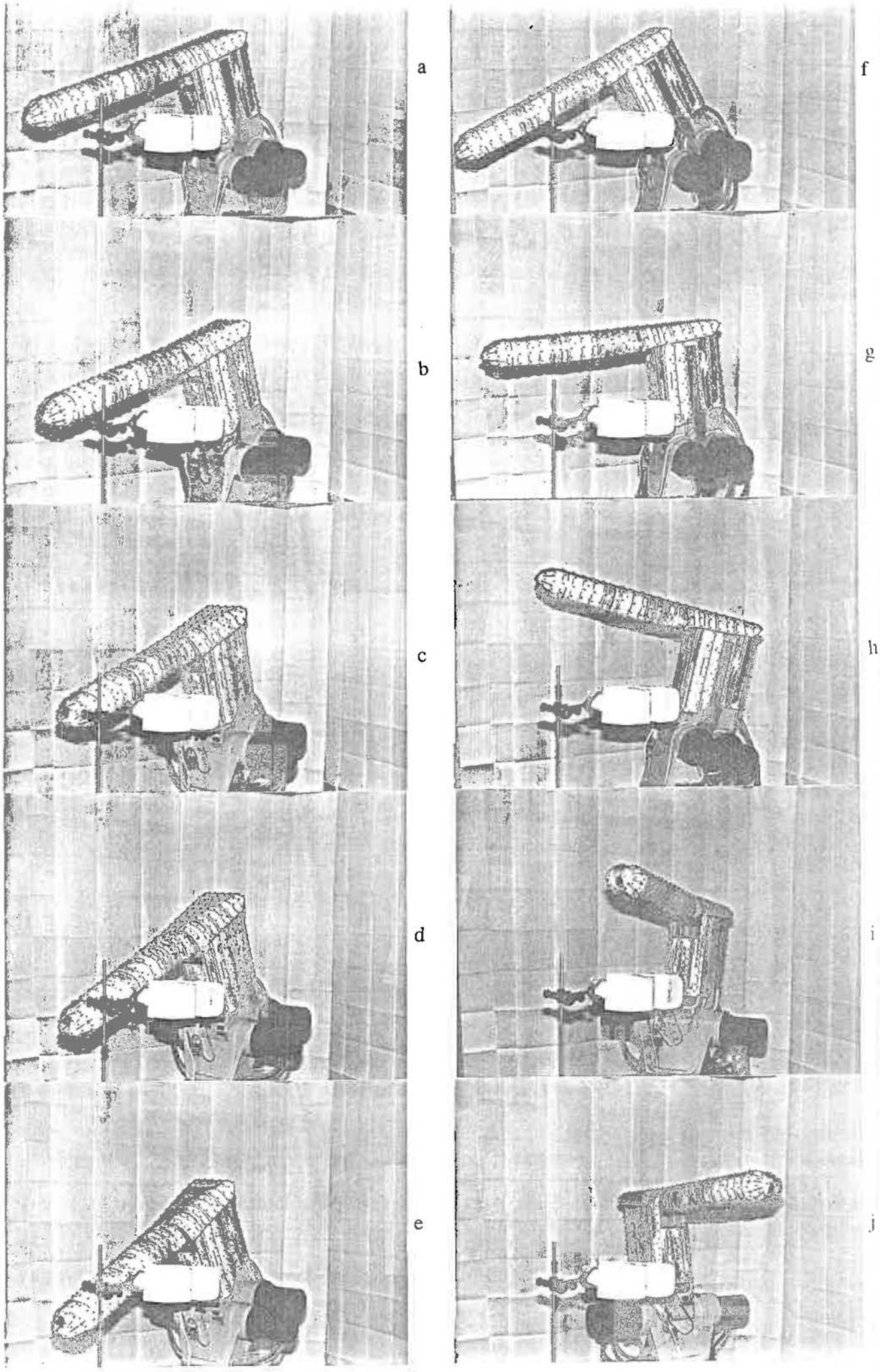


Figure 11.
Experiment 2.

