MULTI-ROBOT CARAVANNING

A Thesis

by

ADITYA MAHADEVAN

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Chair of Committee,	Nancy M. Amato
Committee Members,	Dezhen Song
	Suman Chakravorty
Head of Department,	Nancy M. Amato

December 2013

Major Subject: Computer Science

ABSTRACT

We study *multi-robot caravanning*, which is loosely defined as the problem of a heterogeneous team of robots visiting specific areas of an environment (waypoints) as a group. After formally defining this problem, we propose a novel solution that requires minimal communication and scales with the number of waypoints and robots. Our approach restricts explicit communication and coordination to occur only when robots reach waypoints, and relies on implicit coordination when moving between a given pair of waypoints. At the heart of our algorithm is the use of leader election to efficiently exploit the unique environmental knowledge available to each robot in order to plan paths for the group, which makes it general enough to work with robots that have heterogeneous representations of the environment.

We implement our approach both in simulation and on a physical platform, and characterize the performance of the approach under various scenarios. We demonstrate that our approach can successfully be used to combine the planning capabilities of different agents.

DEDICATION

To my family,

who are always close by

no matter how far away

ACKNOWLEDGEMENTS

This thesis is the culmination of a long but fulfilling journey that I could not completed alone. I am indebted to many people for all the advice and support that I received along the way.

Firstly, I would like to thank Dr. Nancy M. Amato, my thesis advisor, for her direction and encouragement. Without her supervision, I would not have come this far. Under her guidance, I have learned a great deal and become a stronger person.

I would also like to thank my committee members, Dr. Dezhen Song and Dr. Suman Chakravorty, as well as Dr. Dylan Shell, for their kindness and encouragement during this journey.

I have made many friends in the last three years but I am especially grateful for the company of Jory Denny, Andy Giese, Ali Agha, Adam Fidel, Olivier Rojo, Shishir Sharma, Tarun Jain and Shuvra Nath. They have each been mentors in their own way. They have taught me to tread firmly, to live freely and to take chances. Andy and Jory, in particular, toiled with me on this work. It is their work as much as it is mine.

Finally, I reserve my deepest gratitude for my family. Their love is more than I can express or repay.

TABLE OF CONTENTS

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION*

Multi-robot coordination, especially among heterogeneous robots, is becoming commonplace in robotics applications including swarming, flocking, task cooperation, and more. In scenarios such as collaborative surveillance [22], robot soccer [40], and search and rescue [34], heterogeneity presents an advantage because it allows robots with different capabilities to cooperate in manners that homogeneous robot groups cannot. However, communication, coordination [41], and robust task execution [7] among such groups present challenges such as determining what information needs to be combined and how to do so. In this thesis, we explore these benefits and challenges in the context of *multi-robot caravanning*, the problem of directing a team of robots to cooperatively visit a sequence of areas of interest (*waypoints*) in an environment and in a manner that ensures that the robots stay together at all times. An example of an environment with waypoints is shown in Figure 1.1.

The problem of multi-robot caravanning is inspired by the historical role of caravans – collections of travellers journeying together across potentially hostile territory – in human commerce and societal development. For humans, travelling in groups offers benefits such as the distribution of payload among individuals, the sharing of resources such as food and water, and more efficient management of work such as cooking or herding. In addition, it offers safety in numbers against adversarial threats, and allows individuals to better cope with harsh climates or rough terrain. We explore the benefits of caravanning in a robotics context, considering what a team of robots can gain by travelling together as a caravan to complete shared tasks.

*This chapter is reprinted with permission from "Multi-Robot Caravanning" by Jory Denny, Andrew Giese, Aditya Mahadevan, Arnaud Marfaing, Rachel Glockenmeier, Colton Revia, Samuel Rodriguez, and Nancy M. Amato. *Proc. IEEE. Conf. Intel. Rob. Syst. (IROS)*, November 2013.

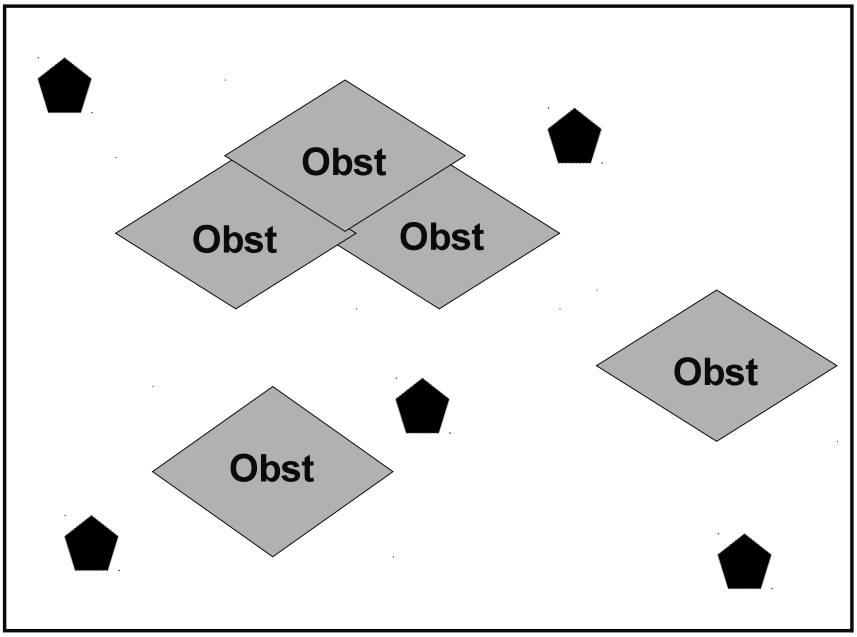


Figure 1.1: An example environment with areas of interest (waypoints) shown as pentagons, and labeled diamonds representing obstacles.

Caravanning arises in scenarios where robots must not only cooperate as a unit for the duration of a given task, but may also encounter multiple instances of the task in different portions of an environment and have to move from task to task together. The requirement to cooperate arises when no one robot has sufficient information or capabilities to complete a given task on its own, either due to heterogeneity in capabilities or information, or some limitation in individual capabilities that can only be overcome by using large numbers of robots. For instance, in a collaborative object transport task [19, 30], a group of robots must cooperate to move a large object from one location to another. In a search and rescue scenario [19], the "object" could be a disaster victim who needs to be transported between waypoints. The combined effort of multiple identical robots is required in order to complete the task, i.e.,

a single robot is insufficient. Another task requiring cooperation is collaborative target tracking [31], whereby multiple robots attempt to maintain observations of a moving target. The robots stay together in order to combine their observations in order to improve their estimates of the target's location and velocity. In the field of service robotics, multiple robots are used in a highway maintenance task in [18]. In this scenario, one robot which has global knowledge serves as a leader and guides other, simpler robots in a workspace. The roles of the leader and the followers are not interchangeable, and both types of robots are required in order to successfully complete the task.

