Applying Kinodynamic Randomized Motion Planning with a Dynamic Priority System to Multi-Robot Space Systems

Christopher M. Clark, Tim Bretl & Stephen Rock
Aerospace Robotics Lab
Department of Aeronautics & Astronautics
Stanford University
Stanford, CA 94305
650-723-3213
chrisc@sun-valley.Stanford.edu
tbretl@sun-valley.Stanford.edu
rock@sun-valley.Stanford.edu

Abstract—This paper presents a new motion planning system that can construct collision-free trajectories for groups of robots in dynamic environments, without global knowledge or high-bandwidth communication. The robots plan within a confined environment that consists of stationary and moving obstacles. Each robot plans independently using Kinodynamic Randomized Motion Planner techniques to construct its own trajectory that is free of collisions with moving obstacles and other robots. To resolve conflicts between robot trajectories, a new Dynamic Priority System (DPS) is introduced which gives the right of way to the robot whose local workspace is the most crowded. The kinodynamic randomized motion planner allows easy integration of the robots nonholonomic constraint into the planning so that only kinematically and dynamically consistent plans are constructed. The speed of the trajectory 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. Results show a clear improvement in the decrease in planning times when the DPS is used instead of a static priority system.

TABLE OF CONTENTS

1. INTRODUCTION

Future space missions will require a fleet of robots to perform tasks such as planet exploration and outpost construction. When large groups of robots and moving obstacles are working together within a designated area, high-level motion planning is required to avoid collisions. In a real system with a large number of robots, it becomes difficult for each robot to estimate the state of all other robots. Continuous communication between all robots may not be feasible, and no system of sensors can provide global knowledge. Also, to function in a dynamic environment with moving obstacles, the system must be able to react quickly. For this type of multi-robot system, a motion planner that does not need global knowledge or high bandwidth communication, but can still plan in real-time, is required. The main objective of this work is to develop a motion planning system that meets this requirement.

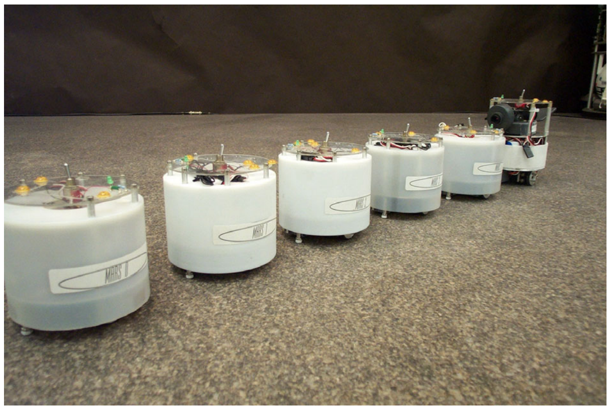


Figure 1 - Robots from the Micro-Autonomous RoverS (MARS) test platform.

The motion planning algorithm presented is based on the planner developed by Kindel and Hsu [4]. Their work demonstrates the use of a Kinodynamic randomized 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.

In this paper, the Kinodynamic Randomized Motion Planner is extended so that it may be used in a multi-robot situation. This extension requires a new communication protocol, a new priority system, and a limit on the number of nodes in a robots trajectory. When robots detect one another using local sensors, they communicate with each other. Using the priority system, the robots coordinate their motion plans to avoid collisions, (see Section 3). Relative to Kindel & Hsu's work, the number of nodes in a trajectory is limited, (see Sections 2). This restricts the amount of information communicated between robots, limiting delays in the system due to communication.

The coordination of robots planning around each other's previously constructed trajectories has been demonstrated before [6], [11], [12]. In [12], Simeon et. al. use a geometric based approach to coordinate previously built trajectories. This algorithm demonstrated effective planning for a large number of robots in a confined area. In [6], 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.

While this research was the first to apply Kinodynamic Randomized Path Planning to a multi-robot system, it should be noted that it is not the first instance of applying other types of randomized motion planning. Svestka et.al. demonstrated the use of Probabilistic Road Maps (PRM) for a multi-robot system in [13]. The algorithm was successful in building trajectories for up to 5 robots with planning times of a few seconds. However, this required a pre-processing step to build a "Road-Map".

Priority systems have also been used extensively to resolve conflicts between robots in decoupled planning systems. Warren uses a priority scheme with potential fields [14]. The reactive nature of potential field planners [7] makes them very fast and they are used in several applications like robot soccer [9]. A major drawback of potential fields is their susceptibility to deadlock. Our planning system is similar to these planners in that it can react quickly to dynamic environments, but is more robust to deadlock situations due to its randomness.

Bennewitz and Burgard [1], [2] use a search routine to find optimal priorities for minimizing path lengths and maximizing distances to obstacles. This algorithm demonstrated it's effectiveness in finding optimal plans in several difficult situations where the correct order of planning was essential to finding a solution.

The motion planning system presented in this paper and in [3], does not search for optimum trajectories. However, unlike the motion planners listed above, it can be applied to dynamic environments where real-time planning is required the robots are restricted to using only local sensing and communication.