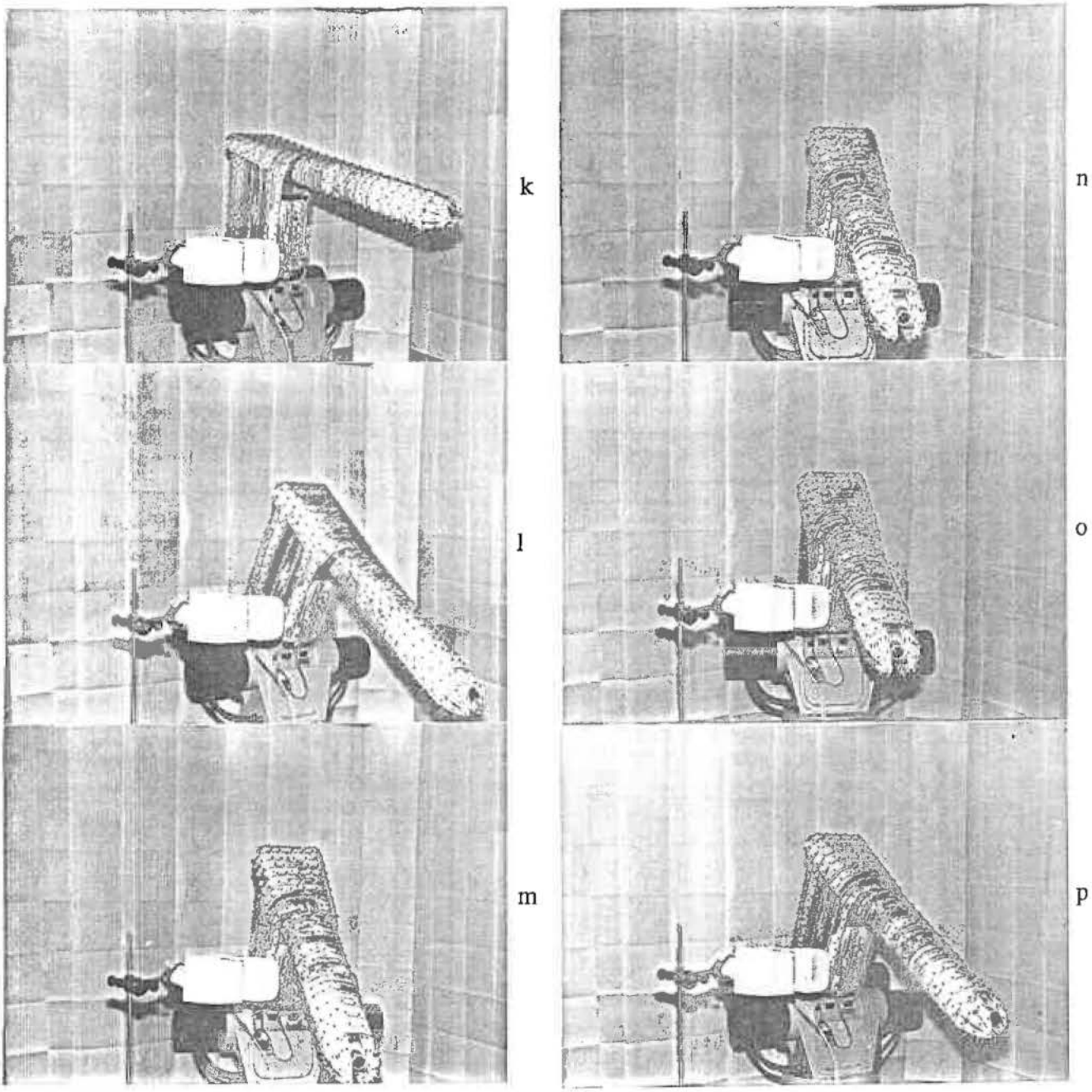
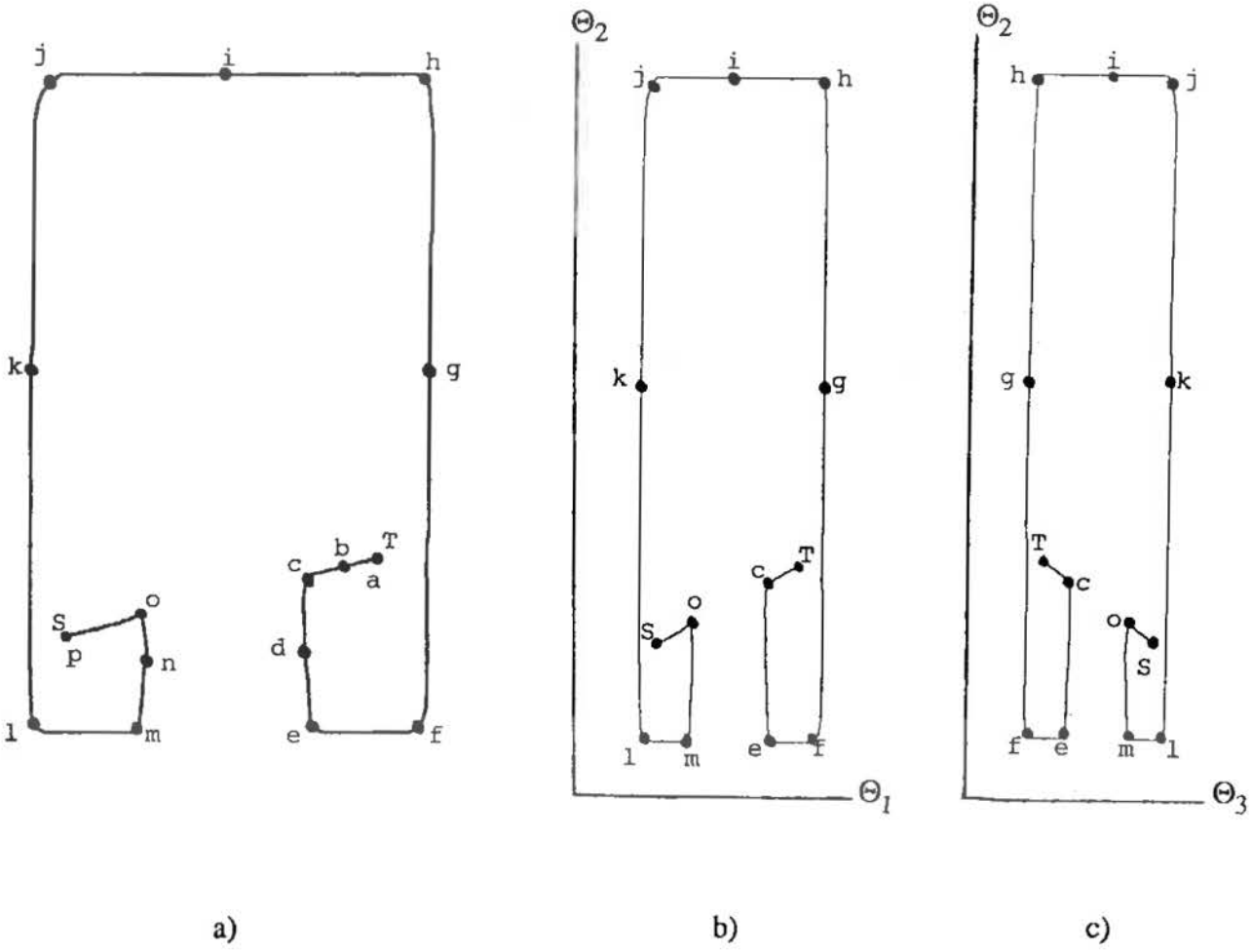