After we formally define the problem, we propose a novel approach to the multi-robot caravanning problem that efficiently exploits the individual knowledge of the robots to benefit the group. The cornerstone of our approach is the use of leader election in conjunction with leader following. The former exploits the differing environmental information of the robots to decide which robot should become the leader, and the latter specifies how robots should follow a leader in order to move from one waypoint to the next. Our solution requires limited communication and sensing ability, and works in scenarios where robots have different representations of the environment.

The contributions of this work are as follows.

- A formal definition of and a scalable solution to the multi-robot caravanning problem that requires minimal explicit communication and sensing ability.
- A novel application of leader election to exploit heterogeneity in representation, which is applicable to robots whose representations are incomplete and/or generated in a distributed manner.
- An implementation of the proposed approach both in simulation and on phys-

ical robots and a characterization of its performance through experiments.

A version of this work [11] with preliminary results has been accepted to appear at the *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* in November 2013.

1.1 Thesis Organization

This thesis is organized as follows. Chapter 2 describes some work related to different aspects of our proposed method. In Chapter 3, we more formally define the caravanning problem and outline our proposed solution. In Chapter 4, we discuss details of its implementation. In Chapter 5, we describe the experimental analysis of our proposed method. Finally, Chapter 6 summarizes and offers conclusions on our work.

2. RELATED WORK*

In this chapter, we review prior work on robotics and motion planning related to the problem and solution that we propose in this work. We cover the problem of motion planning and the approaches used in solving it, with a particular focus on sampling-based methods. We summarize solutions to problems which involve planning for multiple agents or which impose additional requirements on the plans that are generated. We then discuss approaches to coordinate teams of robots through communication (e.g., leader election) and cooperative movement (e.g., leader following).

2.1 Motion Planning

In robotics, motion planning is the problem of finding a valid path in an environment for a robot to go from a start location to a goal location. More generally, motion planning is the problem of finding a valid sequence of transitions for an object between a valid start configuration and a valid goal configuration [23]. A "valid" configuration in this context refers to one that meets a chosen set of problem-specific constraints. A common constraint for robots is that of being collision-free, i.e., not colliding with any obstacles in the workspace. Solutions to this problem are many and varied. We briefly discuss the most relevant and important ones here.

2.1.1 Sampling-based Methods

A robot's *configuration* is the minimal set of parameters (degrees of freedom) necessary to uniqely define the location in the workspace of every point on the robot.

*Portions of this chapter are reproduced or adapted with permission from "Multi-Robot Caravanning" by Jory Denny, Andrew Giese, Aditya Mahadevan, Arnaud Marfaing, Rachel Glockenmeier, Colton Revia, Samuel Rodriguez, and Nancy M. Amato. *Proc. IEEE. Conf. Intel. Rob. Syst. (IROS)*, November 2013. © 2013 IEEE.

The configuration space or *C-space* [29] is the set of all possible combinations of parameters. The dimensionality d of the configuration space is the number of degrees of freedom (DOFs) of the robot. The *C-space* can be partitioned into the set of valid or feasible configurations, *C-free*, and the set of invalid or infeasible configurations, *C-obst*. The motion planning problem can now be described as the problem of finding a valid sequence of configurations in *C-free* between a start and goal configuration.

Sampling-based motion planners [28, 15, 24, 21] constitute an important subset of approaches to motion planning. They work by sampling and connecting configurations in *C-free*. They include both tree-based and graph-based methods.

The rapidly-exploring random tree (RRT) [24] is one such tree-based approach meant for single-query problems. It involves constructing a tree in the configuration space, rooted at the start configuration. During construction, a sample q_{rand} is generated randomly in the configuration space. The closest tree node q_{near} to q_{rand} is determined, and the tree is grown by advancing a node incrementally from q_{near} toward q_{rand}, up to a pre-specified distance. The resulting node, q_{new}, is connected to q_{near}, thereby growing the tree. This process is repeated until a leaf node is sufficiently close to the goal configuration.

Probabilistic roadmaps (PRMs) [21] are an example of a graph-based method for sampling-based planning. A probabilistic roadmap is a graph whose nodes are valid configurations, and whose edges constitute a valid transition between pairs of nodes. Nodes are generated by sampling points in the configuration space and retaining those that meet validity constraints. A local planner is used to attempt connections between pairs of retained points. The most common local planner is a straight line in *C-space* which is valid if every point along it is also valid, up to a resolution. The resulting graph is an approximation of the robot's configuration space. A particular motion planning problem is solved by attempting to connect the

start and goal configurations to the roadmap, and then querying the roadmap for a valid path between the start and goal configurations. RRTs provide solutions to single queries, whereas PRMs can be reused for multiple queries. In addition, for both PRMs and RRTs, many variations on the basic technique attempt to improve sample quality [2, 44, 27], address non-holonomic and dynamic constraints [25], optimality properties [20], uncertainty [1], etc.

2.1.2 Planning for Multiple Agents

The difficulty of planning for a single robot is magnified by the introduction of other robots into the same workspace [36], since each robot must account for the potential execution of the others' plans while creating its own, most commonly in order to prevent or minimize collisions and interference.

Multi-robot planning methods can largely fall on a spectrum between centralized and decoupled approaches. Probabilistic roadmap methods have been extended to define a collection of robots as a single configuration [38]. This centralized approach generalizes the configuration space to include all robots. Although powerful if agents share global information, this technique is not robust to failure because any change in the available set of robots causes a change in the dimensionality of the planning space, necessitating the complete reconstruction of the roadmap. It also suffers from the curse of dimensionality – as the number of robots increases, the planning space becomes infeasibly large.

Decoupled approaches [45, 5] attempt to address these issues by planning for each agent independently, and resolving potential collisions and deadlocks afterward using techniques such as randomization or decomposition. Such approaches tend to trade off optimality for efficiency [38]. However, in [17], an optimal decentralized algorithm is presented in which each agent plans independently, the paths are broadcast to all

agents, and paths are adjusted in a manner that minimizes a global performance objective. The issue of interference between robots arising from decoupled or decentralized planning is explored in [39], where the authors attempt to determine the effect of a parameter that controls the level of decomposition on the efficiency of a distributed foraging task.

In [35], a decentralized motion control system is developed for robots subject to formation constraints, based on reactive controllers that are selected based on the relative position between a robot and its neighbors. This system serves to address the problem of planning for multiple robots while addressing constraints in addition to avoiding collisions.

2.1.3 Coverage and Inspection Planning

The domain of coverage planning, synonymous with inspection planning, provides a motivation to the problem we consider in this thesis, that of planning for robots to visit specific locations in the environment. Coverage planning is the problem of planning for one or more robots to pass over or observe every point in its free space, and has applications such as lawn-mowing, harvesting and oceanographic mapping [10, 9]. In [13], the problem of multi-robot boundary coverage is introduced. In this problem, a group of robots must inspect all points on the boundary of an environment in two dimensions. The proposed solution generates inspection routes that provide full coverage while balancing the inspection load, i.e., the total distance travelled. In [43], the same authors propose a path revision algorithm that handles changes to the environment and/or the team size.

2.2 Coordination

In this section we discuss some prior work on coordination among multiple robots, particularly approaches in which one or more of the robots serve as leaders. We

discuss the process of designating a leader, known as leader election, as well as approaches that involve moving relative to a leader, which include flocking and leader following.