The paper is organized as follows. Details of the individual Motion Planners are given in section 2. Algorithms for coordinating the robot plans are given in Section 3. Section 4 describes the Micro-Autonomous RoverS test platform that was used for simulations and real robot experiments. Results from the experiments are presented here. Section 5 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 [5]. As described in [4] and [8], this algorithm can be modified to accommodate any kinodynamic constraints by building a roadmap in the state x time space. This is known as Kinodynamic Randomized Motion Planning. Also shown in [4], 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 independently constructs trajectories using the algorithm described below. These trajectories are constructed to be collision-free with any obstacles in the robot's local area. When robots enter each other's local area, they must coordinate their trajectories to ensure they will remain collision-free. This is discussed in Section 3.

Road Map Construction

The state of the robots in the MARS test platform can be described by $x = (x_1,x_2,\theta) \in \Re^3$ representing the position and attitude of the robot with respect to the inertial frame. Milestones are specified by both the state and time the robot reaches that state (x,t).

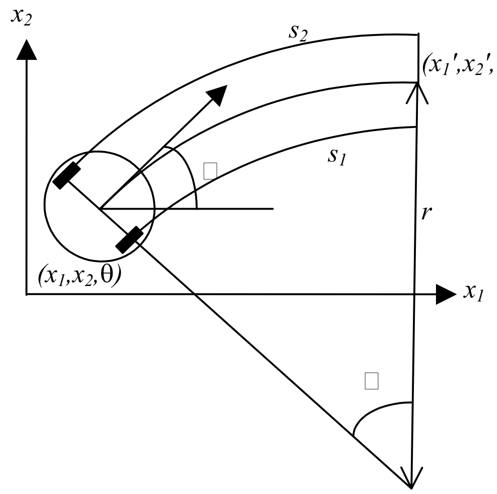


Figure 2 - State space model of the MARS robot

Let the tree T be a set of milestones. Initially T contains only the milestone (x^s,t^s), where 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 below). 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, (i.e. the region of the configuration space for which there must exist a collision-free arc connecting every point in the region to the goal), 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.

Selecting new Milestones

When selecting a new milestone *(x',t')*, consideration must be taken for the nonholonomic constraint described by:

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

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

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

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

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

To select a new milestone in the road map, the velocities v_1 and v_2 could be randomly selected 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.

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

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

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

$$r = \frac{s_1 + s_2}{-s_1 + s_2} \qquad (4a)$$

$$\varphi = \frac{s_1 + s_2}{2r} \qquad (4b)$$

$$\theta' = \theta + \varphi \qquad (5a)$$

$$x_1' = x_1 + r(\sin\theta' - \sin\theta) \qquad (5b)$$

$$x_2' = x_2 + r(-\cos\theta' + \cos\theta) \qquad (5c)$$

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 φ less than 90 degrees.

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. Coordinating Multi-Robot Planning

To deal with the intractability of planning for *n* different robots, the following technique was developed. Each robot creates a plan with knowledge of only the few obstacles 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. When two robots encounter one another (i.e. they detect each other using local sensing), they proceed to 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 that is free of collision. The high priority robot will continue along its original path knowing the other robot will avoid it.

Two different priority systems are presented here. First, the "Static" Priority System originally used in [3]. Second, a new "Dynamic" Priority System (DPS) is introduced.

Static Priority System

Before the experiment begins, each robot is given a priority number distinct from all others. To facilitate this priority system, each robot must store a list of all robots within its 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 its new trajectory to all the local robots of lower priority so that they can replan, (see Figure 3). 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 other's field of view, communication between robots is infrequent.

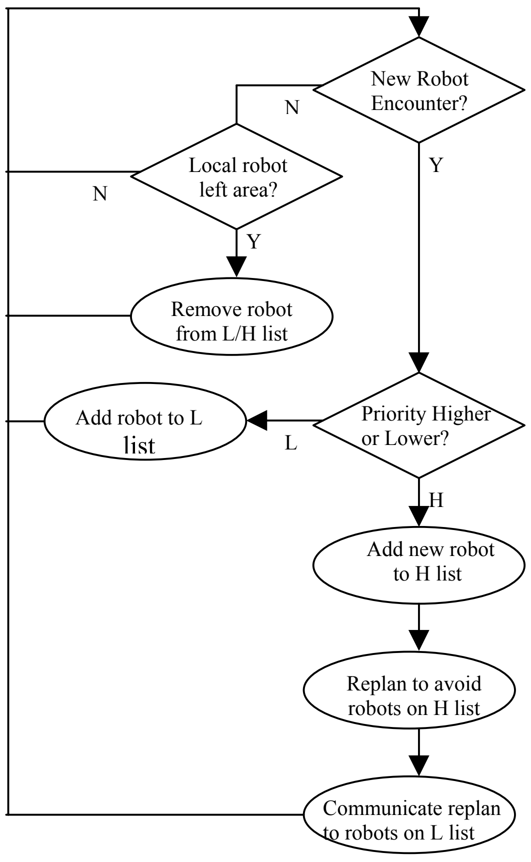

New Robot Encounter?
N
Y
Local robot left area?
N
Y
Remove robot from L/H list
Priority Higher or Lower?
L
H
Add robot to L list
Add new robot to H list
Replan to avoid robots on H list
Communicate replan to robots on L list

Figure 3 -Robot Coordination Algorithm for the Static Priority System

Dynamic Priority System

The purpose of using a dynamic priority system is to modify robot priorities, based on the current workspace configuration, in attempt to create less complex planning problems for the individual robots and more evenly distribute motion planning responsibilities among the robots.

