DYNAMIC ROBOT NETWORKS: A COORDINATION PLATFORM FOR MULTI-ROBOT SYSTEMS

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF AERONAUTICS AND ASTRONAUTICS
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Christopher Michael Clark
May 2004

Abstract

A large number of tasks, from manufacturing to planetary exploration, have been successfully accomplished using single robot systems. Many of these tasks could be completed faster, more reliably, and on a larger scale using a cooperating team of autonomous mobile robots. However, robots must be able to coordinate their actions before cooperation is possible.

This work aims to enable robots with the ability to coordinate their actions for safe navigation in dynamic, unknown environments. Specifically, the work focuses on: 1) the coordination of multiple robots when sensing and inter-robot communication are limited and 2) multi-robot motion planning in dynamic, unknown environments.

First, a new coordination platform is introduced - *Dynamic Robot Networks* - that facilitates centralized robot coordination across ad hoc networks. As robots move about their environment, they dynamically form communication networks. Within these networks, robots can share local sensing information and coordinate the actions of all robots in the network.

Second, a fast motion planner called within robot networks is presented. The planner is a probabilistic roadmap (PRM) motion planner augmented with new sampling strategies. These strategies decrease the planner's run time to enable on-the-fly planning - a key requirement for navigation in environments that are unknown a priori and contain moving obstacles.

Simulations and real robot experiments are presented that demonstrate: 1) centralized robot coordination across dynamic robot networks, 2) on-the-fly motion planning to avoid moving and previously unknown obstacles, and 3) autonomous robot navigation towards individual goal locations.

Acknowledgments

Throughout my time at Stanford University, I have been lucky to be a member of the Aerospace Robotics Laboratory (ARL). Professor Robert Cannon first invited me to join the ARL in 1998, and I am indebted to him for this reason. He is a great engineer, role-model and story-teller.

Students in the ARL made it a great place to work. As both researchers and collaborators, the ARL gang made my Ph.D. experience truly unique. In particular Tim Bretl and Jack Langelaan deserve to be recognized for their friendship and integral contributions during the most difficult and frustrating years of this process. Tim has been both a great climbing partner and research partner. Jack is one of the most dependable people I know.

The staff of the ARL as well as the Aeronautics and Astronautics Department are a wonderful group of people, who do an incredible amount of work for the students. In particular, I want to thank Godwin Zhang, Jane Lintott, Sally Gressens, Aldo Rossi, Shreann Ellsworth and Dana Parga for all their help and for their encouragement.

I want to thank Professor Rock for advising me and funding my research throughout my time at Stanford. He gave me the freedom and the encouragement to pursue my own ideas and to manage my own research. Most notably, he provided a family-friendly environment that helped me balance my family life with my studies.

Professor Jean-Claude Latombe provided me with the inspiration for my research directions. He is an incredible source of knowledge, and provided excellent guidance on technical details and publication writing. I am especially thankful for all the time he provided me in reworking this dissertation.

Professor Claire Tomlin sat on both my defense and reading committees. She was able to provide in-depth analysis of my work, despite my late hour requests. She is also my favorite instructor at Stanford.

I am grateful to Professor Gunter Niemeyer for sitting in on my defense committee with such short notice. I appreciate the time and effort as well as his insightful comments.

Various organizations have provided resources, financial and otherwise, to support my research. I would like to thank the people at Real-Time Innovations (RTI), whom I also got a chance to work with. In particular, I would like to thank Arnab, Rajive, Gerardo and Stan. I sincerely appreciate their support.

Those who deserve my greatest thanks are my family. My mother Elaine provided an incredible support net while growing up. My father Albert showed me how to get here by setting a perfect example. My brothers and sisters: Eric, Kevin, Laurie, Raymund and Sen-Mei have all encouraged me along the way.

My second parents, Romel and Lilia, have always helped out whenever asked. They visited us on a regular basis and helped watch our kids whenever I had an important deadline.

My two children were both born at Stanford. My son Jaden gave me inspiration and happiness whenever times were difficult. Everyday he made me laugh. My recently born daughter, Sequoia, is a true joy.

Most importantly, my wife Christine, gave love, support and sacrifice during what most consider a very selfish endeavor. Without her support, I wouldn't have even applied to Stanford. Her counselling always helped me put things in perspective and she always took care of our family (including me) when times were tough. She made my Stanford experience enjoyable, fun and full of love.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Multi-robot systems provide an exciting solution to many real-world problems. Multiple robots can cooperate to manipulate large objects, survey large areas in a short amount of time, and provide system redundancy. These functionalities make them applicable to a variety of tasks including large-scale construction [1], hazardous waste cleanup [52], and planetary exploration [56].

To enable multiple robots to cooperate and gain additional functionality over single robots, several technical difficulties must be overcome. These include enabling dependable inter-robot communication, fusing sensor data from multiple robots, establishing a group architecture that allows for the desired cooperation, designing a software architecture to implement the group architecture, providing a user interface to command robots, and providing a method for coordinating robot actions.

Of these difficulties, this dissertation focusses on robot coordination - the execution of simultaneous robot actions without conflict. For example, in robot *motion planning*, robots must execute their maneuvers simultaneously without colliding.

Robot coordination is especially difficult within environments that are dynamic and unknown a priori. For robots to coordinate within such environments, two key issues must be addressed: limitations in robot sensing and limitations in robot communication. This dissertation presents 1) a new robot coordination platform called Dynamic Robot Networks to enable coordination despite such limitations, and 2) a new motion planner that operates within that platform.

1.1 Related Work

There exists a large body of literature on multi-robot systems. Some research has focused on system architectures (e.g [59, 44]). Other research has focused on enabling specific functionalities. Examples include coordinating robots for large object manipulation [56, 23], searching large areas for sites of interest with robot formations [16, 19, 53], sensor network deployment[65, 17], and large area mapping [22, 63]. Most related to this dissertation is research on motion planning [3, 5, 6, 8, 9, 14, 24, 31, 37, 50, 57, 58, 62, 64], (see Chapter 3 for a more thorough review).

Regardless of the purpose, the method of coordinating robots will depend heavily on the group architecture of the multi-robot system. Different architectures allow for different coordination algorithms. For example, it is impossible to implement a coordination algorithm that plans actions for all robots, when the architecture does not permit communication between all robots.

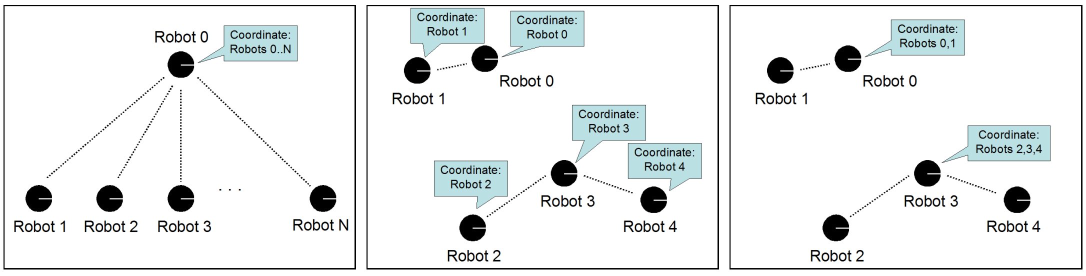
The following section discusses such problems and how group architectures affect robot coordination in general.

1.1.1 Coordination within a Group Architecture

The group architecture of a system "provides the infrastructure upon which collective behaviors are implemented and determines the capabilities and limitations of the system" [12]. Thus the selection of an appropriate architecture is essential to mission success and will depend on the application of interest.

Desired is an architecture that is scalable, fault-tolerant, and allows for centralized robot coordination. When coordination is centralized, the actions of all robots can be taken into account when planning the actions of individual robots. This ensures the avoidance of any robot conflicts (e.g. robots will not collide).

These desired attributes are directly related to two characteristics of an architecture design: centralization/decentralization and communication structure.



(a) (b) (c)

Figure 1.1: Centralization versus Decentralization

Centralized vs. Decentralized

Most system architectures are classified as being centralized or decentralized. In centralized architectures, there exists a single agent that controls the robots. In decentralized architectures, control responsibilities are divided among the robots.

Within a centralized architecture, a single central agent will have information about the entire system and will control all agents in the system [40, 52]. Because the central agent has complete information, centralized coordination algorithms can be used. Figure 1.1a) provides an illustration of a centralized architecture in which the central agent, Robot 0, is using centralized coordination to plan actions for all robots. Examples include the SCOUTS developed for nuclear site inspection [52], and the NANOWALKERS for nano-scale manipulation and inspection [40]. Unfortunately, centralized architectures are usually not scalable because a single agent is responsible for communicating with and processing the control over every other robot. They suffer from single-point failures in that the whole system will fail if the central agent fails. They are also not practical for many applications where no single agent has complete knowledge of the environment and the other agents, as is the case when limitations in communication are present.

Within decentralized architectures, control responsibility is distributed and each agent uses local sensing and communication for control [14, 24, 61, 41, 44]. Figure 1.1b) provides an illustration of a decentralized architecture in which each agent plans its own actions based on information about neighboring robots, (i.e. they use a type of decentralized coordination). These approaches have been shown to be scalable and fault-tolerant. One example is Behavior-Based Systems [41], in which robots are equipped with a set of primitive behaviors (e.g. corridor-finding). If individual robots employ the appropriate behavior(s), desirable group behaviors can result. Related to this approach are Robot Ant Colony systems [61]. Robots within these systems have been shown to cooperate and accomplish complex tasks, despite the fact that individual robots are simple (i.e. they have limited sensing, communication and computation capabilities). The main issue is that robots don't generally have complete system information or communication with all robots in the system. For example, in Figure 1.1 b), no communication link exists between two groups of robots. This makes it impossible to implement centralized robot coordination.

Beneficial would be a method for maximizing the centralization of coordination in systems which suffer from limitations in communication. Figure 1.1c) provides an illustration of a decentralized architecture in which centralized coordination is implemented. Communication limitations prohibit any communication link between the two groups of robots. While centralized coordination can not occur between all 5 robots, centralized coordination can occur within each of the two distinct groups of robots. Also, because a decentralized architecture is used, the system is scalable and fault-tolerant to single-point failures.

Communication Structure

The type of communication used between robots is usually classified as being *implicit* or *explicit*. Implicit communication occurs through sensing of the world, and is usually the side-effect of some other action. For example, unmanned aerial vehicles attempting to maintain a formation can use sensors to detect the flow disturbances caused by the actions of other vehicles and react accordingly. Explicit communication occurs directly, usually through a wireless medium (e.g. radio). Some researchers try to do without explicit communication to allow the use of simple, cost-effective robots. One example is [4], where motor-schema-based techniques were implemented to provide a behavior-based strategy that produced globally coherent cooperative behavior in forage tasks. Other researchers have showed the relative advantages of using explicit communication to improve group behavior in multi-robot systems [41].

While many systems use explicit communication, they are still limited because robots can only communicate with robots in their local vicinity. Recently, there has been research in using Mobile Ad-Hoc Networks [10] for multi-robot systems. Equipped with this type of communication capability, robots can act as routers in a network to pass information between robots which might not otherwise be able to communicate, e.g. robots 2 and 4 in Figure 1.1b). This can be used to provide robots with more information about the system. However, robots are still not guaranteed to have information about all robots in the system, e.g. robots 0 and 2 in Figure 1.1b).

Research that specifically investigates the application of ad hoc networks to mobile robots has focused on sharing local information to improve performance in the deployment of robots as sensor networks [65], and on facilitating behavior-based or reactive multi-robot systems [41]. The research demonstrated improvements in global behavior made possible by exchanging local sensing information.

Centralized coordination across an ad hoc network (e.g. Figure 1.1c) could benefit robots operating in dynamic, unknown environments where sensing and communication are limited. This research presents, for the first time, *probabilistic roadmap* (PRM) motion planning [27] that is coordinated in ad hoc robot networks. Several issues must be resolved to ensure centralized coordination is 1) fault-tolerant to network communication drops caused by network breaks, 2) tolerant to communication delays caused by information having to hop through the network, and 3) equipped with a planning algorithm that is fast enough to be run on-line.

1.2 Problem Statement

The problem is to enable safe navigation for multi-robot systems in which robots have limited sensing and communication and operate in environments that are dynamic, and unknown.

In this problem, a multi-robot system is comprised of N robots that share a common workspace. Robots are assigned individual goal locations to which they must navigate autonomously. Goal locations can be assigned in several ways. An autonomous agent or human operator can assign goal locations on-the-fly in response to sensing information. An agent/operator can also download a series of goal locations for a robot to visit. This can only occur when a robot is close enough to the agent/operator such that communication is possible.

Navigation towards a goal location is accomplished by first constructing robot trajectories. Based on information about the environment that is available at the time of planning, the trajectories are constructed to be collision-free. Robots will then follow their trajectories. In doing so, they will continually gain new information by sensing the environment and communicating with each other. Robots respond to this new information by replanning new trajectories to ensure the robot motion is free of collision. Robots must also operate under the following conditions:

1. **Unknown Environment** – The workspace is unknown a priori.

2. **Dynamic Environment** – Objects in the workspace may be moving.

3. **Limited Communication** – Robots are equipped with limited communication capabilities. They can only communicate directly with other robots that are within a local region R_C of the workspace, where R_C depends on the robot and obstacle locations within the workspace. That is, a robot can communicate directly with any other robot that lies within its region R_C, but cannot communicate directly with any robot outside the region. Because robots will move in and out of each other's communication regions, they will only be able to communicate directly with one another for intermittent periods of time.

4. **Limited Sensing** – Robots are equipped with limited sensing capabilities. They can only sense and detect objects in a local region R_S of the workspace, where R_S depends on the robot and obstacle locations within the workspace. That is, a robot can sense any object that lies within its region R_S, but cannot sense any object outside the region.

5. **Dynamic Goals** – Robot goal locations are re-assigned on-the-fly. New goal locations can be assigned by an autonomous agent/human operator in response to new knowledge of the environment. For example, a robot moving to site A

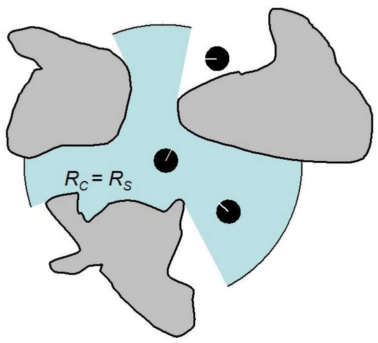


Figure 1.2: A Communication Region Example: Dark circles denote robots and grey shapes denote obstacles in the workspace. The communication region of the centrally located robot is illustrated (denoted as R_C). In this example the communication has limited omnidirectional range and suffers from obstacle occlusions. Hence, only the two lower robots can communicate with one another.

detects site B. Since site B is of greater interest, the robot re-assigns its goal location to be that of site B. For goal locations to be reassigned through a human operator, the robot must be close enough to an operator such that they can communicate. In such situations, the operator could download a list of goal locations to visit. After each location on the list is visited, the robot is assigned the next goal location on the list.

6. **Kinodynamic Constraints** – Robot plans must satisfy any kinematic or dynamic constraints on the robot's motion.

Note the size limitations on the region R_C can result in intermittent communication, (an example in which not all robots can communicate is provided in Figure 1.2). Robots will move in and out of each other's regions, causing communication links to form and break respectively. In finding a solution to the navigation problem, this dissertation does not rely on these regions being of any particular shape or size. Instead, the proposed solution will be one that functions despite the fact that communication is intermittent. More specifically, the solution will exploit the local inter-robot communication whenever possible, and be robust to situations where this same communication is infeasible.

1.2.1 Particular Implementation

The Micro-Autonomous RoverS (MARS) test platform was used to implement the operating conditions. The test platform includes a large 4m x 3m granite table upon which six rovers (0.1m diameter) operate. This test platform meets the above criteria:

1. **Unknown Environment** – Robots in the MARS test platform are given no knowledge of the other robots or obstacles a priori. Once they begin operation, they begin forming a model of the environment that consists of a list of all objects on the table including their effective diameter, their state, and their predicted trajectory.

2. **Dynamic Environment** – The test platform includes several constant-velocity moving obstacles that float on air-cushions.

3. **Limited Communication** – Limited Communication is simulated. All robot processing is done off-board and communication between robots is accomplished across a wired local area network. To simulate on-board wireless communication, the communication is filtered such that robots may only communicate directly with those robots that are within some radial distance r_C of one another. This simulates a circular communication region.

   It is possible to simulate communication occlusions,(e.g. those shown in Figure 1.2). However, the system's functionality does not depend on the actual shape or size of the communication region, but on the intermittent communication caused by the limited size of the regions.

4. **Limited Sensing** – Limited sensing is simulated. Sensing in the test-platform is accomplished with an overhead vision system that can provide the position and velocity for any object (robot or obstacle) on the table. To simulate on-board sensing, a robot only receives the state information of those objects that are within some radial distance r_S. This simulates a circular sensing region. As with R_C, one could include sensing occlusions but without benefit (as above).

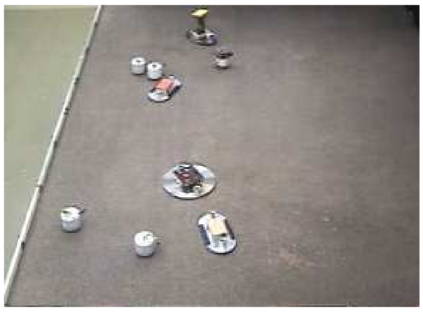


Figure 1.3: Rovers avoiding obstacles on the MARS test platform.

5. **Dynamic Goals** – The MARS test platform offers a Graphical User Interface (GUI) upon which new robot goal locations may be commanded at any time.

1.3 Proposed Approach

To enable safe navigation within multi-robot systems operating under the conditions outlined above, a solution is proposed based on centralized robot coordination through Dynamic Robot Networks. In different parts of the workspace, those robots which can communicate form communication networks to facilitate information exchange, coordination, and cooperation. Within these networks, centralized motion planning is invoked to construct feasible, collision-free robot trajectories.

1.3.1 Dynamic Robot Networks

Dynamic Robot Networks is a new coordination platform, i.e. a communication infrastructure that defines how robots can coordinate their actions through data exchange. The platform functions within a decentralized group architecture, but maximizes the centralization of robot coordination.

Dynamic Robot Networks are mobile ad hoc communication networks in which the robots become nodes in the network and can act as routers to relay information through the network. Such networks are formed by robots establishing communication links whenever possible. This can result in many different networks of robots located in different parts of the workspace. The networks are dynamic in that they can break or merge with other networks over time.

Within these networks, information is distributed to the point where all robots in a network share a common model of the world, (although each network in the workspace will have a different model). Over time, this model will change as new information about the environment is gained from on-board sensing. In response to these changes in the model, robots may adapt their navigation plans. In such cases the network of robots will respond as a whole, by replanning coordinated motion for all robots in that network.

The benefits of using this robot coordination platform include:

- **Centralized Coordination within Networks** – The Dynamic Robot Network platform allows centralized coordination within each individual robot network. This increases plan feasibility since plans are constructed with more knowledge of the environment. Moreover, when coordination is centralized, the actions of all robots in a network can be taken into account when planning the actions of individual robots. This prohibits conflicts between robots in a network (e.g. robots will not collide).

- **Increased Scalability** – In centralized architectures, a single agent is required to communicate with all robots in the system. The addition of more robots can increase the communication responsibility of this agent beyond its capabilities. The addition of more robots to a Dynamic Robot Network system, where the architecture is decentralized, will only increase such responsibilities in situations where a large number of robots are communicating directly with one another, (i.e. when robots are relatively close to one another.)

- **No Single Point Failure** – In centralized systems, there exists a single central agent which is responsible for controlling and communicating with all other robots. If this robot fails, the entire system will fail. Because no central agent exists in decentralized systems, this type of single-point failure does not exist.

- **Robust to Intermittent Communication** – Because robots have limited communication capabilities, they will only be able to communicate when close enough to one another. As the robots move around the environment, they will move in and out of communication range of one another yielding intermittent communication. Dynamic Robot Networks allow for the establishment of communication networks under such conditions. Distributed and centralized coordination that is robust to intermittent communication can be implemented within Dynamic Robot Networks by way of a new communication protocol. The protocol is designed to ensure that robot coordination algorithms can be called on-the-fly in response to the changes of the environment, but will not be interrupted by network merges or breaks.

- **Robust to Asynchronous Communication** – When robots detect changes in the environment that require them to adapt their coordination plans, they will propagate the information through the network so that each robot can learn this new information. Because this propagation requires information to hop through nodes in the network, delays will be incurred. This results in different nodes (i.e. robots) learning of new information at different times. A new communication protocol to be used within Dynamic Robot Networks is designed such that robot coordination can occur despite such delays.

1.3.2 Motion Planning

Within each robot network that forms, a randomized planning algorithm is invoked to construct collision-free trajectories for all robots in the network. The algorithm is a modified Probabilistic Road Map (PRM) planner.

The benefits of using this planner include:

- **Speed** – The PRM algorithm presented in [34] as a single robot planner has been modified to provide on-the-fly trajectory construction for multiple robots. Average planning times are on the order of 20 ms.

- **Kinodynamic Constraints** – The algorithm considers any significant kinematic or dynamic constraints when generating plans.

- **Probabilistic Completeness** – The probability of not finding a plan decreases exponentially to zero with the number of iterations. This has been proven for single robot planning, and demonstrated empirically for multiple robots in this work.

1.3.3 Coordination and Cooperation

One challenging form of robot coordination is cooperation. In this case, the completion of a high-level goal is desired. This high-level goal will be achieved only after the coordinated completion of several individual robot goals.

The Dynamic Robot Network coordination platform can be applied to various types of robot coordination, including instances that involve cooperation. This dissertation focusses on the application of dynamic robot networks to one particular type of coordination - robot motion planning, (see Example 1 below). To illustrate how the platform can be applied to instances of robot coordination that involve cooperation, two examples are provided (Example 2 and Example 3).

All three examples illustrate how information exchange can benefit coordination within robot networks. However, the examples differ in the manner in which this information is used for their particular type of coordination.

Example 1: Motion Planning

The purpose of this example is to demonstrate why robot coordination is necessary, and how Dynamic Robot Networks can be used for coordination.

This example illustrates motion planning through Dynamic Robot Networks. The information that is exchanged within networks is used to allow centralized motion planning that ensures robot trajectories are collision-free.

In Figure 1.4(a), all three robots are at their initial locations. The two left robots are in communication range of one another and establish a network. If robots are assigned goal locations, their centralized planners create coordinated collision-free trajectories that lead to the goal locations (b). The right robot forms a network by itself, and its trajectory is planned independently from the other two. As the

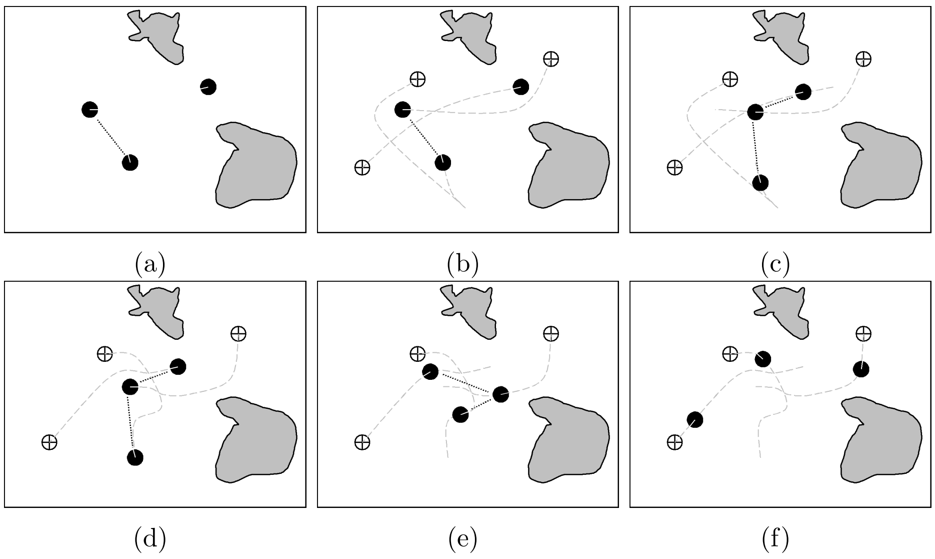


Figure 1.4: Motion Planning Example: Top-down view of a robot motion planning example with three robots (grey circles). In each of the fours snapshots, the illustration on the left shows the robots following their trajectories to their respective goal locations (cross-hairs). Dotted lines indicate communication links exist because robots are within communication range of one another.

robots move along their trajectories (c), the middle robot and the right robot enter communication range with each other, and the two networks merge to form a larger network. Robots within this larger network exchange information such that all robots share a common model of the world. Based on this model, each robot constructs a new plan, consisting of trajectories (one for each robot), and the robots select the best of the three plans to execute (d). They follow these trajectories as shown in (e). In (f), as robots move along their new trajectories, they leave communication range of each other and network links are broken. They continue to follow the planned trajectories.