2.2.1 Leader Election

Leader election is the task of selecting a coordinator from a group of candidates. As detailed in subsequent chapters, leader election and leader following in conjunction form the cornerstone of our proposed method to solve the caravanning problem.

Although many algorithms have been proposed to perform election, in this thesis we use a variation of the *Bully Algorithm* [14] for its simplicity. The original algorithm works as follows: at a call for an election, each member of the group broadcasts their processor ID (PID). The one with the highest PID notifies the group that it will be the leader. In this work, we replace the PID with a more meaningful metric based on the available paths between a given pair of waypoints.

Leader election is an important precursor to the solutions of many problem in distributed robotics. In [6], leader election is considered in the context of stable self-organization of a team of robots. However, in that work the robot with the smallest angle to two other robots becomes the leader, whereas in our approach the leader is the one with the best path between two areas of interest.

2.2.2 Flocking

The multi-robot planning approaches in 2.1.2 largely deal with global planning for robots with independent goals. Flocking [37] is an instance of multi-agent motion planning in which potentially large groups of agents with *shared* goals react to *local* forces exerted by their neighbors, resulting in the emergence of cohesive, coordinated movement in a shared direction. Incidentally, this method has been successfully combined with the global solution offered by roadmaps [4].

2.2.3 Leader Following

Leader following [7, 8] is another approach to planning for robots with shared goals, in which a leader with a valid plan leads a group of agents, and one or more followers attempt to follow the leader agent. This technique has applications in formation control [12], multi-robot planning [3], and cooperative task execution [18]. Generally, the follower aims to stay at a given distance from the leader, adjusting its relative angle to keep the leader within its field of view. Although very reliable for a small number of robots, the method usually does not scale as the number of robots increases, particularly in the presence of obstacles: as the number of agents increases, so does the number of possible flock formations and the difficulty of avoiding collisions among all the agents and with obstacles. In this thesis, we employ a variation of this technique in which a follower, rather than attempting to maintain its relative distance and bearing to the leader's current position, instead attempts to retrace the leader's steps.

The current state of the art in cooperative movement assumes that all robots have global knowledge or that robots with global knowledge are designated as leaders beforehand. For instance, in [18], the leader is assumed to have global knowledge and precision positioning, e.g., using GPS, while followers use only simple sensors and perform simple computation to follow the leader. In this approach, it is impossible to recover from a failure of the leader.

In contrast, our approach requires none of the robots to have global knowledge, and any robot could be elected the leader (provided it has sufficient environmental information to find a path between a given pair of waypoints). Thus, our approach advances the state of the art by generalizing to scenarios in which *all* robots have incomplete or overlapping information. Moreover, while prior approaches typically

address the problem of *how* to follow the leader effectively, we focus on *who* should become the leader based on the robots' representations.

Moreover, the authors of [7] address the problem of unexpected transient obstacles that might temporarily break the chain of robots, while we are more concerned with avoiding persistent static obstacles. In [18], robots are divided into two classes – one class of simple robots with sufficient sensors to localize themselves with respect to the leader but not much more, and another class of robots with sophisticated global information who serve as leaders. In contrast, in our approach any robot could be elected leader in a particular portion of the problem, based on the knowledge it has for that portion (even though its global knowledge may be incomplete).

3. MULTI-ROBOT CARAVANNING*

In this chapter, we formally define the *multi-robot caravanning* problem, outline and illustrate the proposed approach, and comment on its limitations.

3.1 Problem Definition

In the following section, we will formally define the *caravanning problem*. Prior to that, however, we introduce and describe the concept of *representation heterogeneity*, the basis of the problem we are trying to address. We demonstrate that representation heterogeneity can arise even in robots that are otherwise identical.

3.1.1 Heterogeneity in representation

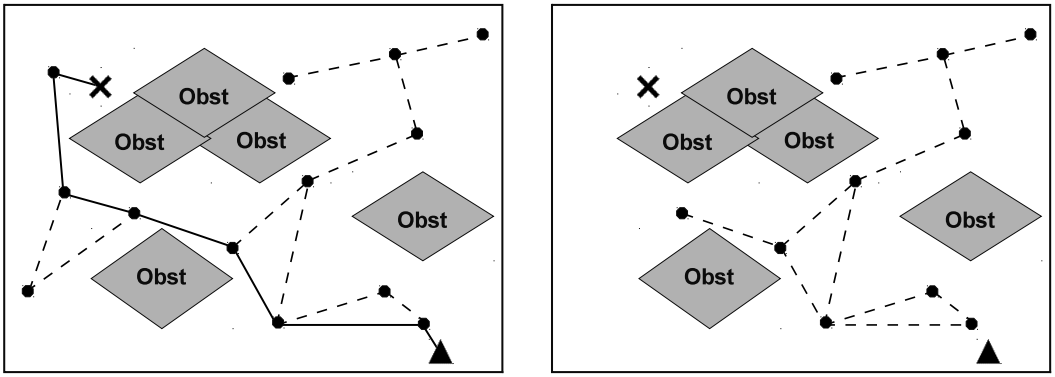
A robot's *representation* of the environment is an *approximation* of its configuration space that incorporates the observations available to it and determines the actions it can take. We say two robots are *representation heterogeneous* if their representations constitute different approximations of the environment.

An important consequence of random sampling in the construction of Probabilistic Roadmap (PRM) representations is that two roadmaps representing the same environment are very unlikely to be the same. Not only are the graphs themselves likely to be topologically distinct, but they may also return homotopically different paths as solutions to a given query. Indeed, one of the two roadmaps may even fail to return an answer to the query, in which case we consider this roadmap to be an *incomplete* representation. The use of different sampling methods, local planners, and construction strategies [32, 42] introduces even more variability. The hetero-

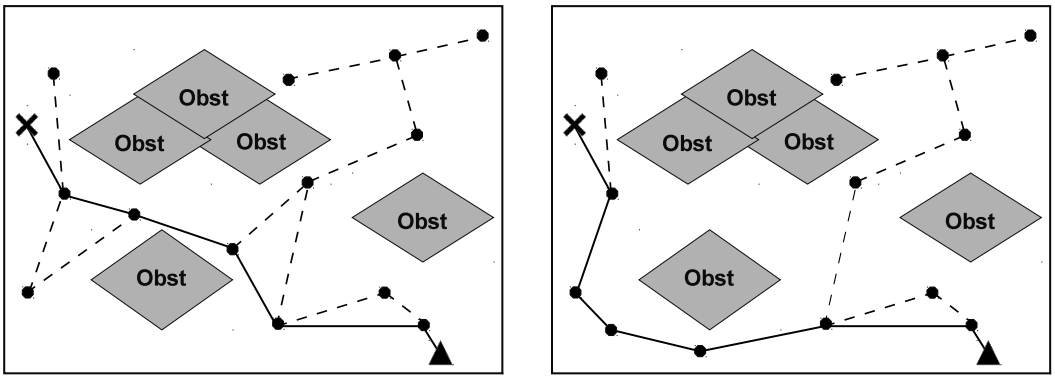
*Parts of this chapter are reproduced with permission from "Multi-Robot Caravanning" by Jory Denny, Andrew Giese, Aditya Mahadevan, Arnaud Marfaing, Rachel Glockenmeier, Colton Revia, Samuel Rodriguez, and Nancy M. Amato. *Proc. IEEE. Conf. Intel. Rob. Syst. (IROS)*, November 2013. © 2013 IEEE.

geneity that naturally arises in constructing PRMs makes them an excellent focus for considering representational heterogeneity in this work.