The complexity of the collision checking routine in the planning problems that each robot solves during a replan is of order $O(p)$, where p is the number of local obstacles it must perform collision checks with at each milestone expansion. By giving priority to robots with higher p, the order of the collision checking routine and hence the overall complexity of the planning problem can be decreased.

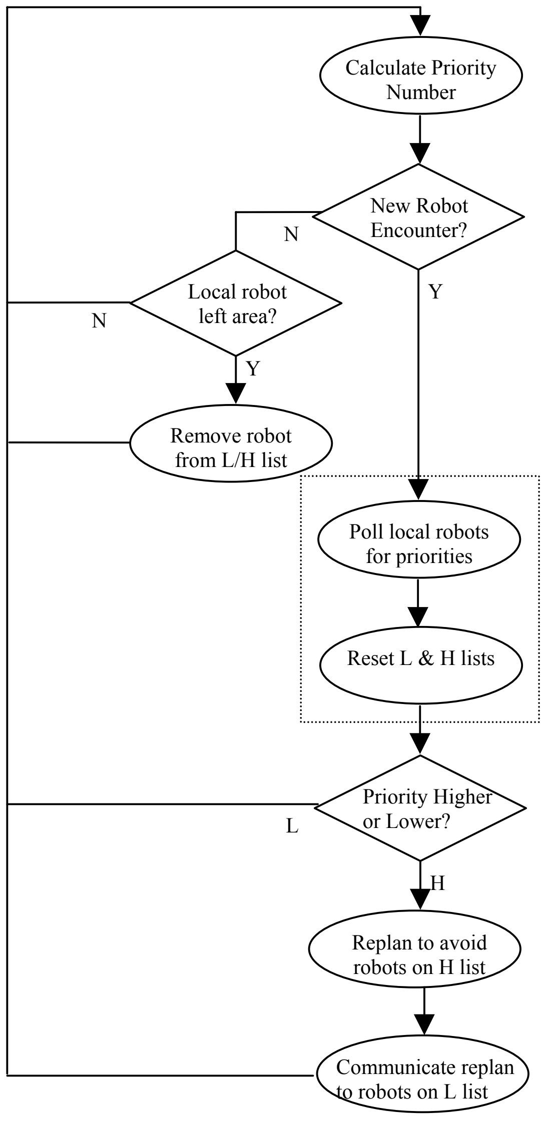

Calculate Priority Number
New Robot Encounter?
N
Y
Local robot left area?
N
Y
Remove robot from L/H list
Poll local robots for priorities
Reset L & H lists
Priority Higher or Lower?
L
H
Replan to avoid robots on H list
Communicate replan to robots on L list

Figure 4 -Robot Coordination Algorithm for the Dynamic Priority System

To implement the dynamic priority system, the robot coordination algorithm described in Figure 4 is used. The main addition to the static priority system algorithm is the function that polls local robots to request their current priority numbers, (see dotted box in Figure 4). This requires an extra communication call between robots and is the main cost of using the dynamic priority system.

Each robot dynamically recalculates its priority number based on how crowded the local search space is. The following equation is used.

$$priority = \sum_{i}^{n} \varsigma_i + 0.001\rho$$
$$\varsigma_i = \begin{cases} 1 \rightarrow i \in A \\ 0 \rightarrow i \notin A \end{cases} \tag{6}$$

The first term equals the number of robots in the local area *A*. This gives the right of way to robots operating in the most crowded environments, forcing the robots operating with the most open space to replan when encounters occur. The second term is the robot's identification number, weighted by a very small number. This term is used to act as a tiebreaker when two robots have the same number of robots in their local area.

4. Planning Experiments

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.

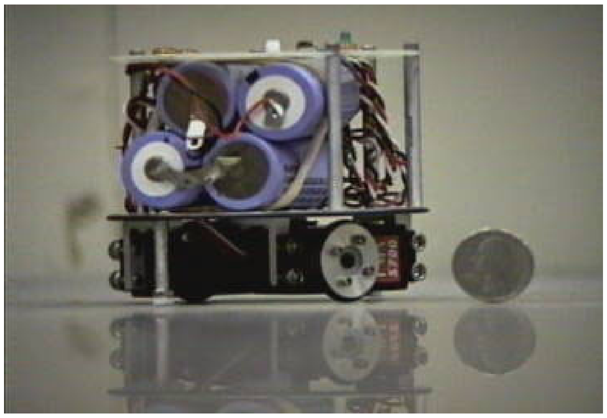


Figure 5 – A rover on the MARS test platform standing beside a quarter.

The MARS Platform

The Micro-Autonomous RoverS (MARS) test platform at Stanford University was used to model the rovers in a two-dimensional workspace. 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.

An overhead vision system is used to provide position sensing. Three cameras with Infrared 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 30Hz.