Example 2: Site Surveillance

The purpose of this example is to demonstrate how cooperation can be used within Dynamic Robot Networks to optimize global task performance.

In this example, robots are given the high-level goal of visiting all sites of interest

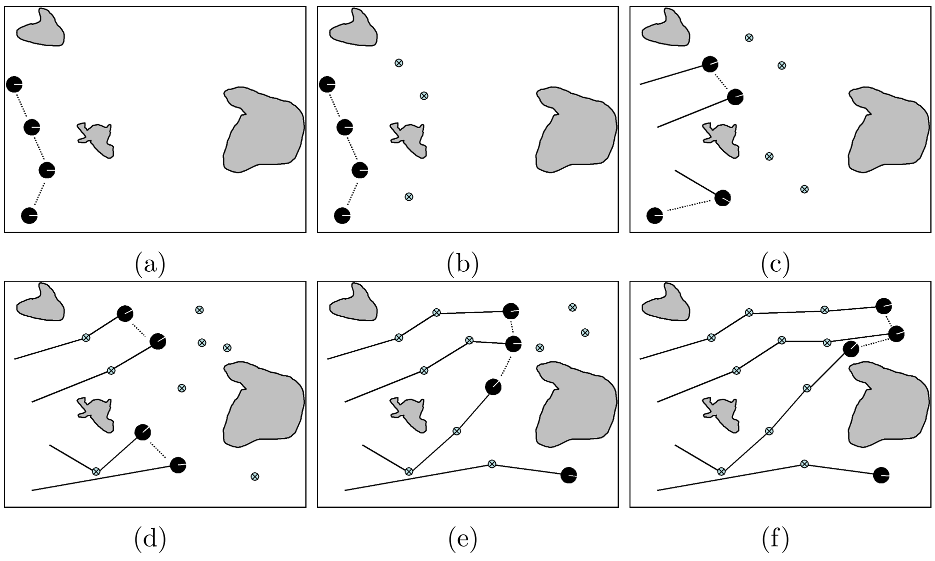


Figure 1.5: Site Survey Example: Top-down view of a robot formation. Robots are denoted by black circles, with dashed lines to indicate communication links. Grey obstacles are scattered throughout the environment. Sites of interest are denoted by cross-hairs.

they find within a large area. A task planner provides autonomous and dynamic assignment of individual robot goals (i.e. site locations). The information that is exchanged within networks is used to optimize goal assignment among robots so that sites of interest are visited as quickly as possible.

As new information about the environment is sensed, new sites of interest within the environment are identified. The task planner will assign these sites as goal destinations to the robots. Figure 1.5 provides an example involving four robots that are searching for sites of interest to investigate. In Figure 1.5 b), the four robots detect three sites of interest. The task planner, which could either be distributed across the network or reside on one robot, allocates the tasks of visiting these sites to three of the four robots. In c), the three robots have moved to their goal destinations which requires the network to break into two smaller networks. At this point, new sites of interest are identified and assigned within each of the two networks. This process repeats itself.

The cooperation within Dynamic Robot Networks is highlighted in Figure 1.5 e). When the two networks merge, information exchange occurs across the network.

Based on this information, the task planner assigns goal locations to robots that minimizes the time to visit the sites of interest (f).

Example 3: Large Object Manipulation

The purpose of this example is to demonstrate how Dynamic Robot Networks can be used for tasks that *require* cooperation.

In this example, groups of robots are assigned the task of manipulating large objects, (e.g. for assembly tasks). The information that is exchanged within networks is used to allow tight coordination between robots carrying the object. This can be accomplished through a leader/follower control scheme [19], where state estimation and control signals are communicated using the ad hoc communication link.

An advantage of using Dynamic Robot Networks is that robots carrying the object can be represented as a single robot when coordinating with other robots in the system. This single robot representation will encode the size and dynamics of the object and robots together.

In Figure 1.6, two pairs of robots are assigned the task of carrying large objects to desired goal locations. In (a), two robots in the upper right corner are completing a manipulation task. The multi-robot manipulation, an example of robot cooperation, is facilitated by the communication link established within the ad hoc robot network. The two robots in the lower left are merging into a network and cooperating to move another object (b). Together, they plan a trajectory to the goal location. In planning, both robots and the object are treated as a single robot. In (c), robots from the upper-right are moving back toward the bottom left. When close enough, one of these robots establishes communication with a robot that is carrying the object. This results in a network merge in which all robots can communicate. As shown in (e), the two robots on the right replan their trajectories to avoid the robots carrying the object. They treat the robots carrying the object, and the object, as a single robot to communicate with and avoid.

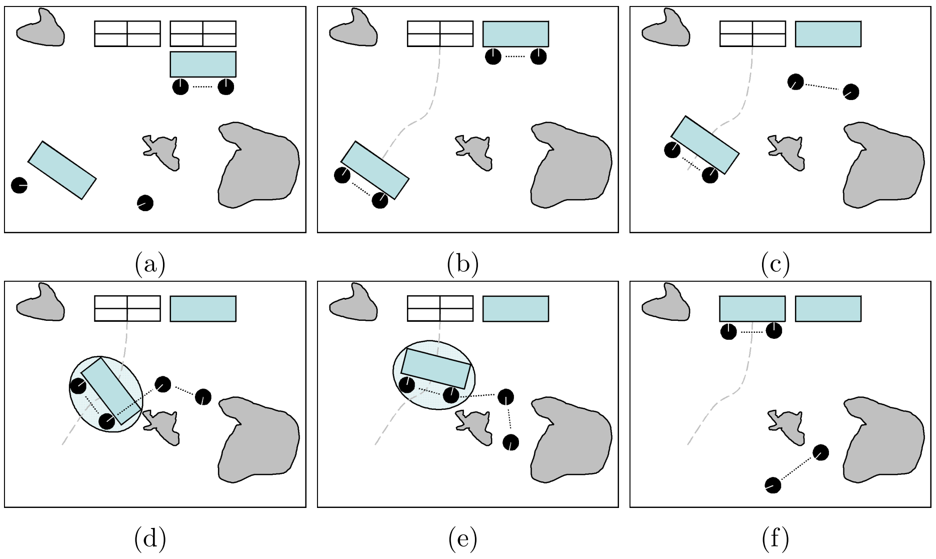

(a) (b) (c) (d) (e) (f)

Figure 1.6: Large Object Manipulation Example: Top-down view of multi-robot manipulation. Robots are denoted by black circles, with dashed lines to indicate communication links. Grey obstacles are scattered throughout the environment. Objects to manipulate are blue rectangles and their goal locations are rectangular cross-hairs.

1.4 Contributions

In developing this new approach to multi-robot systems, several research contributions were made that are summarized below. These contributions are categorized into three areas. The first area, System Control, contains contributions related to high-level robot coordination. The second area, Technical Contributions are strategies to improve motion planning algorithm speed. Last, the System Validation contributions outline the various simulations and experiments that demonstrate the system performance.

1.4.1 System Control

1. Developed the *Dynamic Robot Networks* platform that allows for centralized coordination across ad hoc networks.

2. Developed an application level communication protocol to manage information sharing and multi-robot coordination across Dynamic Robot Networks.

1.4.2 Technical Contributions

1. Identified a method of sampling milestones for roadmap expansion when applying PRMs to multi-robot planning problems.

2. Introduced a method of generating milestones - *serial expansion*, which demonstrates faster roadmap expansion over the traditional method - *parallel expansion* when applying PRMs to multi-robot planning problems.

3. Developed a new endgame region definition, based on velocity-tuning, for applying PRMs to multi-rover planning problems. It was demonstrated through simulation that using the new endgame region increased the likelihood of finding a solution when sampling the PRM. Also, under assumptions specific to this implementation, it was shown that conditions for belonging to the new endgame region are easily-calculated.

1.4.3 System Validation

1. Demonstrated, through simulation, on-the-fly motion planning through Dynamic Robot Networks. Average planning times on the order of 20 ms were achieved in scenarios involving up to 12 robots. Within these scenarios, 20 networks were merged per minute, demonstrating the platform's ability to handle frequent network merges/breaks.

2. Demonstrated, on hardware, on-the-fly motion planning of a group of mobile robots in an unknown, bounded workspace occupied by stationary and moving obstacles. This demonstrated planning on-line, assumptions on system modelling were valid, and practicality of system implementation.

Chapter 2

Dynamic Robot Networks

2.1 Introduction

This dissertation aims to enable multiple robots with the ability to navigate in dynamic, unknown a priori environments using limited communication and sensing capabilities. To navigate safely, robots must be able to coordinate their actions to avoid conflicts (e.g. robot collisions). This chapter presents a new coordination platform - Dynamic Robot Networks - that enables robot coordination under such conditions. Subsequent chapters present the implementation of a particular motion planning algorithm that can be used within Dynamic Robot Networks to allow safe movement of robots towards goal locations.

Dynamic Robot Networks provide a scalable and fault-tolerant coordination platform. A key advantage of this platform is that it enables centralized coordination across ad hoc robot networks. Centralized robot coordination is desired because actions of all robots in a network are taken into consideration when planning any single robot's actions.

The platform is implemented by way of a new communication protocol. This protocol handles data exchange and centralized planning across networks. It is robust to the two main difficulties encountered when coordinating robots across an ad hoc network: asynchronous communication and communication drops.

In this chapter, mobile ad hoc networks are introduced with a focus on their application to robot systems. Dynamic Robot Networks are then described. Key issues to be addressed in developing this coordination platform are identified. Finally, a description of the communication protocol that addresses these issues and allows a particular implementation of Dynamic Robot Networks is presented.

2.2 Ad Hoc Networks for Mobile Robots

A wireless ad hoc network is a collection of autonomous nodes that communicate with each other by forming a multi-hop radio network and maintaining connectivity in a decentralized manner. Each node in a wireless ad hoc network functions as both a host and a router, and the control of the network is distributed among the nodes. The network topology is in general dynamic, because the connectivity among the nodes may vary with time due to node departures, new node arrivals, and the possibility of having mobile nodes. Critical features, (e.g. network settling time), of such networks are outlined in [60].

There are two main categories of ad hoc networks: Mobile Ad Hoc Networks and Smart Sensor Networks. Within Mobile Ad Hoc Networks (MANETs), the nodes of a network are continuously moving. For this research, robots act as nodes in MANETs. Figure 2.1 illustrates an example where robots are forming ad hoc networks.

Classically, two types of routing algorithms exist for MANETs, *table driven* [47] and *source initiated on demand driven* [48]. In the table driven approaches, each node of a network maintains a table that encodes the network topology. By communicating with other nodes, this table can be continually updated as the topology changes. Nodes then route messages based on information extracted from the table. In the source initiated demand driven algorithms, nodes only find routes as they are required. Each time a message route is required, the node will explore the network through communication.

Equipped with MANET communication capabilities, robots can act as routers in a network to pass information between robots which might not be able to communicate directly. This information could be used to improve the performance of any of the core capabilities required by autonomous robots including planning, sensing and control.

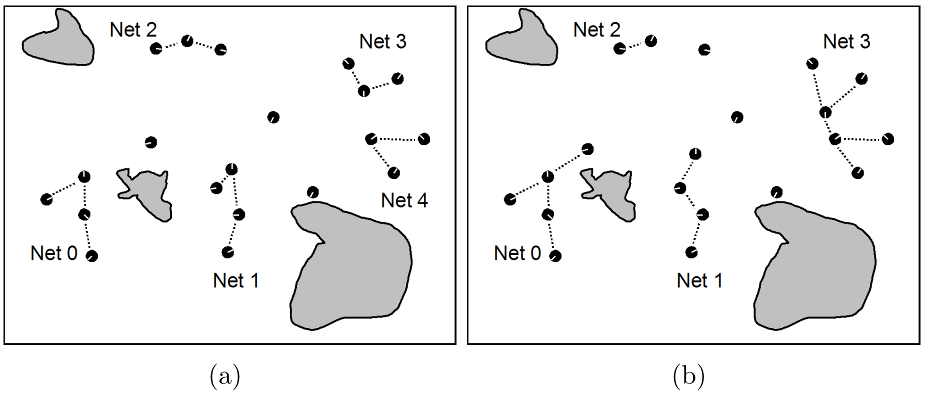


Figure 2.1: Mobile Ad Hoc Networks: Nodes in the network are circular robots. Communication links are indicated by dashed lines. Large gray objects in the environment are also depicted. In (a), many robots have formed ad hoc communication networks. As shown in (b), the mobility of the robots greatly affects the topology of these networks. For example, Net 0 has grown in size because a single robot has moved into communication range and joined the network. Net 1 has maintained the same set of robots, but has changed its topology as a result of one robot's movement (i.e. top-most robot in the network). Net 2 has broken because the right-most robot has moved out of communication range with the central robot. Also, a robot in Net 3 has moved down resulting in the merger of Net 3 and Net 4.

The majority of past research has focussed on sharing local information to improve performance in the deployment of robots as sensor networks [65], and in facilitating behavior-based or reactive multi-robot systems [41]. That research demonstrated the improvement in global behavior made possible by exchanging local sensing information. It should be noted that these projects rely less on the well-established MANET routing protocols and simply broadcast information to all other robots who are local, (i.e. flooding the network).

While there exists a large amount of research in multi-robot systems that rely on wireless communication, most assume that communication is reliable throughout the duration of a robot task Examples include [5, 43, 18, 53]. In [53], an algebraic representation of vehicle formations is presented based on a class of triangulated graphs. The representation allows for formation stabilization, collision avoidance and tracking.

A coordination framework based on artificial potentials is used in [43] to control a mobile sensor network in a gradient climbing task. The framework allows for gradient descent towards local maximima/minima of an unknown, noisy environmental (e.g. a temperature field).

Also, in [18], a state space framework for distributed control of spatially-interconnected systems. This framework was applied to formation flight experiments [21] in order to reduce the induced drag.

Coordinated robot planning that requires finite time (as opposed to simpler reactive systems) across ad hoc networks that are dynamic has seen little investigation.

2.3 Dynamic Robot Network Platform

2.3.1 Platform Description

Dynamic Robot Networks provide a new coordination platform that enables centralized robot coordination within ad hoc robot networks. Within this coordination platform, every robot will belong to one network, (which could include only that one robot). As robots move about the environment, they will enter and leave each others communication range. This causes *network merges* and *network breaks* respectively.

By way of ad hoc network routing algorithms, information can be passed between any two robots in a network, (but not between networks). Assuming world models can be encoded in a concise manner, (a possible issue for some applications), robots can use information exchange to share a common world model. This allows for a centralized *coordination process* to occur across the network in which the actions are planned for all robots within that particular network.

A *coordination process* is a defined series of steps that robots must take to coordinate their actions. Steps include Event Detection, Data Exchange, Model Fusion, Planning and Plan Execution. A coordination process can be initiated by any robot in a network, at any time. A robot will initiate such a process in response to changes in the environment (e.g. two robot networks merge). Once the process is initiated, all robots in the network participate in each step of the coordination process. The platform allows for several of these processes to occur concurrently.

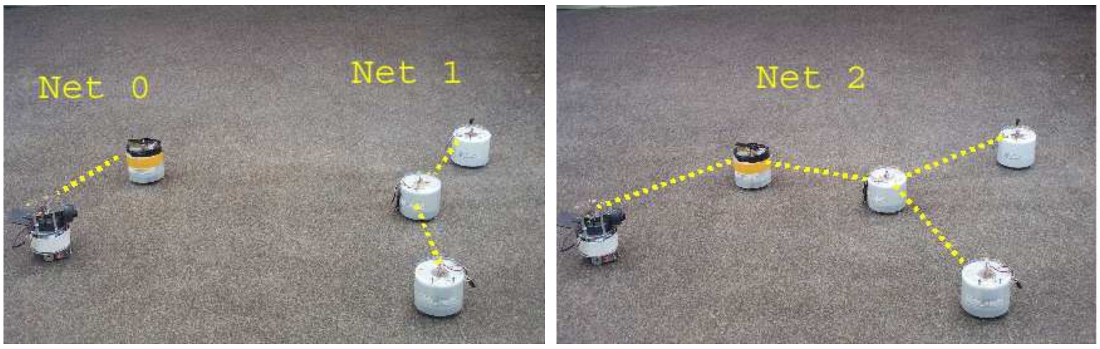


Figure 2.2: Example with 5 robots. Dashed lines between robots depict communication links. In a) the robots form two distinct networks Net0 and Net1. In b), two robots have moved, and the two networks in a) have merged into Net2.

Network Merges/Breaks

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 maximally 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 2.2 a) 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 and objects are moving, the networks are dynamic. The networks may merge and/or break apart (see Figure 2.2 b). Ad hoc network protocols [10] ensure that edges in G are established when possible, and that information can be routed efficiently across these edges. With G established, robots within the network can communicate and conduct a coordination process.

To facilitate information exchange between robots in a network, it is assumed that each robot is assigned a unique identification number. Also, when two networks merge, let the robot with the lower identification number of the two robots that caused the merge be known as the *Lead* robot and the other robot that caused the merge be known as the *Secondary* robot.

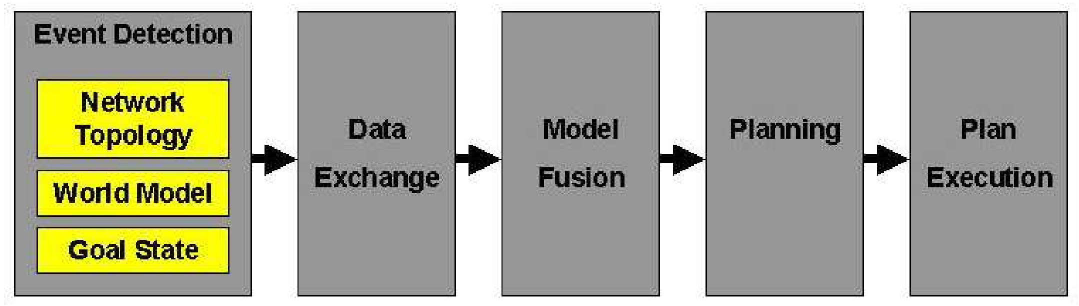


Figure 2.3: Coordination process

Coordination Process

The coordination process that takes place across a robot network is a series of steps as shown in Figure 2.3. The process is initialized with an *Event Detection* step. Such events may include the sensing of new obstacles in the environment, the awareness of new robots within communication range, or a new goal state request. Information regarding the event will be broadcasted across the network to allow the *Data Exchange* step to occur. This information will include world state information with which each robot's world model must be updated. Hence a *Model Fusion* step is required. Along with this information will also be sent a "plan request" message (if required). This informs robots to start constructing a new plan that takes the new event into account. This starts the *Planning* step in which robots construct a plan that schedules actions of all robots in the network. This is followed by robots broadcasting their newly constructed plans to all other robots. Robots will then implement the best plan of those broadcasted to carry out the *Plan Execution* step.

Example

An example of the coordination process involving 5 robots is illustrated in Figure 2.4. Initially, two robot networks are present. Two robots, one within each network, are following trajectories to their respective goal locations (b). Note that these trajectories collide, but this is undetected because robots are not close enough to communicate. As the robots follow their trajectories (c), they eventually can communicate (Event Detection). They begin the Data Exchange step of the process when the *follower robot* broadcasts its world model (d). The *lead robot* then broadcasts a "plan request" message to all robots in the network (e). Upon receiving this message, robots merge the newly acquired information (Model Fusion step) and query their planners (i.e. the Planning step) to construct a set of trajectories for all robots in the newly formed network (f). As each robot completes its plan, it broadcasts it (i.e. the Plan Exchange step) for other robots to receive (g). Once a robot receives a plan from every robot in the network, it picks the best plan based on some established criteria and uses it for motion (h) to complete the Plan Execution step.

2.3.2 Platform Requirements

To enable centralized robot coordination across an ad hoc robot network, the coordination platform must meet the following requirements:

- **On-the-Fly Changes in Network Topology** - Robots in the system must be able to merge and break networks immediately following their detection of one another through communication probing. Additionally, the network topology (i.e. current state of the graph G) must be provided to each robot. These requirements can be met with Mobile Ad Hoc NETworks (MANETS) [51] technology.

- **World Models are Shared** - The platform must ensure that all robots have a common shared world model before coordination is initiated. This requires that world models be concise to allow their quick distribution across the network, while still maintaining all relevant information about the environment. Many methods of encoding a world model exist [28, 29] including that presented in this dissertation, but there is no general method that is appropriate for all applications. This could be a significant issue when implementing the platform on some systems.

- **Responsive Robot Coordination** - The platform must allow robots to coordinate their actions in response to different events that may occur, (e.g. new robots detected, new goal location assigned). Such responsive coordination is

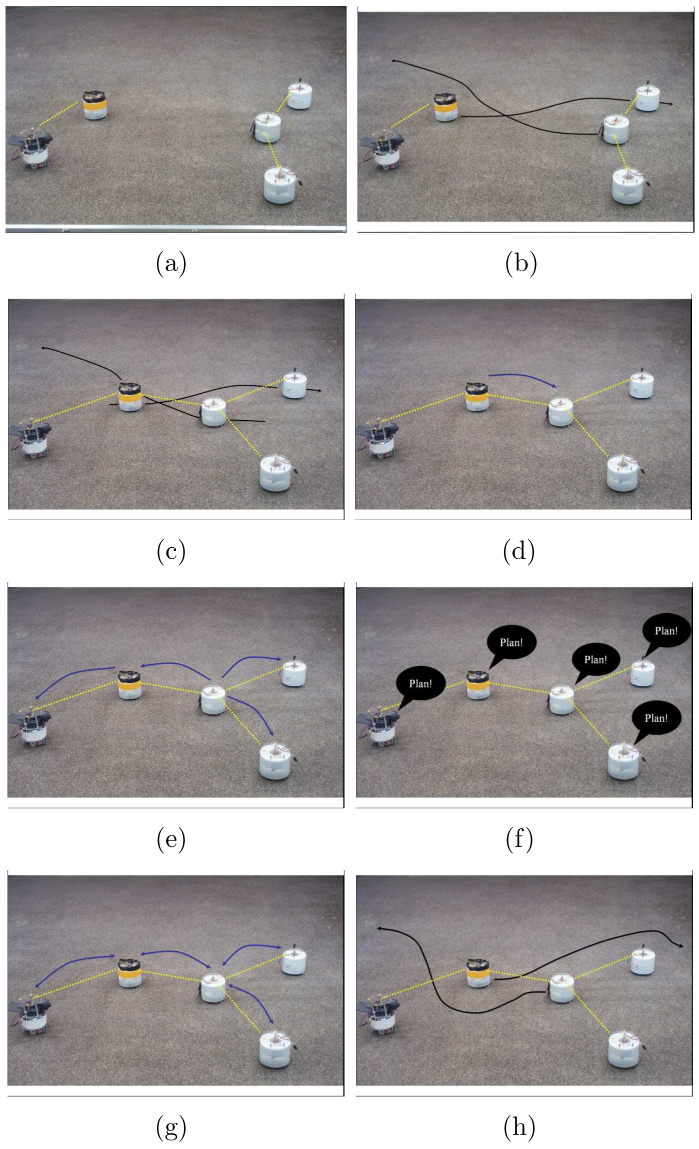


Figure 2.4: Robot Networks Merging.

possible given two requirements are met. First, the coordination process must be fast enough to keep up with frequent events. This is largely dependent on the coordination algorithm's running time. Second, robot coordination must occur promptly in response to events that occur while a previous coordination process is underway.

