Application of the CIRSSE Cooperating Robot Path Planner to the NASA Langley Truss Assembly Problem

Jonathan M. Weaver and Stephen J. Derby
Rensselaer Polytechnic Institute
Troy, New York

Application of the CIRSSE Cooperating Robot Path Planner to the NASA Langley Truss Assembly Problem

Jonathan M. Weaver
Research Assistant

Stephen J. Derby
Associate Professor

Center for Intelligent Robotic Systems for Space Exploration
Department of Mechanical Engineering, Aeronautical Engineering, and Mechanics
Rensselaer Polytechnic Institute, Troy, New York 12180-3590

Abstract

A method for autonomously planning collision free paths for two cooperating robots in a static environment has been developed at CIRSSE. The method utilizes a divide-and-conquer type of heuristic and involves non-exhaustive mapping of configuration space. While there is no guarantee of finding a solution, the planner has been successfully applied to a variety of problems including two cooperating 9 dof robots.

Although developed primarily for cooperating robots, the method is also applicable to single robot path planning problems. A single 6 dof version of the planner has been implemented for the truss assembly task at NASA Langley's Automated Structural Assembly Lab (ASAL). The results indicate that the planner could be very useful in addressing the ASAL path planning problem and that further work along these lines is warranted.

1 Introduction

The robot path planning problem involves determining if a continuous and obstacle avoiding path exists between start and goal positions, and, if so, to find such a path. The complexity of the path planning problem has been shown to be exponential in the number of dof [1, 2]. A review of the many path planning techniques is beyond the scope of this paper. Reference [3] presents a recent survey paper on the subject.

A method has been developed at Rensselaer's Center for Intelligent Robotic Systems for Space Exploration (CIRSSE) to autonomously plan collision free paths for two robots working cooperatively in a known, static environment [4, 5, 6]. Cooperation refers to the case whereby both robots simultaneously grasp and manipulate a common, rigid, payload. The planner is based around a divide-and-conquer heuristic aimed at traversing c-space while performing selective mapping on an as-needed basis. This path planner has been applied to the CIRSSE testbed. The testbed consists of two 9 dof robots, each of which consists of a 3 dof platform and a 6 dof Puma. A sample path found by the cooperating 9 dof planner is shown in Figure 1. This example required approximately 10 minutes solution time on a SparcStation 1.

Although developed primarily for the cooperating robot case, the c-space traversal heuristic around which the planner is based may also be applied to single robot path planning problems. This paper discusses a single arm version of the planner which was implemented for the truss assembly task at NASA Langley's Automated Structural Assembly Lab (ASAL). The purpose of the implementation was to assess the potential usefulness of the planner for the ASAL path planning problem.

The ASAL path planning problem is described in Section 2. Our path planning strategy is discussed in Section 3. Sections 4 and 5 present some implementation details and results for application of the planner to the ASAL path planning problem. Section 6 presents some conclusions and areas for future work.

2 Problem Statement

A CimStation model of NASA Langley's ASAL is shown in Figure 2 (CimStation model provided by NASA Langley). The system consists of a 6 dof Merlin robot, shown in Figure 3, mounted to a xy-positioning table (referred to as the carriage), and a turntable. The turntable includes a triangular platform which can rotate around a vertical axis through its center. The Merlin robot is kinematically similar to a Puma. The objective of the ASAL is to assemble truss structures consisting of 102 2 meter long struts. Such a truss is illustrated in Figure 4. The truss is assembled upon the turntable of the ASAL by positioning the carriage and the turntable such that the Merlin may take each strut from a canister near the base of the Merlin and install it in its final position in the assembly.

The ASAL path planning problem as addressed herein is defined as follows: Given a carriage and turntable position for each strut, determine a suitable path for the Merlin to safely move the robot and its payload from a start position to a prescribed goal position. The start position is above the canister holding the as-yet unassembled struts. The goal position for each strut is taken as 10 cm from the final position in the negative of the approach direction. The assembly sequence is as specified by NASA Langley.

It is assumed that feasible and collision free start and goal joint configurations of the robot are known.