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.

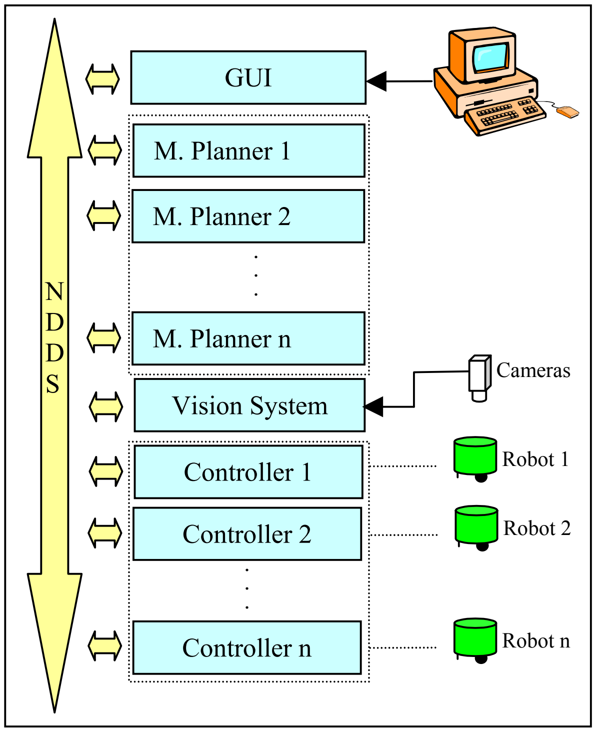


Figure 6 - Data Flow in the MARS test platform.

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 6), 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.

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 its corresponding Motion Planner to construct the robot's trajectory. A PD control scheme is used to track the desired heading and position of sampled points of the trajectory.

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.

A simple example of a simulation is illustrated in Figures 7a-f. In this example, a single robot starts in the lower left corner of the workspace and has a desired goal location situated behind two obstacles. The robot is denoted by the black circle and the goal is denoted by the black cross, (see Figure 7a). The robot initially plans its trajectory, (see Figure 7b,) without any knowledge of the two obstacles (one stationary and one moving). However, as time progresses, it encounters the moving obstacle and constructs a new trajectory around it, (see Figures 7c, 7d). As it closes in on the goal, it finally sees the stationary obstacle and replans again to avoid it, (see Figure 7e). Finally, in Figure 7f, the robot approaches the goal. Time steps between screenshots are not equal.

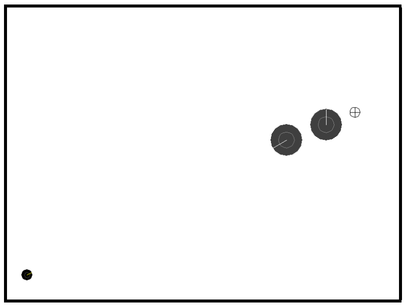


Figure 7a) - Simulation at time T1

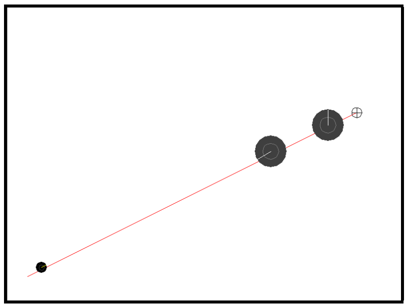


Figure 7b) - Simulation at time T2

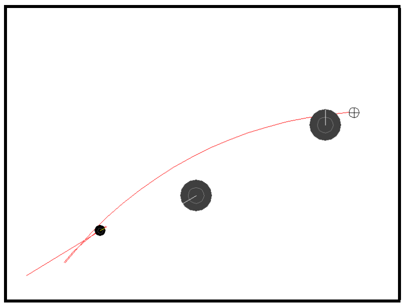


Figure 7c) - Simulation at time T3

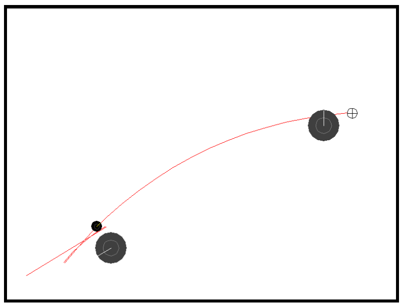


Figure 7d) - Simulation at time T4

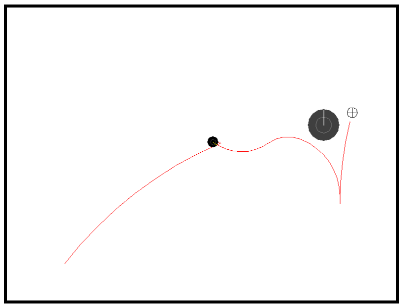


Figure 7e) - Simulation at time T5

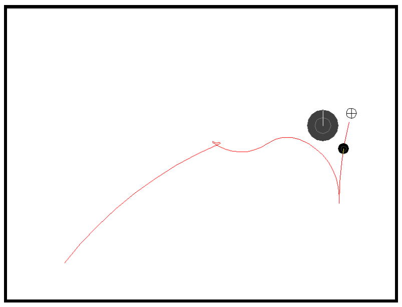


Figure 7f) - Simulation at time T6

Table 1 lists the results from 3 different simulation sets. 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 workspace with a constant velocity and don't stop or interact with other obstacles. Between each simulation set, the number of robots and obstacles were varied. Each set was run 20 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)	38.57	96.36	4.66
Average Maximum Plan Time (ms)	102.47	276.52	44.44

As shown in Table 1, the motion planning system can provide real-time 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 Table 2, the planning times are compared for simulations using the dynamic and static priority systems. Simulations demonstrated large reductions in both the average plan time (approximately 78%), and the maximum plan time (approximately 71%), when using the dynamic versus static priority systems.

