Randomized Motion Planning for Groups of Nonholonomic Robots

Christopher M. Clark chrisc@sun-valley.Stanford.edu
Stephen Rock rock@sun-valley.Stanford.edu

Department of Aeronautics & Astronautics
Stanford University
Stanford, CA 94305, U.S.A.

KeyWords Motion Planning, Robotics, Nonholonomic, Group Robotics.

Abstract

This paper presents a technique for motion planning which is capable of planning trajectories for a large number of nonholonomic robots. The robots plan within a two dimensional environment that consists of stationary/moving obstacles, and fixed boundaries. Each robot uses randomized motion planner techniques based on Probabilistic Road Maps (PRM's) to construct it's own trajectory that is free of collisions with moving obstacles and other robots. The randomized motion planner allows easy integration of the robots nonholonomic constraint into the planning so that only kinematically consistent plans are constructed. It is important to include this constraint in the planning problem since the majority of planetary surface robots are nonholonomic. The speed of the road map construction allows planning in real-time, enabling the robot to maneuver safely in a dynamic environment. Communication between robots is infrequent since robots only communicate on a "need to know basis". To verify the planner's effectiveness, it was tested using both simulation and experiment.

1 Introduction

Future missions to Mars will require a fleet of surface robots to carry out planet exploration. When large groups of robots are working together within a designated area, high-level motion planning is required to avoid collisions. The main objective of this work is to develop a motion planning system that can handle these types of situations.

The motion planning algorithm presented is based on the planner developed by Kindel and Hsu[1]. Their work demonstrates the use of a randomized kinodynamic motion planner for a single robot maneuvering around stationary and moving obstacles. Their planner benefited from fast planning times, allowing the robot to rebuild trajectories in the presence of changes to the environment.

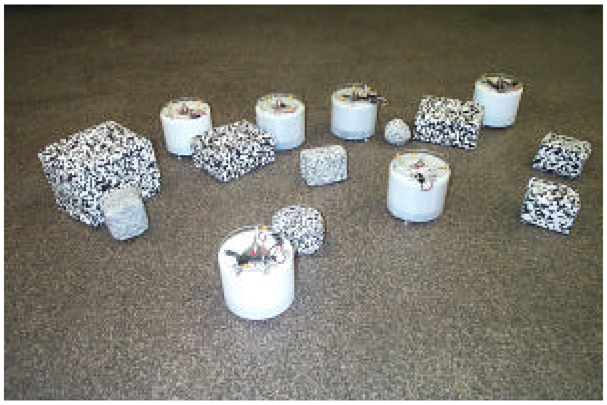
Figure 1: Robots and obstacles from the Micro-Autonomous RoverS (MARS) test platform.

In this paper, a randomized motion planner is applied to a multi-robot situation. The planning is decentralized in that each robot constructs it's own path independently, (see sections 2.1–2.4). When robots encounter one another, they communicate with each other. Using a priority system, the robots coordinate their motion plans to avoid collisions, (see Section 2.5).

The coordination of robots planning around each other's previously constructed trajectories has been demonstrated before[3],[7]. In [3], the trajectory for one robot is constructed irrespective of the other robots trajectory. To avoid collisions, the robots maintained the same path they constructed earlier, but altered the velocities along their paths. A drawback of this method is that confining robots to their original paths can eliminate a large amount of feasible solutions from the search space. Our planning system does not have this restriction.

A good majority of the motion planners for multi-robot systems are based on potential field methods[4],[8]. The reactive nature of potential field planners makes them very fast and hence are used in several applications like robot soccer[6]. A major drawback of potential fields is their susceptibility to dead-lock. Our planning system is similar to these planners in that it can react quickly to dynamic environments, but is more robust to dead-lock situations due to its randomness.

The paper is organized as follows. Details of the individual Motion Planners and how they coordinate are given in Section 2. Section 3 describes the Micro-Autonomous RoverS test platform that was used for simulations and real robot experiments. Results from the experiments are presented here. Section 4 discusses future work on the MARS test platform and possible heuristics to improve performance of the planner.

2 Motion Planner

Probabilistic Road Maps are constructed by randomly selecting milestones from the robot's configuration space and connecting the milestone pairs whose connection paths are collision-free [2]. As described in [1] and [5], this algorithm can be modified to accommodate any kinodynamic constraints by building a roadmap in the state x time space. Also shown in [1], is that under reasonable assumptions on the free space, the probability of failure decreases exponentially to 0 as the number of sampled milestones increases. They demonstrated how randomized motion planners can successfully build kinodynamically consistent trajectories in real-time.