- **Coordination is Distributed** - By running coordination algorithms in parallel, the processing can be distributed among the different robots in a network. This has been shown to be advantageous for randomized motion planning [13], where different methods of implementing Rapidly-exploring Random Tree (RRT) algorithms in parallel were compared. While all methods sped up the planning, it was assumed that complete communication was available at all times. Required is a platform that can take advantage of parallel coordination within ad hoc communication networks.

- **Handle discontinuities in communication** - The coordination process must be robust to robots continually entering and leaving each others range of communication. If network connections are lost or established during any stage of the coordination process, robots must still complete the process successfully.

- **Minimize communication requirements** - As the number of robots in a system increases, so will the communication. This can result in large communication delays, slowing down the entire coordination process and limiting the system's ability to respond to changes in the environment. For this reason, both the overhead of handling messages and the quantity of information communicated must be minimized. Some multi-robot systems that use ad hoc networking broadcast messages by flooding the network (e.g. [65]). This is effective for applications using decentralized coordination strategies. In such applications, robots don't require information from all other robots in the network before acting. With centralized coordination, where robots must exchange information with all other robots in the network, better routing algorithms are required to reduce the quantity and size of messages sent and received by robots.

2.4 Communication Protocol

The core communication requirements listed above (i.e. On-the-fly network maintenance) can be handled through MANET technology [51]. Table driven MANET routing approaches (e.g. [47]) are preferred in which each node in a network stores a copy of the network topology (i.e. the graph G). This information is required by robots for two reasons. First, this information allows a robot to construct coordination plans that consider all robots in the network. Second, knowing the network topology allows for intelligent data delivery that can reduce the amount of information broadcasted (i.e. minimize communication requirements).

The challenge then, is to enable responsive, parallel robot coordination across a robot network in which the network topology can change and communication delays cause coordination algorithms to run asynchronously. To meet this challenge, an application level communication protocol has been developed and is described below. The protocol is described as a step-by-step coordination process that occurs across a network.

2.4.1 Single Coordination Process

The following subsections detail how each stage of the coordination process can be implemented on a robot system.

Event Detection

To initiate a coordination process, several triggers are monitored by each robot in the system. Each trigger is detailed below.

- **Network Topology Change** - By monitoring the routing table provided by a MANET table driven routing algorithm, robots can become aware of changes in the network topology. In particular, the plan manager of a robot can detect the merging of two networks. When this occurs, two robots (one from each network) will detect one another. The robot with lower identification number will broadcast a “network merge request” message, accompanied by its world model, to the other robot. The other robot will receive this request and will initiate a coordination process. The process is initiated by sending out a "plan request" message, accompanied by world model information, to all other robots in the network (see Data Exchange section).

- **World Model** - Using local sensors, robots can monitor the environment and produce a world model upon which coordination is based. As robots move about the environment, they will continually update this model with new state estimates. Robots must monitor these changes of state that occur in the world model, to ensure that previously constructed plans are consistent with the environment. For example, if the velocity of a moving obstacle has changed significantly from that with which the previous plan was based, then a new coordination process must be initiated. This is accomplished by broadcasting a "plan request" message that includes the new world model information.

  Also, as robots move closer to any particular object, they will presumably obtain more accurate sensor readings. They can then use these more reliable state estimates for future planning. In this manner, uncertainties in sensing are compensated for by enabling robots with the ability to initiate a coordination process when better estimates are provided.

- **Goal State** - Within this coordination platform, robots must have the ability to accomplish individual goals autonomously. In dynamic environments, these individual goals will change with time. These goals can originate from some high level task manager that responds to changes in the environment, or from some human operator. Robots must respond to new goal requests and trigger a coordination process by broadcasting a "plan request" message that includes the new goal location.

Data Exchange

Once an event trigger occurs, the robot that detected the trigger must broadcast the relevant information (e.g. a change in network topology or new desired goal state),

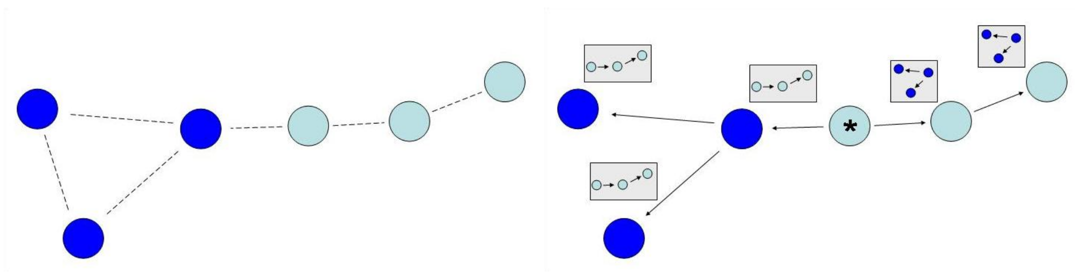


Figure 2.5: Network Routing Topology: Using a table-directed routing algorithm, robots can establish how to send world models of minimum size to reduce communication delays.

to all other nodes in G. This will insure that network robots have available complete and updated world models.

In networks with large numbers of robots, bandwidth will be limited. Also, communication latencies will diminish the system's ability to plan on-the-fly. For these reasons, it is desirable to broadcast as little information as possible.

Using the network topology information gained from implementing a table driven routing algorithm, the amount of information broadcasted can be minimized. Consider the case where two networks merge and information sharing is required. Recall the robot with the lower identification number of the two robots that caused the merge is known as the *Lead* robot and the other robot that caused the merge is known as the *Secondary* robot. With this terminology, the following rules can be used to minimize the amount of information broadcasted through the newly formed network.

1. **Broadcast** two separate messages from the Lead robot. The first message will be sent to all the Lead robot's children in the graph G, and will contain information about the Secondary robot and its children. Conversely, the second message will be sent to the Secondary robot and its children in G, but will contain information about the Lead robot and its children.

2. **If** any robot receives a new message from a parent robot

   Then save the information and rebroadcast the message to all children in the graph G.

Many mobile ad hoc network communication systems are applied to robots for use as sensor networks or to produce some optimal global behavior (e.g. [65]. In these applications, a network flooding of information was implemented by broadcasting information to all who listened. However, by following the rules listed above, each robot is only sent state, goal and trajectory information about the robots it has no knowledge of yet. On average, this will reduce the delays on the order of $rd/2$, where r is the number of robots in the newly formed network and d is the maximum depth of the routing tree.

An example scenario is depicted in Figure 2.5. In a), possible communication links (i.e. edges for G) are presented after two networks have merged. The resulting graph G is shown in b). The Lead robot responsible for the merge (denoted by the star) will initiate the coordination process by distributing world models. Note that only the unknown portions of the robot are received by each robot (depicted by topology nets shown above each edge in b).

Model fusion

When robots receive world model information obtained from other robots, they must fuse it with their own world model. This is an important step to ensuring all robots share a common world model so that centralized coordination can occur.

Describing the world model in a concise but useful form is necessary to allow for information sharing between robots in the same network. As mentioned above, the ability to accomplish this is not available to a general system. In the experimental system described in this dissertation, world models consist of a list of robots and their descriptions, and a list of obstacles and their descriptions. Table 2.1 outlines the information stored in each list.

The most recent update time is used for data fusion. When multiple state estimates received from different robots, the most recent information is used.

The information source is a robot ID that indicates which robot sensed (or communicated with) the object. It is used to determine if an object is currently being sensed by a robots in the network, or if it state estimates were obtained by a robot that no longer belongs to the network.

Table 2.1: 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

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

- Each object is approximated as a circular object. This allows 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 [12], if at any later time its trajectory is found to diverge by more than some threshold from the predicted trajectory, then the robot that detects this divergence initiates a new coordination process within its network. This could occur because the obstacle did not move at constant velocity, or because the error in the velocity estimate was too high.
- All objects in the environment are easily identifiable by robot sensors, which can precisely estimate their positions and velocities. Any discrepancy between two local world models can be easily resolved.

The first 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., [54]). In [26], the second 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 (Chapter 4). In the future, it will be important to relax this assumption by using more general sensing systems and data fusion techniques [42].

Planning

When robots receive a "plan request" message, they will query an algorithm to plan the actions of all robots in the network. As the number of robots increases, so does the complexity of the coordination problem. This motivates the use of parallel processing to conduct robot coordination across the network.

For the implementation presented in this dissertation, parallel processing is used to solve the motion planning problem (i.e. construct coordinated robot trajectories). Upon receiving a plan request, Each robot in the network will query a randomized motion planning algorithm. To parallelize the search for a solution, each robot will seed its random number generator differently. This results in each robot generating a different solution to the same motion planning problem.

Once a robot has generated its plan, it will send the plan to all other robots in the network. After a robot receives the plans created by all other robots, it will select and implement the best plan out of all those plans. The selection criteria will be based on some easily-calculated, predetermined cost function. This optimization is a clear advantage of the system.

Plan Execution

After a robot has selected the optimal plan, it will send this plan to a low-level controller for execution. Many controllers exist for mobile robot trajectory tracking and a good resource can be found in [36]. In this reference, feedback linearization techniques are used to achieve global stabilization of the trajectory tracking error to zero when implemented on a car-like robot.

2.4.2 Multiple Coordination Processes

One of the main challenges of implementing centralized coordination across an ad hoc network is that the robots are continuously moving and hence the network topology is dynamic. Difficulties arise when robots enter and leave one another's communication range within a short period of time, (e.g. less than a second.) In these cases, continuous network communication might not be possible throughout the entire coordination process which can last on the order of 500ms. The planning system must be robust to such difficulties. What follows is a description how such events are handled, so as to continue providing responsive distributed planning across the network.

Network Breaks

In the case where a network breaks into two different networks of reduced size, the coordination process must continue. Because messages are queued and processing of them is synchronized, it can be assumed that the plan manager will not realize such

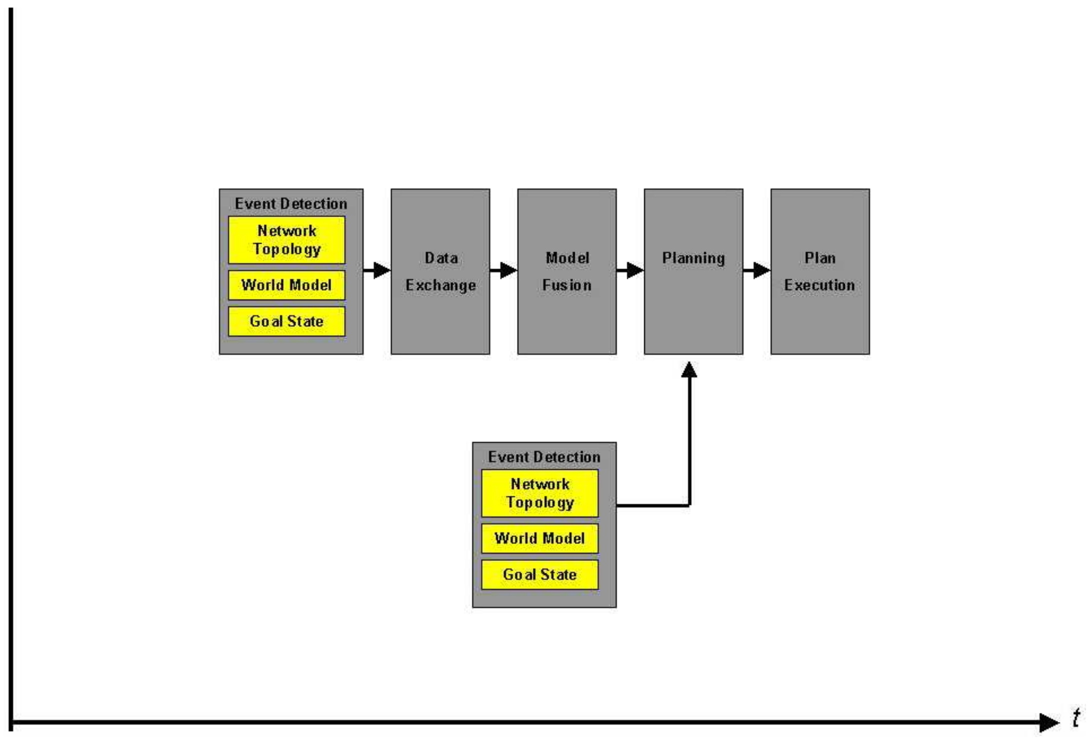


Figure 2.6: Network Break Time line: The plan manager will detect an event through network maintenance monitoring. In this case, a network break is detected and the network topology records are updated. Thus, the plan manager is informed that it should not wait to hear any plans from robots no longer within the current network of reduced size.

a break until after a robot begins its actual planning (i.e. it has queried the planning algorithm).

At this point the robot's planner will continue constructing trajectories, even for those robots that no longer belong to the same network as the robot. However, once the robot finishes planning, it waits to receive plans from only those robots that are currently in its new reduced network. For example, if five robots in a network are planning and one robot leaves, then the four remaining robots will distribute their plans and implement the best of the four. The fact that the plans consist of trajectories for five robots will not hinder the coordination process. Note that this does require robots to update the network with the information that another robot has

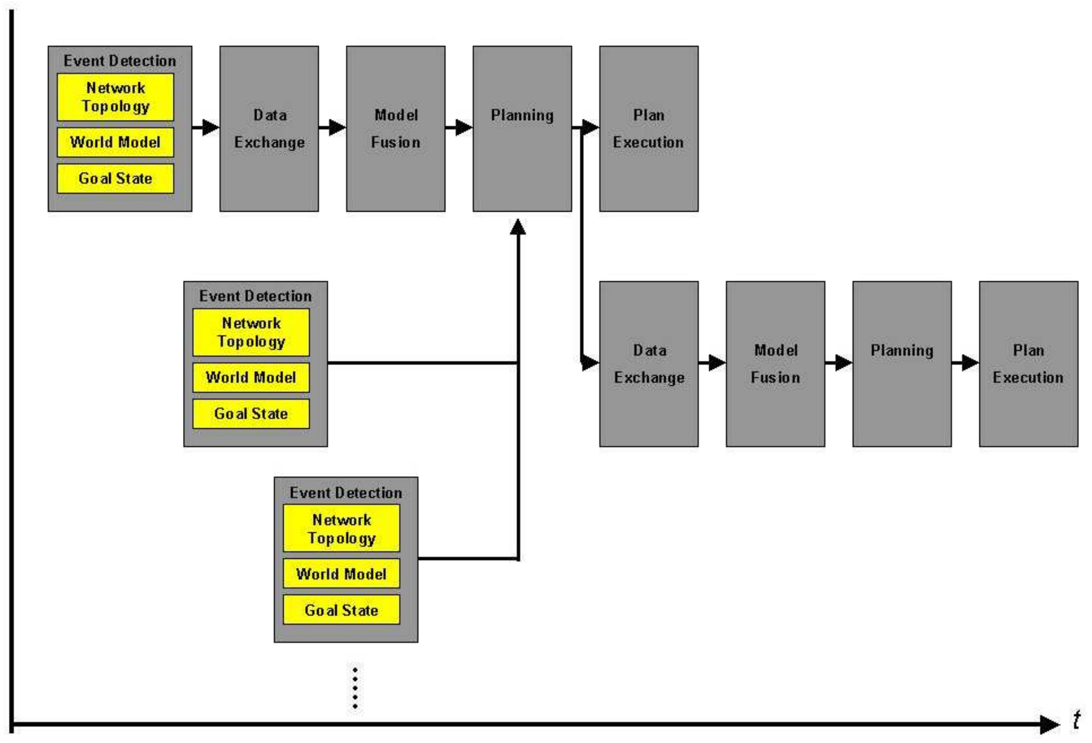


Figure 2.7: Multiple Network Trigger Time line: The plan manager will detect an event through network maintenance monitoring, local sensing and task management. In any of these cases, a plan trigger is detected and the plan manager stores this information until the first plan is received. At this point a new coordination process is triggered.

left communication range and robots should not wait to receive a plan from it. This can be accomplished through means of a network level routing algorithm protocol as discussed above.

If the network breaks after plans are completed (i.e. during the plan execution phase of a coordination process), there will be no ill effects. Each robot executes only its own plan and doesn't consider the other robot plans at this point.

New Plan Triggers

It is possible for a new plan trigger (i.e. new desired goal state, new network merge, or new object state estimates), to occur during a coordination process. In these cases, it is desirable to plan with this new information as soon as possible. However, robots cannot simply halt their current coordination process to start a new process based on the most recent information. This can lead to endless planning with no plan execution, (i.e. the system may repeatedly halt plan searches as a robot continually receives new plan triggers.)

The solution presented here is pictured in Figure 2.7. As new triggers occur during a coordination process (or any time after a coordination process has been initiated), they are stored until the first completed plan from the original coordination process is received. At this point the robots execute the first plan and initiate the next coordination process which takes into account all stored trigger information. This ensures that plans are given time to finish, but starts the next process promptly.

This system allows for several new triggers to be stored until the next coordination process begins. Also, it allows for different triggers to be heard by different robots at different times. Consider an example where two robots, located at opposite ends of a network, each detect a different plan trigger. Each robot will initiate a separate coordination process and send out its own "plan request" message with information regarding the trigger event it detected. Each robot will also begin the planning stage for the coordination process it initiated. As each robot receives the other robot's plan request, it will store it until it gets the first solution to its own plan request. Once receiving this first plan, it will begin executing the plan and immediately start planning again to incorporate the trigger received from the other robot. In this manner, each robot will execute a plan that responds to the trigger it detects, then construct and execute a plan that responds to both triggers.

For this protocol, the maximum time before a plan is executed for any given trigger is always less than double the time to carry out one coordination process. This may occur if a new trigger is detected immediately after the start of a coordination process initiated by an earlier trigger. This ensures a finite planning time for any new trigger.

Note that due to communication delays, numerous completed plans for a coordination process may have been sent after the first plan, only to be received after a new coordination process has begun. In these cases, robots will simply implement them if they are better than the first, without interrupting the new coordination process.

Chapter 3

Multi-Robot Motion Planning

3.1 Introduction

This dissertation is motivated by the need for navigation capabilities that enable multiple robots to operate in dynamic, unknown environments. In Chapter 2, a new coordination platform was described, *Dynamic Robot Networks*, that facilitates centralized planning within ad hoc networks. Discussed here is motion planning, an essential capability for safe robot navigation.

Motion planning is the construction of collision-free trajectories that connect robots to their individual goal destinations. Motion planning performance can be characterized by the following algorithm properties: speed, completeness, and optimality. For robots to operate in dynamic, unknown environments where planning must occur on-the-fly, the primary requirement is algorithm *speed*.

For multi-robot motion planning, centralized planning is beneficial because the motion of each robot can be planned while considering the motion of all robots. Given the Dynamic Robot Network platform, the main difficulty is in developing a centralized planner that meets the speed requirement.

In [27], a *probabilistic roadmap* (PRM) planner was introduced that could construct feasible, collision-free trajectories for single robots operating in dynamic environments. In this chapter, new sampling strategies are presented that decrease the PRM planner's run time when applied to multi-robot motion planning problems.

First, an appropriate method of selecting milestones in a PRM is identified. Second, a new method of generating PRM milestones is described. Finally, a new endgame region for multi-robot PRMs is presented. What follows is an overview of related motion planning research, a description of the PRM algorithm, a description of the new sampling strategies, and simulation results.

3.2 Related Work

Many approaches have been taken to multi-robot motion planning. They are usually compared based their the algorithm's speed, completeness and optimality. For complex problems, it is difficult to meet all of these requirements. In recent years, Probabilistic Road Map (PRM) planners have gained popularity because of their speed. However, effective sampling strategies are crucial to achieving successful PRM planning. Presented below is an overview of multi-robot motion planning, PRMs, and PRM sampling strategies.

3.2.1 Multi-Robot Motion Planning

Multi-robot motion planners are usually classified according to whether the planning is *decoupled* or *centralized* [3, 55], (see Figure 3.2). Decoupled planners construct plans for each robot separately before coordinating the individual plans [3, 5, 30, 31, 39, 45, 58]. The coordination step can be accomplished by tuning the robot velocities along their respective paths (e.g. [30]). Consider the two robots in Figure 3.1. If both these robots follow their paths with the same velocity, they will collide. However, by tuning velocities so one robot slows down and the other robot speeds up to pass by, a collision-free pair of trajectories results. This coordination can be done globally, in which complete information is available to the planner, or locally (i.e. when robots come close to one another) [44].

A variant of decoupled planning, 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 [9, 14, 20].

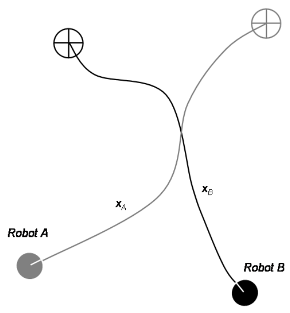


Figure 3.1: Velocity Tuning Example: If both these robots follow their paths with the same velocity, they will collide. However, by tuning velocities so one robot slows down and the other robot speeds up to pass by, a collision-free pair of trajectories results.

Decoupled planning algorithms can be advantageous because they don't require robots to have complete system information and are generally fast enough for planning on-the-fly. However, they are inherently not complete and often can not find solutions when robots must be tightly coordinated [55].

Centralized planning considers all robots together as if they were forming a single multi-body robot [6, 11, 37, 46, 54, 62, 64]. Centralized planning is beneficial because the motion of each robot can be planned while considering the motion of all robots. Unfortunately, centralized planning is often slow and requires that at least one robot be provided with complete system information.

In Chapter 2, a new coordination platform was described - *Dynamic Robot Networks*, that facilitates the information exchange necessary for centralized robot motion planning within ad hoc networks. Given this coordination platform, the main difficulty is in developing a centralized motion planner that can plan quickly despite searching configuration spaces with many degrees of freedom.

Recently there has also been research into using mixed integer linear programming to solve multi-robot path planning (e.g. [8, 50]). These methods result in optimal trajectories, but still require longer planning times not practical for some on-line implementations.

In [57], a non-linear model predictive control (NMPC) is used for the control of autonomous helicopters. Simulation results exhibited trajectory generation for helicopters operating in complex 3D environments, multiple vehicle collision avoidance, and predator evasion. Computation times ranged from 41 to 173 seconds.

To handle the requirement for speed, a probabilistic roadmap (PRM) planner is proposed. These planners have been shown to be fast enough to handle dynamic, unknown environments (e.g. [34]).

3.2.2 Multi-Robot Planning with PRMs

Probabilistic roadmaps (PRMs) have been used to solve path planning problems with many degrees of freedom successfully [33, 54, 55]. They have also been shown to construct plans that satisfy various constraints (e.g. dynamic, nonholonomic etc.) [34]. They are not complete in the traditional sense. However, under certain assumptions (e.g. the free space is *expansive* [27]), they are *probabilistically complete*. That is, the probability of failure decreases exponentially to zero with algorithm iterations.

PRMs have been applied to multi-robot motion planning problems, many of which use decoupled planners. One example is [14], where a single-query PRM algorithm is used with prioritized planning. Each robot calculates a priority number based on the occupancy of its neighborhood, (i.e. the more robots/obstacles in its neighborhood, the higher the planning priority). As robots move into one another's neighborhood, the robot with lower priority plans to avoid the higher priority robot. The higher priority robot continues on its original path. Results demonstrate on-the-fly planning for up to 15 robots in a cluttered environment.

One example of a centralized approach is presented in [62], where a multi-query PRM is used. First, a roadmap is constructed for one robot. Then, several of these roadmaps are combined into a roadmap for the composite robot. The approach worked well in planning for up to 5 car-like robots in static environments, and has the advantage of being probabilistically complete.

