Proceeding of the 7th International Symposium on Artificial Intelligence, Robotics and Automation in Space: i-SAIRAS 2003, NARA, Japan, May 19-23, 2003

Dynamic Networks for Motion Planning in Multi-Robot Space Systems

Christopher M. Clark & Stephen M. Rock
Aerospace Robotics Lab
Department of Aeronautics & Astronautics
Stanford University
{chrisc, rock}@sun-valley.stanford.edu

Jean-Claude Latombe
Department of Computer Science
Stanford Universty
latombe@cs.stanford.edu

KeyWords: Motion Planning, Robotics, Networks

ABSTRACT

A new motion planning framework is presented that enables multiple mobile robots with limited ranges of sensing and communication to maneuver and achieve goals safely in dynamic environments. The framework is applicable to both planetary rover and free-floating space robot applications. To combine the respective advantages of centralized and de-centralized planning, this framework is based on the concept of centralized planning within dynamic robot networks. As the robots move in their environment, localized robot groups form networks, within which world models and robot goals can be shared. Whenever a network is formed, new information becomes available to all robots in this network. With this new information, each robot uses a fast, centralized planner to compute new coordinated trajectories on the fly. Planning over multiple robot networks is decentralized and distributed. The applicability of the framework to planetary rovers is demonstrated in both simulations and real robot experiments. Also, the framework's applicability to free-floating robots in a 3D space environment is demonstrated in simulation.

1. INTRODUCTION

Concepts for future space robotic systems involve many robots under the direction of a few human operators, (e.g. assembly of large space structures [1], human-robot colonies [24]). To enable this type of human-robot operation, robots must be given a high degree of autonomy for completing tasks. Many challenges must be overcome to achieve this level of autonomy. This research focuses on one of these challenges: multi-robot motion planning.

When many robots operate in the same environment, high-level motion planning is required for the robots to accomplish tasks autonomously. They must be able to reach their goals while avoiding collisions among themselves and with static and moving obstacles. In unknown or partially known environments, it is unlikely that a system of sensors can provide global knowledge. In addition, continuous inter-robot communication is usually not feasible. Instead, only robots that are sufficiently close to each other can exchange information, e.g., share their goals and local world models.

This paper introduces a new planning framework that exploits the changing communication links between robots, as the robots move, to combine the respective advantages of centralized and decentralized planning.

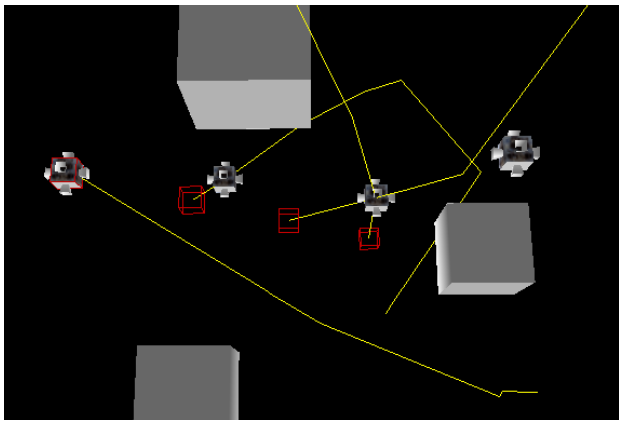


Figure 1: Motion Planning for 4 free-floating robots in a 3D space environment. Yellow lines denote robot trajectories that end at goal locations denoted by red cube lattices. The large gray cubes denote obstacles.

More precisely, our approach is based on *dynamic robot networks* that are capable of: 1) forming dynamically whenever communication and sensing capabilities permit; 2) sharing world models and robot goals within each network; and 3) constructing "on the fly" coordinated trajectories for all robots in each network using a fast centralized motion planner.

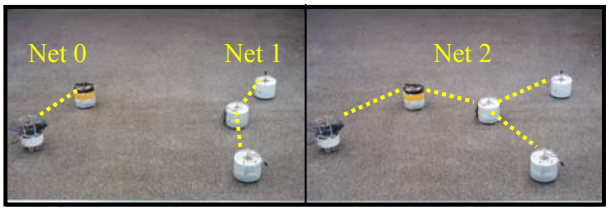
A brief overview of this approach is presented in Section 2. Then, a background review (Section 3) justifies the choices made in our approach. We then describe some aspects of our framework in more detail, namely the representation of partial world models (Section 4) and the planning technique used (Section 5). Section 6 presents the Micro-Autonomous RoverS (MARS) test-platform and discusses experiments involving rovers in a 2D workspace. In Section 7, results from free-floating space robot simulations are provided.

2. PLANNING IN DYNAMIC NETWORKS

2.1 Network Formation

When any two robots are within communication range of each other, they establish a communication link. Define G to be the graph whose nodes are the robots and edges are the communication links. A *network* of robots is any group of $k \geq 1$ robots forming a maximal connected component of G. So, any two robots in a network can communicate through one or several communication links, but two robots from different networks can not.