For the cases referenced, simulations and experiments were carried out successfully with only a single robot. While the planners could be modified to include more robots by increasing the size of the configuration space searched, the planning times would increase to the point where real-time implementation would become infeasible.

In the new multi-robot planning system presented, each robot plans trajectories as described in sections 2.1 through 2.3. To decrease the complexity of the problem, robots only plan trajectories around other robots that are within a specific range. This is discussed in section 2.4.

2.1 Selecting new Milestones

The state of the robots in the MARS test platform can be described by $x = (x_1,x_2,q) \in \Re^3$ representing the position and attitude of the robot with respect to the table. Milestones are specified by both the state and time the robot reaches that state *(x,t)*. When selecting a new milestone *(x',t')*, we must consider the nonholonomic constraint described by:

$$\tan q = \frac{\dot{x}_2}{\dot{x}_1} \qquad (1)$$

The constraint can be reformulated in terms of the wheel velocities v_1 and v_2 of the robot.

$$\dot{x}_1 = \frac{(v_1+v_2)}{2}\cos q \qquad (2a)$$

$$\dot{x}_2 = \frac{(v_1+v_2)}{2}\sin q \qquad (2b)$$

$$\dot{q} = v_1 - v_2 \qquad (2c)$$

To select a new milestone in the road map, we could randomly select velocities v_1 and v_2 from $\{ 0, v_{max} \}$. However, further restrictions on the search space can be incorporated to increase the probability of finding a solution. The search space is restricted so that from one milestone to the next, the robot will not rotate more than 90 degrees. This inhibits the robot from spinning in circles. The distance the robot travels between milestones is also restricted to decrease the probability of selecting milestones located beyond the boundaries of the workspace. To incorporate these restrictions, two randomly selected variables are introduced: *range* which is selected from $\{-range_{max}, range_{max}\}$ and *difference* which is selected from $\{-difference_{max}, difference_{max}\}$. From these two variables, the distance traveled by each wheel s_1 and s_2 can be determined.

$$s_1 = range + difference \qquad (3a)$$

$$s_2 = range - difference \qquad (3b)$$

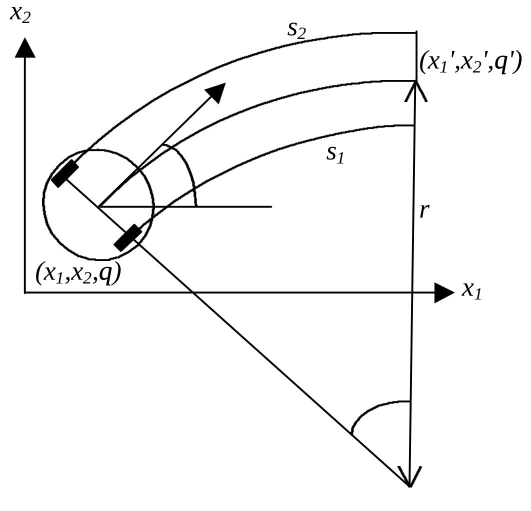

Figure 2: State space model of the MARS robot

This method corresponds to randomly selecting an arc of radius r and angle q. The new milestone $x' = (x_1', x_2', q')$ can be calculated as follows:

$$r = \frac{s_1 + s_2}{-s_1 + s_2} \tag{4a}$$

$$j = \frac{s_1 + s_2}{2r} \tag{4b}$$

$$q' = q + j \tag{5a}$$

$$x_1' = x_1 + r(\sin q' - \sin q) \tag{5b}$$

$$x_2' = x_2 + r(-\cos q' + \cos q) \tag{5c}$$

2.2 Road Map Construction

Let the tree T be a set of milestones. Initially T only contains the milestone (x^s, t^s), were x^s and t^s are the starting position and starting time respectively. The Roadmap is constructed using an iterative algorithm that adds new milestones to the set T at every step.

At each iteration, a milestone (x,t) is randomly selected from T for expansion. From this milestone, the tree is expanded to several new randomly selected milestones, (see section 2.1). If the arc connecting (x,t) to a new milestone (x',t') is collision-free, then it will be added to the tree T and the milestone (x,t) will be stored as it's parent. If (x',t') also lies within the endgame region, then the algorithm has found a solution and halts. The final trajectory is constructed by linking milestones to their parent milestones, starting with the goal milestone.

To avoid over-sampling in any one area of the workspace, procedure of selecting a milestone for expansion is modified. The workspace is divided up into a grid of cells. Let C denote the set of all cells in which a milestone from T is located. To select a new milestone in T for expansion, a cell c_{exp} is randomly selected from C. Then from within c_{exp}, the next milestone to expand from is randomly selected.