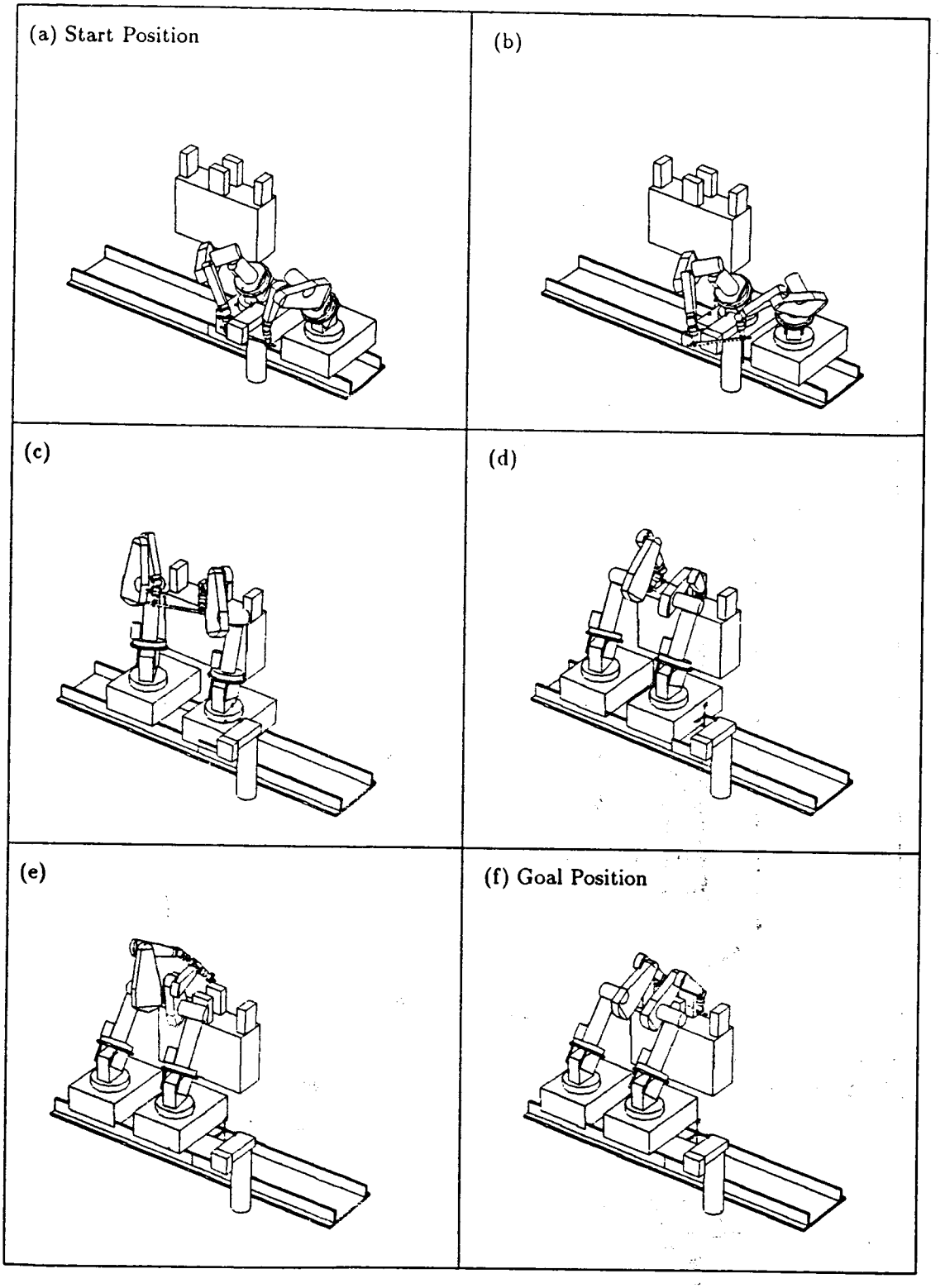
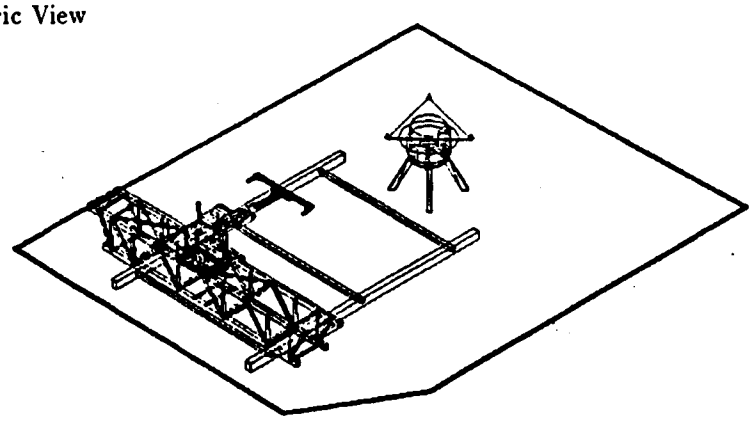