In [55], centralized and decoupled planning are compared using PRMs. Both approaches were applied to test scenarios involving 2-6 robot manipulators (12-36 degrees of freedom). Given those scenearios, decoupled planning often failed to find

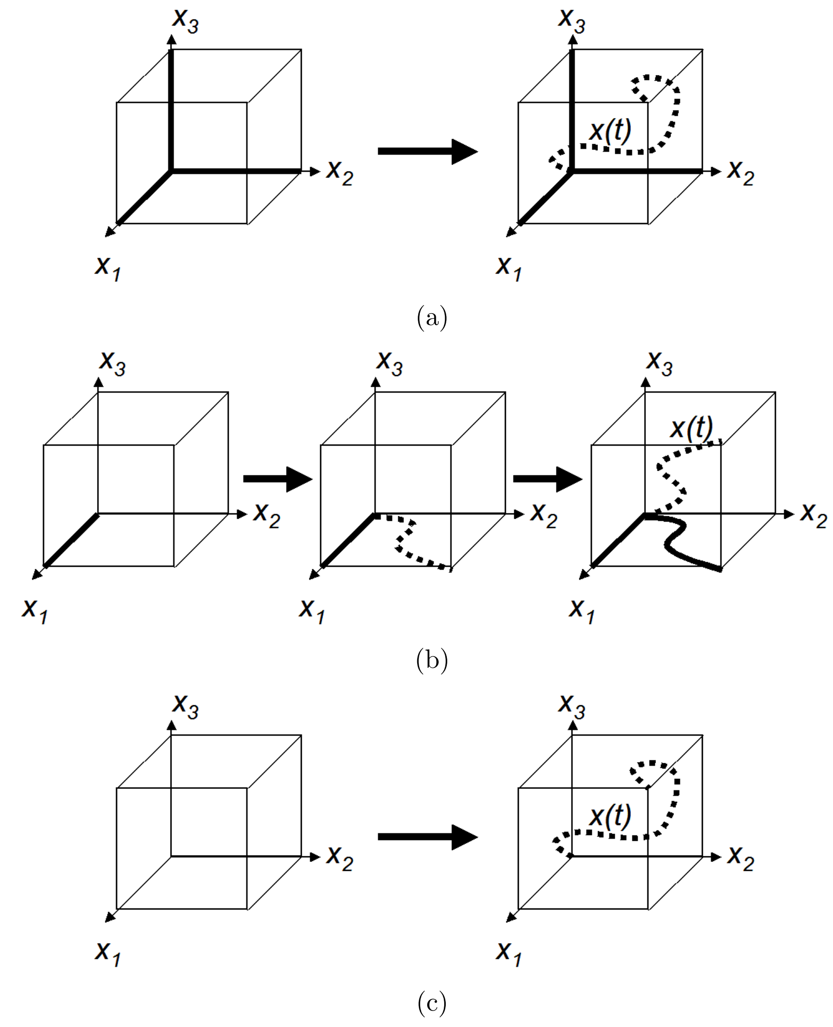


Figure 3.2: The trajectories of three robots are constructed and coordinated using three different methods. To visualize the differences between methods, plotted along each axis is a trajectory representation for the respective robots. Decoupled planning is illustrated in (a), where three trajectories are constructed independently, and then coordinated. Within the trajectory space of all three robots, the resulting three trajectories are represented as $x(t)$. In (b), prioritized planning is used. Trajectories are constructed robots one at a time, using the previously constructed trajectories as obstacles. In (c), centralized planning is depicted where the trajectories of all three robots are constructed simultaneously.

any solution. This research demonstrated the advantage of centralized planning when the motion of multiple robots requires tight coordination.

Given the large amount of research in PRMs, few have investigated how different sampling strategies can affect planning for multiple robots. Presented below are descriptions of PRM planning algorithms, some sampling strategies used within these algorithms, and new sampling strategies specific to multi-robot motion planning.

3.3 Background on PRMs

Methods of sampling the configuration space to generate PRMs are usually classified according to whether they are *single-query* or *multi-query.* To construct a multi-query PRM, the configuration space is sampled and all resulting configurations that lie in the free space are retained. These configurations are stored as *milestones* and are connected locally by *edges* to form a *roadmap* of the free space. This roadmap can be queried multiple times for different start/goal configuration pairs. First, the start and goal configurations are connected to a pair of milestones in the roadmap, say m_s and m_g. Then, a fast graph search of the roadmap is used to find a path that connects m_s to m_g.

The multiple-query PRM planner described above is practical for situations in which the roadmap need only be constructed once, (i.e. the environment is static). Queries are very fast, but roadmap construction is slow because the roadmap must cover the entire configuration space. For many applications, the roadmap construction step is too slow for on-line implementation (e.g. to avoid moving obstacles).

Another strategy is to use a single-query PRM planner, in which a new roadmap is constructed for each query. In these planners, less time is spent constructing the roadmap because only a restricted subset of the configuration space is sampled. This is usually accomplished by a *single-directional* search or a *bidirectional search.* For a single-directional search, a tree of milestones in grown from the initial configuration until a connection is found with the goal configuration. Two trees are grown for a bi-directional search, one from the initial configuration and one from the goal configuration, until a connection between them is found.

Presented by Hsu [26], is a single-query PRM planner that has shown significant success in planning trajectories for a robot operating in dynamic environments. Results demonstrated on-the-fly planning for real robots that are operating among moving obstacles. For this reason, Hsu's algorithm was selected as the core algorithm for this motion planning research. Then, to increase the algorithms speed when applying it to multi-robot motion planning problems, new sampling strategies were developed.

Hsu's algorithm is represented as Algorithm 1. In this representation, the motion of the robot is governed by the Equation 3.1. The state of the robot is x such that $x \in X$, an n-dimensional manifold called the state space. Control inputs to the robot are represented as u.

$$\dot{x} = f(x, u) \tag{3.1}$$

A milestone is defined by $m = (t, x)$ where x represents the state of the robot r at time t. The initial milestone m_0 defines the initial state of the robot at time zero.

Algorithm 1 Single Query PRM Planner

```
1.    Add initial milestone $m_0$ to roadmap $M$
2.    **Until** timeout
3.        Randomly select a milestone $m$ from $M$
4.        $m_{new}$ = PROPAGATE($m$)
5.        Add $m_{new}$ to the roadmap $M$
6.        **If** $m_{new}$ is connected to goal state
7.            **Return** plan connecting $m_o$ to the goal state
8.    **Return** null
```

To start, the roadmap M is rooted at m_0 by adding it as the first milestone in M (step 1 in Algorithm 1). The algorithm iteratively tries to expand M by first selecting an existing milestone m from M and then propagating it to a new milestone m_{new} (step 4). Within the PROPAGATE function, a candidate path from m is generated by integrating Equation 3.1 with randomly selected values for u. The function iterates until a collision-free path is found, whereby it returns a milestone m_{new} defined by the path endpoint. In step 5, m_{new} is added to the roadmap M. If there exists a simple path from m_{new} to the goal state, then planner returns a path connecting m_0 to the goal state (step 6).

This algorithm can be extended to planning for multi-robot planning using a centralized or decoupled planning approach, (e.g. [14]). In this dissertation, a coupled planning approach is taken in which all robots are planned for at once (e.g. [15]). Specifically, the milestones must define the configuration of all robots being planned for, $m = (t, x_1, x_2, ...x_R)$ where x_r represents the state of robot r at time t. This approach will be substantially slower (due to the increased size of the configuration space) but maintains the property of *probabilistic completeness.*

The remainder of this chapter concerns the development of new sampling strategies that decrease the algorithm's running time when a coupled approach is taken.

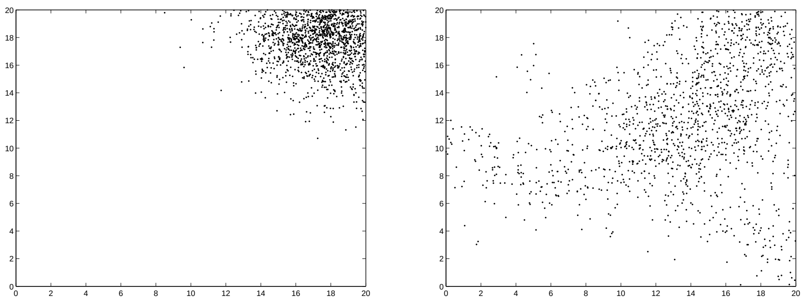
3.4 PRM Sampling Strategies

In PRM planning, a large amount of time is spent collision-checking. One way to reduce the amount of collision checking is use better sampling strategies. These strategies avoid milestone generation in uninteresting areas of the free space. Connecting new milestones to the roadmap in such areas requires costly collision-checks, without greatly expanding the roadmap.

Examples of different sampling strategies that have been applied to multiple-query PRM planners include multi-stage strategies [32], obstacle-sensitive strategies [2], and narrow-passage strategies [25]. In [32], a multi-stage strategy approach is taken. For the initial stage, a uniform distribution of milestones is generated and connected with edges. In subsequent stages, additional milestones are generated around milestones that have few or no connections.

An obstacle-sensitive strategy is taken in [2]. Those configurations sampled in the non-free portion of the configuration space are retained. They are then used as base points from which to cast rays in a random direction. Along these rays, a search for free configurations along the free space boundary is conducted.

One example of narrow-passage sampling strategy is found in [25]. First, a roadmap of the "dilated" free space is constructed in which narrow passages are



(a) (b)

Figure 3.3: Unweighted versus Weighted milestone selection strategies: The initial milestone is in the upper right corner of both configuration spaces. As the roadmap expands, and more milestones are added, the unweighted approach exhibits a slow expansion. Clustering is indicated by the high density of milestones located around the initial milestone. The weighted approach on the right, which was allowed to expand for the same amount of time, exhibits a more uniform expansion that leads to greater coverage of the configuration space.

widened. Then, collision-free milestones are generated via local resampling for those milestones which belong to the dilated free space, but not the original free space.

Several sampling strategies have also been applied to single-query PRM planners. Both single-directional and bi-directional searches require *diffusion* strategies to avoid over-sampling certain areas of the free space. More specifically, the roadmap must eventually diffuse through the reachable component of the free space, and result in a uniform distribution of milestones across the components. This uniform distribution is a requirement in proving the planner's fast convergence property [27].

To understand how diffusion strategies can affect the speed of coverage, consider the two examples of roadmap expansions depicted in Figure 3.3. In (a), no sampling strategy is invoked. Each milestone in the roadmap is given equal probability in being selected to expand from. This leads to a non-uniform cluster of milestones that slowly expands to fill the free space. In (b), a particular sampling strategy is invoked that weights the milestone selection. These weights give milestones in less densely populated neighborhoods a higher probability of being selected, leading to faster diffusion.

One diffusion strategy is presented in [35], where a configuration q is randomly selected from the configuration space. Then, the milestone m which is closest to q is obtained. Finally, a new milestone m_{new} is selected along the line connecting m to q.

Another technique is presented in [27]. First, a milestone m is selected from the roadmap with probability inverse to the density of milestones in the neighborhood of m. Second, a new milestone m_{new} is obtained with a random but uniform sampling of the neighborhood of m. To speed up the selection of m, milestone density calculations are approximated through a discretization of the configuration space, (see section 3.5.1).

A similar method ([54]) is applied to planning the motion of multiple robot manipulators with many degrees of freedom N_{dof}. First, h degrees of freedom are randomly selected, where $h << N_{dof}$. Then, local milestone densities are calculated based only on the closeness of milestones within the h degrees of freedom. Using these densities for weighting milestone selection, a milestone m is picked to generate m_{new}.

3.5 New Sampling Strategies

This research adds new techniques to improve upon the sampling strategy found in [27], when applied to coupled multi-robot planning. Each of these techniques are implemented as one distinct step of Algorithm 1:

- **Selecting milestones from the roadmap for expansion (Step 3)** - to ensure fast configuration space coverage and sampling uniformity (a requirement for probabilistic completeness).
- **Generating new milestones for the roadmap (Step 4)** - in a fast manner despite the increased number of robots.
- **Checking for endgame region inclusion (Step 6)** - that is large enough to improve the chance of finding a solution, yet still easily calculated to reduce computation time and increase speed.

The next three sections provide details on these techniques.

3.6 RoadMap Milestone Selection

This research invokes a sampling diffusion strategy for single-query PRM planning based on one presented in [27], where new milestones are generated in vicinities of the roadmap that have a low density of milestones. This strategy is a two step process, first requiring the random selection of a milestone m from the roadmap, followed by the generation of new milestone in the neighborhood of m.

This section investigates the first step of this process: randomly selecting a milestone from the roadmap (step 3 of Algorithm 1). Presented below are several candidate techniques of selecting milestones from the roadmap. Desired is a technique that leads to a fast, uniform expansion. The faster the expansion, the faster a milestone will be sampled that has a connection to the goal state. Hence the faster a solution will be found.

3.6.1 Milestone Selection Techniques

Hyper-Grid Milestone Selection

In this technique, the configuration space is divided into a grid of cells HG. Those cells that are occupied by milestones form a sub-grid called $HG_{occupied}$. A milestone is selected by 1) randomly selecting a cell c from $HG_{occupied}$, and 2) randomly selecting a milestone from within c. An example is provided in Figure 3.4 (a).

This technique has been shown to work well for single robot PRM planning (e.g. [34]), and has been extended here for multi-robot planning. This is accomplished by producing the hyper-grid HG that is the joint configuration space of all robots. If each robot has D degrees of freedom that are divided into K cells, then a hyper-grid for R robots will contain K^{DR} hyper-cells.

Given that K must be large enough to provide uniform sampling (e.g. 5 to 10), a large number of cells would be required to grid the entire configuration space. However, in many cases, only a small portion of the configuration space need be searched before a solution can be found. This requires dynamic allocation of memory for cells. As shown here, this can be accomplished via *hashtables* that allow for

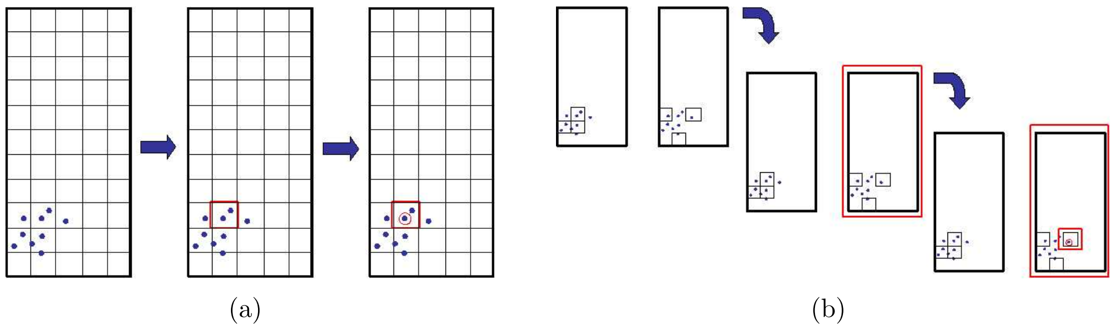


Figure 3.4: Hyper-grid milestone selection for a single robot with 2DOF is shown in (a): The configuration space is divided into a grid, (see left figure). Blue dots denote states corresponding to milestones in the roadmap. Of the nine occupied cells in the case presented, one is randomly selected (denoted by the red square in central figure). From within that cell, one of the milestones is randomly selected, (denoted by the red circle in right figure). In (b), weighted hyper-grid selection is used for a single 2DOF robot: Two partial hyper-grids are created one with cells occupied by only a single milestone and one with cells occupied by more than one milestone, (see left figure). One of these two hyper-grids is randomly selected, (denoted by red box in central figure). Then, from within that grid, one of the cells occupied by milestones is randomly selected, (denoted by the small red square in right figure). From within this cell, a milestone is randomly selected.

optimal gridcell referencing, (insertion and selection is logarithmic).

Hashtables can also provide an efficient means of weighting the gridcells further. For example, in Figure 3.4 (b) the gridcells occupied by only a single milestone are given much higher weighting than all other gridcells.

This is accomplished using two partially allocated hyper-grids. When a new milestone is generated, it is added to the first of the two hyper-grids HG_1 only if it will be the sole milestone to occupy a gridcell, otherwise it is added to the second hyper-grid HG_{2+}. When sampling to obtain a new milestone, a random (weighted) selection of one of the two hyper-grids HG_i is made, followed by selecting a milestone from HG_i as described above.

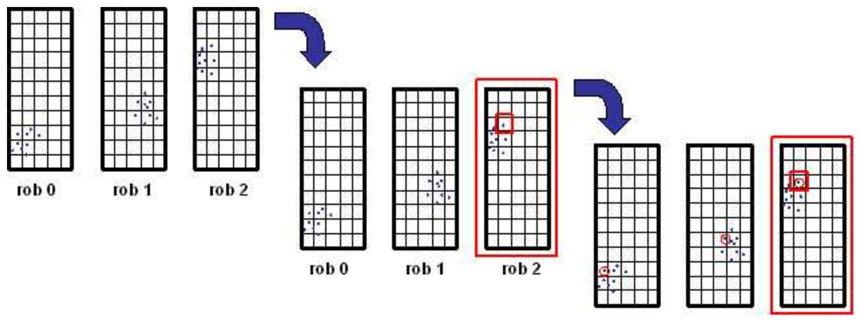


Figure 3.5: Multi-Grid Milestone Selection for three 2DOF robots: Each robot is allotted a grid to characterize the coverage of its configuration space, (see left figure with three grids). Blue dots denote states corresponding to milestones in the roadmap. Of the three robots in the case presented, one is randomly selected (the grid for robot 2 in the central figure). Then, from within that grid, one of the cells occupied by milestones is randomly selected, (denoted by the small red square in right figure). From within this cell, a milestone is randomly selected. This milestone has a corresponding robot state denoted by red circles in each of the three grids.

Multi-Grid Milestone Selection

This technique was designed to weight the expansion effectively for multi-robot PRM planning [14]. The hyper-grid technique is modified such that each robot is assigned its own grid of cells to characterize a coverage of its configuration space. As milestones are added to the roadmap, these grids are updated to represent the milestone coverage particular to that robot. To select a milestone for expansion, a robot is randomly selected followed by random selection of an occupied cell in that robot's grid. Finally, a milestone from this cell is selected randomly.

Within each of the R robot grids, there will be K^D grid cells, where D is the degrees of freedom of the robot that are divided up into K cells. In total there will be R x K^D cells.

Random Point Milestone Selection

Used often for Rapidly-exploring Random Trees (RRTs) [38], (a variant of PRM planning), is random point milestone selection. The idea is to randomly pick a point in the configuration space, then find the milestone in the roadmap to which it has

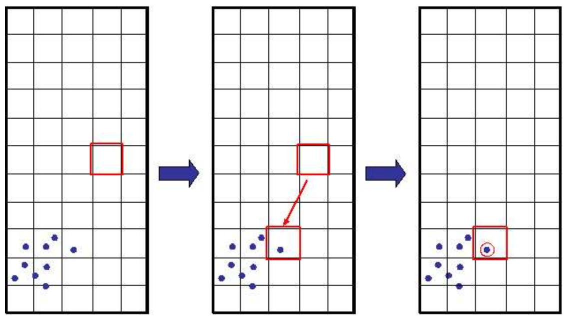


Figure 3.6: Random Cell Milestone Selection for a single 2DOF robot: Within the 2D configuration space, a cell is chosen randomly, (denoted by the red square in left figure). Next, the closest cell from a randomly subsampled list of cells is selected, (denoted by red boxes in central figure). Finally, from all milestones in this closest cell, one is randomly selected (denoted by the red circle in right figure).

the smallest Euclidean distance. A drawback of this technique is that a search for the milestone with the shortest distance must be done for each expansion.

One way to minimize the effects of this drawback is to only consider a small sample of randomly selected milestones in the roadmap for each expansion. Also, instead of picking a point, one can randomly select a gridcell c_{random} from a discretized grid of the configuration space, then find the occupied gridcell c that is closest to c_{random} using the Manhatten distance metric. From c, a milestone is selected randomly.

3.6.2 Results

To compare the different techniques of milestone selection, simulations were conducted involving three robots, each with one degree of freedom. Hence, if each robot i is defined by state x_i whose feasible set is $[x_{min}, x_{max}]$, then the configuration state of the system can be defined by $[x_1, x_2, x_3]$, and the product of the three individual configuration spaces is a cube. Inter- robot collisions are simulated by adding non-permissible *collision regions* to the cube.

At the start of each simulation, a point in the configuration space is randomly selected to be the initial milestone and is added to the roadmap. During each iteration of the simulation, a milestone in the roadmap is selected (using one of the

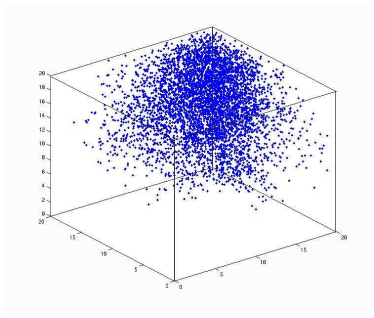


Figure 3.7: Visualization of milestones sample from a cubic configuration space: This simulation has three 1DOF robots. Blue dots denote milestones in this configuration space.

previously discussed techniques). To this milestone, small random variations are applied resulting in a candidate milestone. If the candidate lies within the boundaries of the configuration space, (i.e. if x_i lies within $[x_{min}, x_{max}]$ for all i) and does not lie in one of the collision regions, then it is added to the roadmap as a new milestone. In this manner, milestones are continually added to the roadmap which expands over the configuration space.

To establish a comparison metric, the joint configuration space (i.e. the cube) is divided into 3375 smaller *occupancy cubes*. The coverage of the configuration space is then measured by the number of these smaller cubes occupied by at least one milestone.

Illustrated in Figure 3.8 are the average configuration space coverages from expanding a roadmap using each of the above mentioned sampling techniques. Aside from the unweighted case, each technique demonstrates an initial region of fast expansion, followed by a region of slower expansion. However, the ratio of these two regions differs greatly between sampling techniques. The multi-grid approach tapers off quickly to a very slow expansion. The random cell technique (from RRT) provides a good rate of coverage, especially when considering the composite of three planners running in parallel. The hyper-grid techniques, (including the dynamically allocated hyper-grid), demonstrated superior performance. It was not until a majority of the configuration space was covered before their rate of expansion decreased significantly.

A second metric for comparing these sampling techniques is the uniformity of the

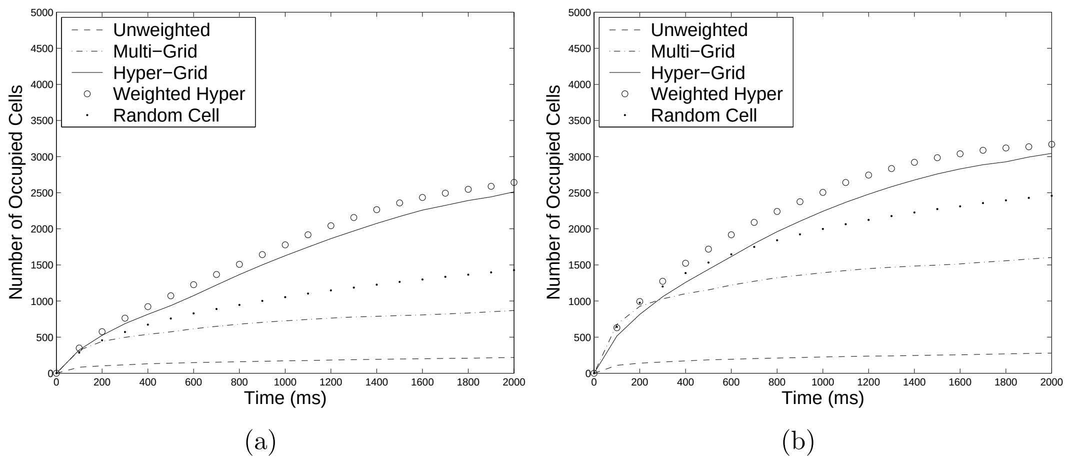


Figure 3.8: Milestone Selection Selection Techniques - Coverage: The coverage of 3375 cells hyper cube are shown for various sampling techniques. In a), the coverage from a single planner is plotted. In b), the composite coverage of three different planners running in parallel is plotted.

expansion. To measure uniformity, the variance of *occupancy cubes* - the square of the average difference between the occupancy of the cubes and the average occupancy, was used. In Figure 3.9, the variance of occupancy cube milestone density is plotted as a function of time. It is clear that the unweighted approach leads to a very non-uniform milestone expansion. The variance increases with time indicating that some *occupancy cubes* are occupied by many more milestones than others. Other techniques demonstrated a slightly increasing variance, indicating a more uniform milestone expansion (i.e. most areas of the configuration space have generally the same density of milestones).