Table 2 – Comparing Static and Dynamic Priority Systems

Experiment Set	**Static Priority System**	**Dynamic Priority System**
Robots	15	15
Stationary Obstacles	0	0
Moving Obstacles	0	0
Average Plan Time (ms)	4.66	1.03
Average Maximum Plan Time (ms)	44.44	15.06

Figure 8 illustrates the distribution of the number of plans each robot constructs during experiments. Plotted is the average number of plans constructed by each robot in the experiments. It is clear that the number of plans per robot is more evenly distributed when using the dynamic priority system as opposed to the static priority system. However, there is still a slight rise in the number of plans per robot as the robot number increases. This is due to the second term in the priority equation (6).

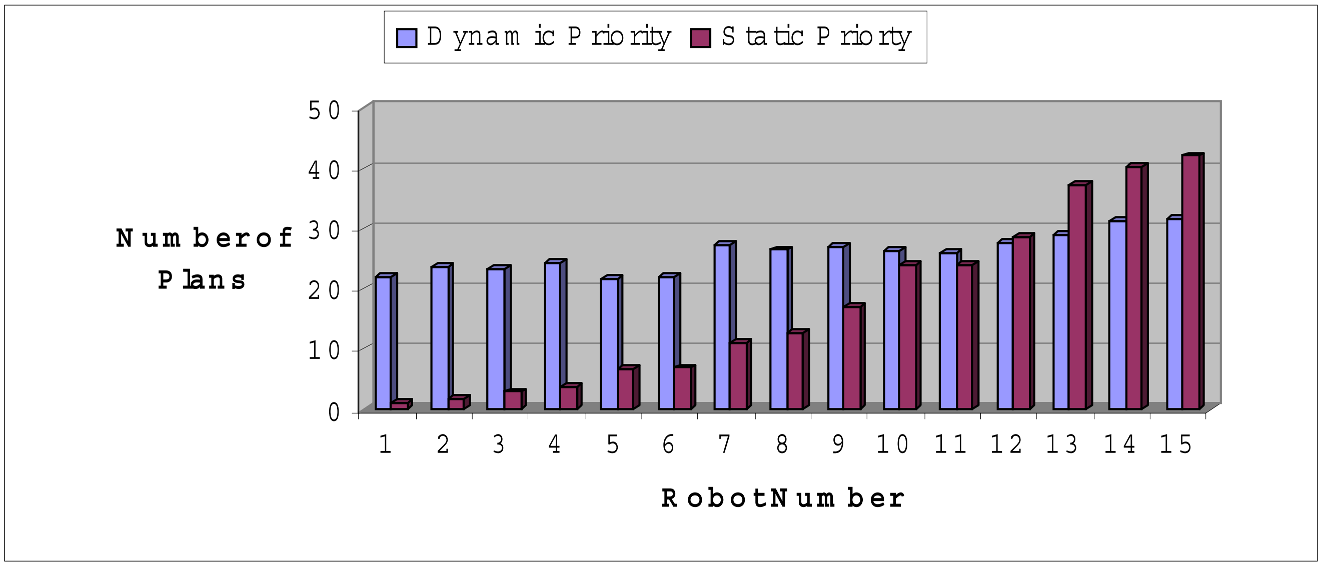


Figure 8 - Robot Planning Responsibilities for Static and Dynamic Priority Systems

In the simulations presented, the average replan times are significantly faster than the average maximum 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 less than 100 ms, allowing real-time planning.

An example of one such simulation is represented in the Figures 9a)-9f). 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.

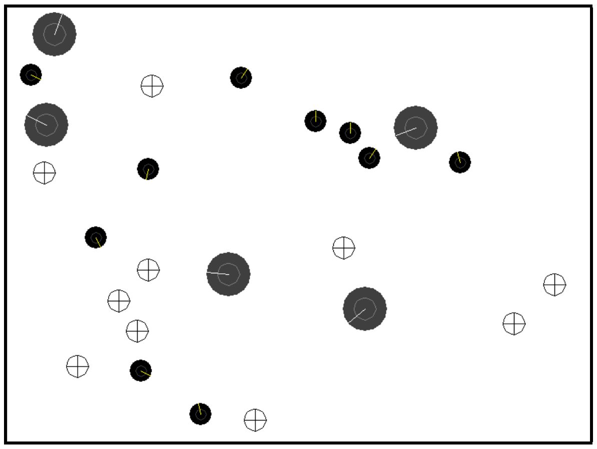


Figure 9a) - Simulation at time T1:
Rovers , goals and obstacles before the simulation.