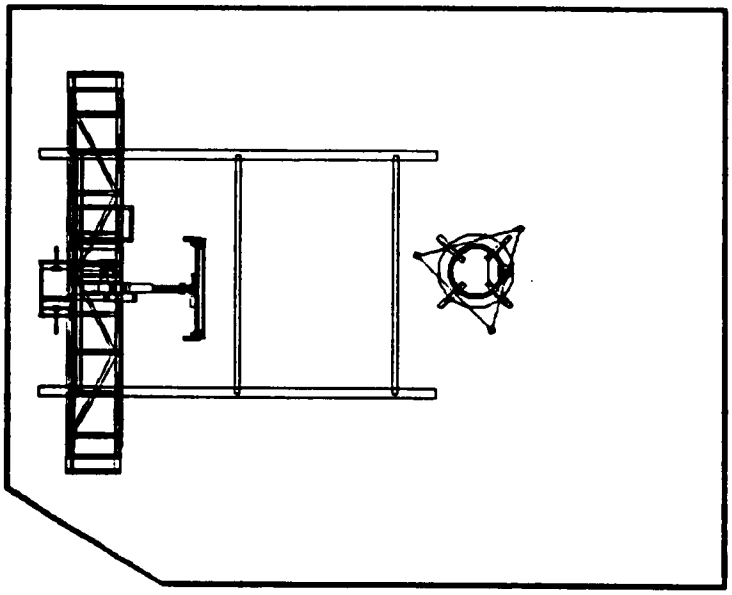