Figure 2a shows an environment with 5 robots, where 2 networks have formed. In Net1, the top and bottom robots can exchange information via their communication links with the middle robot. Because robots are moving to achieve their goal locations, the networks are dynamic. Robots may leave networks and/or form new networks (see Figure 2b). An application level protocol ensures that at any time robots in each network can access the local sensing information of all other robots in the same network, and hence share a common world model.



a) *b)*

Figure 2: *Example with 5 robots. Dashed lines between robots depict communication links. In a) the robots form two distinct networks Net0 and Net1. In b), two robots have moved, and the two networks in a) have merged into Net2.*

2.2 Planning Process

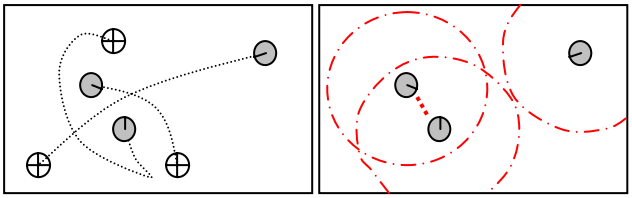
Motion planning in a network *N* is triggered by any one of the following events:

- *N* just got formed, i.e., two robots from different networks entered one another's communication range.
- A significant change in the world model occurs, e.g., a robot in *N* senses a new obstacle.
- A new goal location is requested for one or several robots in *N*.

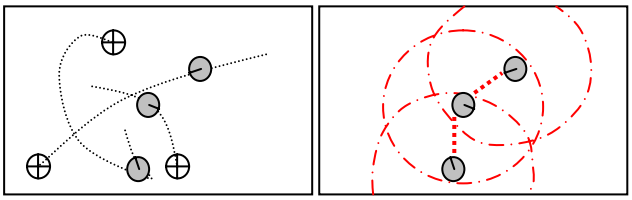
When such a triggering event occurs, data is exchanged between the robots in *N*, so that each one gets an updated world model that combines the local world model and goal of every robot. Once robots have shared this information, each robot runs its own copy of a centralized motion planner to construct coordinated trajectories for all robots in the network. When the planner terminates, each robot broadcasts its plan to all other robots in the network. Each robot selects the same best plan and immediately starts executing its trajectory in this plan. The planner is a single-query probabilistic-roadmap (PRM) planner similar to the one presented in [14] (see Section 5).

This process is illustrated in Figure 3 on a simple example involving 3 robots, with no obstacles. A triggering event automatically occurs at the start of the process, as the first networks get formed.

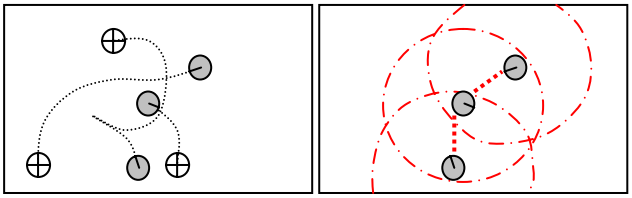
Since robots also have limited sensing, the world model shared through a network is partial. Planning is done using this model. As robots move, their sensors may detect previously unknown obstacles or a change in the trajectory followed by a known obstacle. Such an event triggers a re-plan operation within the network where the new obstacle or change of trajectory was detected.



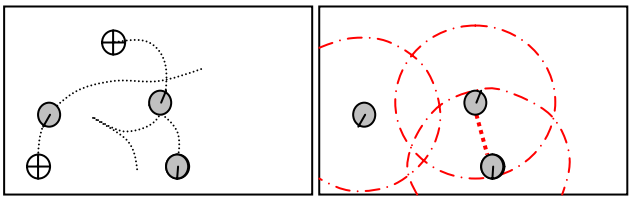
a) All three robots (grey circles) are at their initial locations. The two left robots are in communication range and form a network. Their centralized planners create coordinated collision-free trajectories for them toward the goals (cross-hairs). The right robot forms a network by itself, and its trajectory is planned independently from the other two. The robots start moving along these trajectories.



b) As the robots move along their trajectories, the middle robot and the right robot enter communication range with each other, and all three robots now form a larger network.



c) A new plan is made for all three robots in the network. This plan consists of collision-free trajectories for all three robots.



d) As robots move along their new trajectories, they leave communication range of each other and some network links are broken. They keep following the planned trajectories.

Figure 3: Top-down view of a planning example with three robots. In each of the fours snapshots, the illustration on the left shows the robots on their trajectories to their respective goals (cross-hairs). The diagram on the right depicts the communication range of each robot and the existing communication links.