2.3 Endgame Region

In our algorithm, the final goal state x^g is underspecified in that only the position and not the attitude is specified. This is based on the assumption that once a robot has reached its goal location, it can rotate on the spot to reach any desired attitude. This underspecification on the goal state increases the size of the endgame region, hence increasing the probability of finding a solution. The endgame region E in our algorithm is defined as the subspace that includes all states x^e, such that the arc connecting x^e to the goal state has an angle j less than 90 degrees.

2.4 Coordinating Multi-Robot Planning

To deal with the intractability of planning for n different robots, the following technique is proposed. Each robot will create a plan with knowledge of only the few objects surrounding it. By planning around only those objects within the robot's local area, the motion planning problem is greatly simplified leading to decreased planning times. When new objects enter the robot's field of view, a re-plan is called for to ensure that the robot's trajectory is collision-free.

A priority system is used to determine how robots plan around each other. Before the experiment begins, each robot is given a priority number distinct from all others. When two robots encounter one another (i.e. enter each others field of view), they communicate with each other. The robots exchange data including the milestones of their roadmap and the robot's priority number. The robot with the lower priority number will immediately replan. When re-planning, the milestones received from the high priority robot are used to estimate its trajectory. The low priority robot can then construct a plan which is free of collision. The high priority robot will continue along its original path knowing the other robot will avoid it.

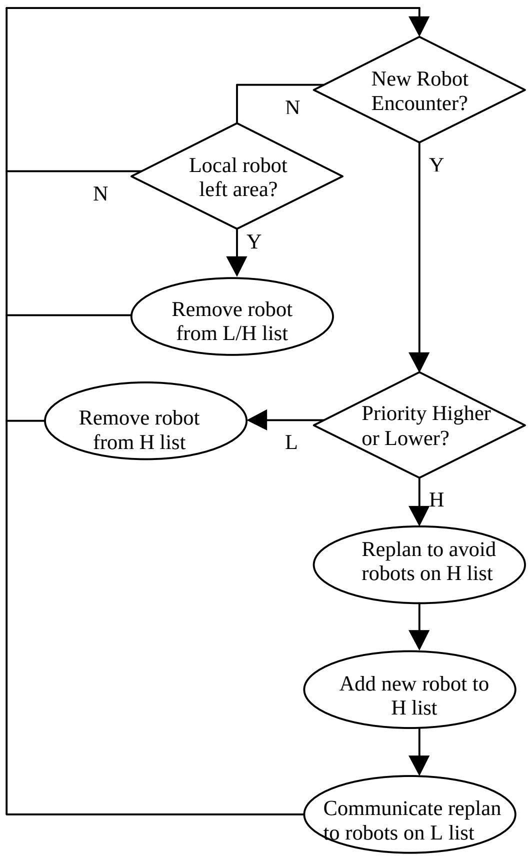
New Robot Encounter?
N
Y
Local robot left area?
N
Y
Remove robot from L/H list
Remove robot from H list
L
Priority Higher or Lower?
H
Replan to avoid robots on H list
Add new robot to H list
Communicate replan to robots on L list

Figure 3: Robot Coordination Algorithm

To facilitate this priority system, each robot must store a list of all robots within it's field of view who have lower priority, and a list of all robots within it's field of view who have higher priority. When the robot must replan because it encounters a robot with higher priority, it must communicate it's new trajectory to all the local robots of lower priority so that they can replan. For example, Robot A has priority 1, robot B has priority 2, and robot C has priority 3. If robot B and C encounter one another at time t_1, robot C will build a new trajectory so as to avoid robot B. If at time t_2, robot A and robot B encounter one another, then robot B will replan and communicate it's new trajectory to robot C who must also replan.

Since robots communicate only when they enter each others field of view, communication between robots is infrequent.

2.5 Decreasing Path Distances

The randomness of the planner leads to trajectories that are non-optimal. However, the randomized motion planner offers decreased planning times (on the order of 0.1 seconds), allowing the robots to plan in real-time. A method of improving the trajectories is to plan *m* (>1) times consecutively, and use the trajectory with the shortest path.

3 Experiment

The purpose of this research is to develop a system that can plan for large groups of robots. Presented below are simulations and experiments that verify the system's ability to build trajectories for many robots in constrained environments. First, the Micro-Autonomous RoverS test platform is introduced, followed by descriptions of how the platform is used in both simulations and real robot experiments.

3.1 The MARS Platform

The Micro-Autonomous RoverS (MARS) test platform at Stanford University was used to model the rovers in a two-dimensional work-space. The platform consists of a large 3m x 2m flat, granite table with six autonomous robots that move about the table's surface.