Figure 3.1(a) and Figure 3.1(b) illustrate incompleteness and heterogeneity in roadmaps respectively.



(a) Roadmap completeness. The incomplete roadmap on the right is unable to find a path between the given start and goal (triangle and cross, respectively). The roadmap on the left is complete because there is a path for every given pair of points in the environment.



(b) Roadmap heterogeneity. The roadmap on the left and the roadmap on the right return different paths to the same query.

Figure 3.1: Roadmap completeness and heterogeneity.

3.1.2 Problem Definition

In this section, we define the multi-robot caravanning problem, as well as related concepts such as the environmental representation and path.

Definition 1 *A* **path** *is a sequence of valid configurations connecting a given start and goal configuration.*

Definition 2 *A* **representation** *is a data structure, or collection of data structures, that an individual robot can query to obtain a path from its start position to a goal position such that if the robot follows the path it will be guaranteed to arrive at the goal within a finite expected time.*

The representation is assumed to include a source of observations that the robot can use to verify that it has arrived at its goal and has not collided with an obstacle.

Definition 3 *A representation is* **incomplete** *if there exists a start and goal in the environment for which it is unable to return a valid path for the robot to transition from the start to the goal.*

Definition 4 *Two robots are* **representation heterogeneous** *if there exists a goal for which their representations return different paths from one another.*

Note that by this definition, two robots are also representation heterogeneous if one representation returns a path but the other fails.

Definition 5 *A* **waypoint** *is a coordinate in the robot's configuration space.*

A group of robots may share a set of waypoints that represent, for instance, task locations or locations that must be inspected. We assume each waypoint is reachable,

i.e., there exists a path to it from every portion of the environment. However, the robot's environmental representation may be incomplete for some or all of the waypoints.

Definition 6 *A* **caravan** *is a group of robots that operate while meeting a visibility or cohesion constraint that applies to the group.*

The constraints may require, for example, that all robots in the group stay within a predefined distance of one another or to the group's centroid. In our implementation, each robot must be able to see at least one other robot and the graph of visibility between robots must not be disjoint. Robots that have failed are not considered part of the group.

We are now ready to define the multi-robot caravanning problem:

Definition 7 *Given a group of n representation heterogeneous robots* $r = \langle r_1, r_2, \ldots, r_n \rangle$, *and a set of waypoints* $W = \langle w_1, w_2, \ldots, w_m \rangle$, *the* **multi-robot caravanning (MRC) problem** *is to generate a valid path for each* r_i *to visit all the waypoints in* W *such that the robots visit each waypoint as a caravan.*

Informally, the MRC problem is the problem of planning for a group of agents to visit a sequence of locations in the environment (waypoints) as a group.

3.2 Approach

We propose a novel solution to the MRC problem. Our solution divides the MRC problem into stages. At each stage, a leader is elected and a leader following approach is used to move robots between one waypoint and the next. The novelty of our approach lies in the application of leader election to decide which robot should become the leader. For every pair of waypoints in the sequence, the robot with the

"best" path according to some metric (for instance, lowest path length or highest path clearance) becomes the leader. The other robots follow the leader until the next waypoint, where the process is repeated.

Prior approaches that perform leader following tend to differentiate between leaders and followers offline, based on heterogeneity in capabilities [7]. For instance, followers have just enough sensing and communication ability to localize themselves with respect to the leader in order to follow it, while the leaders have sophisticated global knowledge [18]. Moreover, prior approaches that perform leader election either elect a random robot as the leader, or rely on robots' IDs (e.g. selecting the robot with the lowest or highest ID) or relative positions [6].

In contrast, we perform leader election both dynamically and in a problem-specific manner. Doing so has several benefits:

- The use of a path metric in performing leader election allows us to handle scenarios in which robots have different, even incomplete, representations of the environment. This scenario arises frequently in problems that involve generating or storing the representation in a distributed manner, such as in a distributed simultaneous localization and mapping (SLAM) scenario, where agents representations may be heterogeneous and/or incomplete.

- Only limited communication is required since robots only communicate at waypoints and never communicate their representations or even their paths to one another, but only the path metric.

- Our solution can exploit overlap between representations. If a robot with the best path has already failed or been lost, the one with the next best path will be elected.

Algorithm 1 Agent Algorithm Overview

Input: Waypoints $W = \langle w_1, w_2, \ldots, w_m \rangle$, Roadmap R

```
1: for all w_k ∈ W do
2:    p = R.FindPath(w_k, w_{k+1})
3:    result = ElectLeader(p)
4:    if result == "leader" then
5:       SwitchLeader()
6:       Traverse p while localizing
7:       Call for leader election
8:    else
9:       repeat
10:         FollowLeader()
11:      until Leader election call
12:   end if
13: end for
```

The overall algorithm is shown in Algorithm 1 (failure conditions omitted). Each stage can be explained in terms of three steps. In the Leader Election step, one robot that knows a path between the current and next waypoint is chosen as the leader and will be responsible for traversing its path. In the Leader Switching step, the newly elected leader travels to a designated position near the current waypoint, from which it will begin to traverse its path. In the Leader Following step, all other robots follow the leader by maintaining a constant position and orientation relative to it. We now explain these steps in detail.

3.2.1 Leader Election

In the first step, one robot is selected (the leader) which will be assigned the responsibility for executing a plan from waypoint w_k to waypoint w_{k+1}. This is achieved using a slightly modified version (Algorithm 2) of the Bully algorithm [14] for leader election. First, each robot queries its environmental representation for a path between waypoints w_k and w_{k+1}. It then broadcasts a *path metric* based on

the result of its query to the other robots, together with its ID. If no robot finds a valid path (i.e., all robots broadcast an invalid metric), the algorithm terminates and returns failure. If exactly one robot finds a path, it is elected the leader by default. If two or more robots find a path, the one with the better path metric is elected leader; in the case of a tie, the robot with the lower ID is chosen. The path metric is any scalar value that summarizes the quality of the candidate path. In this work, we choose to use path length as the metric; the shortest path is the most desirable. Other possible metrics include path clearance, path smoothness, etc.

Algorithm 2 ElectLeader

```
1: Broadcast ID and path metric
2: Receive M as a map of IDs to path metrics
3: bestID = arg max_{id∈M} M[id]
4: if bestID ≡ myID then
5:     Broadcast end of leader election
6:     return leader
7: else
8:     return follower
9: end if
```

3.2.2 Leader Switching

If an agent decides that it has been selected to be the leader, it will need to move to a designated leader position. In our implementation, it creates a Rapidly-exploring Random Tree (RRT) [25] from its current position to a position along the path between the current waypoint w_k and the next. Next, the robot traverses the path from the RRT and turns to face the waypoint. As explained in Section 3.2.3, this step is necessary to update the formation that the robots assume in the leader following step.