Figure 1: Sample Results for Cooperating 9 DOF Robots

(a) Isometric View



(b) Top View



(c) Side View

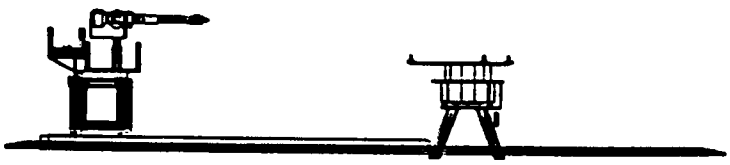


Figure 2: NASA Langley's Automated Structural Assembly Lab

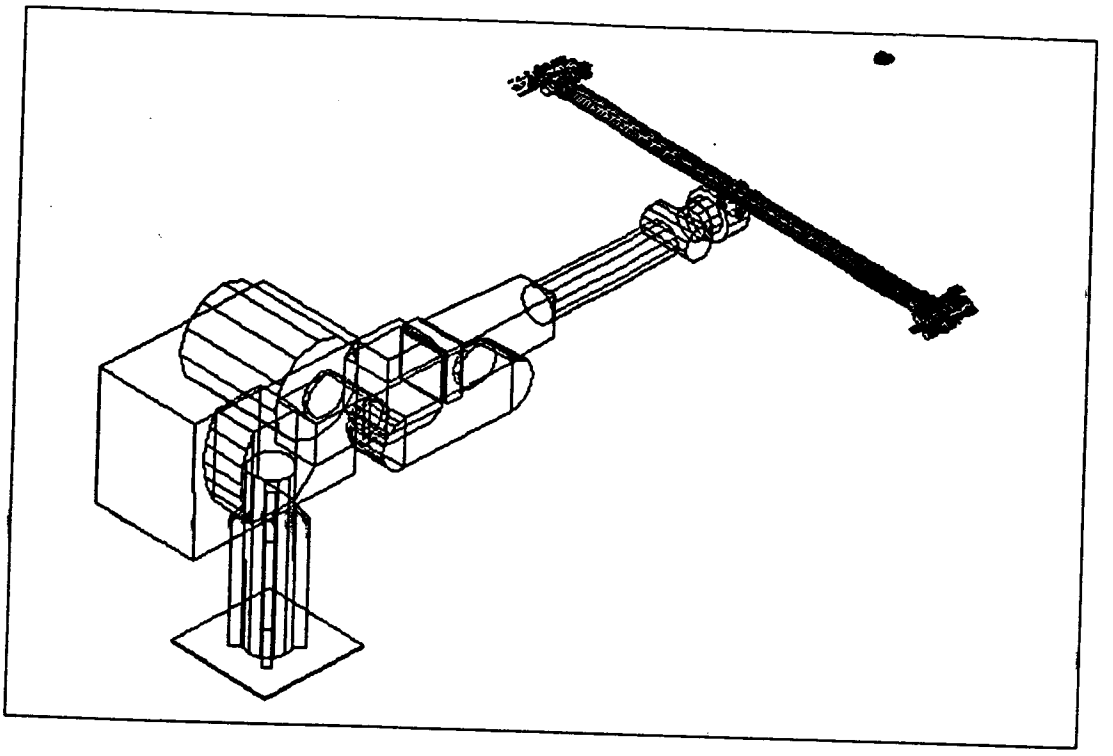


Figure 3: 6 DOF Merlin Robot with End Effector for Truss Assembly

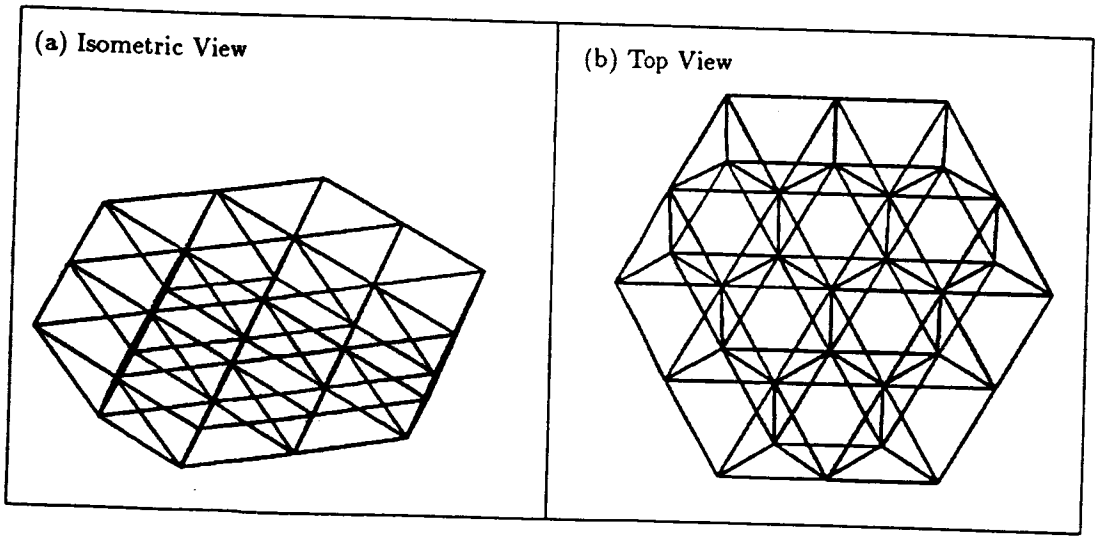


Figure 4: 102 Strut Truss Structure

3 Strategy

Like the single robot planner presented by Dupont [7], the principle strategy of our planner is to minimize the computationally expensive mapping of configuration space by performing mapping on an as required basis. The approach is based around a divide-and-conquer style heuristic for traversing through c-space. Computationally expensive precomputations and exhaustive c-space mappings are avoided. The approach is applicable regardless of the number and type of joints in the robot and for any number of obstacles in the workspace. A *string tightening* algorithm may be applied to modify any safe path found by the planner into a more efficient one, where efficiency is measured by joint space trajectory length.