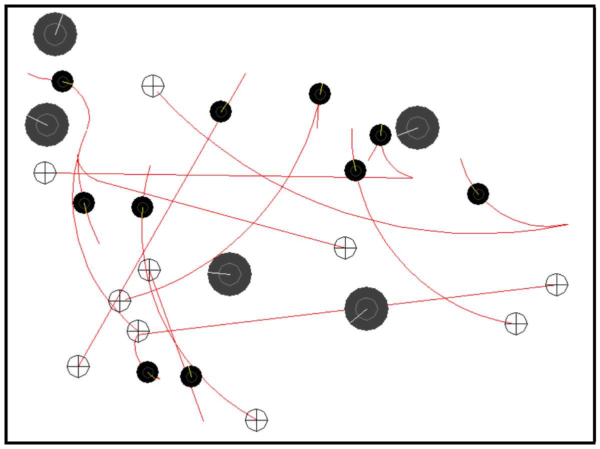


Figure 9b) - Simulation at time T2:
Rovers after constructing their first plans

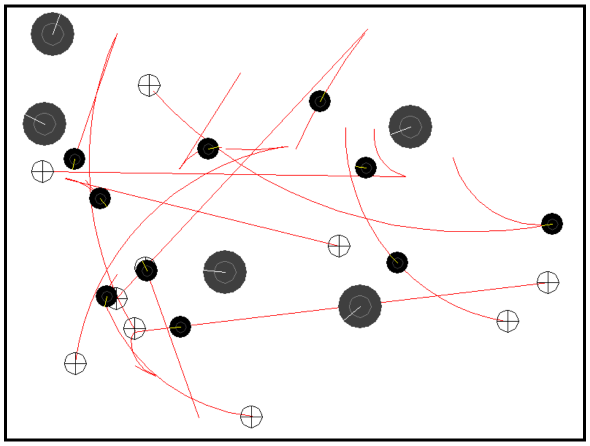


Figure 9c) - Simulation at time T3:
Rovers replanning on the fly to avoid each other

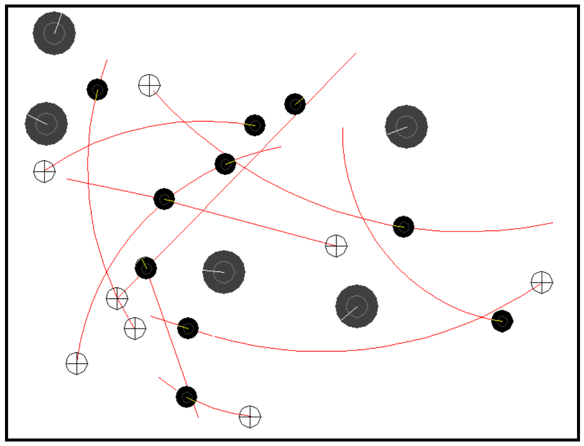


Figure 9d) - Simulation at time T4:
Rovers following their trajectories

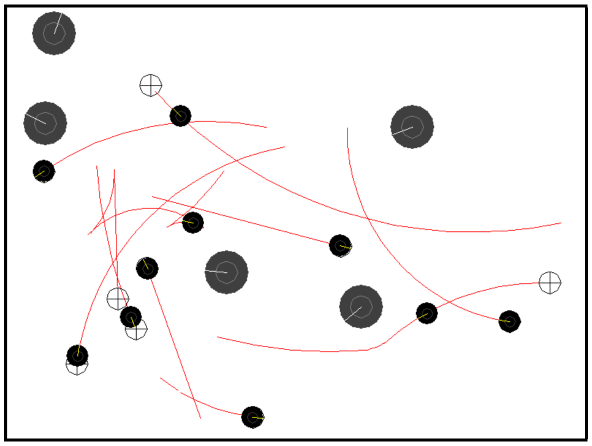


Figure 9e) - Simulation at time T5
Rovers heading towards their respective goals .

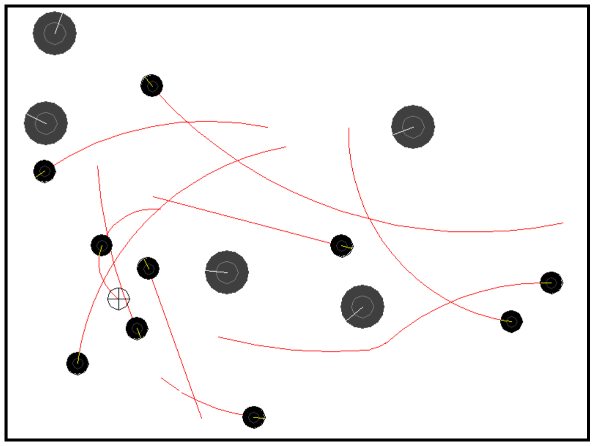
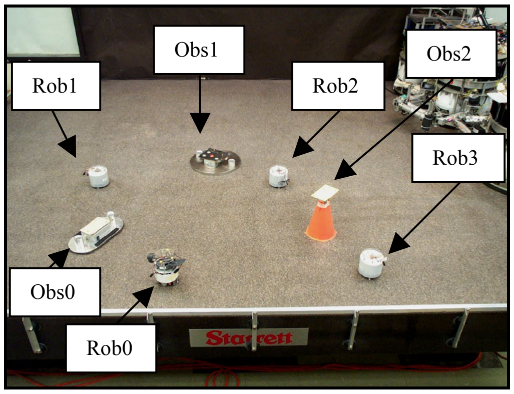


Figure 9f) - Simulation at time T6:
All but one rover having reached their goal location.