Algorithm 3 SwitchLeader

Input: Waypoint w

```
1: repeat
2:    g = CreateGoal(w)
3:    s = GetCurrentPosition()
4:    P = GetRRTPath(s, g)
5:    for all p ∈ P do
6:       V = GetNewlyVisibleRobots()
7:       for all v ∈ V do
8:          O = O ∪ AddTempObstacle(v)
9:       end for
10:      if IsInCollision(p, O) then
11:         break
12:      else
13:         MoveToPoint(p)
14:      end if
15:   end for
16: until AtGoal(g)
17: RemoveTempObstacles(O)
```

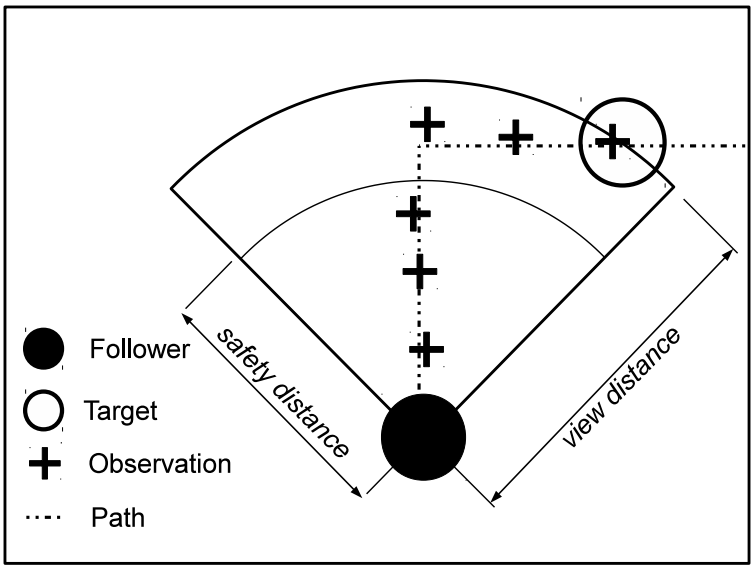
The leader switching process is outlined in Algorithm 3. All other robots stay stationary until the leader has successfully reached its designated position or notified them of failure. The leader initially creates the RRT plan without taking into account any of the other robots' positions. However, as each robot is seen for the first time, the leader updates its list of obstacles to include the new agent. The leader evaluates each next point along its RRT path before it moves along it; if any point is in collision because of a change to the list of obstacles, a new RRT is created from the leader's current position. At the end of the leader switch, the obstacles representing robots are removed from the list. If the leader is unable to find a path, it notifies the other robots that it has failed and the algorithm terminates with failure.

3.2.3 Leader Following

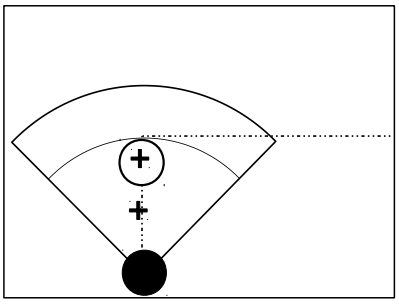
In the leader following step, the leader executes the plan by following its queried path and all other robots move relative to the leader, and attempt to maintain visibility to it or to one another.

At the start of this step, we maintain the invariant that the *visibility graph* of active robots (i.e., all robots that have not failed) is connected and at least one robot is at or near the current waypoint. The nodes of the visibility graph are robots. There is an edge between every pair of robots that can observe one another.

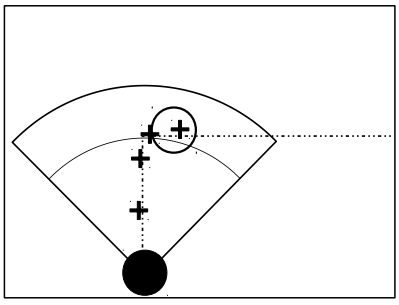
A number of different flocking or formation techniques could be employed at this stage. We employ a simple leader-following approach, in which robots form a chain that is headed by the leader. Each robot tracks the one in front of it (its target), attempting to visit each of its target's observed positions in sequence. This approach has several advantages over other flocking techniques. Firstly, each robot is guaranteed to follow a path along the leader's roadmap, which is known to be valid. This also means robots need not employ any kind of obstacle sensor or rangefinder. Moreover, this technique is scalable to a large number of robots since each robot's movements depend only on its observations of the robot in front of it (and are therefore independent of the number of robots in the chain). This step ends when the next waypoint is reached.



(a) Follower and target. *Safety distance* and *view distance* are shown.

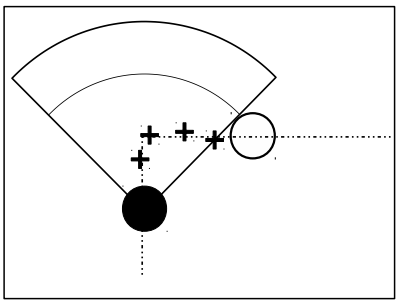


(b) While target is closer than *safety distance*, follower does not move, but adds observations to the queue.

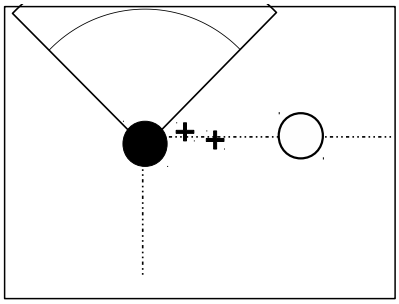


(c) Target eventually becomes further than *safety distance*. Follower approaches first point in queue and continues adding observations.

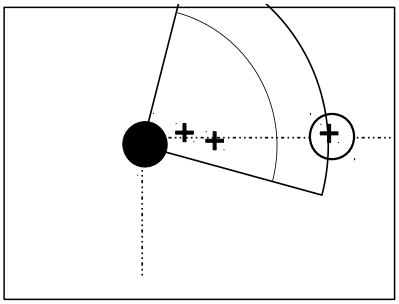
Figure 3.2: Target tracking parameters and process.



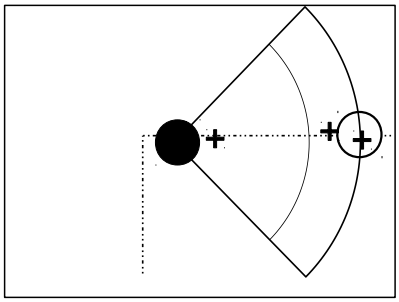
(d) Target may be unobserved temporarily because of follower's limited *view angle*.



(e) Follower reaches bend in (observed) path. Target is still unobserved.



(f) Follower turns to face next point in queue and observes target once more.



(g) Follower continues following target until next waypoint (not shown) is reached.