The path planning method involves first attempting to traverse a c-space vector from the start to the goal of one of the robots. If this vector passes through unsafe space, the hyperspace orthogonal to and bisecting the unsafe segment of the vector is systematically searched to identify an intermediate goal point for consideration as a via point. An attempt is made to traverse from the last safe point to the intermediate goal point. This process is repeated as necessary until the attempted traversal to the newest intermediate goal point is entirely safe. At that point, progression is attempted toward all previous guide points in the opposite order in which they were found, where guide points include not only previous intermediate goal points but also the safe points found on the goal end of each unsafe region which invoked a search. When progression to a particular guide point is not entirely safe, that point is permanently dismissed and progression is attempted toward the next guide point in the specified sequence. The progression continues until an attempt has been made to progress to the global goal point. If that attempted progression is not entirely successful the overall process is repeated until the global goal point has been safely traversed to.

In 2D, the hyperspace orthogonal to an unsafe vector (the space which the heuristic searches) is simply a line. For 2D problems, the initial search is performed equally in both directions until a safe point is found. Subsequent searches will first exhaustively search in the direction which has a component in the previously successful search direction. Only when no safe point can be found in that direction will the other direction be considered. A 2D example of the c-space traversal heuristic is shown in Figure 5. This example involves non-disjoint safe space and requires multiple searches. More 2D examples and a vector description of the heuristic may be found in [6].

In the general nD case, the search space will be $n-1$ dimensional. In this case, several approaches were considered for computing search directions. The most effective method found involves considering all combinations of ± 1 and 0 (except all zeros) times a set of orthogonal basis vectors for the subspace. This yields $3^{n-1}-1$ search directions for an n dof problem. The following vectors may be calculated in the sequence shown and then normalized to yield one such orthogonal basis:

$$\begin{aligned}
\mathbf{B}_1 &= (1, h_1, 0, \ldots, 0) \\
\mathbf{B}_2 &= (b_{1_1}, p_2, h_2, 0 \ldots, 0) \\
\mathbf{B}_3 &= (b_{2_1}, b_{2_2}, p_3, h_3, 0 \ldots, 0) \\
&\vdots \qquad\quad \vdots \qquad\qquad \vdots \\
\mathbf{B}_{n-1} &= (b_{n-2_1}, b_{n-2_2}, \ldots, b_{n-2_{n-2}}, \\
&\qquad p_{n-1}, h_{n-1})
\end{aligned} \tag{1}$$

where the p_i are chosen so that the $\mathbf{B}_i$ and $\mathbf{B}_{i-1}$ are orthogonal, then the h_i are chosen so that the $\mathbf{B}_i$ lie in the search hyperplane.

Initial searches favor all directions equally, whereas subsequent searches sort the i directions $\mathbf{S}_i$ into g equal breadth bins by the following rule:

$$\mathbf{S}_i \in bin(j) \;\; if \;\; \frac{j-1}{g} \leq \frac{dp_i - dp_{min}}{dpmax - dp_{min}} \leq \frac{j}{g} \tag{2}$$

where dp_i is the dot product of $\mathbf{S}_i$ with the previously successful search direction, and dp_{min} and $dpmax$ are the minimum and maximum dp_i, respectively.

Searches then exhaust $bin(i)$ before considering $bin(i+1)$.

3.1 Completeness

Unfortunately, this path planning method is not complete, i.e., it cannot guarantee finding a solution even if one exists. Though certainly undesirable, this lack of completeness does not seem unreasonable since researchers have thus far been unable to develop algorithms which achieve both completeness and practicality for reasonably difficult yet practical path planning problems for more than a few degrees of freedom. We sacrificed completeness in exchange for the possibility of solving some practical yet potentially difficult problems as quickly as possible.