The robots are cylindrical in shape and use two independently driven wheels that allow them to rotate on the spot, but inhibit lateral movement so as to induce the nonholonomic constraint. Each robot has it's own Motion Planner located off-board. Control signal processing is also done off-board, and the control signals are sent to the individual robots via a wireless RC signal.

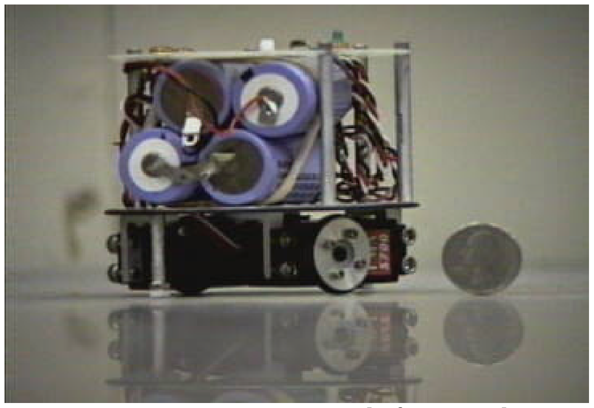

Figure 4: A rover MARS test platform standing beside a quarter.

An overhead vision system is used to provide position sensing. Three cameras with Infra-Red filters are used to detect LED's mounted on the top surface of robots and obstacles. Each robot/obstacle has a distinct pattern of LEDs to distinguish it from other robots/obstacles. The vision system updates the robot's position and velocity at a rate of 60Hz.

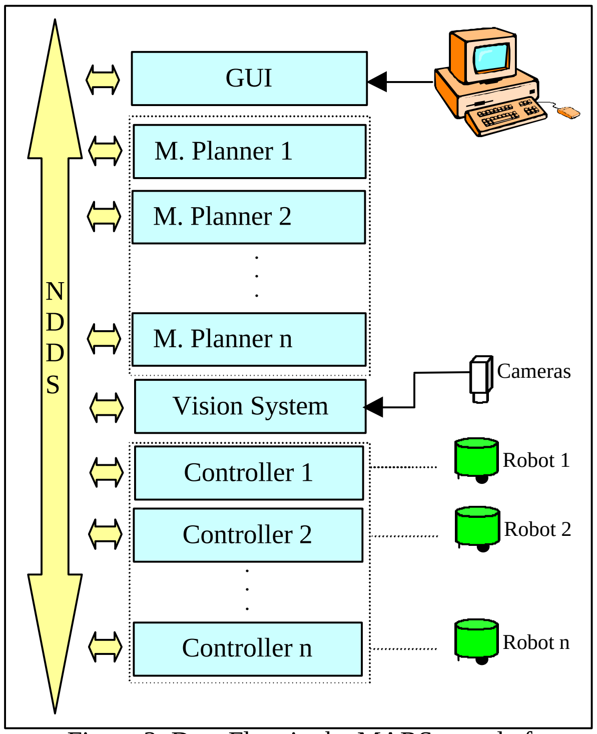

Figure 3: Data Flow in the MARS test platform.

The test platform features a Graphical User Interface (GUI) designed in Java/Swing. It provides a top-down view of the table including graphical representations of robots and obstacles, (see Figures 7,8). Setting robot goal locations is accomplished with a drag and drop system. New goal locations are sent to the respective motion planner so trajectories can be constructed.

All communication within the MARS platform is accomplished with Real Time Innovation's Network Data Delivery Service (NDDS) software. NDDS is based on a publish/subscribe architecture. Figure 3 illustrates the data flow in the platform.

The platform can be modified to allow for multi-robot simulations. The Vision System, the Controller, and the robot, (i.e. The two lower blocks in Figure 3), can be replaced by a software simulation program. Therefore the same Graphical User Interface(GUI) and Motion Planner are used for both physical experiment and simulation.

3.2 Integration of the planner

Data Flow

As mentioned above, NDDS works on a publish/subscribe architecture. Hence every node on the network can send and receive different data types.

The GUI subscribes to the vision data being published so that it may display the current locations of objects on the table. It publishes any command signals and desired goal locations requested by the user.

The Motion Planners subscribe to the vision data and to the command signals being published. Upon receiving a new command signal, it immediately constructs a new trajectory which it then publishes. To limit the amount of data sent across the network, Motion Planners only publish the milestones of the trajectory.

The Controllers subscribe to the vision data and the trajectory data published by their corresponding Motion Planner. They don't publish any information on the NDDS, but send control signals to their corresponding robots via an RC signal.