Figure 11.
Experiment 2.



a) b) c)

Figure 12: Experiment 2. C-space image of the arm path shown in Figure 11: (a) the path in P-plane; (b) projection of the path onto the reference plane (Θ_1, Θ_2); (c) projection of the path onto the reference plane (Θ_2, Θ_3). Note that a large part of the path passes along the "obstacle" that represents the joint limits of the arm. The path segments (c,d,e) and (m,n,o) correspond to the arm maneuvering around the obstacle.

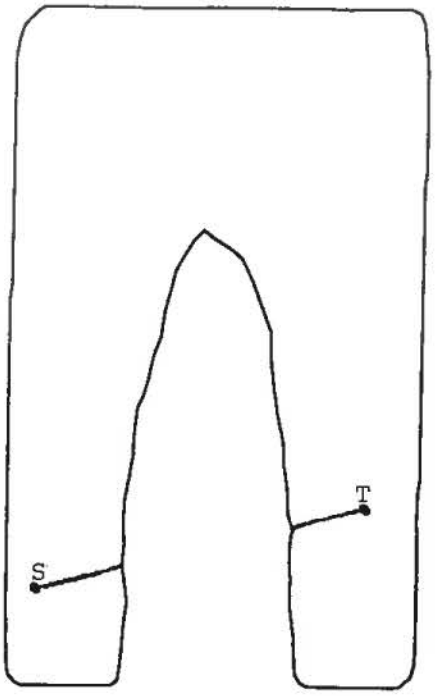


Figure 13. P-plane presentation of the sum of paths generated during Experiments 1 and 2.

References

1. E. Cheung and V. Lumelsky, Development of Sensitive Skin for a 3d Robot Arm Operating in an Uncertain Environment, Proc. 1989 IEEE Conference on Robotics and Automation, Scottsdale, AZ, May 1989.

2. J.P. Karlen, J.M. Thompson,J.D. Farrell, H.I. Vold, Reflexive Obstacle Avoidance for Kinematically-Redundant Manipulators, NASA Conference on Space Telerobotics, Pasadena California, January, 1989.

3. S. Gordon and W.Townsend, Integration of Tactile-Force and Joint Torque Information in a Whole-Arm Manipulators, Proc. 1989 IEEE Conference on Robotics and Automation, Scottsdale, AZ, May 1989.

4. H. Hemami, Differential Surface Model for Tactile Perception of Shape and On-Line Tracking of Features, IEEE Journal of Systems, Man, and Cybernetics, March/April 1988.

5. V.Lumelsky and A. Stepanov, Path Planning Strategies for a Point Mobile Automaton Moving Amidst Unknown Obstacles of Arbitrary Shape, Algorithmica, Springer-Verlag, 2, 1987.

6. E. Cheung and V. Lumelsky, Motion Planning for Robot Arm Manipulators with Proximity Sensors, Proc. 1988 IEEE Conference on Robotics and Automation, Philadelphia, PA, April 1988.

7. K.Sun and V. Lumelsky, Motion Planning with Uncertainty for a 3d Cartesian Robot Arm, 5th International Symposium on Robotics Research, Tokyo, Japan, August 1989.

8. R. Sedgewick, Algorithms, Addison-Wesley Publishing Co. Reading, MA, 1984.