3.2 String Tightening

Once a safe path is found, it may be modified to reduce the joint space trajectory length of the path. This process is referred to as string tightening [7]. Since the path planner produces discretized paths, the objective during string tightening is to reduce the following cost function:

$$L_1^N = \sum_{i=1}^{N-1} \sqrt{\sum_{j=1}^{n} (\theta_j(i+1) - \theta_j(i))^2} \tag{3}$$

where:

L_1^N	=	the joint space trajectory length
N	=	number of knot points in path
n	=	number of dof
$\theta_j(i)$	=	i^{th} knot point for joint j

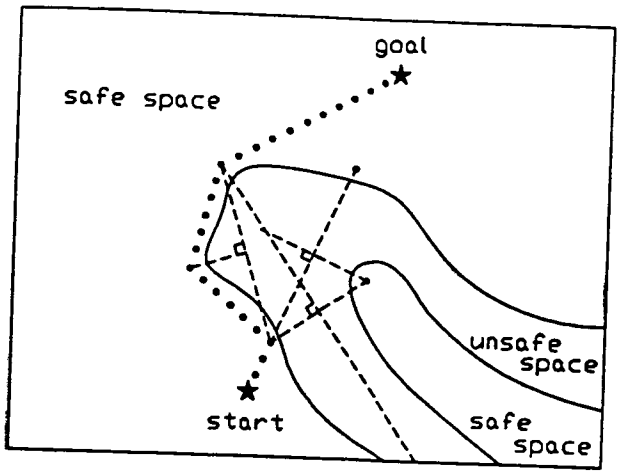


Figure 5: 2D Example of C-Space Traversal Heuristic

The tightening algorithm involves examining each sequence of three adjacent knot points and performing whichever of the two options below produces the most desirable effect on L_1^N:

1. Make no changes to the knot points.

2. Modify the second knot point so that the three knot points are straight in the robot's joint space (if not already so).

The feasibility of option 2 must be determined by checking the configuration for interference. These local adjustments are continued along the length of the path until no significant improvement can be obtained from further adjustments.

4 Implementation

In addition to having been implemented for single and cooperating robot path planning problems for the CIRSSE testbed, the path planning strategy described in this paper has been implemented for the ASAL path planning problem described in Section 2. The programs are written in C and utilize sections of code developed by Schima [8]. They also invoke methods and code developed by Hamlin and Kelley [9, 10]. The polytope representation scheme was chosen because it permits accurate modeling of the robots and typical obstacles in the workcell while enabling relatively fast interference checking. Paths are visually simulated using CimStation [11]. The implementation uses 242 search directions and 5 bins for search direction prioritization.

Since this was a preliminary implementation intended to evaluate the possible usefulness of the path planner for the ASAL path planning problem, some simplifications were made:

- Nodes were not modeled.
- In the ASAL, panels are installed (the first set after the 60^{th} strut). These panels were not modeled (except for one particular strut as a case study).

5 Results

The path planner quickly found paths for the first 21 struts since there is little possible interference at that stage. Due to symmetry, the assembly of the remaining 81 struts can be accomplished using only 21 unique trajectories for the Merlin with the appropriate carriage and turntable positions for each strut. The path planner was able to find feasible paths for all 102 struts with solution times ranging from 1 to 30 minutes on a SparcStation 1, with the vast majority of solution times in the 2 to 5 minute range.

The 61^{st} strut is possibly the most difficult from a path planning perspective due to the confined location of the goal position and due to the presence of an installed panel above the goal position. Although this implementation generally ignored the panels, a panel was modeled as an obstacle for this strut. In spite of the panel, a path was found without requiring any intermediate carriage/turntable positions. The path found for this strut is illustrated in Figure 6.