Time Synchronization

Robots are building trajectories based on the trajectory information of other robots. In order to ensure one trajectory is collision-free of another, all processors must have their clocks synchronized. This is accomplished by sending out an initial start signal from the GUI. When the start signal is received by any processor connected to the NDDS network, the processor's clock will be set to time zero. The time delay induced by the time it takes for the signal to travel across the network is compensated for by over constraining the collision checking.

Trajectory Following

Each Controller uses the milestones from the corresponding Motion Planner to construct the robot's trajectory. A Proportional Derivative (PD) control scheme is used to track the desired heading and position of sampled points of the trajectory.

3.3 Simulation

To simulate the MARS rovers and their environment, a Java application was developed which replaces the vision system, controllers, and robots. The simulations demonstrate the success of the motion planner for a large group of robots in a confined environment.

In each simulation, robots are initialized with randomly selected starting points and goal locations. Obstacle locations and orientations are also selected randomly. To simplify the implementation, obstacles move through the work space with a constant velocity and don't stop or interact with other obstacles. Shown in Table 1 are the results from 3 different simulation sets. Between each set, the number of robots and obstacles were varied. Each set was run 10 times with different randomly selected starting points. These simulations were run on a Sun Sparc Ultra5 with a 333 MHz processor and 128 MB RAM.

Table 1: Simulation Results

Experiment Set	**1**	**2**	**3**
Robots	5	10	15
Stationary Obstacles	5	5	0
Moving Obstacles	5	0	0
Average Plan Time (ms)	19.21	9.304	37.52
Average # of Plans	54.0	125.8	216.6
Average Maximum Plan Time (ms)	250.78	177.80	749.46
Average 1st Plan Time (ms)	58.92	44.69	44.61
Average Replan Time (ms)	8.67	5.59	15.31

As shown in Table 1, the motion planning system can provide motion planning solutions for experiments with up to 15 robots in an obstacle-free, bounded workspace, as well as for experiments with only 5 robots, 5 moving obstacles and 5 stationary obstacles.

In the simulations presented, the average replan times are significantly faster than the average first plan times. This can be attributed to the fact that replans first check to see if their original trajectory is collision-free with a newly encountered obstacle. If the original trajectory is safe, the planner will return it as the solution, and no new trajectory is required. The average first plan times are all less than 0.050 seconds, allowing planning in real time.

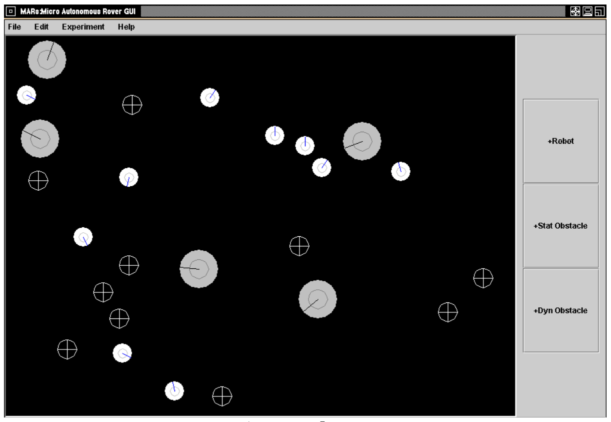

Figure 7a) Simulation at time T1:
Rovers , goals and obstacles before the simulation.