3. Background Review

Most previous work on multi-robot motion planning can be grouped into *centralized* and *decentralized* planning [3,27]. While centralized planning considers all robots together as if they were forming a single multi-body robot [5,8,26,19,30,31], a decentralized planner plans for each robot separately before coordinating the individual plans by tuning the robot velocities along their respective paths [2,4,11,16,17,22,25,29]. A variant of decentralized, called *prioritizing planning*, plans for one robot at a time, in some sequence, considering the robots whose trajectories have already been planned as moving obstacles [6,12].

Centralized planners can be advantageous because they allow the possibility of completeness and global optimization. For example, it was shown in [27] that a centralized planner based on PRM techniques can reliably solve problems requiring the tight coordination of multiple articulated arms, while decentralized planners based on similar PRM techniques fail often. On the other hand, centralized planning may take more time due to the high dimensionality of the configuration spaces that are searched. A worse drawback is that they require all information (partial world models and robot goals) to be centralized in one single place, which is only possible if the robots have unlimited communication abilities. This is not the case in many practical settings.

A major advantage of decentralized planning is that it allows for distributed planning. Each robot can then plan its own trajectory using its own partial model of the environment. If two robots eventually get close to one another and risk colliding, simple velocity-tuning techniques or other reactive techniques can be used to locally coordinate their motions. However, such a fully distributed approach fails to exploit the fact that localized groups of robots can exchange information to improve planning

By searching several configuration spaces of smaller dimensionality, decentralized planning is potentially less computationally intensive. But it cannot offer any completeness or optimality guarantee. Various attempts have been made to improve the outcome of decentralized planners (e.g., [4,6,13]). In particular, a negotiation scheme between localized groups of robots is used in [4] to assign priority orders to robots, which allow the decentralized planner to compute trajectories of reduced lengths. This negotiation scheme demonstrates the benefits of localized inter-robot communication, and is the technique most closely related to the robot network planning framework presented in this paper. However de-centralized planning remains intrinsically incomplete.

The planning approach presented in this paper exploits the respective advantages of centralized and decentralized planning. In each robot network, it uses a centralized single-query PRM planner to increase completeness and still provide fast on-the-fly planning. However, planning is distributed over the various networks – hence, planning over multiple networks is decentralized – to accommodate the fact that robots from different networks cannot share information. The triggering event caused by the merging of two previously distinct networks into a single network leads the robots in this new network to take advantage of the information they now share by centrally re-planning their coordinated trajectories.

Planning with incomplete world models and on-the-fly re-planning when a sensor detects the presence of a still unknown obstacle or a change in an obstacle's trajectory have previously been described in [14, 18] for a single robot. We use similar techniques, but extend them to multiple robot networks.

4. WORLD MODEL

Describing the world model in a concise but useful form is necessary to allow for information sharing between robots in the same network. In the experimental system that we have built, world models simply consist of a list of robots and their descriptions, and a list of obstacles and their descriptions. The following table outlines the information stored in each list:

World Model Description

1) List of Robot Descriptions
   - State (position and velocity)
   - Size (Radius)
   - Most Recent Update Time
   - Information Source
   - Goal position
   - Current Trajectory

2) List of Obstacle Descriptions
   - State (position and velocity)
   - Size (Radius)
   - Most Recent Update Time
   - Information Source

Robots report their own size and state, while obstacle sizes and states are estimated by robot sensors. The most recent update time is useful when updating world models with information received from other robots. The information source is a robot identification number that indicates which robot sensed (or communicated with) the object. It is used to keep track of which robots are currently in the network.

Several assumptions were made to allow such a concise world model:

- Each robot has access to its own state relative to a global coordinate system (e.g., GPS).
- Each object is approximated as a circular object to allow its geometry to be described by a single parameter, its radius.
- Each obstacle has constant linear velocity estimated by a robot's sensor. As in [11], if at any later time its trajectory is found to diverge by more than some threshold from the predicted trajectory (either because the obstacle did not move at constant velocity, or because the error in the velocity estimate was too high), then the robot that detects this divergence calls for the construction of a new plan within its network. The planner "grows" the obstacles (and the robots) to allow for some errors in predicted trajectories of the objects.
- All objects in the environment are easily identifiable by robot sensors, which can also precisely estimate their positions and velocities. Any discrepancy between two local world models can be easily resolved.

The second assumption is rather easy to eliminate, as it has been shown before that PRM planners can efficiently deal with geometrically complex robots and obstacles (e.g., [26]). In [14], the third assumption has been shown to be quite reasonable, even when obstacle velocities change frequently, provided that (re-) planning is fast enough. The last assumption is more crucial. In our experimental system, it is enforced by engineering the vision system appropriately (see Section 6.2). In the future, it will be important to relax this assumption by using more general sensing systems and data fusion techniques [23].