Some particular comments regarding this implementation follow:

- The path planner has no trouble with goal positions placing the load or robot in very close proximity to obstacles.
- The path planner performs well even with a large number of obstacles. For example, the final few struts of the assembly involve over 100 workspace obstacles. The additional collision checks required near the end of the assembly seem to increase execution time by a factor of approximately two.

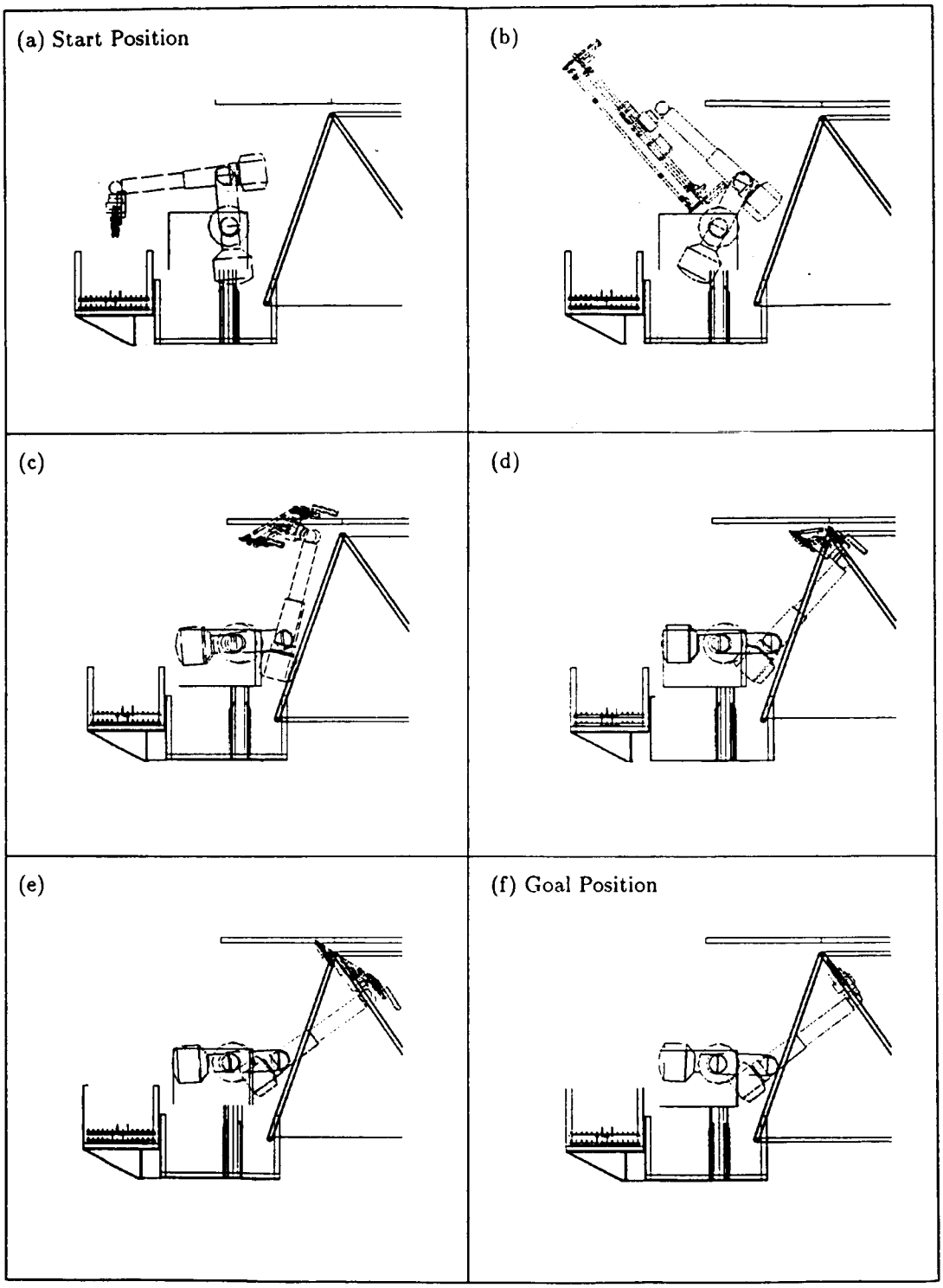


Figure 6: Sample Results for 61^{st} Strut (Side Views)

- The paths found typically include segments which are obstacle boundary tracing. Because of the close tolerances involved, it is not practical to simply model the objects larger than actual size to provide a safety margin since doing so would often result in an unsolvable problem.