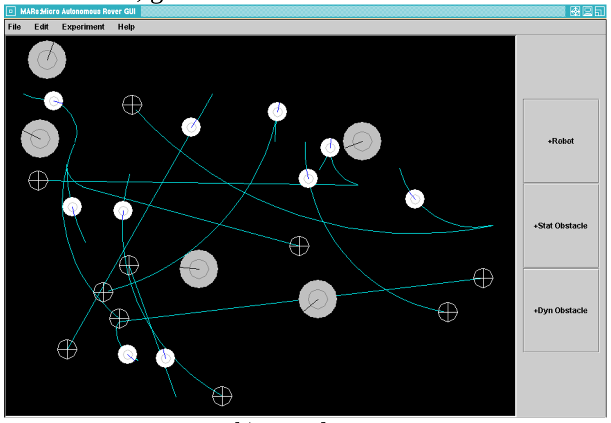

Figure 7b) Simulation at time T2:
Rovers after constructing their first plans

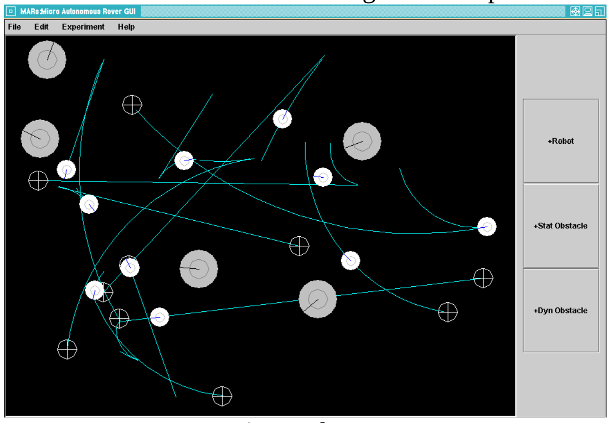

Figure 7c) Simulation at time T3:
Rovers replanning on the fly to avoid each other

An example of one such simulation is represented in the Figures 7a)-7f). This particular simulation involves 10 rovers, and 5 stationary obstacles. Smaller circles represent the micro-rovers as viewed from above, while crosses represent goal locations and larger circles represent obstacles. Trajectories constructed by each robot's motion planner are indicated with lines that lead to goal locations. Note that the trajectories change as the simulation progresses, indicating the replanning in real time.

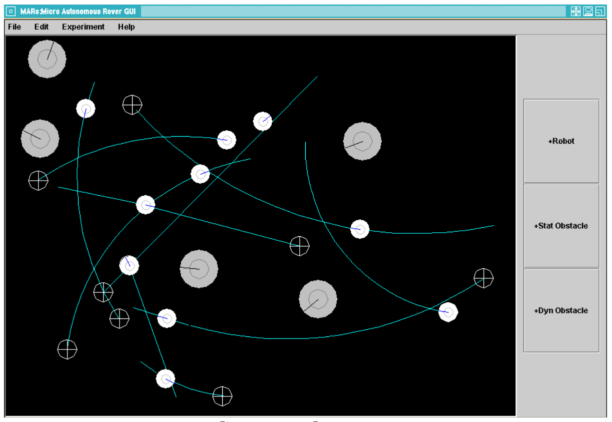

Figure 7d) : Simulation at time T4
Rovers following their trajectories

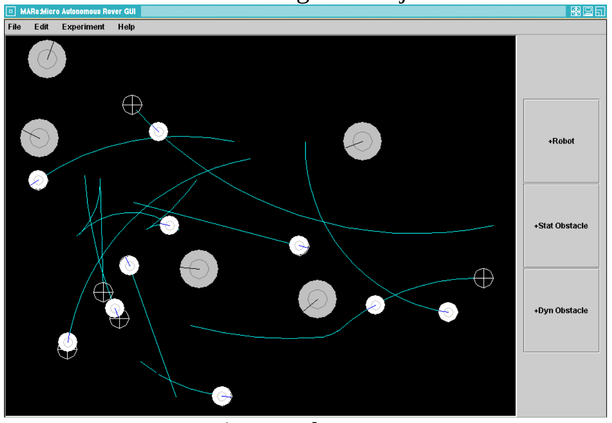

Figure 7e) : Simulation at time T5
Rovers heading towards their respective goals .

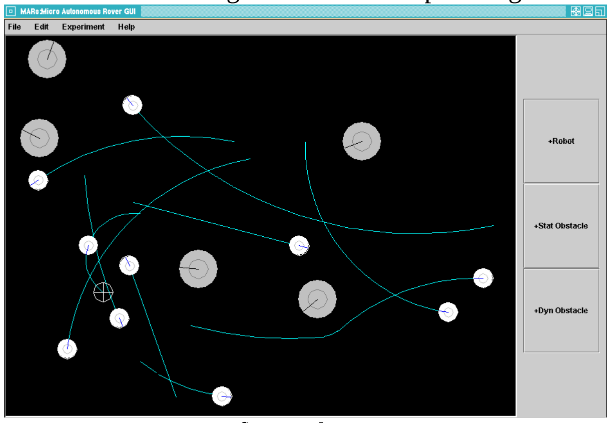

Figure 7f) Simulation at time T6:
All but one rover having reached their goal location.