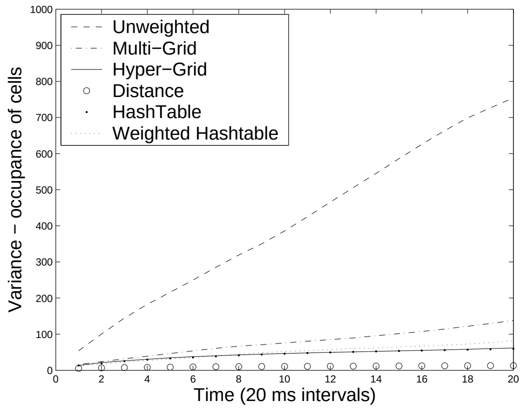


Figure 3.9: Roadmap Sampling Techniques - Variance: The uniformity of the configuration space coverage is measured as the variance of the occupancy of cells.

3.7 Milestone Generation

In [27], a two-step sampling diffusion technique was introduced where new milestones are generated in vicinities of the roadmap that have a low density of milestones. Discussed in the previous section was the first step: the random selection of a milestone m from the roadmap. This section investigates the second step: the generation of new milestone in the neighborhood of m. Within Algorithm 1, this is referred to as the PROPAGATE function, (see step 6). This section presents a new method of generating milestones, called *serial expansion*, that increases the likelihood of success. This decreases the number of required collision-checks and speeds up planning.

3.7.1 Serial Vs. Parallel Expansion

Within the PROPAGATE function of Algorithm 1, several candidate paths from m are generated by integrating Equation 3.1 with randomly selected values for u. The function iterates until u induces a collision-free path, whereby it returns a milestone m_{new} defined by the path endpoint. The order in which the different control inputs of u are randomly selected can affect the number of collision-checks necessary to successfully generate a new milestone.

Previous research for single robot planning has used a parallel approach to milestone generation in that all control inputs are selected simultaneously, followed by collision checking [34]. If the trajectories connecting states in the existing milestone to states in the newly generated milestone are collision-free, then the new milestone is added to the roadmap. For multi-robot planning, the likelihood of successfully generating new milestones decreases with the number of robots, which in turn slows down the roadmap expansion.

In this research, a serial approach is introduced. For each robot, the control inputs are randomly selected and collision-checking is carried out between it and all previously expanded robots, (refer to Algorithm 2). For example, consider generating a new milestone by expanding a milestone defined by $m(t, x_A, x_B, x_C)$ for robots A, B and C. First, the amount of time Δt between milestones is randomly selected. Second, a new state x'_A is generated by applying random inputs to state x_A. Then x'_B is generated and a check is made to ensure that the trajectory from x_B to x'_B is collision-free with the trajectory from x_A to x'_A. Random inputs are continually used to obtain a new x'_B until collision-free trajectories are obtained. Finally a new state x'_C is generated and a check is made to ensure that the trajectory between x_C and x'_C is collision-free with the trajectories from x_A to x'_A and from x_B to x'_B. Again, candidate states for x'_C are randomly generated until collision-free trajectories are obtained. What results is a collision-free milestone defined by $m'(t', x'_A, x'_B, x'_C)$, where $t' = t + \Delta t$.

As shown in Algorithm 2, there is also a timeout check. This is used to ensure that the algorithm does not get stuck in a particularly difficult expansion. For example, the first robot state expanded could result in a trajectory for which all other robot state expansions will lead to collision.

The purpose of using serial expansions over parallel expansions is that information from previous failed state expansions is used for future expansion attempts. That is, as each individual robot state is expanded, the previous successful robot state expansions are reused. In contrast, parallel expansion throws out this information at every expansion attempt.

To further justify the use of serial expansions, equations that predict its superior

Algorithm 2 Serial Milestone Expansion

```
1.   Set $\Delta t$ to be a random time from $[0, \Delta t_{max}]$
2.   **For** $i = 1$ to $R$ robots
3.       Randomly select a robot $r$ that hasn't been selected yet
4.       **While** true
5.           Randomly select control inputs $u_r$
6.           Generate new state $x'_r$ by applying $u_r$ to $x_r$
7.           **If** path connecting $x_r$ to $x'_r$ is collision-free
8.               Add $x'_r$ to new milestone $m_{new}$
9.               **Break** while loop
10.          **If** timeout
11.              **Return** null
12.  **Return** new milestone $m_{new}$
```

performance are presented. These equations are based on the following definitions:

Definition Let m be a milestone selected for expansion, where m is defined by the states of R robot at time t:

$$m = m(t, x_1, x_2, ...x_R) \tag{3.2}$$

Definition Let m' be a milestone resulting from the random propagation of m, where m' is defined by a set of R robot states at time t:

$$m' = m'(t', x'_1, x'_2, ...x'_R) \tag{3.3}$$

Definition Given the states of two robots, x_i and x_j, define the probability that the trajectories produced from random propagations to new states x'_i and x'_j are collision-free as:

$$p_{ij}\left(x_i, x_j\right) \tag{3.4}$$

Definition Given the states of two robots, x_i and x_j, and a propagated state x'_i, define the probability that the trajectory produced from a random propagation from x_j to a new states x'_j will be collision-free with the previously constructed trajectory from x_i to x'_i as:

$$q_{ij}(x_i, x_j, x'_i) \tag{3.5}$$

These two probabilities can be related by summing over all N possible propagations for robot i, that have probability $r(x_i')$.

$$p_{ij} = \sum^{N} q_{ij}(x_i, x_j, x_i')r(x_i') \tag{3.6}$$

This equation can be simplified by noting that each possible propagation for robot i will have equal probability $1/N$.

$$p_{ij} = \sum^{N} q_{ij}(x_i, x_j, x_i')\frac{1}{N} \tag{3.7}$$

$$= q_{ij,avg}$$

Using these definitions, the average number of collision checks necessary for expansion can be calculated and compared for parallel and serial expansions, (see Appendix A for calculations).

The average number of collision checks necessary for a successful parallel expansion of R robot states is approximated in Equation 3.8. In this expression, p_i is the probability that the i^{th} pair of state expansions is collision-free.

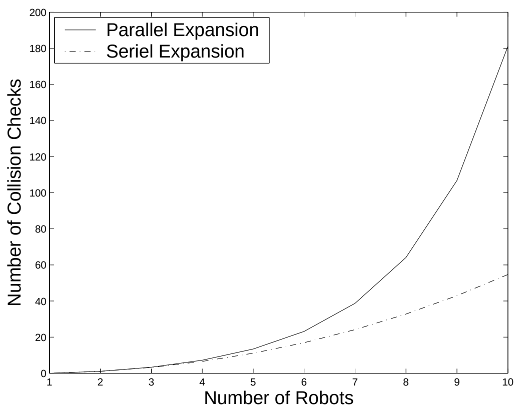
$$C_{avg,parallel} = \left(\frac{1}{\prod_{i=1}^{C_{max}} q_{i,avg}} - 1\right)\left(\frac{\sum_{l=1}^{C_{max}} lQ_l}{\sum_{l=1}^{C_{max}} Q_l}\right) + C_{max} \tag{3.8}$$

Where:

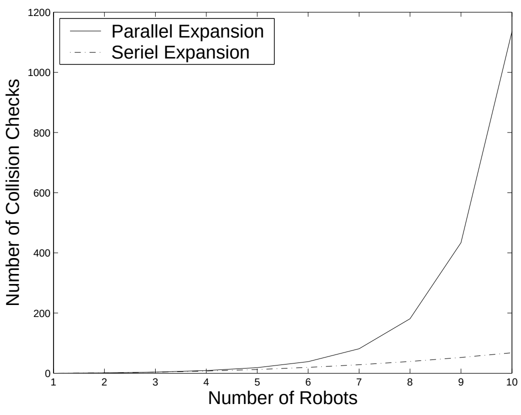
$$C_{max} = \frac{1}{2}R(R-1) \tag{3.9}$$

$$Q_{l,parallel} = \left(\prod_{i=1}^{l-1} p_{i,avg}\right)(1 - q_{l,avg}) \tag{3.10}$$

The average number of collision checks necessary for a successful serial expansion of R robot states is approximated in Equation 3.11. This equation is a function of q_i,



(a)



(b)

Figure 3.10: The average number of collision checks required for a successful expansion. In (a), $q = q_{avg} = 0.95$. In (b), $q = q_{avg} = 0.90$

the probability that the i^{th} pair of state expansions is collision-free,

$$C_{avg,serial} = \sum_{m-1}^{R-1} \left(\left(\frac{1}{\prod_{i=1}^{m} q_i} - 1 \right) \left(\frac{\sum_{l=1}^{m} l Q_l}{\sum_{l=1}^{m} Q_l} \right) + m \right) \tag{3.11}$$

Where:

$$Q_{l,serial} = \left(\prod_{i=1}^{l-1} q_i \right) (1 - q_l) \tag{3.12}$$

Figure 3.10 illustrates how both of these functions scale with the number of robots. To compare them, the probabilities q_i is approximated as $q_{i,avg}$. Under this approximation, the $C_{avg,parallel}$ increases more rapidly with the number of robots R than $C_{avg,serial}$.

3.7.2 Simulation Results

To compare the two methods of expansion, 50 simulations were run in which a roadmap was expanded continuously for 0.5 seconds. At each milestone expansion, both the parallel and serial methods were implemented.

For each simulation, data was recorded including q_i, $q_{i,avg}$, and the number of collision checks during each expansion. With this information, the average number of collision checks necessary for a successful expansion were predicted based on theorems

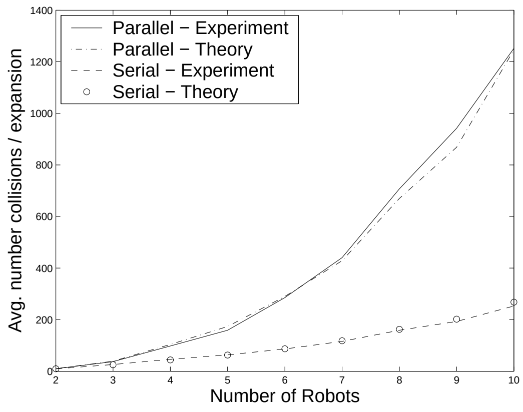


Figure 3.11: Parallel Vs. Serial Expansion.

A.1.3, A.1.4 and compared with the recorded number for each expansion. Results are plotted in Figure 3.11.

There are two points to be observed from these plots. The first is that for both the serial and parallel expansions, the theorems accurately predicted the average number of collision-checks necessary for a successful expansion.

The second point to be observed is the difference in scalability between serial and parallel expansion methods. That is, as the number of robots increases, the number of collision-checks required with parallel expansion grows more quickly than with serial expansion. Note that there is a direct correlation between the number of collision checks necessary for an expansion and the time taken to complete an expansion. Thus, on average, serial expansions take less time than parallel expansions.

Note that these results are based on single-body mobile robots, where the majority of collision checks in an expansion are those between different robots. The reduction in expansion time experienced with Serial expansions is a result of decreasing the number of these inter-robot collision checks. When planning for multi-body robots, the majority of time might be spent performing collision-checks between different parts of the same robot. In these situations, the effects of Serial expansion could be diminished.

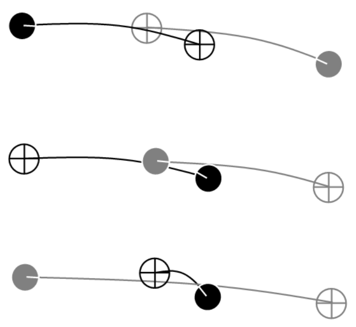


Figure 3.12: Velocity Tuning Counter Examples: Three examples of paths that can not use velocity tuning to become collision-free.

3.8 Defining the Endgame Region

For single-query PRM planning using a single directional search, a tree of milestones is grown until it connects with the goal state. Whether or not the tree connects to the goal state is determined by how one defines the *endgame region* E: a region of the free space in which all configurations have a simple connection with the goal configuration. This region is not calculated explicitly. Instead, admissibility tests are conducted to determine if any configuration belongs to the endgame region.

The method in which an endgame region is defined for a specific planning problem can significantly alter the success of the planner. A key to successful planning is to enlarge the endgame region as much as possible [34]. This increases the possibility that a roadmap will intersect with the endgame region and provide a feasible solution, i.e. the larger the endgame region, the higher the probability a milestone in the roadmap will belong to the endgame region and hence the higher the probability of finding a solution.

A second desired characteristic of the endgame region is that the admissibility test be easily calculated. This test will occur for every new milestone added to the roadmap and will greatly affect the speed of the planner.

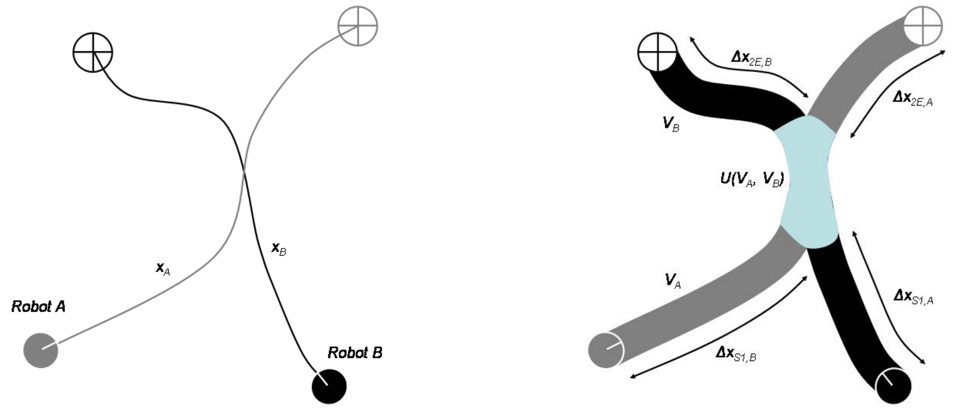
3.8.1 Proposed Endgame Region

Previous approaches to defining the endgame region fail to meet the above mentioned requirements when applied to multi-robot planning problems. In [7], the endgame region is defined to be a ball of small radius centered at the goal. This works well for configuration spaces of low dimensionality. However, as the dimensionality increases, the likelihood of sampling a milestone within the ball of fixed radius decreases rapidly.

For some robots, it is possible to analytically compute one or several canonical control functions that exactly connect two given points while obeying the kinodynamic constraints (e.g. [49]). If such control functions are available, one can test if a milestone belongs to E by checking if the canonical control function generates a collision-free trajectory connecting m to the goal state. A similar example method is found in [34], where cubic splines take the place of the control function. The cubic splines were generated based on k randomly selected end-times. If any of the k splines were collision-free and satisfied all kinodynamic constraints, the milestone was said to belong to the endgame region.

This section presents a new endgame region for multiple mobile robot planning that exploits some geometric properties of a multi-rover system. In doing so, it provides a region that is not only larger than that described in [14], but easily calculated. The endgame region presented is based on the concept of *velocity-tuning* - prescribing a time parameterization to path to produce collision-free trajectories [30]. This is accomplished by discretizing the path into trajectory points defined by both space and time. Allowable velocities (e.g. $v \in [0, v_{max}]$) must be considered in carrying out this parameterization.

The new endgame region presented here aims to include those milestones from which the simple paths that connect them to goal states can be velocity-tuned to produce a collision-free trajectory set. Specifically, to check if a candidate milestone m belongs to the endgame region, a test is done to see if the simple paths connecting robot states in m to their respective goal states can be velocity-tuned. It is essential that this test rule out non-admissible cases (see Figure 3.12), but still be fast so as not to slow down the roadmap expansion.



(a) (b)

Figure 3.13: Defining variables for *Leadability*

The test is based on the property of *Leadability*, defined below, that indicates when paths can be velocity-tuned. Simply stated, robot paths are *Leadable* if one robot can take the lead and pass through the intersection(s) of the paths before the other robot. Provided below are two easy-to-calculate conditions that sufficiently (not necessarily) demonstrate *Leadability* for the implementation described in this dissertation. These conditions are used to develop the endgame region test.

Nomenclature

x_i: candidate path for robot i

V_i: volume of the workspace swept by the path x_i

$U(V_i, V_j)$: the union of V_i and V_j

$t_{i,U-}$: time robot i enters $U(V_i, V_j)$

$t_{i,U+}$: time robot i leaves $U(V_i, V_j)$

Definition Consider a pair of paths $\{x_A, x_B\}$ for robots A and B. The paths intersect at $U(V_A, V_B)$, the union of volumes V_A and V_B swept out by the respective robot paths. The path pair $\{x_A, x_B\}$ is said to be (A, B)*Leadable* if there exists a time parameterization for the paths in which robot A can pass through $U(V_A, V_B)$ before robot B enters it, thus forming a collision-free trajectory set.

Given initial states of the robots are far enough away from $U(V_A, V_B)$, and given that enough variability exists in their velocities, then it is fairly easy to show whether or not a path pair is (A, B)*Leadable*. The core requirement is that finite values for times $t_{A,U+}$ and $t_{B,U-}$ exist such that $t_{B,U-} > t_{A,U+}$. That is, the time at which robot B enters $U(V_A, V_B)$ is after the time at which robot A leaves $U(V_A, V_B)$.

For this implementation it is assumed that robots have allowable velocity $v \in [0, v_{max}]$. Furthermore, it is also assumed that robots have infinite acceleration and that robots can change velocity instantaneously (e.g. stop on the spot). Under these assumptions, it is straightforward to show that sufficient (not necessary) conditions for a path pair $\{x_A, x_B\}$ to be (A, B)*Leadable* are:

1. Robot A's path end location $x_{A,end}$ does not intersect V_B.
2. Robot B's path start location $x_{B,start}$ does not intersect V_A.

While this property helps determine whether two paths can be velocity-tuned, it alone will not provide information on whether a set of $R > 2$ paths can be velocity tuned to be collision-free. For this reason, the definition of Leadability is generalized to any number of robots:

Definition A path set $\{x_A, x_B, x_C, ...x_R\}$ for R robots is said to be $(A, B, C, ...R)$ *Leadable* if there exists a time parameterization for the paths in which each robot in the list $x_A, x_B, x_C, ...x_R$ can pass through their path union $U(V_A, V_B, V_C, ...\ V_R)$ before the next robot in the list enters the union, thus forming a collision-free trajectory set.

To check whether a milestone belongs to the new velocity-tuneable endgame region, a test is made as to whether the simple paths connecting robot states in the milestone to the goal states make up a path set that is Leadable. While no formal proof is presented, it should be clear that a path set is Leadable if each path pair in the set is Leadable. For example, the conditions (Q, R)*Leadable*, (Q, S)*Leadable* and (R, S)*Leadable* would imply that the path set $\{Q, R, S\}$ is (Q, R, S)*Leadable*.

To accomplish the endgame region test on a milestone, several steps are carried out on the set of paths that connect the robot states to thier goal states. First, each path within the set must be tested for collisions with obstacles in the environment. If a collision exists, the milestone is rejected.

Second, each pair of paths $\{x_i, x_j\}$ within the set is checked whether or not it is (i, j)*Leadable* or (j, i)*Leadable*. If it is neither, the milestone is rejected. Moving obstacles are also considered in this step as robots that can only be Leadable in one direction (i.e. the obstacle must lead the robots).

Finally, if all the pairs are Leadable in at least one direction, then the test continues to see if the set is Leadable. For each path pair that is only Leadable in one direction, a consistency check is made to ensure that no circularity would prevent the set from being Leadable (e.g if the only lead conditions are (Q, R)*Leadable*, (R, S)*Leadable* and (S, T)*Leadable*, then $\{Q, R, S\}$ is not a Leadable set). If a circularity exists the milestone is rejected, otherwise the milestone is determined as belonging to the endgame region.

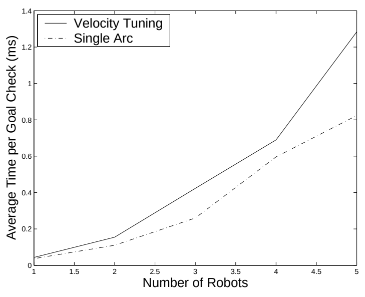
The endgame region is summarized below. Note that only once the set is determined as being Leadable (i.e. a solution to the planning problem is found) does the planner actually assign a velocity profile to the paths.

Definition Let the *Endgame Region* be defined as the set of all milestones such that the arc paths connecting robots to their respective goals form a *Leadable* set. The following criteria must be satisfied to determine if a milestone belongs to the endgame region:

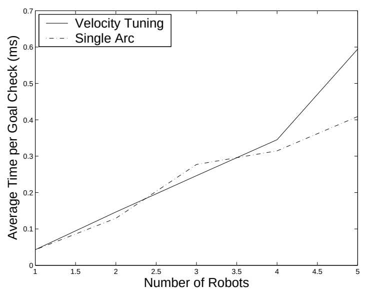
1. Each arc path connecting a robot to its respective goal is collision-free with obstacles.

2. Each pair of arc paths connecting robot states to their respective goals are Leadable.

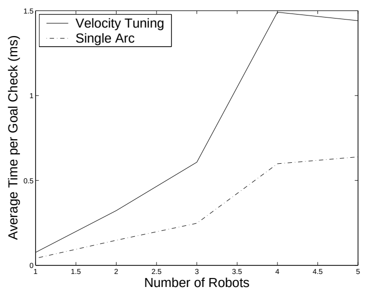
3. The ordering of robots produced from leadability constraints is not circular.



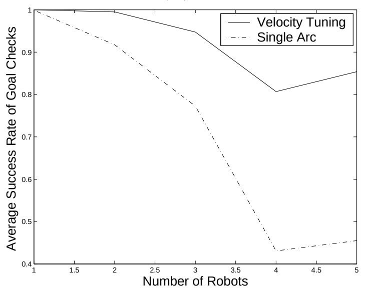
(a)



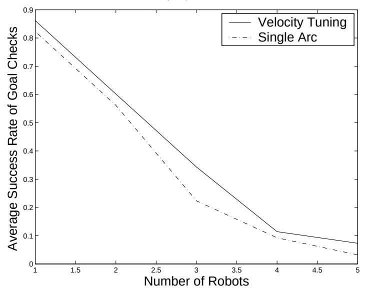
(b)



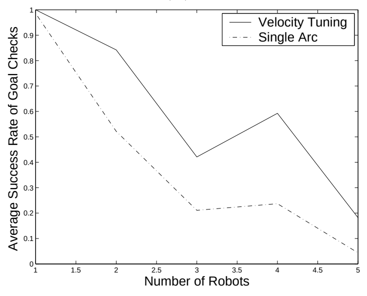
(c)



(d)



(e)



(f)

Figure 3.14: Endgame Region Comparison: In Figures (a), (b) and (c), the time taken for goal checking when using both velocity-tuned and non velocity tuned arc paths is presented. In a), no obstacles were present. Stationary obstacles were present in b), and moving obstacles were present in c). In figures (d), (e), and (f), the percentage of candidate milestones found in the endgame region when using both velocity-tuned and non velocity tuned arc paths is presented. In (d), no obstacles were present. Stationary obstacles were present in (e), and moving obstacles were present in (f).

3.8.2 Results

Two scenarios were used to evaluate the use of velocity-tuned endgame regions. In the first scenario, simulations with up to 5 robots were run in which robots and obstacles were added to random locations of the workspace. In each case the planner was run for 0.5 seconds, and the number of expanded milestones that belong to the respective endgame regions was recorded.