- Panels and nodes were not modeled. As a result, some of the paths might collide with the panels or nodes if the paths were used in an actual assembly. This could be remedied simply by modeling the panels and nodes and including them in the collision checking routine. Due to the small size of the nodes it is expected that including them would have little impact on the difficulty of the path planning problems. Although the panels will typically represent significant obstacles to be avoided, a strut for which the panels would seem to interfere the most was solved with the relevant panel modeled.

- In a few cases the path planner was not able to solve the problem quickly in the forward direction but could quickly solve the problem in the opposite direction. Although a very confined goal position makes it likely that solving in reverse may prove easier, trial and error was the only sure way to decide which direction would yield better performance.

- Return paths for the robot after inserting a strut were not planned.

6 Conclusions and Future Work

This implementation of the path planner for the ASAL assembly task illustrates the potential usefulness of the path planning technique developed at CIRSSE for solving the practical and potentially very difficult ASAL path planning problem. Based on the results of this study, additional work appears warranted towards applying this planning technique to the path planning problems at the ASAL. Some particular issues which would need to be addressed before paths created by the planner could be executed on the actual hardware are as follows:

- Nodes and panels need to be modeled.

- An improved string tightening algorithm or an alternate method of path modification is required to provide paths with adequate clearances. This could be done by modifying the current string tightening cost function to include a penalty on clearance or by utilizing the path found by the planner as input to a potential fields based smoothing algorithm.

Acknowledgments

This work was supported by NASA grant NAGW-1333.

References

[1] John H. Reif. Complexity of the mover's problem and generalizations extended abstract. In *Proceedings of the 20th Annual IEEE Conference on Foundations of Computer Science*, pages 421–427, 1979.

[2] J.T. Schwartz and M. Sharir. On the 'piano movers' problem ii. general techniques for computing topological properties of real algebraic manifolds. *Computer Science Technical Report No. 41*, February 1982. Courant Institute, New York University.

[3] Y.K. Hwang and Narendra Ahuja. Gross motion planning - a survey. *ACM Computing Surveys*, 24(3):219–291, September 1992.

[4] J.M. Weaver and S.J. Derby. A method for planning collision free trajectories for two cooperating robots. AIAA Space Programs and Technologies Conference, 1992. Paper No. AIAA-92-1722.

[5] J.M. Weaver and S.J. Derby. A divide-and-conquer method for planning collisions free paths for cooperating robots. In *Robotics, Spatial Mechanisms, and Mechanical Systems, ASME*, volume DE-45, pages 461–471, 1992.

[6] J.M. Weaver and S.J. Derby. A divide-and-conquer method of path planning for cooperating robots with string tightening. In *Proceedings of the Fourth Annual Conference on Intelligent Robotic Systems for Space Exploration*, pages 30–40, Sponsored by NASA CIRSSE, Troy, NY, 1992.

[7] Pierre E. Dupont. *Planning Collision-Free Paths for Kinematically Redundant Robots by Selectively Mapping Configuration Space.* PhD thesis, Rensselaer Polytechnic Institute, Troy, NY, 1988.

[8] Francis J. Schima. Two arm robot path planning in a static environment using polytopes and string stretching. Master's thesis, Rensselaer Polytechnic Institute, Troy, NY, 1990.

[9] A representation scheme for rapid 3-d collision detection. CIRSSE Document No. 9, 1988.

[10] G.J. Hamlin and R.B. Kelley. Efficient distance calculation using the spherically-extended polytope (s-tope) model. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2502–2507, Nice, France, May 1992. Vol. 3.

[11] Cimstation user's manual, cimstation 4.3. Silma Inc., Cupertion, CA, 1992.