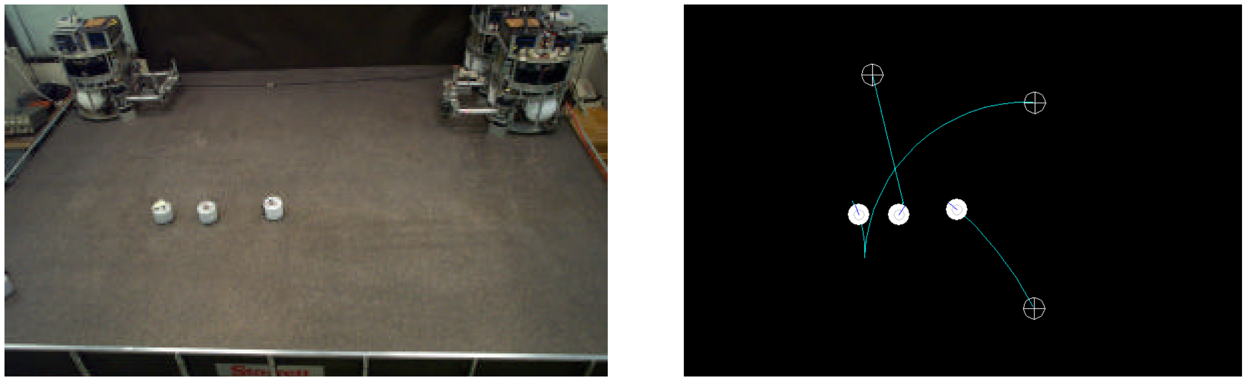
Figure 8a)Experiment on the MARS test platform: Initial trajectory generation.

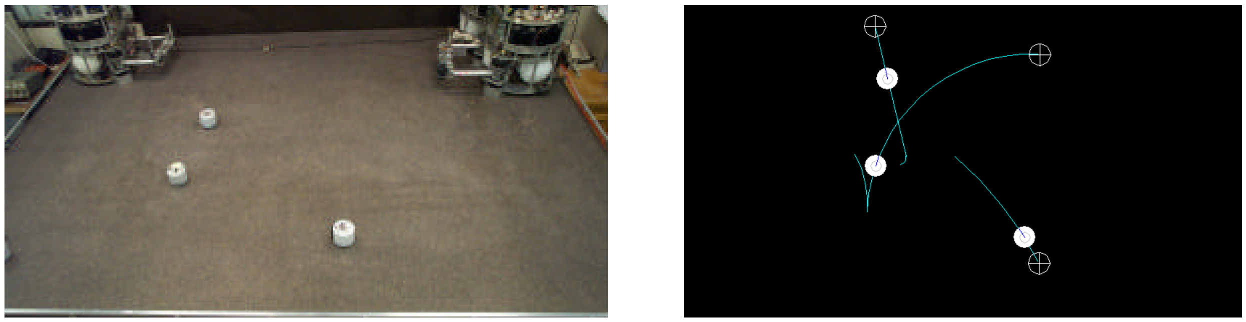
Figure 8b)Experiment on the MARS test platform: Following the trajectories.

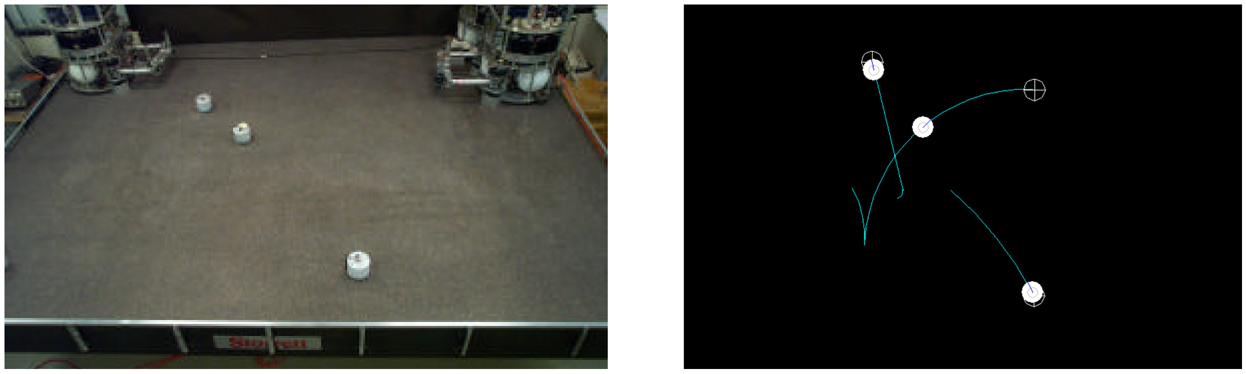
Figure 8c)Experiment on the MARS test platform: Approaching the goal destinations.