5. MOTION PLANNING ALGORITHM

As indicated earlier, motion planning within a robot network is done using a centralized single-query PRM planner (more precisely, several copies of this planner running in parallel). This planner searches the joint state×time space C of the k robots in this network. The state of each robot is defined by the two coordinates of its center and two velocity parameters, so C has $4k+1$ dimensions. This representation can easily be extended to other robots. For instance, we have implemented a version of the planner for robots in three-dimensional space [10]. The planner searches C for a collision-free trajectory from the initial state of the robots to their goal state. The resulting trajectory defines the coordinated motions of the robots to their respective goals.

Our planner searches C by incrementally building a tree of milestones (the roadmap), as described in [14,15,19]. At each iteration, it selects a milestone m in the current roadmap, generates a collision-free state m' at random in a neighborhood of m in C and, if the path from m to m' tests collision-free, installs m' as a new milestone in the roadmap. The search terminates when m' falls into an "endgame" region around the goal. See [14] for details.

As in [14,28], our planner satisfies kinodynamic constraints as follows: to generate each new milestone m', it picks a control input at random and integrates the equations of motion of the robots over a short duration.

We name our planner Kinodynamic Randomized Motion Planning (KRMP). As shown in [14], under reasonable assumptions on the free space, the probability of not finding a plan when one exists decreases exponentially to 0 as the number of milestones increases. This is a major advantage over our previous work in [9,11], which used a decentralized prioritized planning approach**.** Note, however, that the fact that the planner is probabilistically complete does not imply that the entire system is also probabilistically complete. The robots use partial world models and thus need to re-plan their trajectories when they encounter discrepancies in their model, (e.g. new obstacles). Since there is no guarantee that a series of complete plans is itself a complete plan, the robots are not guaranteed to find a global plan if one exists. While it is unclear to what extent the notion of completeness applies when planning for global goals with only partial knowledge of the environment, it is still desirable to achieve completeness in the system's components whenever this is possible.

The work in [14] also demonstrated empirically that the above techniques successfully compute trajectories for a single robot with kinodynamic motion constraints, in real-time, (i.e. fast enough to be run on the fly). To enable motion planning within robot networks, KRMP extends this previous work to accommodate several robots. Modified techniques are needed to 1) select milestones for expansion, 2) generate new milestones, and 3) define the endgame region. Below we present only the technique we use to generate a new milestone m'. Not all modifications are presented in this paper.

When planning for multiple robots, one may generate m' using the following "parallel" approach: first, pick the control inputs for all the robots at random; next, integrate the motions of all the robots concurrently; if no collision is detected, then record the endpoint as a new milestone, otherwise pick another set of control inputs. We found that this technique yields a high rejection rate, especially in tight space. This led us to develop the following "sequential" approach: consider the robots in some order, pick the control input for each robot and integrate its motion (while considering the previous robots as moving obstacles); if the motion collides, pick another control or change the motion of a previous robot. Our experiments show that this sequential approach makes it possible to get each new milestone much faster, without affecting the probabilistic completeness of the overall planner.

Finally, we take advantage of the various processors available in a robot network by concurrently running a separate copy of KRMP on each robot of the network. Each copy uses a different seed of the random number generator, hence constructs different roadmaps. We set the same timeout constraint (typically, a small fraction of a second) on every robot. Each robot then returns a plan or its failure to generate one. The same best plan is selected by the robots and each robot immediately switches to executing its new trajectory. This is made possible because we use a PRM planning approach.

6. ROVER PLANNING

To validate our planning approach, it was implemented on the MARS test-platform. This section describes the hardware used for rover experiments, followed by a brief summary of experimental results. For details about the implementation of the planner on planetary rovers, refer to [11].

6.1 Micro-Autonomous RoverS Test-Platform

Located in the Aerospace Robotics Lab at Stanford University, the Micro-Autonomous RoverS (MARS) test-platform is used to model mobile robots in a two-dimensional workspace. The platform consists of a large 12' x 9' 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 (nonholonomic constraint). Each robot is equipped with its own planner (copy of KRMP) and controller that are located off-board.

An overhead vision system is used to track the states of all objects on the table. The vision system processor calculates these states and publishes them to all applications that subscribe (see Figure 4). This makes global state information available to all robots. To simulate the limited sensing range that would occur when sensors are mounted on robots, the object states are filtered such that robots only receive state information regarding objects within some predetermined range of the robot.

Figure 4 shows the computer/network architecture of the MARS test-platform. All the processing is done off-board. Two processors are assigned to each robot, respectively for planning and control. These computers are connected through a LAN. All communication within the LAN is accomplished with Real Time Innovation's Network Data Delivery Service (NDDS) software. Because a LAN is used for inter-robot communication instead of a wireless medium, there are no physical barriers to limit the range of communication. Hence the communication barrier is simulated.

NDDS is based on a publish/subscribe architecture. To broadcast messages by flooding a robot network, the sender will publish a message to which all robots subscribe. Before robots can receive their subscriptions, the messages are filtered so that only robots within some predetermined range of the sender will receive the message. This effectively simulates a discrete physical communication range.