The process by which each robot tracks the one in front of it is illustrated in Figure 3.2. Tracking is achieved by maintaining a queue of observations of the target. Whenever a new observation is made, it is added to the back of the queue. The robot always moves to the observed point at the front of the queue, removing it from the queue when it reaches to within a given radius of the point, known as the *reach distance*. In order to prevent collisions with the target, a robot will execute a waiting behavior until the target has moved to be greater than a certain distance, the *safety distance*, away.

Observations and robot motions are corrupted by noise. Hence, successive observations of even a stationary target are likely to be distinct. Moreover, a robot can

never expect to visit an observed position with perfect accuracy, and may have to turn back if it overshoots the next observed position. In order to prevent the follower from turning back frequently and reduce the chances of the target moving away unobserved, the queue is smoothed by discarding observations that are too close to the immediately preceding one or which would cause the robot to have to turn in the opposite direction.

3.3 Limitations on Heterogeneity

The approach described in this thesis is applicable if a number of assumptions are satisfied, which impose limitations on the heterogeneity of the robots:

- **Shared coordinate system.** The fact that all robots share the set of waypoints necessitates that they must also have a common coordinate system. This, in turn, requires that robots share some knowledge of the environment, even if their representations are different.

- **Path validity for all robots.** This approach assumes that a path that is valid for the leader is also valid for all the other robots. This assumption may not be satisfied if robots are greatly different in size or shape, such as when the leader is much smaller than the follower(s). In such a scenario, the leader may find a path through a narrow portion of the environment that it can follow, but which is too narrow for followers to fit through. This can be overcome to some extent if each robot can build a representation that only returns paths that are valid for all robots. One way to achieve this for probabilistic roadmaps is to generate a roadmap for a robot geometry generated from the Minkowski sum [26] all the robots.

- **Ability to sense relative positions of target.** In order to follow a target's path, a follower must have sufficient sensing ability to allow it to observe the target's positions over time, relative to itself. In other words, a follower must be able to localize itself relative to its target.

4. IMPLEMENTATION*

In this section, we describe an implementation of our proposed algorithm both on physical robots and in simulation. We first describe the robot platform and hardware. We then describe our methods for localization, state estimation and communication, followed by a discussion of how the same framework used for physical robots is adapted to simulation.

4.1 Robot Hardware

The robot platform we use is Asus EEE PC netbook equipped with an on-board webcam and wireless networking capability, mounted on an iRobot Create (Figure 4.1) and controlling it through the Player robot interface [16].

Figure 4.1: iRobot Create with mounted Eee PC netbook for webcam use. Markers placed around the robot are used by neighboring robots to determine relative position and orientation. © 2013 IEEE.

*Parts of this chapter are reprinted with permission from "Multi-Robot Caravanning" by Jory Denny, Andrew Giese, Aditya Mahadevan, Arnaud Marfaing, Rachel Glockenmeier, Colton Revia, Samuel Rodriguez, and Nancy M. Amato. *Proc. IEEE. Conf. Intel. Rob. Syst. (IROS)*, November 2013. © 2013 IEEE.

The camera has a maximum resolution of 640x480 pixels. The Creates are two-wheel differential drive unicycle robots with a maximum speed of about $0.5m/s$ and a minimum speed of about $0.1m/s$, below which their motion is highly unreliable. Internal robot odometry information is highly inaccurate, especially when rotating, and at particular speeds. To compensate for this, we require accurate observations of environmental features.

4.2 Localization and Robot Detection

We rely on frequent localization using visual markers. For robust marker creation and detection, we utilize the ArUco marker detection library from the University of Córdoba [33].

Markers are placed along the walls in the environmnet at roughly regular intervals. Each marker has a known unique ID, and known absolute position and orientation in the environment.

Robots localize themselves by calculating their relative pose to the markers and transforming it into global coordinates based on the markers' known positions and orientations. Markers are also used by robots to detect other robots' poses. For this purpose, each Create's perimeter is covered with markers whose relative positions to its centroid are known.

The size of the markers determines the minimum and maximum range, as well as the maximum angle, from which they can be fully captured by the camera. The wall markers used for localization are larger, hence they are visible from further away (up to $2m$ and 30 degrees from the normal); however, when they are too close, they are not fully captured by a webcam image. The smaller markers on the robots are visible from a closer minimum and maximum distance (up to $1.5m$) but also a wider angle (45 degrees) since their centroids are visible from a larger angle before they

become occluded.

The movement of each robot's camera relative to both wall markers and markers on other robots introduces camera blur, leading to intermittent failures in observations. To mitigate this, every movement of the robot is accompanied by a brief period in which the robot is stopped, but still making observations. Typically, each robot will stop for $0.1s$ after every $0.15m$ of movement. This temporarily minimizes camera blur, allowing the robot to make observations of both wall markers and other stopped robots.

4.3 State Estimation

An Extended Kalman Filter (EKF) is used to estimate the robot's state, accounting for uncertainty in both movement and sensor observations. Motion uncertainty is caused by uneven tiles on the floor of the environment, slippage of the wheels, and variations in the length of time controls are applied. Observation uncertainty results from variations in the intrinsic parameters of the netbook cameras, latency between the movement of the robot and the detection of the next image, and intermittent failures in observation due to motion blur of the camera.

The intermittent failures in observation described in 4.2 lead to an increase in the covariance of the state estimate maintained by the EKF. To correct for this, when the covariance is sufficiently high, the leader stops, rotates until it can detect a marker a wall marker, and updates its state estimate using the EKF until the error covariance is acceptably low before continuing towards its previous goal.

4.4 Communication

A number of steps in our proposed algorithm involve communication, such as the broadcast of path metrics during leader election. Our algorithm involves a distributed communication model, i.e., each robot can communicate with any other

robot independently. However, at present the messages are *routed* through a central server to which any robot can connect or disconnect.

4.5 Software-in-the-loop Simulation

In addition to implementation on a physical robot, we also run the algorithm in a simulated environment for quicker debugging, parameter tuning and experiments.

In an effort to make the simulation as accurate an approximation of reality as possible, we follow a software-in-the-loop simulation approach, implementing a thin simulation layer that replaces the real robot hardware and environment. The controls normally sent to the iRobot Create as well as the observations processed from the camera image are instead diverted to/from the simulation layer. In order to process controls and generate observations in simulation, it is necessary to keep track of the true state of each robot, while isolating it from direct access by the algorithm. This is done by the simulation layer. Whenever a robot applies a control, its true state is updated in the simulation layer to reflect the result (after artificially adding noise to the control). Likewise, observations received by the robot are generated from the true state maintained by the simulation layer, corrupted with artificial noise.

The true state is cached by all the computational processes used to represent robots. A dedicated server (different from the one used for the algorithm) is used to broadcast updates to the true state across processes to maintain consistency. The rates at which the true state is updated and observations are made from it can be controlled somewhat, making it possible to simulate the delays in observation seen in the real-world system.

These design choices have resulted in a high-fidelity simulation of real-world conditions, making it possible to tune the algorithm in simulation with confidence that the modifications are applicable to the physical robots.