Figure 3.14 illustrates the average relative size of the endgame regions for these experiments. In (a), (b) and (c), the average time it takes to check if the path from a milestone to a goal state is velocity-tuneable is greater than the average time it takes to check if that path is collision-free at some nominal velocity. However, the

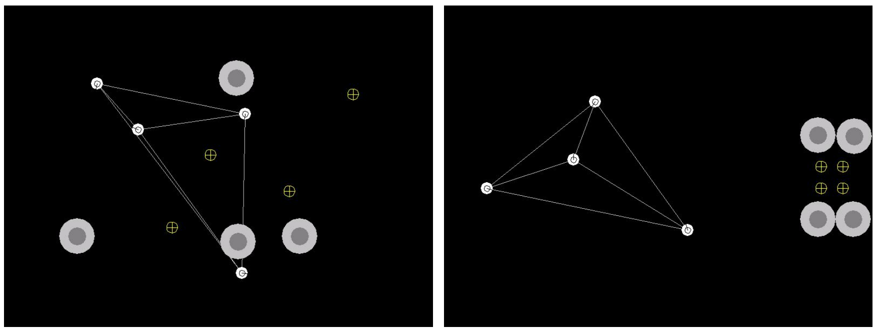

(a) (b)

Figure 3.15: Velocity-Tuned Endgame Region: Sample scenarios used to illustrate increased size of the endgame region attained when using velocity-tuning. The scenarios are illustrated as top-down views of environments involving 4 robots (white circles) and 4 obstacles (gray circles). Goal locations are depicted as gold cross-hairs. Given the easier scenario in (a), the planner using a velocity-tuned endgame region produced only 1.3 times more milestones. However in scenario (b), the planner using a velocity-tuned endgame region produced 22 times more milestones.

average number of milestones with paths to goal states that are velocity tuneable is greater than those that have collision-free paths with nominal velocity (see (d), (e) and (f)). Hence, the velocity-tuned endgame region appears larger, but takes longer to calculate for the average case.

The above experiments show that using a velocity-tuned endgame region yields only a small relative increase in performance for the average scenario. To highlight the true advantage, results from two planning scenarios are compared in which one goal state is more confined than the other. The two scenarios are depicted in Figure 3.15, in which the environment in (a) has been created by randomly selecting robots, obstacles and goal locations. In (b), a more constrained goal state was created. In 0.5 seconds of roadmap expansion, the average planner for case (a) produced 111 milestones belonging to the non-velocity-tuned endgame region, and 144 milestones belonging to the velocity-tuned endgame region. However, in case (b), the average planner produced 1.5 milestones belonging to the non-velocity-tuned endgame region, and 33 milestones belonging to the velocity-tuned endgame region. This example illustrates the advantage of using a velocity-tuned endgame region when tight-coordination is required to attain the goal state.

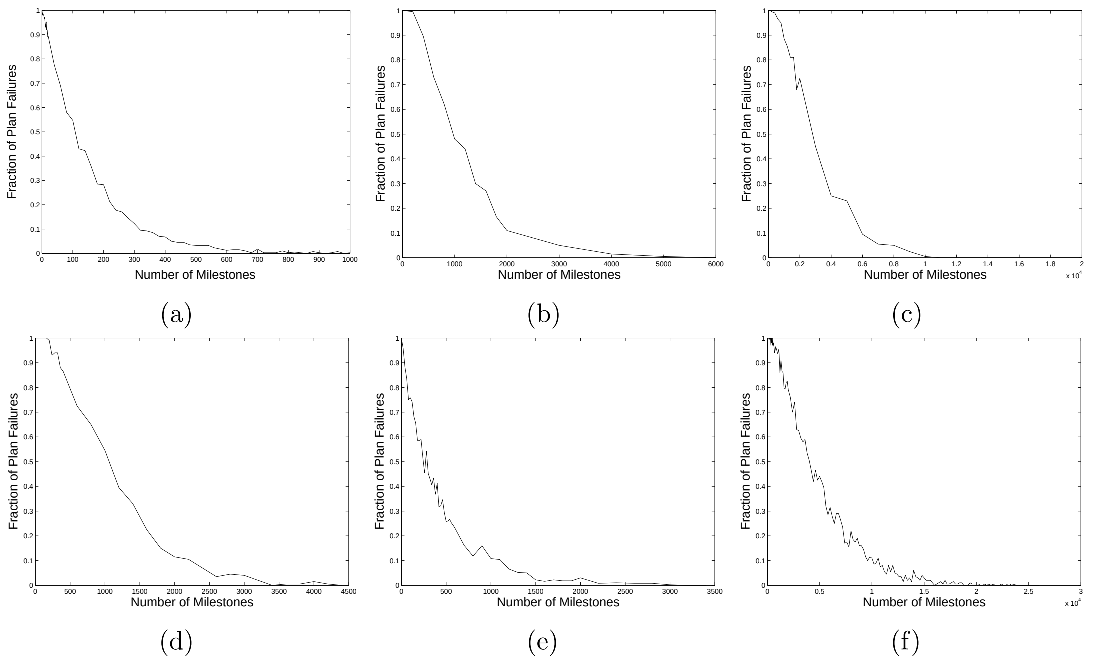


Figure 3.16: Exponential Decay of Planner Failure

3.9 Probabilistic Completeness

Given certain assumptions, Hsu's algorithm is proven to probabilistically complete [27]. That is, it has an exponentially fast convergence for general motion planning problems, including multi-robot planning problems. The analysis is based on two simplifying assumptions: that the configuration space is expansive, and that the coverage converges to a uniform distribution over the configuration space. Because these assumptions are difficult to verify, experiments have been conducted to demonstrate the exponential convergence rate of the planner presented in this dissertation.

Simulations were run for 6 different scenarios of varying complexity, involving up to 5 robots and 10 obstacles within in a 2D workspace. For each simulation, the planner was allowed to expand until x milestones were added to the roadmap, with 100 searches run for each value of x.

A summary of simulation results are plotted in Figure 3.16 as the ratio of failure for increasing values of x. As expected for probabilistic complete planners, there is an exponential decay in the failure rate.

3.10 Summary

A motion planning algorithm that can be queried by robots within Dynamic Robot Networks must be fast enough to plan trajectories on-the-fly. Presented here is a probabilistic roadmap planner with new sampling strategies that increase the algorithm's running time.

First, an appropriate milestone selection strategy was identified. The *hyper-grid* strategy proved to demonstrate a quick uniform roadmap expansion over the free space. Second, a new method of generating milestones for the roadmap was presented. This method, called *serial expansion*, proved faster than the traditional *parallel expansion* method for cases with more than 2 robots. Third, a new endgame region was defined that increases the likelihood of finding a solution for every new milestone sampled.

With these sampling strategies, a PRM algorithm which has been demonstrated empirically to be *probabilistically complete* has been developed that allows for on-the-fly, centralized planning within a Dynamic Robot Network.

Chapter 4

Experiment Implementation

4.1 Introduction

The purpose of this research is to allow multiple mobile robots to navigate in dynamic unknown environments. In Chapters 1 through 3, a solution has been proposed based on using centralized motion planning within Dynamic Robot Networks. To validate this solution, the Dynamic Robot Networks coordination platform and motion planner have been implemented on the Micro-Autonomous RoverS (MARS) test platform at Stanford University.

The MARS test platform consists of 6 mobile robots, several obstacles for robots to avoid, an overhead sensing system, a graphical user interface, and several workstations to handle off-board processing (i.e. motion planning and control signal processing).

This chapter first describes the hardware including the robots, their communication system and their sensing systems. Second, a software architecture based on robot software agents is described, highlighting the inter-agent communication. Finally, this chapter provides specifics on implementing the motion planning algorithm with two particular robots: 1) the MARS rovers and 2) simulated 3D free-floating robots.

With the fully integrated system, experimental results can be obtained that demonstrate the effectiveness of the proposed solution (see Chapter 5 for results).

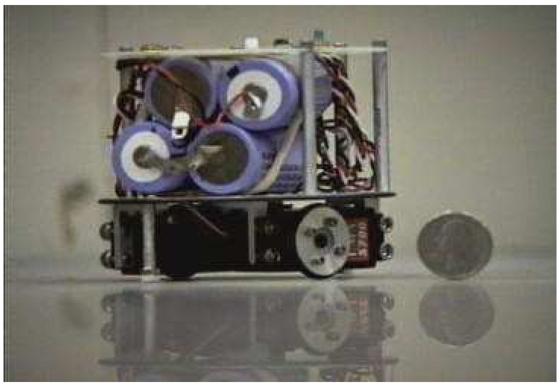


Figure 4.1: A rover on the MARS test platform standing beside a quarter.

4.2 Hardware 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. 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.

Rovers

Rovers were custom built within the Aerospace Robotics Lab (ARL). They are cylindrical in shape with diameter 0.10 m and height 0.10 m. They are built upon a circular metal base, raised 2 cm off the ground by two independently driven wheels and two balance posts. The wheel configuration allows them to rotate on the spot, but inhibits lateral movement so as to induce the nonholonomic constraint.

Upon the base sits a 6V rechargeable battery pack, and RC receiver. Control signals are sent via radio signals to the receiver, which relays them to servo motors that produce maximum speeds of 0.10 m/s . The control signals are sent from an off-board controller.

A circular metal tray sits on top of the robot. Embedded into this tray is a distinct pattern of three LEDs, so that the robot can be tracked by overhead cameras.

Sensing

An overhead vision system is used to provide a surrogate sensing system. Three Pulnix B/W 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 output of each of the three cameras is fed into a Matrox Meteor II frame grabber board that sits within a windows workstation. Camera signals are processed to track LEDs and estimate robot/obstacle states. The vision system updates object positions and velocities at a rate of 15Hz. This state information is sent over the network to any application that requires it.

Interface

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 Figure 4.2). Setting robot goal locations is accomplished with a drag and drop system. New goal locations are sent to the appropriate robot motion planner so trajectories can be constructed.

Controller

Each robot has a low-level control module that sits upon a designated workstation. The module receives trajectories from the motion planner, computes control signals to follow the trajectory, and sends the control signals to the robots. Control signals are computed with a closed-loop Proportional Derivative (PD) control scheme which tracks the desired heading and position of sampled points of the trajectory.

Control signals are sent to robots at a rate of 15 Hz using a wireless radio transmitter. The digital control signals must be converted to analog through a D/A interface board located in the workstation.

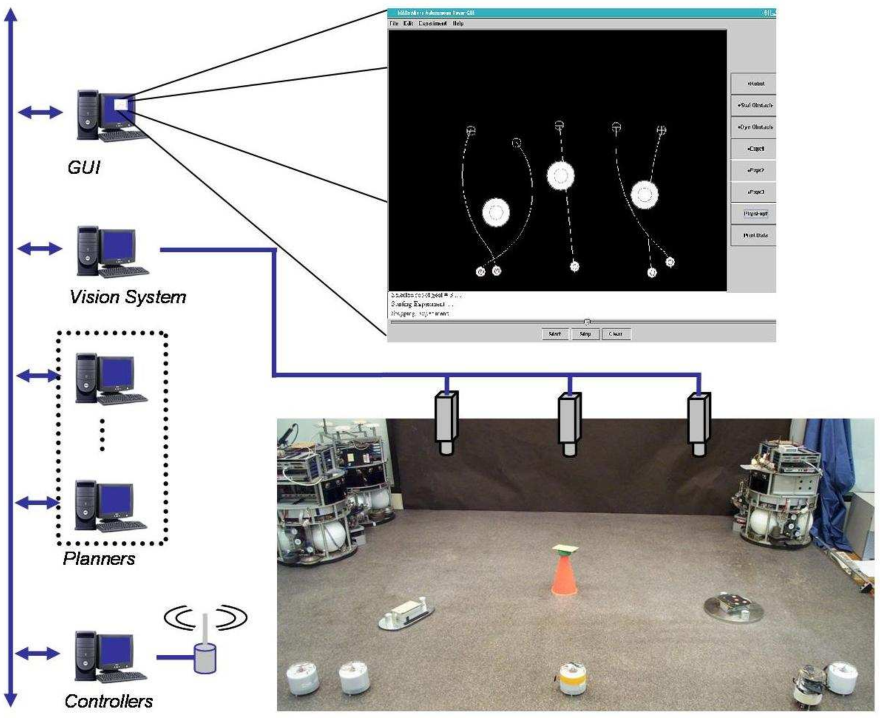


Figure 4.2: MARS: Micro Autonomous Rovers test platform.

Communication

All communication within the MARS platform is accomplished over a wired ethernet Local Area Network (LAN). Figure 4.2 illustrates the data flow in the platform. To facilitate inter application communication, a middle-ware software package is used called Network Data Delivery Service (NDDS). NDDS is a publish/subscribe middleware that sits between applications and the TCP/IP stack.

Simulator

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 4.2, 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.

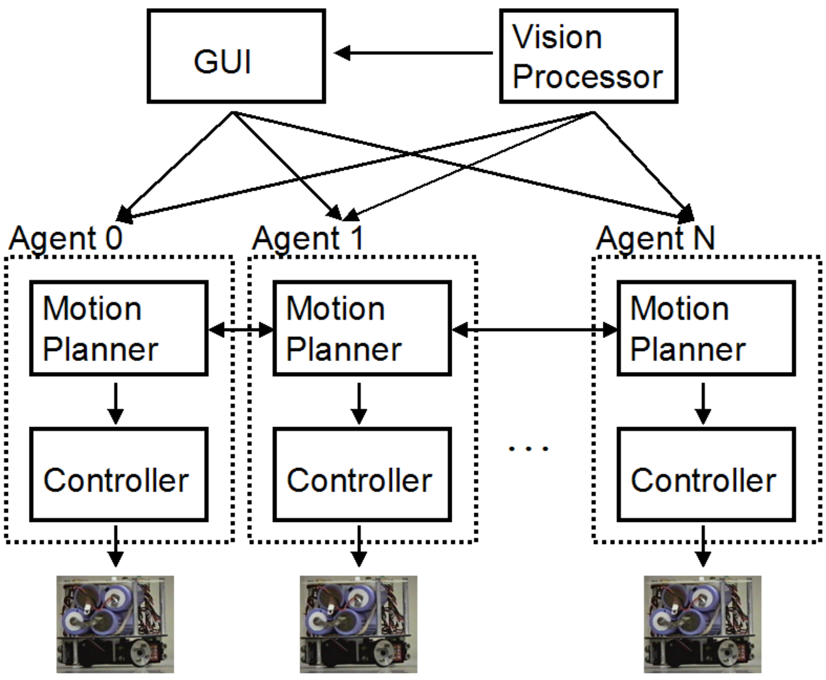


Figure 4.3: Software Architecture

4.3 Software Architecture

Within the MARS test platform, all computer processing for the robots is done off-board by way of robot software agents. Each software agent runs on a workstation and represents the computer processor of a single robot on the table. Inter-robot communication is simulated by inter-agent communication that is accomplished over wired ethernet connections.

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. Figure depicts the data flow between agents in the software architecture.

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 initiates a planning process across the motion planners of robots who belong to the same robot network. To coordinate this planning process, communication between motion planners is required in which planning information and trajectories are published/subscribed to by the motion planners. Final trajectories are also published to be received by controllers. 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 wireless radio 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.

4.4 Motion Planner Implementation

4.4.1 Rover Implementation

When implementing a motion planner for the MARS rover, the most significant constraint to be satisfied is the nonholonomic constraint. In this section, this constraint is described through a mathematical model. This model is used when generating new milestones in the PRM roadmap, and for the endgame region admissibility tests.

Rover Model

The PRM motion planner constructs feasible trajectories based on a robot model that takes into account any significant dynamic or kinematic constraints. The MARS rovers have an extremely fast acceleration, and hence a controller can track any desired wheel velocity step response (within the range $[v_{min}, v_{max}]$) with a very short settling time. Thus the main physical constraint is not the dynamics of the rover, but the nonholonomic kinematics described by Equation 4.1. In this equation, θ represents the robot's heading, while $\dot{x_1}$ and $\dot{x_2}$ represent velocities within a 2 dimensional Cartesion coordinate system, (see Figure 4.4).

$$\tan\theta = \frac{\dot{x}_1}{\dot{x}_2} \tag{4.1}$$

This constraint can be reformulated as a function of the left and right wheel velocities v_{right} and v_{left}:

$$\dot{x_1} = \frac{v_{right} + v_{left}}{2}\cos\theta \tag{4.2}$$

$$\dot{x_2} = \frac{v_{right} + v_{left}}{2}\sin\theta$$

$$\dot{\theta} = v_{right} - v_{left}$$

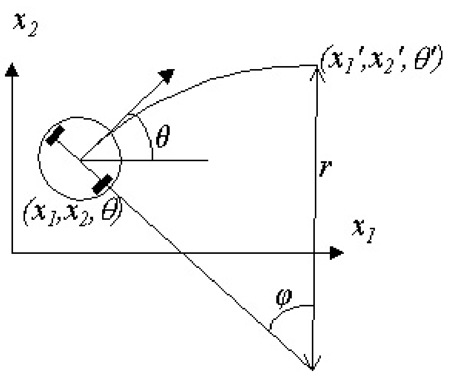


Figure 4.4: Rover description.

Milestone Generation

One of the main advantages of using the PRM planner presented in Chapter 3 is that it takes nonholonomic constraints into consideration while planning. This consideration occurs at the milestone generation stage of planning, (refer to Algorithm 1 in Chapter 3).

A milestone m is described by a set of R robot states:

$$m = m(t, X_1, X_2, X_3, ..., X_R) \tag{4.3}$$

where each state is described by two cartesian coordinates and orientation.

$$X_i = X_i(x_1, x_2, \theta) \qquad i = 1..R \tag{4.4}$$

The generation of a new milestone is initiated by selecting an existing milestone (i.e. the parent milestone) from the roadmap. This milestone is propagated to a new milestone (i.e. the child) by applying randomly selected piecewise control inputs U to the parent milestone for a random amount of time $\delta t \in [t_{min}, t_{max}]$. Each robot state within the parent milestone is propagated by:

$$X_{i,child} = f(X_{i,parent}, U_i, \delta t) \qquad i = 1..R \tag{4.5}$$

For any rover i, control inputs are left and right wheel velocities:

$$U_i = [u_{i,j} \;\; u_{i,j}] \qquad j = right,\ left \tag{4.6}$$

$$u_{i,j} \in [v_{min}, v_{max}]$$

By applying these control inputs, the nonholonomic constraint dictates that the propagation function f in Equation 4.5 produce circular arc paths with constant radius of curvature r. The radius r (as seen in Figure 4.4) can be calculated with geometry as follows:

$$r_i = \frac{u_{i,right} + u_{i,left}}{-u_{i,right} + u_{i,left}} \tag{4.7}$$

The new state of the i^{th} robot in a candidate milestone can then be described by:

$$x_{1,child} = x_{1,parent} + r(+\sin\theta_{child} - \sin\theta_{parent}) \tag{4.8}$$

$$x_{2,child} = x_{2,parent} + r(-\cos\theta_{child} + \cos\theta_{parent})$$

$$\theta_{child} = \theta_{parent} + \frac{u_1 + u_2}{2R}\delta t$$

$$t_{child} = t_{parent} + \delta t$$

Endgame Region

Once a candidate milestone is added to the roadmap, a admissibility test is done to see whether or not it lies within the endgame region. To accomplish this, an endgame milestone is constructed whose robot states are defined by the individual goal positions $x_{1,goal}$ and $x_{2,goal}$ of the robots.

$$X_{i,goal} = g(X_{i,child}, x_{1,goali}, x_{2,goali}) \qquad i = 1..R \tag{4.9}$$

To calculate $X_{i,goal}$, (i.e. to implement g), a circular arc path is constructed that connects the child state $X_{i,child}$ to the goal state $[x_{1,goal} \quad x_{2,goal}]$. Geometry provides the radius of curvature:

$$r_{goal} = \frac{\Delta x_1^2 + \Delta x_2^2}{2\Delta x_2 \cos\theta - 2\Delta x_1 \sin\theta} \tag{4.10}$$

$$\Delta x_1 = x_{1,goal} - x_{1,child}$$

$$\Delta x_2 = x_{2,goal} - x_{2,child}$$

With the radius of the robot c known, the remaining terms in the goal state of the robot can be calculated:

$$\theta_{goal} = 2 \arcsin \frac{\sqrt{\Delta x_1^2 + \Delta x_2^2}}{2r_{goal}} \tag{4.11}$$

$$t_{child} = t_{parent} + \delta t$$

If the arc connecting $X_{i,child}$ to $X_{i,goal}$ is collision-free for $i = 1..R$, then the new child milestone belongs to the endgame region and a solution is found.

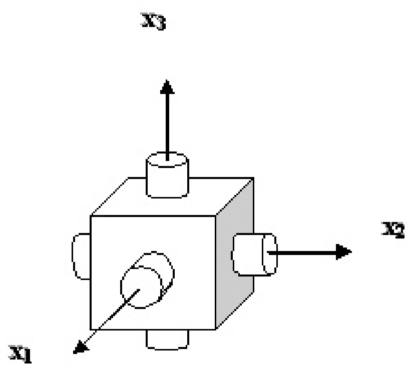


Figure 4.5: State-space model of the free-floating robot.

4.4.2 Space Robot Implementation

This section details the planner implementation for free-floating space robots operating in a 3D environment. First, the robot is modelled and dynamic equations are provided. Using these dynamics, the method for milestone generation is provided, followed by a description of the endgame region.

Free-Floating Robot Model

The free-floating robot is modelled as a simple cube-shaped robot equipped with 6 independent on/off thrusters. Future work could include additional actuators to allow roll, pitch and yaw variation. The state of the R robots can be described by X representing the position with respect to the inertial frame. Milestones are specified by both the state of the R robots and the time robots reach those states $(X_0, X_1, ..., X_R, t)$.

$$X_i = X_i(x_1, x_2, x_3) \in \Re^3 \qquad i = 1..R \tag{4.12}$$

The dynamics of the free-floating robot are those of a $1/s^2$ plant.

$$M\ddot{x}_j = F_{thruster\ j+} - F_{thruster\ j-} \qquad j = 1, 2, 3 \tag{4.13}$$

Table 4.1: Mapping the random variable u_{act} to thruster actuation.

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

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. To prevent the possibility that two opposite-facing thrusters will both be enabled at the same time, only one random variable will be used for both of them. That is, for each pair of opposite-facing thrusters, a control input variable uact is selected where:

$$u_{act} \in [-1, 0, +1]$$

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

$$x_{i,new} = \frac{u_{act,i}}{2M} t_{act}^2 + \dot{x}_i t_{act} + x_i \tag{4.14}$$

$$\dot{x}_{i,new} = \frac{u_{act,i}}{M} t_{act} + \dot{x}_i \tag{4.15}$$

Endgame Region

The endgame region E is defined as the subspace of the configuration space that includes all milestones m_e in which robots are propagated without collisions from states defined by m_e to their respective goal location via a bang-off-bang control

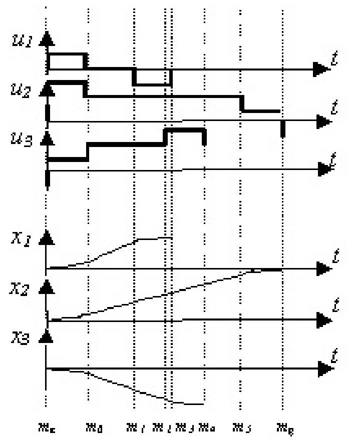


Figure 4.6: Example of actuation required to move one robot from $(0, 0, 0)$ to a goal state. The series of milestones required is $\{m_p, m_0, m_1, m_3, m_4, m_5, m_g\}$.

sequence. An advantage this sequence has over a bang-bang sequence is that it allows us to limit the velocity of the robot, making it easier to replan 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. An example control sequence is provided in Figure 4.6.

Chapter 5

Results

5.1 Introduction

This chapter presents results that demonstrate the success of the Dynamic Robot Network platform in enabling multiple robots to navigate autonomously towards their individual goals. Results were obtained by implementing the Dynamic Robot Networks coordination platform and a PRM planner (see Chapters 2 and 3 respectively) into the Micro-Autonomous Rovers test platform (see Chapter 4).

First, simulations of a particular motion planning problem are presented. Results demonstrate the motion planner's fast running time, and the coordination platform's ability to conduct coordination that is tolerant to network breaks and merges. Also provided are visualizations of rovers navigating through a walled-in, multi-level environment. Within these scenarios, robots conduct clock-driven, rather than event-driven, robot coordination. This type of coordination exemplifies the platform's ability to handle frequent replans as well as providing an optimization strategy.

Second, simulations of free-floating space robots are presented that motivate the use of Dynamic Robot Networks use for such applications. These simulations demonstrate how the motion planner can be extended for operation within 3D environments.