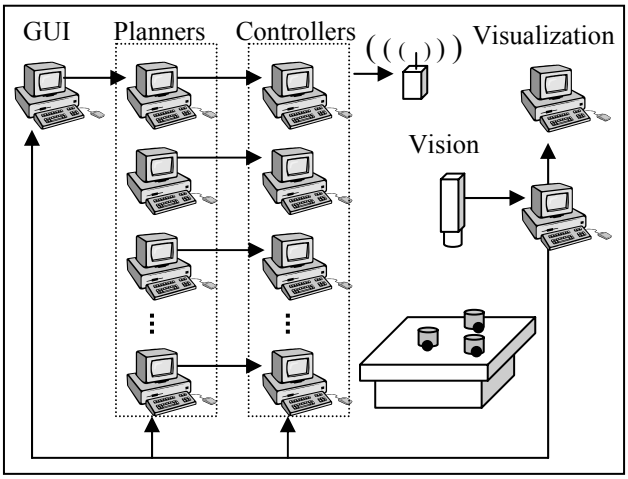


Figure 4: Network architecture of MARS test-platform

6.2 Planetary Rover Experiments

To illustrate the applicability of the planner to a physical system, real robot experiments with up to 5 robots have been carried out. One example of such an experiment is illustrated in Figure 5. The top photo is a screenshot of the GUI taken at one point in the experiment. The bottom photo shows the physical hardware, and was taken at the same time as the GUI screenshot. In the GUI, robots and objects are depicted as small and large circles, respectively. Robot goal locations are indicated by cross-hairs, and lines leading to the goal locations depict the trajectories. When robots form a network as described in Section 2, it is indicated by a color change. Hence robots within a network have a common color, and this color will differ between networks.

In the experiment presented, all five robots are initially located at the near end of the table (i.e. bottom of the GUI screen). Communication and sensing ranges were limited to 0.75 m. Robot colors indicate that 2 networks have formed, one with the 2 robots in the bottom left and one with the 2 robots in the bottom right. As the experiment progresses, the robots follow their trajectories to reach their goal locations at the far end of the table. Throughout the experiment, robots planned an average of 3.4 times, and planning times were an average of 9 ms.

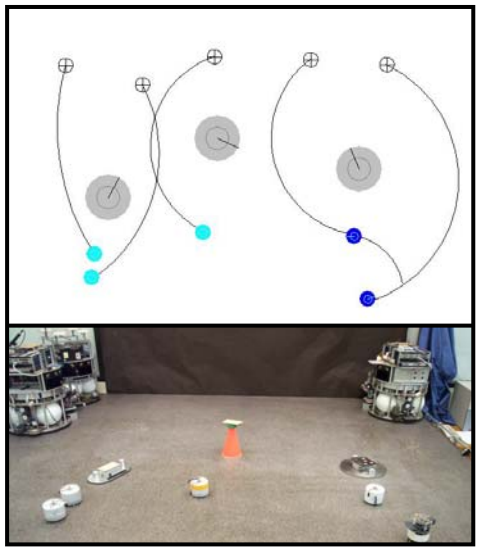


Figure 5: Example experiment on the MARS test-platform involving 5 robots and 3 obstacles.

Successful planning was demonstrated in more complex simulations involving up to 12 planetary rovers in a bounded workspace that contained 12 static and moving obstacles, (see [10]).

7. FREE-FLOATING ROBOT PLANNING

This section details planner implementation issues, provides a brief note on the 3D visualization, and gives results from simulations of free-floating robots in a 3D environment.

7.1 Free-Floating Robots Planner Implementation

Free-Floating Robot Model - A simple cube-shaped robot equipped with 6 independent on/off thrusters was used to model a space robot in simulation. Note this

does not allow for any change in orientation. Future work will include additional thrusters to allow roll, pitch and yaw variation.

The state of the robots can be described by $X = (x_1, x_2, x_3) \in \Re^3$ representing the position with respect to the inertial frame. Milestones are specified by the state of the k robots at a particular time, $(X_0, X_1, \ldots, X_k, t)$.

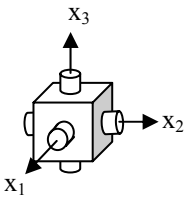


Figure 6: State-space model of the free-floating robot

Milestone Generation - To generate a new milestone for the road map, thruster control inputs are randomly selected that will propagate robots to new states. First, the time for which the thrusters will be actuated, (t_{act}), is randomly selected where:

$$t_{act} \in [t_{\min}, t_{\max}]$$

Next, the control inputs (ON/OFF) are randomly selected for each thruster. We assume that only one of two opposite-facing thrusters should be enabled at the same time. This can reduce the number of random variables. That is, for each pair of opposite-facing thrusters, a control input variable u_{act} is selected where:

$$u_{act} \in \{1, -1, 0\}$$

u_{act}	1	-1	0
Thruster 1	ON	OFF	OFF
Thruster 2	OFF	ON	OFF

Table 1: Mapping the random variable u_{act} to thruster actuation.