5. EXPERIMENTAL EVALUATION*

In this section, we test the feasibility of our proposed approach both on physical robots and in simulation. Through our evaluation, we attempt to determine how the parameters of the robot tracking method and the quality of the robot's roadmap affect the rates of success.

We first describe our experimental setup, and then the parameters tested and experiments that are run, before showing and discussing our results for each experiment.

5.1 Experimental Setup

5.1.1 Environments

The environment used for testing the physical robots is the fourth floor of the H.R. Bright Building of Texas A&M's campus in College Station, TX. A floorplan can be seen in Figure 5.1. The floor spans 40m of hallways 2m wide on average.

In simulation, two environments are used. The first is a reproduction of the fourth floor environment. It is used to characterize the effect of different parameters on the performance of the proposed approach. The second is an office-like environment that is larger and more open. It is used to characterize the scalability of the approach to increasing numbers of robots.

5.1.2 Experiments

In all experiments, robots are required to visit a set of waypoints in a prescribed order (all robots start in a chain formation, facing the first waypoint). In order

*Parts of this chapter are reprinted with permission from "Multi-Robot Caravanning" by Jory Denny, Andrew Giese, Aditya Mahadevan, Arnaud Marfaing, Rachel Glockenmeier, Colton Revia, Samuel Rodriguez, and Nancy M. Amato. *Proc. IEEE. Conf. Intel. Rob. Syst. (IROS)*, November 2013. © 2013 IEEE.

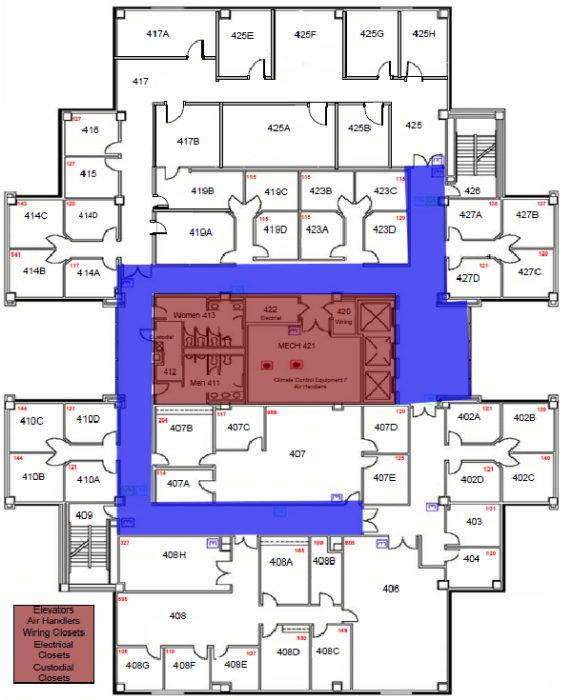


Figure 5.1: H.R. Bright Building Fourth Floor Floorplan with hallways highlighted blue. © 2013 IEEE.

to visit a waypoint, a robot must be able to query its roadmap for a path to the waypoint from the previous waypoint. The experiments test the ability of the robots to cooperate using the caravanning approach to visit every waypoint. A robot is successful if it can visit all the waypoints and does not collide with obstacles. The waypoints are known beforehand and are common to all the robots, but do not influence the construction of each robot's roadmap.

The proposed method does not guarantee that all robots will reach the final waypoint but is tolerant to some types of failure. The likelihood of failure depends on a number of factors related to the sensing and planning capabilities of the robots.

Success rate metric The success rate is a measure of the likelihood that a chain of robots given a set of waypoints manages to sucessfully visit the waypoints. There are numerous possibilities for failure during the course of the experiment:

- A given Leader Following robot may lose track of its target.
- A robot may collide with other robots or obstacles in the environment during the Leader Following stage due to deviations caused by uncertainty. Both the leader and the followers are vulnerable to this.
- A newly elected leader may collide with other robots during the Leader Switching step.
- A newly elected leader may fail to find a path to the head of the chain.
- All robots may fail to find a valid path during the Leader Election.
- Robots may stall indefinitely because their targets have failed. This may be in part due to the prior failure of a robot that knows a valid path.

It is apparent that these failures are not independent of one another. During the Leader Following step, the occurrence of one failure of a particular robot in the chain may have consequences for robots behind it. During the Leader Election step, the prior failure of a robot with a valid path may have consequences for the whole group.

To account for these nuances, we define success rate as the *average number of waypoints visited* by robots during a run (apart from the first, which they start from). Under these terms, in a completely sucessful experiment, the number of waypoints visited is $n \times m$, where n is the number of robots and m is the number of waypoints. This measure gives us an indication of how early and how often failures occur.

Parameters We focus on the sensing capability. In simulation, we vary parameters affecting the Leader Following behavior – namely, the maximum view radius, view angle and the safety distance, and their effect on the success rate.

- *View distance.* The *view distance* directly affects the observations available to the robots; indirectly, it affects the paths taken by the leader and follower, as well as the frequency of localization of the leader.

- *Safety distance.* The *safety distance* in the robot tracking behavior used in the Follow Leader step determines how far away a robot must allow its target to get before it advances to the next point on its queue of tracked paths. Indirectly, this also affects the observations that will be made of each target. If the *safety distance* is smaller, the target is allowed to be closer, increasing the range of observations, but the chances of colliding with the target are expected to be higher.

- *Reach distance.* The *reach distance* specifies how close a follower must get to its next observed target point before it is considered to have reached that point. It is essentially a smoothing parameter that determines how much the follower is allowed to deviate from the leader's path. Some deviation around sharp turns in the tracked path allows the target to more quickly face the leader's latest position. However, too much deviation from the roadmap path is expected to cause the follower to collide, particularly around corners.

We run two different experiments to explore the effect of these parameters. In the first, we vary the *view distance* and *safety distance* to explore their effect on the success rate. This experiment is done for two different roadmaps produced using different sampling methods in order to demonstrate the effect of differences in the

underlying representation. In the second, we explore the effect of *reach distance* on the success rate.

We also perform an experiment to analyse the scalability of the approach to larger numbers of robots as well as the effect of the ordering of robots on the ability to exploit redundancy to compensate for failures of individual robots.

We also perform a qualitative evaluation on physical robots. In particular, we are interested in how the approach scales as the number of physical robots increases. We are interested in determining whether there is a number past which the method consistently fails and if so, whether there is a consistent cause of failure.

5.2 Results

5.2.1 Effect of View Distance and Safety Distance on Success Rate

In this experiment, we explore the effect of varying both the *view distance* and the *safety distance* on the success rate for an environment with 6 waypoints and 10 robots. Due to uncertainty and asynchronicity in the movement of the robots, both on the physical platform and in simulation, no two runs are the same. We therefore take the average of 6 runs for each value of *view distance* and *safety distance* used. The roadmap used by each robot directly affects the path taken by the group of robots between waypoints, and thereby indirectly affects the success rate. In order to explore the effect of the roadmap, we run the experiment for different roadmaps. We ensure no robot has a complete roadmap so that robots are forced to switch at some or all waypoints.