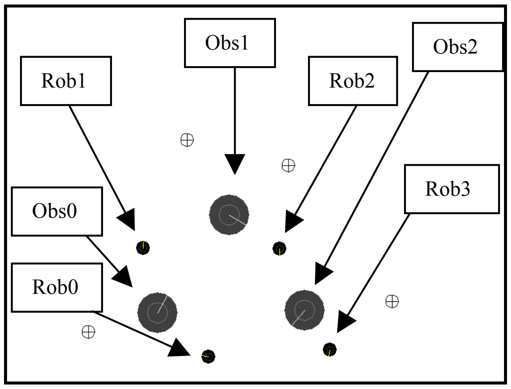


Figure 10a) - Experiment at time T1: Physical Hardware photo and corresponding GUI screenshot.

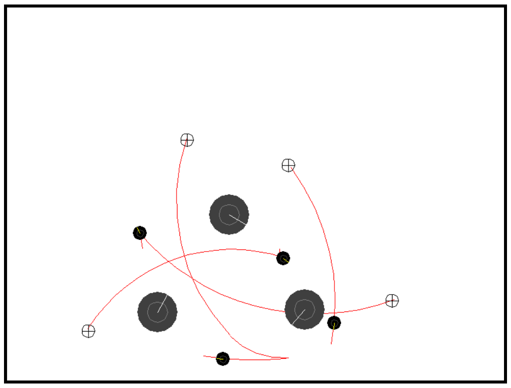


Figure 10b) - Experiment at time T2

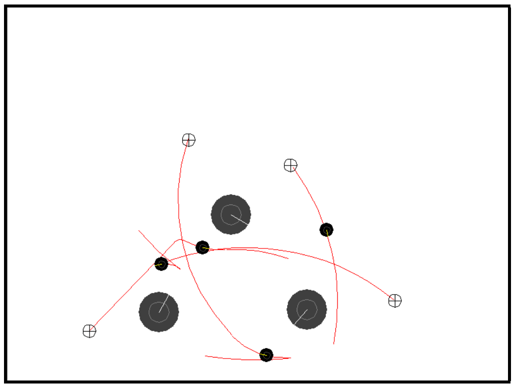


Figure 10c) - Experiment at time T3

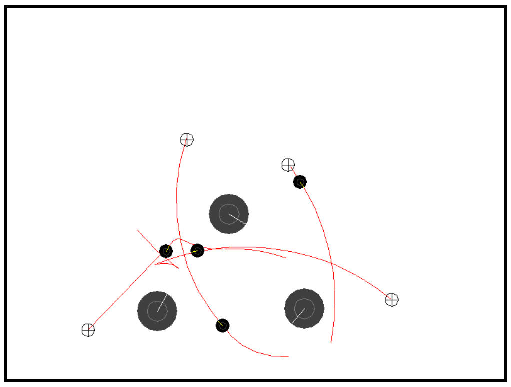


Figure 10d) - Experiment at time T4

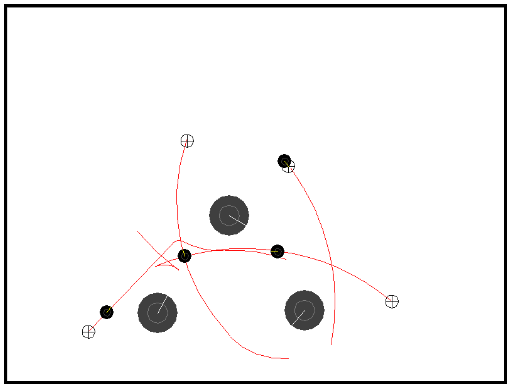
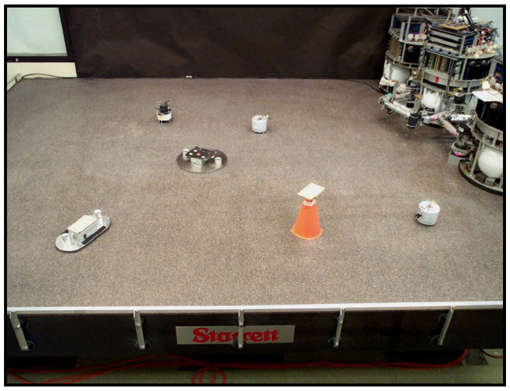


Figure 10e) - Experiment at time T5



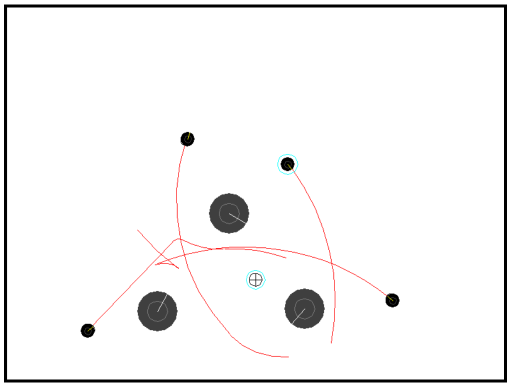


Figure 10f) - Experiment at time T6: Physical hardware and corresponding GUI screenshot.