With the random variables selected, a candidate milestone m can be generated. Given any parent milestone m_p, and using $1/s^2$ dynamics, robot states in m can be easily calculated:

$$x_i = \frac{u_{act\,i}}{2M} t_{act}^2 + \dot{x}_{i,p} t_{act} + x_{i,p}$$

$$\dot{x}_i = \frac{u_{act\,i}}{M} t_{act} + \dot{x}_{i,p}$$

Endgame Region - *The* endgame region E in this implementation is defined as the subspace that includes all milestones m_e, such that all robots can be propagated without collisions from states defined by m_e to their respective goal location via a bang-off-bang control sequence. An advantage this sequence has is that it allows us to limit the velocity of the robot, making it easier to re-plan in the future.

To implement this in practice, one must create a list of milestones to get from m_e to m_g, the milestone defining the goal states of each robot. Each milestone in this list corresponds with the change in actuation necessary for obtaining the bang-off-bang control sequence.

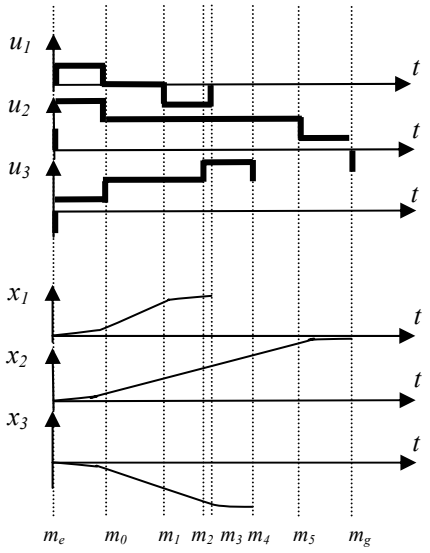


Figure 7: Example of actuation required to move one robot from (0, 0, 0) to a goal state. The series of milestones required is {m_p, m_0, m_1, mm_3, m_4, m_5, m_g}

7.2 Free-Floating Robot Simulations

To simulate motion planning experiments that involve free-floating robots maneuvering in a 3D space environment, the test platform described in Section 6 was augmented to incorporate a 3D visualization. The application was coded in C++ and OpenGl. A screenshot from the application can be seen in Figures 1 and 9. As depicted in Figure 4, the application acts only as a listener to receive state information.

The applicability of the planner to a 3D environment was validated with simulations that include up to 8 robots and 8 obstacles. A test scenario is provided in which robots must cross paths several times. A GUI screenshot of the scenario is provided in Figure 8, (Note that the third dimension is not displayed here.)

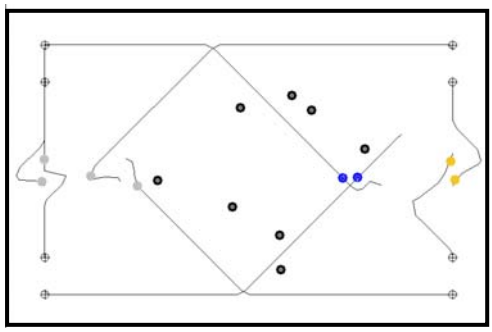


Figure 8: A test scenario involving 8 robots and 8 obstacles. On the left, four robots denoted by light gray circles have formed a network. On the right, two networks have formed, each with two robots, (denoted by blue and yellow circles). Lines indicate trajectories. Obstacles are depicted as dark circles.

The test scenario was simulated 25 times to produce the results in Table 2. From these results it is clear that the planner was capable of planning on the fly with average planning times of 67 ms. An average of 12.2 networks were formed throughout each simulation.

Avg. number of plans made by each robot	4.77
Avg. number of robots in each plan	1.84
Avg. planning time (ms)	67.0
Avg. number of networks formed	12.2

Table 2: Data from 25 simulations of the test scenario depicted in Figure 8.

Provided in Figure 9 is a visualization of a simulation of 4 free-floating robots planning in a bounded 3D environment that contains 4 obstacles. The robots have formed a network and carried out centralized planning to construct trajectories to their respective goals. The view has been rotated between screenshots in a clockwise direction to provide different points of view of the simulation.

8. CONCLUSIONS

The motion planning framework presented has demonstrated its effectiveness in planning for multiple mobile robots within a bounded workspace. It plans with a high probability of success in environments involving robots, stationary obstacles and moving obstacles. Planning times of less than 100 ms allowed the robots to re-plan on the fly and react in real-time to changes in the environment.

Future work includes incorporating more sophisticated methods of modeling the environment into the communication system. Another future direction will be to investigate the effects of varying the ratio between sensor range and communication range. For the application to the three-dimensional workspaces, planning for all degrees of freedom should be incorporated.