Finally, real robot experiments are documented that demonstrate: 1) on-the-fly network merges/breaks, 2) on-the-fly centralized robot coordination within robot networks, 3) avoidance of moving and previously unknown obstacles, and 4) autonomous

Table 5.1: Rover simulation test scenario data.

Average Number of Robots per plan	2.12
Average Planning Time (ms)	17.3
Average Number of plans per robot simulation	5.07
Average number of networks formed per simulation	49.4

robot navigation towards individual goal locations.

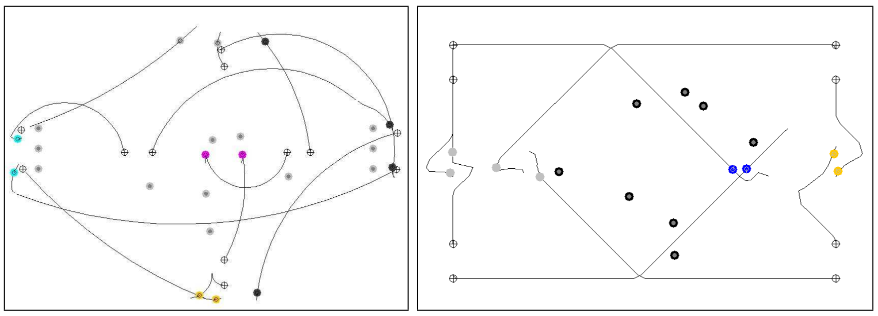
5.2 Robot Simulation

5.2.1 Rover Test Scenarios

Because hardware experiments could only be run with a limited number of robots and obstacles, simulations were run to characterize the performance of the system. To accomplish this, a particular test scenario was chosen that highlights the characteristics of the coordination platform and motion planner.

In this scenario, 12 rovers of diameter 5cm are operating in a 2m x 3m flat workspace amidst 6 stationary and 6 moving circular obstacles of diameter 7cm. To add complexity to the scenario, 4 of the moving obstacles were directed towards a network of 2 robots with little room to maneuver, (see middle of Figure 5.1). Also, 2 networks of 2 robots were placed between a row of 3 obstacles and a workspace boundary. The scenario was run 25 times with different initial random seeds. Despite the apparent difficulty of the scenario, the planner demonstrated fast planning times (an average of 17.3 ms), while planning for up to 5 robots in a network. This fast planning time enables the on-the-fly planning capabilities required for operation in dynamic, unknown environments.

To provide an idea of the level of complexity, robots formed on average 49 different networks throughout simulations that lasted several minutes. This illustrates the ability for centralized coordination despite the merging and breaking of networks to merge and break over time.



(a) (b)

Figure 5.1: Simulation Test Scenarios

In Figure 5.2, a visualization of robots navigating in a walled-in, multi-level environment is provided. Within these scenarios, robot coordination within networks is not only triggered through event detection, but by a single robot that requests new coordination plans with a set frequency. Not only does this demonstrate the platform's ability to coordinate robot actions at a frequent rate, but that replanning can be used to attain better trajectories (according to some pre-determined cost-function).

The example involves 4 rovers. The goal locations for the rovers are located in the middle of the environment's central platform. As shown in Figure 5.2, initial robot trajectories lead robots over drop-offs in unexplored regions of the environment. However, as the rovers traverse these areas, they learn more about the environment. With new information, robots construct new plans that allow for safe movement. This process continues until robots eventually reach their goals.

In attempt to optimize trajectories, one robot within each network (e.g. that with the lowest priority number), calls for a new plan every 2.0 seconds. Robots compare the newly constructed plan with the currently implemented plan. They implement the better of these plans, where the better plan is determined by some predetermined cost function. This assumes the previous constructed plan is still feasible. If not, then no comparison is carried out and the new plan is implemented.

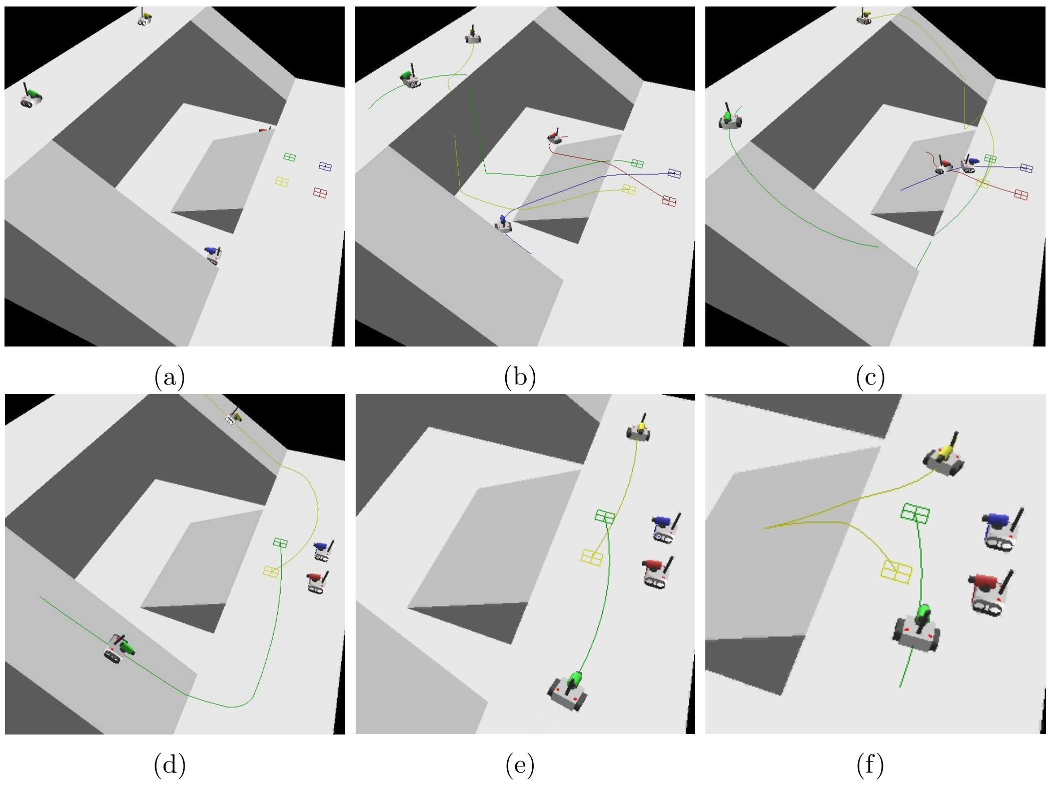


Figure 5.2: A multi-level rover scenario: In (a), four rovers are tasked with the problem of navigating towards the goal locations marked on the mid-level platform, (middle-right in figure). Initial trajectories depicted in (b) are shown to collide and lead robots over drop-offs. As rovers come together, they merge networks and learn more about the environment. New trajectories are then constructed with this information. As rovers move along their paths, they continue to replan in search of shorter paths to their respective goal, (see (d)). In (e), all four rovers merge into one network to generate the final trajectories that lead robots to their goals (d).

Table 5.2: Space Robot simulation test scenario data.

Average Number of Robots per plan	1.84
Average Planning Time (ms)	67.0
Average Number of plans per robot simulation	4.77
Average number of networks formed per simulation	12.2

5.3 Space Robot Simulation Scenarios

To illustrate the applicability of the planner to a 3D environment, simulations with up to 8 free-floating space robots and 8 obstacles were carried out. A test scenario is provided in which robots must cross paths several times. A GUI screen-shot of the scenario is provided in Figure 5.1, (Note that the third dimension is not displayed here.) 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, demonstrating the complexity of the problem.

In comparison with the rover simulation data, the planning times were slower despite planning for fewer robots. This can mostly be attributed to the endgame region definition outlined in the previous chapter. The bang-off-bang control sequence produced efficient trajectories, but the overhead in calculating them was substantial. In the future, it is recommended that robots use a spline function to connect candidate milestones to the goal state as done in [26].

In Figure 5.3, a visualization of a simulation involving 4 robots and 4 obstacles is provided. Large gray cubes denote the obstacles. Trajectories are denoted by yellow lines that end at robot goal locations, (denoted by red cube lattices). Note that in this simulation, all robots are continually in communication range of each other (no communication lines are drawn) and hence only one network is formed at the beginning.

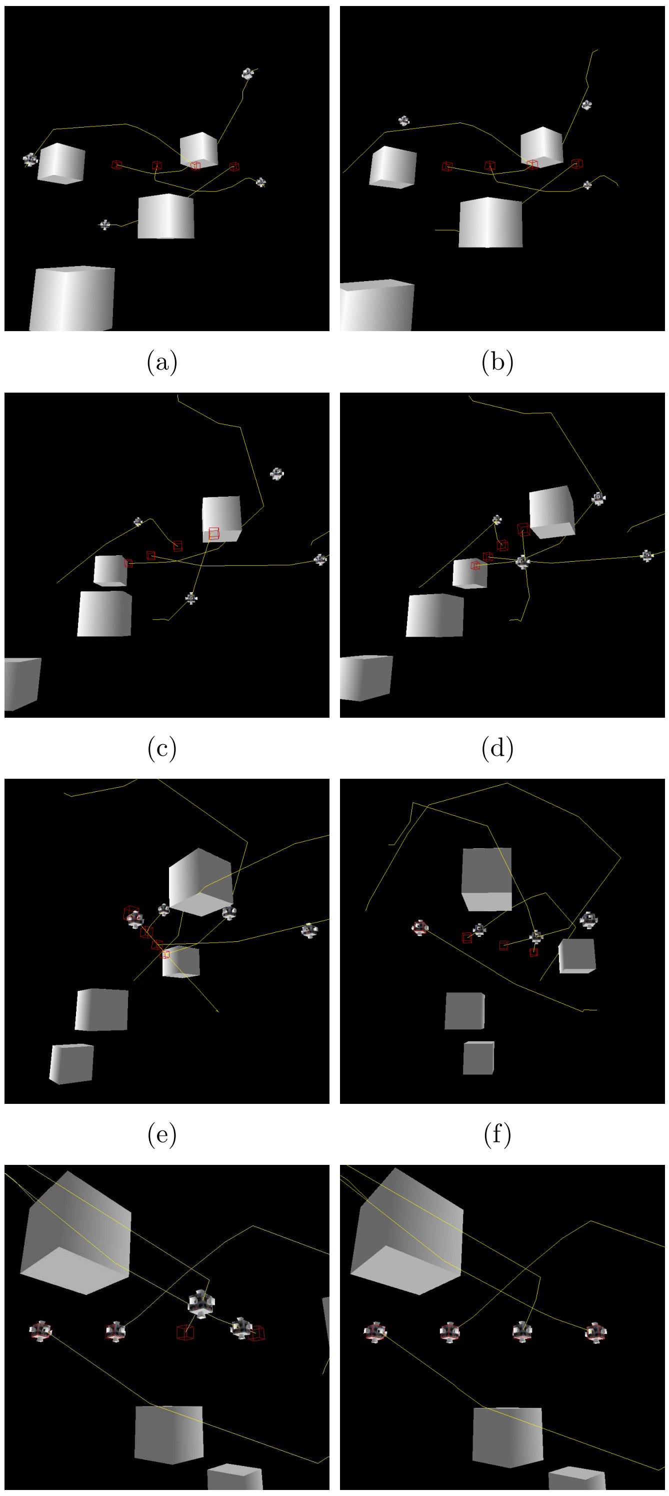

(a) (b) (c) (d) (e) (f) (g) (h)

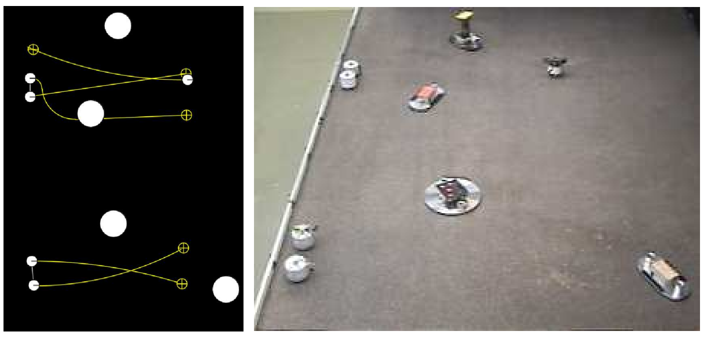
Figure 5.3: Visualizing a 3D space robot simulation.

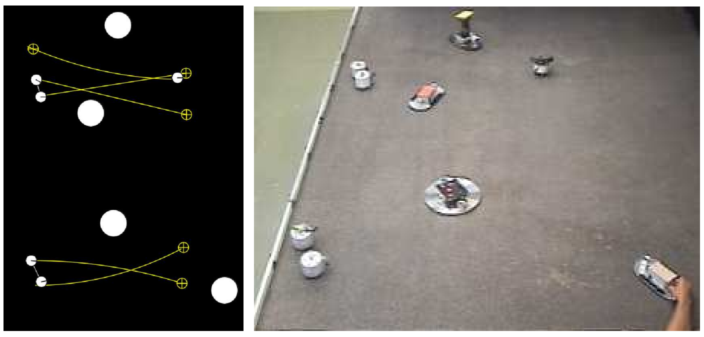
5.4 Robot Experiment

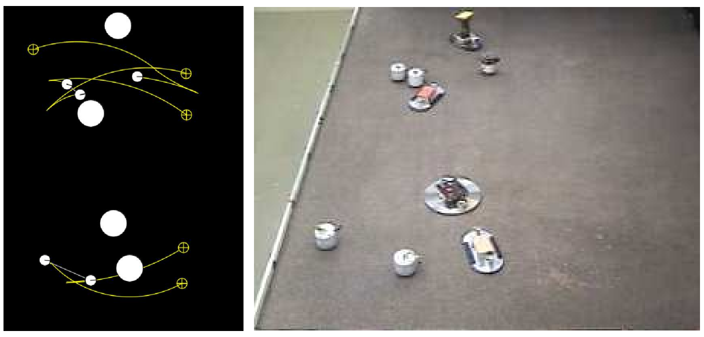
In the simulations above, it was shown how the platform could produce coordination plans quickly to avoid robot collisions and allow robots to achieve goal states. However, to truly validate the Dynamic Robot Network platform's ability to handle on-the-fly robot coordination in unknown, dynamic environments, hardware experiments are required. Such experiments demonstrate the following necessities for real world implementation:

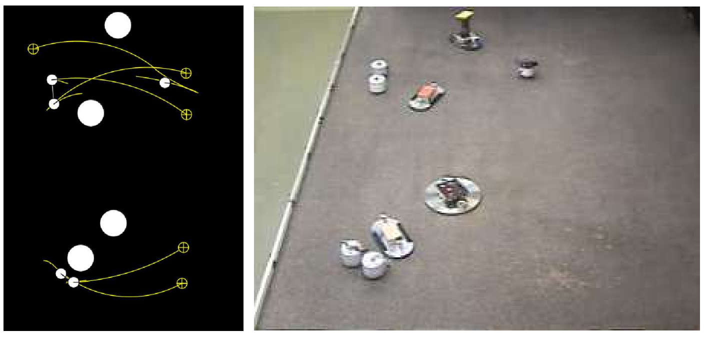
- **Valid System Modelling**- The system dynamics and kinematics used to construct trajectories must be accurate enough such that trajectories will be trackable.
- **Valid Assumptions**- Any assumptions, such as the conciseness of the world model, must remain true for the system to work.
- **Practical Implementation**- The motion planning system must be easy to implement.

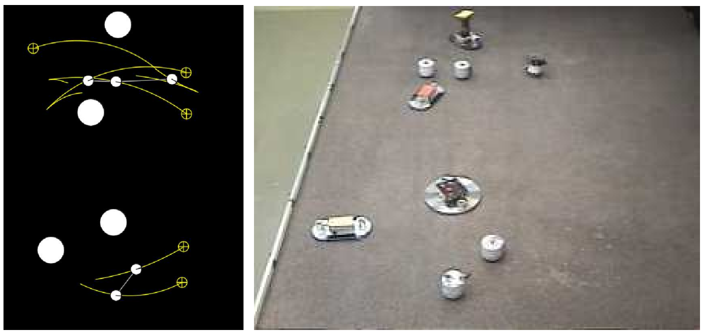
To exemplify the system's ability to function on real hardware, an experiment is documented below in which five rovers and four obstacles were placed on the MARS test-platform. The experiment is depicted in Figure 5.4, where a series of screen-shots of the GUI are on the left with the corresponding hardware photos on the right. As shown in the screen-shots, four of the robots are lined up on the left rail of the test-platform and their goals are located in a line on the right side. The top two of these four robots are close enough to form a local communication network. The goal locations for these two robots are located on the other side of the platform, but swapped such that the lines connecting these two robots to their goal locations will intersect. Likewise, the bottom two of these four robots are also close enough to form their own network and have a similar "swapped" goal configuration. The fifth robot, located in the upper right, has a goal location in the upper left so that its direct route will cross paths with the top left robots. Initially, there are three static obstacles in a line down the middle of the test-platform, and another obstacle located in the bottom right that will be set moving across the table once the experiment begins.


(a)

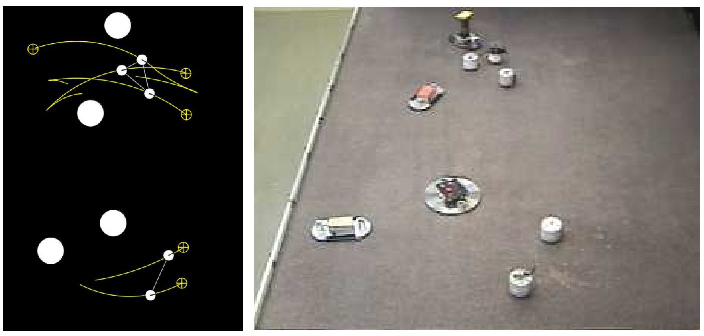

(b)


(c)

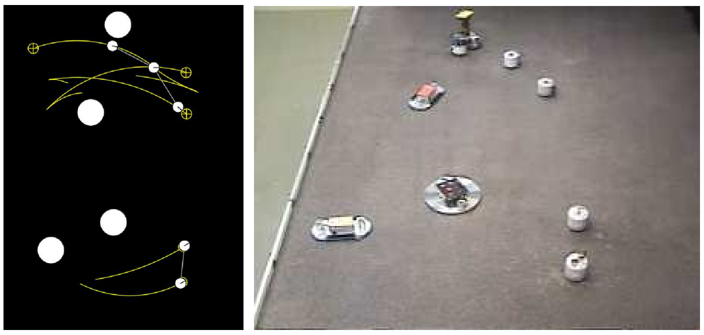

(d)



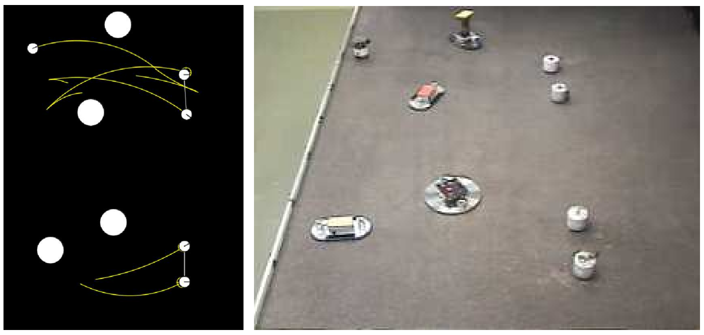
(e)



(f)



(g)



(h)

Figure 5.4: Dynamic Robot Network Experiment

Once the experiment begins, the two robots in the bottom portion of the screen-shots form a network and construct a pair of trajectories. These trajectories illustrate the use of a velocity-tuned endgame region where the uppermost robot of the pair simply waits for the other robot to pass by along it's trajectory, (see Figures 5.4a though 5.4c). In Figure 5.4c, the bottom robot is finally close enough to detect the moving obstacle heading directly towards it. At this point, it communicates this information with the other robot in its network (i.e. they both update their world models) and construct a new set of trajectories that avoid the obstacle (see Figure 5.4d). With their paths clear, the bottom robots follow their new trajectories successfully to their goal destinations (see Figures 5.4e though 5.4h).

In the top portion of the screen-shots, the two robots on the left form a network and construct their initial trajectories with no knowledge of the surrounding obstacles(see Figure 5.4a). At this point the trajectory for the top robot passes directly through the central obstacle. Once the lower robot in the pair moves close enough to sense this obstacle, it communicates its world model with the upper robot and they construct a new set of trajectories (see Figure 5.4b).

The robots continue to follow these trajectories until the the right-most robot of the pair comes within communication range with the fifth robot that started in the upper-right corner. Once these robots can communicate, they merge networks to a new larger network that includes all three robots. After the merge, robots construct new trajectories, (see Figure 5.4c). The new plan requires the robots to break this recently formed network. That is, as the robots follow the trajectories, they move out of communication range of one another (see Figure 5.4d). When the network reforms (Figure 5.4e), no replanning is required since the robots remember their trajectories are already collision-free. In Figures 5.4f through 5.4h, the three robots continue to their goal destinations.

To summarize, not only does this experiment illustrate that the planner can function on real robots (thus meeting the above mentioned criteria), but it highlights the following characteristics of the robot coordination platform:

1. **On-the-Fly Network Merges/Breaks** - For example, the three robots in the top portion of the screen-shots merged into a network, broke this network, and then re-merged as robots moved in and out of communication range.

2. **On-the-Fly Centralized Coordination** - Planning times were all less than 50 ms which enabled robots to plan new trajectories as they moved. One example of this occurred between Figures 5.4a and 5.4b, when the top two robots on the left had to replan to avoid the middle stationary obstacle that was initially out of sensing range.

3. **Avoidance of Moving and previously unknown Obstacles** - The two bottom robots within planned together within their network to avoid an obstacle heading directly for them, (see bottom of Figure 5.4d).

4. **Autonomous Robot Navigation** - Through Dynamic Robot Network coordination, all robots were able to successfully attain their goal state, (see Figure 5.4h).

Chapter 6

Conclusions and Future Research

6.1 Conclusions

Multi-robot systems have received much attention because of their potential to accomplish a variety of complex tasks through cooperation. Example tasks include large-scale construction, hazardous waste cleanup, and planetary exploration. However, to deploy a team of autonomous robots, they must be able to navigate safely.

This dissertation presents a new coordination platform that allows multiple mobile robots to navigate in environments that are both unknown and dynamic. The development of this platform required a new method of robot coordination, new multi-robot motion planning techniques, and validation through experiment.

Robot Coordination

For multiple robots to navigate safely, several issues must be addressed. Two of the key issues are: discontinuous communication and limited sensing. These limitations make it difficult for robots to exchange information and coordinate their actions.

To resolve these issues, a new coordination platform for multiple mobile robots is introduced - *Dynamic Robot Networks*. That is, when robots are within communication range of one another, they establish a communication network in which local sensing information is shared, and robot motion is coordinated through centralized planning. An application level communication protocol is used to manage information sharing and the coordination process across the network.

Results indicate the platform functions well even when frequent network merges or breaks occur. Successful robot coordination was carried out successfully under such conditions, allowing robots to achieve their goal states.

While these results highlighted successful motion planning across ad hoc robot networks, the platform does facilitate other types of coordination. In the future, instances of coordination that involve cooperative behavior will be implemented. Examples include robots working together to construct large structures or search large areas. In both of these examples, coordinating actions across ad hoc robot networks is beneficial when robots have limited communication capabilities.

One of the most valuable aspects of this research is that the platform can be used for coordination between different types of autonomous devices. The platform makes use of a growing technology, ad hoc communication networks, that is finding its place in autonomous devices everywhere, (e.g. passive sensors, unmanned aerial vehicles). As heterogeneous devices become required to work together, such a coordination platform will prove invaluable.

Consider autonomous rescue vehicles merging on the scene of an accident, all trying to accomplish their portion of the rescue task in an efficient manner. Coordinating the actions of all devices will be crucial to a successful rescue. Ideally the type of coordination implemented among Dynamic Robot Networks will translate to such applications.

Motion Planning

Required for safe navigation is motion planning, the construction of collision-free trajectories that lead robots to their individual goal locations. To operate in dynamic, unknown environments, the motion planning must be fast enough for on-line implementation.

To meet this requirement, a randomized algorithm with high processing speed is used. Originally presented in [26], the algorithm is a probabilistic roadmap (PRM) planner. This research has augmented the planner with sampling strategies specifically developed for multi-robot planning problems.