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 10 shows a series of snap-shots taken from an experiment on the MARS test platform. In this experiment, four 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. Time steps between screenshots are not equal. What follows is an explanation of this particular experiment.

Figure 10a) depicts the rovers in their initial configuration. They are sitting stationary, awaiting the "Start" command from the GUI.

In Figure 10b), the rovers have received the "Start" command. They proceed by constructing their initial trajectories and following them.

In Figure 10c), Rovers 0 at the bottom and 3 at the right continue to follow their initial trajectories. Rover 1 has replanned because it encountered Obstacle 2 located in the bottom right. Subsequently, Rovers 1 and 2 encounter one another, leading Rover 2 (located in the middle) to replan around Rover 1.

In Figure 10d) Rovers 0 at the bottom and 3 at the right continue to follow their initial trajectories. Rover 1 (now located in the middle) has passed Rover 2 and has a clear path to its goal located at the far right. Rover 2 is following its new trajectory around Rover 1 and is heading to its goal located at the bottom left.

Finally, as depicted in Figure 10e), all rovers are finally on collision-free paths to their goal locations. Rover 3 has already reached its goal located at the top right.

5. Conclusions

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, stationary obstacles and moving obstacles. 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.

A significant improvement in performance was observed when using the new Dynamic Priority System instead of the original Static Priority System. At the cost of doubling the communication bandwidth, decreases of greater than 70% in the planning times were attained. Also, the Dynamic Priority System more evenly distributed the motion planning computation among the robots.

Future work will include investigation into other Dynamic Priority Schemes that take into account both the local and global search space. The current Dynamic Priority system calculates priorities based on the local search space only.

References

[1] M. Bennewitz and W. Burgard, "A Probabilistic Method for Planning Collision-free Trajectories of Multiple Mobile Robots," *14th European Conference on Artificial Intelligence*, 2000.

[2] M. Bennewitz, W. Burgard and S. Thrun., "Optimizing Schedules for Prioritized Path Planning of Multi-Robot Systems," *Proceedings of the International Conference on Robotics and Automation*, 2001.

[3] C. Clark and S. Rock, "Randomized Motion Planning for Groups of Nonholonomic Robots," *International Symposium of Artificial Intelligence, Robotics and Automation in Space*, 2001.

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

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

[6] 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.

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

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

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

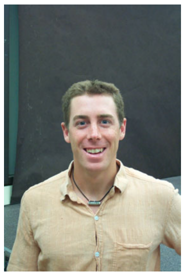
[10] V. J. Lumelsky and K. R. Harinarayan, "Decentralized Motion Planning for Multiple Mobile Robots: The Cocktail Party Model," *Autonomous Robots Journal*, No. 4, pages 121-135, 1997.

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

[12] T. Simeon, S. Leroy and J.P. Laumond, "Path Coordination for Multiple Mobile Robots: a geometric algorithm," *Proceedings of the International Joint Conference on Artificial Intelligence*, 1999.

[13] P. Svestka, and M. H. Overmars, "Coordinated Path Planning for Multiple Robots," *Technical Report UU-CS-1996-43, Utrecht University, The Netherlands*, 13, 1996.

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



__Christopher Clark__ is currently a Ph.D. candidate in the Aerospace Robotics Lab, located in the Department of Aeronautics & Astronautics, Stanford University. After finishing a B.Sc. in Engineering Physics at Queen's University, he worked as a Control Systems Designer at Sterner Automation, Toronto, Canada. He received his M.Sc. in Mechanical Engineering at the University of Toronto where he focused on the application of Neural Networks to PID controlled Robots.

__Tim Bretl__ is currently a Ph.D. candidate in the Aerospace Robotics Lab, located in the Department of Aeronautics & Astronautics, Stanford University. He graduated in 1999 with a B.S./B.A. from Swarthmore College. He received his M.S. from Stanford University in 2000. He now works in the areas of distributed planning and control.

__Dr. Stephen Rock__ received his S.B. and S.M. degrees in Mechanical Engineering from MIT in 1972. He received his Ph.D. in Applied Mechanics from Stanford University in 1978, and joined the faculty of the Aeronautics and Astronautics department of Stanford in 1988. He is the Director of the Aerospace Robotics Laboratory where his research focus is to extend the state-of-the-art in robotic vehicle control. His interests include the application of advanced control techniques for robotics and vehicle systems. Areas of emphasis include remotely operated vehicles for both space and underwater applications. Dr. Rock teaches several courses in dynamics and control at Stanford. Prior to joining the Stanford faculty, Dr. Rock managed the Controls and Instrumentation Department of Systems Control Technology, Inc.