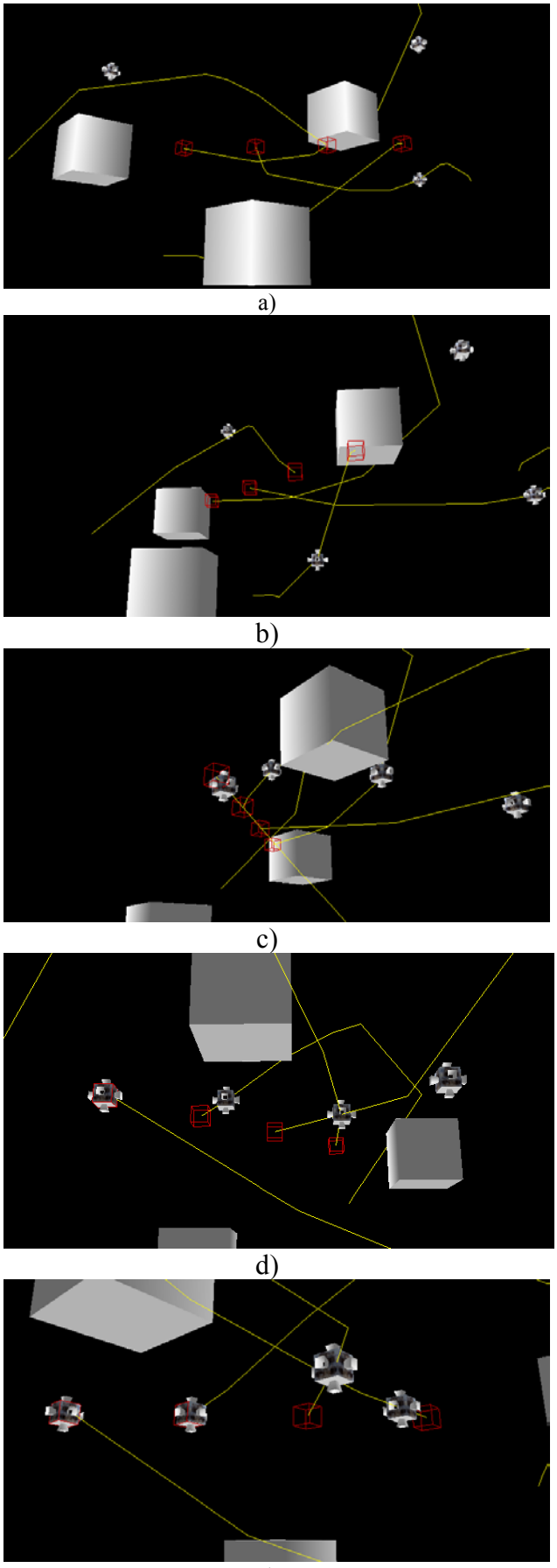


Figure 9: Visualizing a simulation involving 4 robots and 4 obstacles. Large gray cubes denote the obstacles. Trajectories are denoted by yellow lines that end at robot goal locations, (denoted by red cube lattices).

BIBLIOGRAPHY

[1] D. L. Akin, M. L. Bowden. EVA, Robotic, and Cooperative Assembly of Large Sapce Structures, *Proc. IEEE Aerospace Conference*, 2002.

[2]R. Alami, F. Robert, F. Ingrand, & S.Suzuki. Multi-Robot Cooperation Through Incremental Plan-Merging, *Proc. IEEE Int. Conf. on Robotics and Automation*, p. 2573-2678, 1995.

[3] T. Arai & J. Ota. Motion Planning of multiple mobile robots. *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Syst.*, p. 1761-1768, 1992.

[4] K. Azarm & G. Schmidt. Conflict-Free Motion of Multiple Mobile Robots Based on Decentralized Motion Planning and Negotiation, *Proc. IEEE Int. Conf. on Robotics and Automation*, p. 3526-3533, 1997.

[5] J. Barraquand, B. Langlois, & J.C. Latombe. Numerical Potential Field techniques for Robot Path Planning, *IEEE Tr. On Syst., Man, and Cyb.*, 22(2):224-241, 1992.

[6] M. Bennewitz, W. Burgard & S. Thrun. Optimizing Schedules for Prioritized Path Planning of Multi-Robot Systems, *Proc. Int. Conf. on Robotics and Automation*, 2001.

[7] Z. Bien & Jihong Lee. A Minimum-Time Trajectory Planning Method for Two Robots, *IEEE Tr. on Robotics and Automation*, pg 443-450, 1992.

[8] S.J. Buckley. Fast Motion Planning for Multiple Moving Robots. *Proc. IEEE Int. Conf. on Robotics and Autom.*, p. 1419-1424, 1989.