Results indicate decreased planning times over previous sampling strategies. When implemented within the Dynamic Robot Network platform, average planning times were on the order of 20 ms. This enabled on-the-fly planning for avoidance of moving obstacles.

With the improved planning times achieved by the sampling strategies presented in this dissertation, there is still an upper bound on the number of robots that can be planned for at once.

Experiment

To validate the performance of the Dynamic Robot Network platform, it was implemented on the Micro Autonomous Rovers (MARS) test platform. The platform consists of small rovers, several obstacles for rovers to avoid, an overhead sensing system, a graphical user interface (GUI), and several workstations to handle off-board processing.

Experiments involving up to 5 robots demonstrated on-the-fly network merges/breaks, centralized robot coordination within robot networks, avoidance of moving and previously unknown obstacles, and autonomous robot navigation towards individual goal locations. Moreover, experiments indicated that system modelling was relatively accurate, assumptions on the system were valid, and the platform is practical in that it can be implemented easily.

In using the MARS platform, several simplifying assumptions are made on the communication and sensing capabilities of robots. Most notably, a global sensing system was used to estimate all object states. This information was then distributed to robots according to whether objects were close enough to the robots.

While these assumptions do not affect how coordination would occur across networks, they would affect how state estimation and modelling are accomplished. Future work should include an investigation into how different object state estimation algorithms can be incorporated across Dynamic Robot Networks.

6.2 Contributions

Several research contributions were made in developing the Dynamic Robot Network coordination platform. The three categories of contributions include System Control - high-level robot coordination, Technical Contributions - strategies to improve motion planning algorithm speed, and System Validation - various simulations and experiments that demonstrate the system performance.

6.2.1 System Control

1. Developed the *Dynamic Robot Networks* platform that allows for centralized coordination across ad hoc networks.

2. Developed an application level communication protocol to manage information sharing and multi-robot coordination across Dynamic Robot Networks.

6.2.2 Technical Contributions

1. Identified a method of sampling milestones for roadmap expansion when applying PRMs to multi-robot planning problems.

2. Introduced a method of generating milestones - *serial expansion*, which demonstrates faster roadmap expansion over the traditional method - *parallel expansion* when applying PRMs to multi-robot planning problems.

3. Developed a new endgame region definition, based on velocity-tuning, for applying PRMs to multi-rover planning problems. It was demonstrated through simulation that using the new endgame region increased the likelihood of finding a solution when sampling the PRM. Also, under assumptions specific to this implementation, it was shown that conditions for belonging to the new endgame region are easily-calculated.

6.2.3 System Validation

1. Demonstrated, through simulation, on-the-fly motion planning through Dynamic Robot Networks. Average planning times on the order of 20 ms were achieved in scenarios involving up to 12 robots. Within these scenarios, 20 networks were merged per minute, demonstrating the platform's ability to handle frequent network merges/breaks.

2. Demonstrated, on hardware, on-the-fly motion planning of a group of mobile robots in an unknown, bounded workspace occupied by stationary and moving obstacles. This demonstrated planning on-line, assumptions on system modelling were valid, and practicality of system implementation.

6.3 Future Work

6.3.1 Task Planner Implementation

To increase the autonomy of Dynamic Robot Networks, a high-level task planner is required. The platform allows for centralized coordination across ad hoc networks, and is designed to handle more complex tasks than the motion planning examples presented earlier. The implementation of a task planner would allow robots to complete such tasks.

As an example, consider an autonomous construction scenario in which robots are required to survey a remote area, clear the area for construction, relocate parts to the site, and construct a structure. This would involve the completion of a large number of sub-tasks, (e.g. get part A and move it to location X,Y). It would be the responsibility of the task planner to assign these sub-tasks as individual robot goals. Ideally the sub-task ordering and robot assignments would minimize some cost function.

6.3.2 Large Object Manipulation

One ability that multi-robot systems have over single robot systems is they can manipulate larger objects through cooperation. This can prove beneficial in tasks like the remote construction of large structures.

Dynamic robot networks promise to be an excellent platform providing the necessary information for tight coordination between robots carrying an object. As discussed in Chapter 1, the platform has an advantage in that robots carrying an object can be represented as a single robot when coordinating with other robots in the system. This single robot representation will encode the size and dynamics of the object and robots together.

More importantly, the single robot representation should allow for high bandwidth communication between the robots that make up the single robot. A high data rate of control/estimatation signals is required to carry out most manipulation tasks, and the ad hoc communication link established between these robots should meet this requirement.

6.3.3 World Model Fusion

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 described in this dissertation, world models consist of a list of robots and their descriptions, and a list of obstacles and their descriptions. However, the ability to model the world for any general environment is not available.

Required for world model fusion is the combining of environment object state estimates acquired through relative sensing. A key issue to address is the "Correspondence Problem", the difficulty in resolving whether measurements from two sensors (e.g from two different robots) are of the same object. Because the sensing capabilities on the MARS test platform were accurate, the issue was not a problem for our implementation. However, field robots are not usually equipped with such accurate sensing/estimation systems.

One possibility is to implement a Multi-Robot SLAM algorithm (Simultaneous

Localization and Mapping) across the robot networks. Some successful work has been done in fusing state estimates from different vehicles [63], however it has not been implemented across ad hoc communication networks. Such a step would eliminate the need for a GPS style system, which is not always available, (e.g in planetary exploration).

6.3.4 Network Subdivisions

Within a robot network, every robot's sensing information is relayed to every other robot in the network. This is important when robots are relatively close to one another and tight coordination is required. However, if robots on opposite ends of a network are moving apart from one another, it is not clear if their actions need to be coordinated. This prompts the idea of splitting up networks into subdivisions, in which robots from different subdivisions are not explicitly coordinated.

Required would be a method of determining where divisions should be made. This appears to be a difficult problem with no obvious solution. Initial candidate solutions will most like include heuristics. However, further investigation is needed.

Appendix A

Randomized Motion Planning Theory

A.1 Milestone Expansion

Theorem A.1.1 *Given p_{ij} is the probability of no collision between any two robots at states x_i and x_j, the average number of random expansions necessary to achieve a collision-free set of state expansions for R robots using a parallel expansion method is:*

$$k_{avg,parallel} = \frac{1}{\prod_{i=1,j=i+1}^{R} p_{ij}(x_i, x_j)} \tag{A.1}$$

Proof In parallel expansion, the probability of randomly selecting collision-free state expansions for R robots is the product of the probabilities of each pair of robots having collision-free expansions:

$$P_{parallel} = p_{12}p_{13}p_{23}p_{14}p_{24}p_{34}...p_{(R-1)R} = \prod_{i=1,j=i+1}^{R} p_{ij}(x_i, x_j) \tag{A.2}$$

The expected number of random expansions necessary to achieve a collision-free set of state expansions is:

$$k_{avg,parallel} = \frac{1}{P_{parallel}} = \frac{1}{\prod_{i=1,j=i+1}^{R} p_{ij}(x_i, x_j)} \tag{A.3}$$

Theorem A.1.2 *Given q_{ij} is the probability of no collision between robot j's expansion from state x_j to x'_j and the previously constructed expansion of some robot i, the average number of random expansions necessary to achieve a collision-free set of state expansions for R robots using a parallel expansion method is:*

$$k_{avg,serial} = \sum_{j=2}^{R} \frac{1}{\prod_{i=1}^{j-1} q_{ij}(x_i, x_j, x'_i)} \tag{A.4}$$

Proof The probability that the state expansion of robot j will be collision-free with previous state expansions of robots 1 through $j-1$ is:

$$P_j = q_{1j}q_{2j}q_{3j}...q_{(j-1)j} = \prod_{i=1}^{j-1} q_{ij}(x_i, x_j, x'_i) \tag{A.5}$$

On average, the number of random expansions necessary to achieve a state expansion of robot j that is collision-free with state expansions of robots 1 through $j-1$ is:

$$k_{avg,j} = \frac{1}{P_j} = \frac{1}{\prod_{i=1}^{j-1} q_{ij}(x_i, x_j, x'_i)} \tag{A.6}$$

The total number of expansions necessary, on average, can be calculated by summing over j robots.

$$k_{avg,serial} = \sum_{j=2}^{R} k_{avg,j} = \sum_{j=2}^{R} \frac{1}{\prod_{i=1}^{j-1} q_{ij}(x_i, x_j, x'_i)} \tag{A.7}$$

Theorem A.1.3 *Given p_i is the probability that the i^{th} pair of state expansions is collision-free, the average number of collision checks necessary for a successful parallel expansion of R robot states is:*

$$C_{avg,parallel} = \left(\frac{1}{\prod_{i=1}^{C_{max}} p_i} - 1\right)\left(\frac{\sum_{l=1}^{C_{max}} lQ_l}{\sum_{l=1}^{C_{max}} Q_l}\right) + C_{max} \tag{A.8}$$

Where:

$$C_{max} = \frac{1}{2}R(R-1) \tag{A.9}$$

$$Q_{l,parallel} = \left(\prod_{i=1}^{l-1} p_i\right)(1-p_l) \tag{A.10}$$

Proof The average number of collision checks necessary to attain a successful expansion can be broken down into the number of collision checks for failed expansions C_F and the number for the completed expansion C_S.

$$C_{avg,parallel} = (k_{avg,parallel} - 1)C_{F,parallel} + C_S \tag{A.11}$$

For a successful series of collision checks, there must be $C_{max} = R(R-1)/2$ collision checks, one for each pair of robots.

$$C_S = C_{max} = \frac{1}{2}R(R-1) \tag{A.12}$$

The average number of collision checks for a failed series of collision-checks can be calculated by considering the probability of failure for each collision-check. The probability that the k^{th} collision-check between robot j and robot k is unsuccessful is a product of the probability of success between robots $1...j$ and robots $1...k-1$ and the probability of failure between robot j and robot k.

$$Q_{kj,parallel} = (p_{12}p_{13}...p_{1R})(p_{23}p_{24}...p_{2R})...(p_{j(j+1)}p_{j(j+2)}...p_{j(k-1)}(1-p_{jk})) \tag{A.13}$$

$$= \left(\prod_{i=2}^{R} p_{1i}\right)\left(\prod_{i=3}^{R} p_{2i}\right)\cdots\left(\prod_{i=j+1}^{k-1} p_{ji}\right)(1-p_{jk})$$

$$= \left(\prod_{n=1}^{j-1}\left(\prod_{i=n+1}^{R} p_{ni}\right)\right)\cdot\left(\prod_{i=j+1}^{k-1} p_{ji}\right)(1-p_{jk})$$

To simplify this expression, note that the maximum number of collision checks possible is $C_{max} = R(R-1)/2$, i.e. the number of possible collisions between R robots in an expansion. Given the collision check between the two robots is the l^{th} collision check of $i = 1...C_{max}$ maximum collision checks.

$$Q_{l,parallel} = \left(\prod_{i=1}^{l-1} p_i\right)(1-p_l) \tag{A.14}$$

Thus, for a failed expansion, the average number of collision-checks is:

$$C_{F,parallel} = \frac{Q_l + 2Q_2 + 3Q_3 + ...C_{max}Q_{C_{max}}}{Q_l + Q_2 + Q_3 + ...Q_{C_{max}}} \tag{A.15}$$

$$= \frac{\sum_{l=1}^{C_{max}} lQ_l}{\sum_{l=1}^{C_{max}} Q_l}$$

The total number of collision-checks, on average for a successful parallel expansion is:

$$C_{avg,parallel} = \left(\frac{1}{\prod_{i=1}^{C_{max}} p_i} - 1\right)\left(\frac{\sum_{l=1}^{C_{max}} lQ_l}{\sum_{l=1}^{C_{max}} Q_l}\right) + C_{max} \tag{A.16}$$

Theorem A.1.4 *Given q_i is the probability that the i^{th} pair of state expansions is collision-free, the average number of collision checks necessary for a successful serial expansion of R robot states is:*

$$C_{avg,serial} = \sum_{m-1}^{R-1} \left(\left(\frac{1}{\prod_{i=1}^{m} q_i} - 1 \right) \left(\frac{\sum_{l=1}^{m} l Q_l}{\sum_{l=1}^{m} Q_l} \right) + m \right) \tag{A.17}$$

Where:

$$Q_{l,serial} = \left(\prod_{i=1}^{l-1} q_i \right) (1 - q_l) \tag{A.18}$$

Proof The average number of collision checks necessary to attain a successful serial expansion can be obtained by noting that a serial expansion is simply a series of smaller parallel expansions, where the probabilities p_i must be replace by q_i.

$$C_{avg,serial} = \sum_{m-1}^{R-1} C_{avg,parallel}(C_{max} = m, p_i = q_i) \tag{A.19}$$

$$= \sum_{m-1}^{R-1} \left(\left(\frac{1}{\prod_{i=1}^{m} q_i} - 1 \right) \left(\frac{\sum_{l=1}^{m} l Q_l}{\sum_{l=1}^{m} Q_l} \right) + m \right)$$

Bibliography

[1] D. L. Akin and M. L. Bowden. Eva, robotic, and cooperative assembly of large space structures. *Proceedings of the IEEE Aerospace Conference*, March 2002.

[2] N.M. Amato and Y. Wu. A randomized roadmap metho for path and manimpulation planning. *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 113—120, 1996.

[3] T. Arai and J. Ota. Motion planning of multiple mobile robots. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1761—1768, 1992.

[4] R. C. Arkin. Cooperation without communication: Multiagent schema-based robot navigation. *Journal of Robotic Systems*, 9(3):351—364, 1992.

[5] K. Azarm and G. Schmidt. Conflict-free motion of multiple mobile robots based on decentralized motion planning and negotiation. *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3526–3533, 1997.

[6] J. Barraquand, B. Langlois, and J. C. Latombe. Numerical potential field techniques for robot path planning. *IEEE Transactions. On Systems, Man, and Cybernetics*, 22(2):224—241, 1992.

[7] J. Barraquand and J. C. Latombe. Nonholonomic multibody mobile robots: Controllability and motion planning in the presence of obstacles. *Algorithmica*, 10(2), 1993.

[8] J. S. Bellingham, M. Tillerson, M. Alighanbari, and J. P. How. Cooperative path planning for multiple uavs in dynamic and uncertain environments. *Proceedings of the IEEE Conference on Decision and Control*, Dec 2002.

[9] M. Bennewitz, W. Burgard, and S. Thrun. Optimizing schedules for prioritized path planning of multi-robot systems. *Proceedings of the IEEE International Conference on Robotics and Automation*, 2001.

[10] J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu, and J. Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. *Proceedings of the 4th ACM/IEEE Interanational Conference on Mobile Computing and Networking*, 1998.

[11] S.J. Buckley. Fast motion planning for multiple moving robots. *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1419—1424, 1989.

[12] Y. U. Cao, A.S. Fukunaga, A. B. Kahnb, and F. Meng. Cooperative mobile robotics: Antecedants and directions. *Autonomous Robots*, 4:1—23, 1997.

[13] S. Carpin and E. Pagello. On parallel rrts for multi-robot systems. *Proceedings of the 8th Conference of the Italian Association for Artificial Intelligence*, pages 834—841, Sept 2002.

[14] C. Clark and S. Rock. Randomized motion planning for groups of nonholonomic robots. *Proceedings of the International Symposium. of Artificial Intelligence, Robotics and Automation in Space*, 2001.

[15] C. Clark, S. Rock, and J. C. Latombe. Dynamic networks for motion planning in multi-robot space systems. *Proceedings of the International Symposium of Artificial Intelligence, Robotics and Automation in Space*, 2003.

[16] C. M. Clark and T. Barfoot. Motion planning for formations of mobile robots. *Journal of Robotics and Autonomous Systems*, 2004.

[17] J. Cortes, S. Martinez, T. Karatas, and F. Bullo. Coverage control for mobile sensing networks. *IEEE Transactions on Robotics and Automation*, 2002.

[18] R. DAndrea and G. E. Dullerud. Distributed control design for spatially interconnected systems. *IEEE Transactions on Automatic Control*, 48(9), Sep 2003.

[19] J. P. Desai, J. Ostrowski, and V. Kumar. Controlling formations of multiple mobile robots. *In Proceedings of the International Conference on Robotics and Automation*, pages 2864—2869, 1998.

[20] M. Erdmann and T. Lozano-Perez. On multiple moving objects. *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1419—1424, 1986.

[21] J. M. Fowler and R. D'Andrea. Distributed control of close formation flight. *Proceedings of the IEEE Conference on Decision and Control*, Dec 2002.

[22] K. Frengene, R. Madhavan, and L. E. Parker. Incremental multiagent robotic mapping of outdoor terrains. *Proceedings of the IEEE International Conference on Robotics and Automation*, 2002.

[23] A. S. Pereira G., V. Kumar, and M. F. M. Campos. Decentralized algorithms for multirobot manipulation via caging. *Fifth International Workshop on Algorithmic Foundations of Robotics (WAFR)*, pages 242—258, 2002.

[24] Y. Guo and L. E. Parker. A distributed and optimal motion planning approach for multiple mobile robots. *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2612—2619, 2002.

[25] D. Hsu, L. Kavraki, J. C. Latombe, R. Motwani, and S. Sorkin. On finding narrow passsages with probabilistic roadmap planners. *Robotics: The Algorithmic Perspective, A K Peters*, pages 141—153, 1998.

[26] D. Hsu, R. Kindel, J. C. Latombe, and S. Rock. Randomized kinodynamic motion planning with moving obstacles. *International Journal of Robotics Research*, 21(3):233—255, March 2002.

[27] 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.

[28] L. Hughes. Collected grounded reperesentations for robots. *Proceedings of Fifth International Conference on Distributed Autonomous Robotics Systems (DARS 2000)*, pages 79—88, 2000.

[29] D. Jung and A. Zelinsky. Grounded symbolic communication between heterogeneous cooperating robots. *Autonomous Robots*, 8(3):269—292, 2000.

[30] K. Kant and S. Zucker. Toward efficient trajectory planning: The path-velocity decomposition. *International Journal of Robotics Research*, 5(3):72—89, 1986.

[31] S. Kato, S. Nishiyama, and J. Takeno. Coordinating mobile robots by applying traffic rules. *Proceedings of the IEEE/RSH International Conference on Intelligent Robots and Systems*, pages 1535–1541, 1992.

[32] L.E. Kavraki. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *Ph.D. Thesis, Computer Science Dept.,Stanford University*, 1994.

[33] L.E. Kavraki, P. Svestka, J.C. Latombe, and M. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566—580, 1996.

[34] R. Kindel. *Motion Planning for Free-Flying Robots in Dynamic and Uncertain Environments*. PhD thesis, Stanford University, 2001.

[35] J. J. Kuffner. Autonomous agents for real-time animation. *PhD Thesis, Computer Science Dept., Stanford U.*, 1999.

[36] J.P. Laumond. *Robot Motion Planning and Control, previously published as Lecture Notes in Control and Information Sciences 229*. Springer, 1998.

[37] S.M. LaValle and S.A. Hutchinson. Optimal motion planning for multiple robots having independent goal. *IEEE Transactions on Robotics and Automation*, 14:912—925, 1998.

[38] S.M. LaValle and J. J. Kuffner. Randomized kinodynamic planning. *International Journal of Robotics Research*, 20(5):278—300, 1998.

[39] V. J. Lumelsky and K.R. Harinarayan. Decentralized motion planning for multiple mobile robots: The cocktail party model. *Autonomous Robots Journal*, 4:121—135, 1997.

[40] S. Martel and I. Hunter. Nanofactories based on a fleet of scientific instruments configured as miniature autonomous robots. *Journal of Micromechatronics*, 2003.

[41] M. J. Mataric. Using communication to reduce locality in distributed multi-agent learning. *Journal of Experimental and Theoretical Artificial Intelligence*, 10(3):357—369, 1998.

[42] P. Moutarlier and R. Chatila. Stochastic multisensory data fusion for mobile robot location and environment modelling. *Proceedings International Symposium on Robotics Research, Tokyo*, 1989.

[43] P. Ogren, E. Fiorelli, and N. E. Leonard. Cooperative control of mobile sensor networks: Adaptive gradient climbing in a distributed environment. *IEEE Transactions on Automatic Control*, 2003.

[44] L. E. Parker. Alliance: An architecture for fault tolerant multi-robot coordination. *IEEE Transactions on Robotics and Automation*, 14(2):220—240, Apr 1998.

[45] D. Parsons and J. Canny. A motion planner for multiple mobile robots. *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 8—13, 1992.

[46] J. Peng and S. Akella. Coordinating multiple robots with kinodynamic constraints along specified paths. In J. D. Boissonnat, J. Burdick, K. Goldberg, and

S. Hutchinson, editors, *Algorithmic Foundations of Robotics V (WAFR 2002)*, pages 221—237. Springer–Verlag, 2003.

[47] C. E. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers. *Computer Communications Review*, pages 234—244, Oct 1994.

[48] C. E. Perkins and E. Royer. Ad-hoc on-demand distance vector routing. *Proceedings of 2nd IEEE Workshop on Mobile Computing Systems and Applications*, pages 234—244, Feb 1999.

[49] J.A. Reeds and L.A. Shepp. Optimal paths for a car that goes forwards and backwards. *Pacific Journal of Mathematics*, 145(2):367—393, 1990.

[50] A. Richards and J. P. How. Aircraft trajectory planning with collision avoidance using mixed integer linear programming. *Proceedings of the American Control Conference*, May 2002.

[51] E. Royer and C. K. Toh. A review of current routing protocols for ad-hoc mobile wireless networks. *IEEE Personal Communications Magazine*, pages 46—55, Apr 1999.

[52] P. E. Rybski, S. A. Stoeter, M. Gini, D. F. Hougen, and N. P. Papanikolopoulos. Performance of a distributed robotic system using shared communications channels. *IEEE Transactions on Robotics and Automation*, pages 713—727, Oct 2002.

[53] R. O. Saber, W. B. Dunbar, and R. M. Murray. Cooperative control of multi-vehicle systems using cost graphs and optimization. *Proceedings of the 2003 American Controls Conference*, Jan 2003.

[54] G. Sanchez and J. C. Latombe. On delaying collision checking in prm planning : Application to multi-robot coordination. *International Journal of Robotics Research*, 21(1):5—26, Jan 2002.

[55] G. Sanchez-Ante and J. C. Latombe. Using a prm planner to compare centralized and decoupled planning for multi-robot systems. *Proceedings of the IEEE International Conference on Robotics and Automation*, 2002.

[56] P. S. Schenker, T. L. Huntsberger, P. Pirjanian, and E. T. Baumgartner. Planetary rover developments supporting mars exploration, sample return and future human-robotic colonization. *Proceedings of the 10th Conference on Advanced Robotics*, pages 31—47, 2001.

[57] D. H. Shim, H. J. Kim, and S. Sastry. Decentralized reflective model predictive control of multiple flying robots in dynamic environment. *Proceedings of the IEEE Conference on Decision and Control*, Dec 2003.

[58] 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.

[59] R. Simmons, T. Smith, M. B. Dias, D. Goldberg, D. Hershberger, A. Stentz, and R. Zlot. A layered architecture for coordination of mobile robots. *Multi-Robot Systems: From Swarms to Intelligent Automata*, 2002.

[60] M. W. Subbarao. Wireless communications technology group, nist. *White paper - Wireless Communications Technology Group, NIST*, Oct 1999.

[61] J. Svennebring and S. Koenig. Trail-laying robots for robust terrain coverage. *In Proceedings of the International Conference on Robotics and Automation*, 2003.

[62] P. Svestka and M.H. Overmars. Coordinated motion planning for multiple car-like robots using probabilistic roadmaps. *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1631–1636, 1995.

[63] S. Thrun. Multi-robot slam with sparse extended information filters. *Proceedings of the 11th International Symposium of Robotics Research*, Oct 2003.

[64] C.W. Warren. Multiple path coordination using artificial potential fields. *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 500–505, 1990.

[65] A. Winfield. Distributed sensing and data collection via broken ad hoc wireless connected networks of mobile robots. *Proceedings of 5th International Symposium on Distributed Autonomous Robotic Systems*, pages 273—282, 2000.