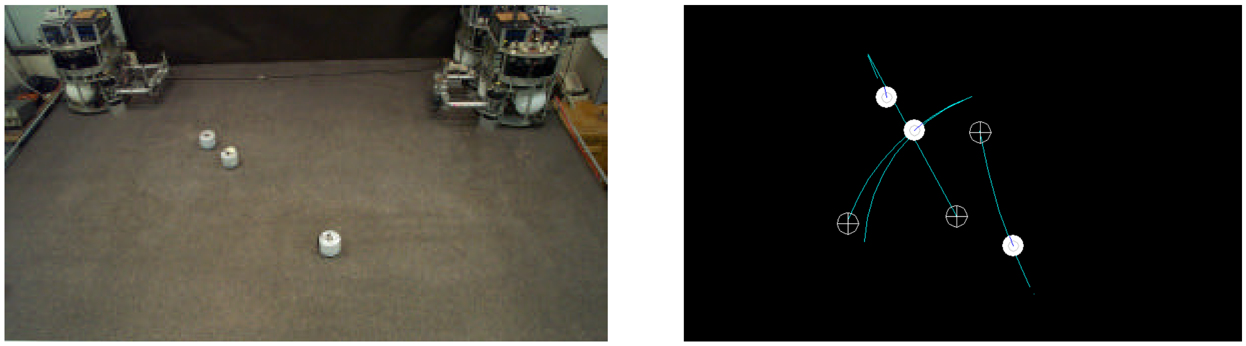
Figure 8d)Experiment on the MARS test platform: Heading towards new goal destinations

3.4 Physical Experiments

Several experiments were run to demonstrate that the system is able to construct collision free trajectories for rovers on a flat, bounded workspace. Tests were performed with various start configurations. Throughout the experiments, goal locations were continually being modified, requiring real-time planning.

Figure 8 shows a series of snap-shots taken from an experiment on the MARS test platform. In this experiment, only three of the rovers are tracking trajectories. The figures on the left are photos of the actual test-bed. The GUI screen shots on the right depict the rovers and the paths they are following. They were taken at the same time as the photos to their left. Figure 8a) shows the rovers immediately after the first trajectories were constructed. Figures 8b) and 8c) depict the rovers tracking these trajectories. In Figure 8d), the user has moved the goal locations. New trajectories were constructed and the rovers began tracking them. The initial plan times for this experiment ranged from 48 ms to 72 ms. Replan times ranged from 3 ms to 6 ms.

4 Conclusion

The motion planner presented has demonstrated its effectiveness in planning for a large number of robots within a bounded workspace. It planned with a high probability of success, even in "cluttered" environments involving 5 to 15 robots. Planning times on the order of 0.1 s allowed the robots to re-plan in real-time and react quickly to changes in the environment. Although the application of the motion planner to a surface rover mission has been discussed, it should be noted that the planner is extendible to three-dimensional workspaces. Hence it is also applicable to aerospace applications.

In the future, the use of a dynamic priority system to increase the probability of finding feasible trajectories in real-time will be investigated. Currently, robots with lower priorities must replan more frequently than robots with higher priorities. Ideally, all robots should replan with the same frequency. Also, robots with lower priorities must plan to avoid more robots than those with higher priorities. Hence lower priority robots have slower planning times. By setting robot priorities online, we hope to balance out the planning responsibilities evenly between all robots. This should increase the speed of constructing trajectories, and thus the probability of finding feasible trajectories in real-time.

References

[1] D. Hsu, R. Kindel, J. C. Latombe, and S. Rock. “Randomized Kinodynamic Motion Planning with Moving Obstacles,” in *Workshop on the Algorithmic Foundations of Robotics*, 2000.

[2] D. Hsu, J. C. Latombe, and R. Motwani, “Path planning in expansive configuration spaces,” in *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2719-2726, 1997.

[3] K. Kant, and S. Zucker, “Toward efficient Trajectory Planning: The path-velocity decomposition,” *The International Journal of Robotics Research*, 5-3, pages 72-89,1986.

[4] O. Khatib, "Real-Time Obstacle Avoidance for Manipulators and Mobile Robots,” *International Journal of Robotics Research*, 5, 1, pages 90-98, 1986.

[5] R. Kindel, D.Hsu, J. C. Latombe, and S. Rock. "Kinodynamic Motion Planning Amidst Moving Obstacles,” in *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 537-544, 2000.

[6] Lee, Lee, and Park, "Trajectory Generation and Motion Tracking for the Robot Soccer Game,” in *Proceedings of the 1999 IEEE International Conference on Intelligent Robots and Systems*, pages 1149-1154 , 1999.

[7] T. Y. Li, and J. C. Latombe, "On-line manipulation planning for two robot arms in a dynamic environment", In *Proceedings of the IEEE International Conference on Robotics and Automation*, 1995

[8] C. W. Warren, "Multiple Path Coordination using Artificial Potential Fields,” in *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 500-505, 1990.