[9] C. Clark & S. Rock. Randomized Motion Planning for Groups of Nonholonomic Robots, *Proc. Int. Symp. of Artificial Intelligence, Robotics and Automation in Space*, 2001.

[10] C. Clark, S. Rock & J. C. Latombe. Dynamic Networks for Motion Planning in Multi-Robot Space Systems, *Proc. Int. Symp. of Artificial Intelligence, Robotics and Automation in Space*, 2003.

[11] C. Clark, T. Bretl, & S. Rock. Kinodynamic Randomized Motion Planning for Multi-Robot Space Systems, *Proc. of IEEE Aerospace Conf.*, 2002.

[12] M. Erdmann & T. Lozano-Perez. On Multiple Moving Objects, *Proc. IEEE Int. Conf. on Robotics and Automation*, p. 1419-1424, 1986.

[13] Y. Guo & L. E. Parker. A Distributed and Optimal Motion Planning Approach for Multiple Mobile Robots, *Proc. IEEE Int. Conf. on Robotics and Automation*, p. 2612-2619, 2002.

[14] D. Hsu, R. Kindel, J.C. Latombe, & S. Rock. Randomized Kinodynamic Motion Planning with Moving Obstacles, *Int. J. of Robotics Research*, 21(3):233-255, March 2002.

[15] D. Hsu, J.C. Latombe, & R. Motwani. Path planning in expansive configuration spaces, *Proc. IEEE Int. Conf. on Robotics and Automation*, p. 2719-2726, 1997.

[16] K. Kant & S. Zucker. Toward efficient Trajectory Planning: The path-velocity decomposition, *Int. J. of Robotics Research*, 5(3):72-89,1986.

[17] S. Kato, S. Nishiyama, & J. Takeno. Coordinating mobile robots by applying traffic rules, *Proc. IEEE/RSH Int. Conf. on Intelligent Robots and Systems*, p. 1535-1541, 1992.

[18] J.J. Kuffner. *Autonomous Agents for Real-Time Animation*. PhD Thesis, Computer Science Dept., Stanford U., 1999.

[19] S.M. LaValle & S.A. Hutchinson. Optimal Motion Planning for Multiple Robots Having Independent Goals, *IEEE Tr. on Robotics and Automation*, 14:912-925, 1998.

[20] S.M. LaValle & J.J. Kufner. Randomized Kinodynamic Planning," *Int. J. of Robotics Research*, 20(5):278-300, 2001.

[21] Lee, Lee, & Park. Trajectory Generation and Motion Tracking for the Robot Soccer Game, *Proc. IEEE Int. Conf. on Intelligent Robots and Systems*, p. 1149-1154, 1999.

[22] V.J. Lumelsky & K.R. Harinarayan. Decentralized Motion Planning for Multiple Mobile Robots: The Cocktail Party Model, *Autonomous Robots J.*, 4:121-135, 1997.

[23] P. Moutarlier & R. Chatila. Stochastic Multisensory Data Fusion for Mobile Robot Location and Environment Modelling. *Proc. Int. Symp. on Robotics Research, Tokyo*, 1989.

[24] P. S. Schenker, T. L. Huntsberger, P. Pirjanian & E. T. Baumgartner. Planetary Rover Developments Supporting Mars Exploration, Sample Return and Future Human-Robotic Colonization, *Proc. 10th Conf. on Advanced Robotics*, p. 31-47, 2001.

[25] D. Parsons & J. Canny. A Motion Planner for Multiple Mobile Robots, *Proc. IEEE Int. Conf. on Robotics and Autom.*, p. 8-13, 1992

[26] G. Sánchez & J.C. Latombe. On Delaying Collision Checking in PRM Planning : Application to Multi-Robot Coordination, *Int. J. of Robotics Research*, 21(1):5-26, Jan. 2002.

[27] G. Sánchez-Ante & J.C. Latombe. Using a PRM Planner to Compare Centralized and Decoupled Planning for Multi-Robot Systems, *Proc. IEEE Int. Conf. on Robotics and Autom..,* 2002.

[28] S. Sekhavat, P. Svetska, J.P. Laumond, & M.H. Overmars. Multilevel Path Planning for Nonholonomic Robots Using Semiholonomic Subsystems. *Int. J. of Robotics Res.*, 17:840-857, 1998.

[29] T. Siméon, S. Leroy, & J.P. Laumond. Path Coordination for Multiple Mobile Robots: a Geometric Algorithm, *Proc. Int. Joint Conf. on Artificial Intelligence*, 1999.

[30] P. Svestka, & M.H. Overmars. Coordinated Motion Planning for Multiple Car-Like Robots Using Probabilistic Roadmaps, *Proc. IEEE Int. Conf. on Robotics and Autom., p. 1631-1636, 1995.*

[31] C.W. Warren. Multiple Path Coordination using Artificial Potential Fields, *Proc. IEEE In. Conf. on Robotics and Autom.*, p. 500-505, 1990.