As explained in Section 4.2, the *view distance* is directly related to the size of the markers on the robots. In the physical environment, the *view distance* is typically $1.5m$ based on our chosen robot marker size, but could be adjusted by increasing or decreasing the marker size slightly. To explore the effect of doing so, we use *view*

distance values of $1.0m$, $1.5m$ and $2.0m$.

For *safety distance*, the range of the values is bounded from below by the radii of the robots (i.e., a *safety distance* less than the sum of the radii of 2 robots is guaranteed to cause collision), so we pick a starting value of $0.4m$. The values are bounded from above by the maximum *view distance*, since a *safety distance* greater than or equal to the *view distance* causes a follower to stop whenever the target is visible, and the follower only attempts to move to that point after the target itself is out of range. We pick $2.15m$ as the upper bound on the *safety distance* value, with intermediate values at increments of $0.25m$.

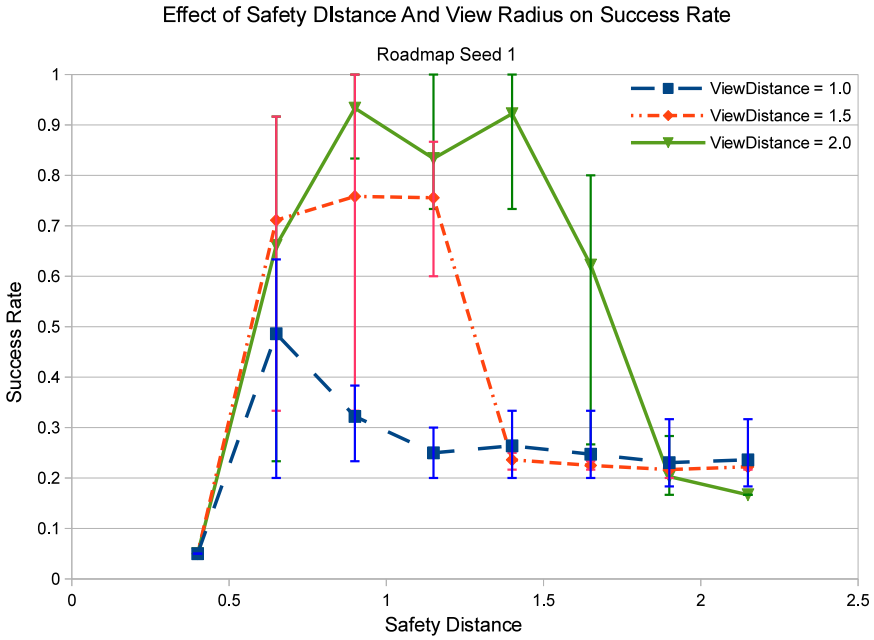
The results are shown in Figure 5.2. They clearly demonstrate the following results:

- Increasing the *view distance* from 1.0m to 1.5m drastically increases the success rate.

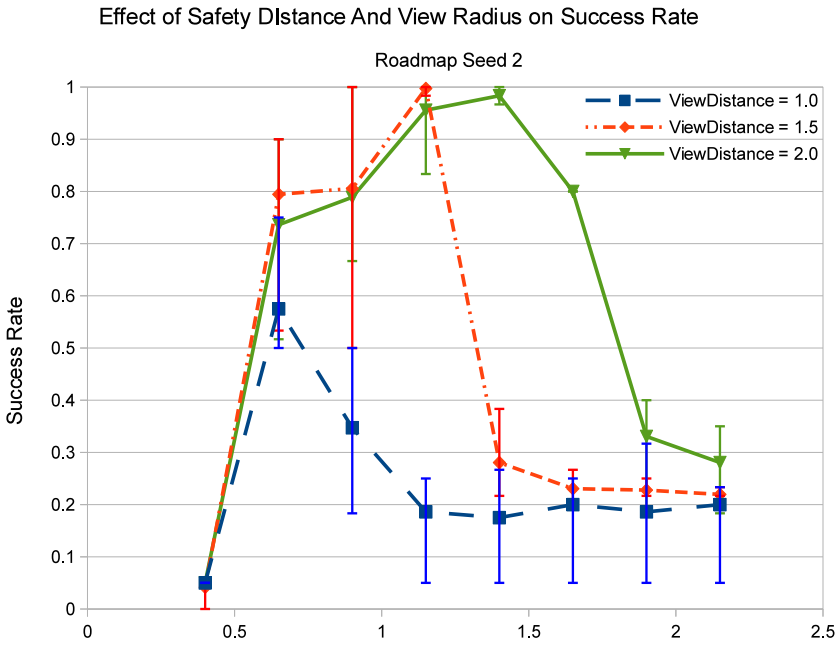
- Increasing the *safety distance* initially increases the success rate, but as the former approaches and then exceeds the *view distance*, the success rate dramatically falls. There is a range of intermediate values of the *safety distance* where the success rate is reliably high.

- The effect of increasing *view distance* and *safety distance* persists across roadmaps with different seeds.

- As expected, a *safety distance* close to the sum of two robots' radii results in almost universal failure since the robots collide almost as soon as the algorithm starts, mostly before they can get to the first waypoint.



(a) Seed 1



(b) Seed 2

Figure 5.2: Effect of *safety distance* and *view distance* on success rate for 10 robots and 6 waypoints.

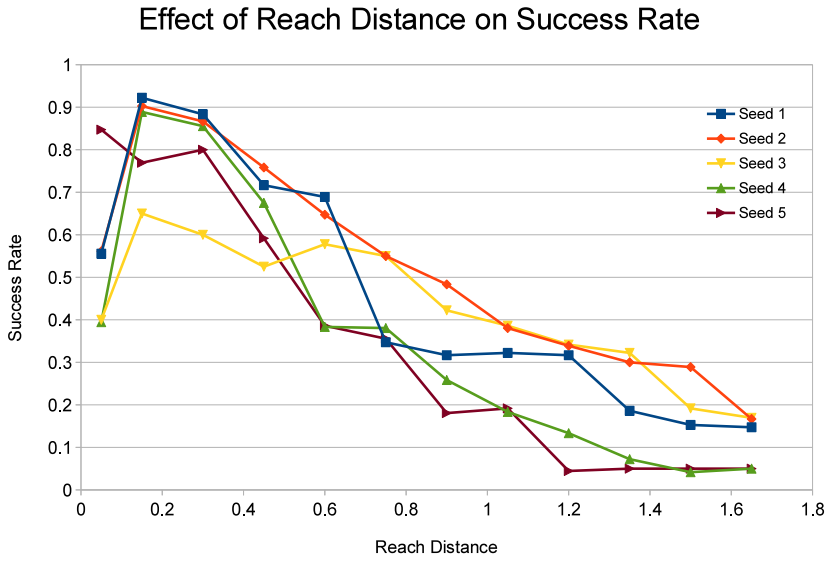


Figure 5.3: Effect of *reach distance* on success rate for 10 robots and 6 waypoints.

5.2.2 Effect of Reach Distance on Success Rate

We use *reach distance* values from 0.15m to 1.65m, in 0.15m increments. The experiment is run for 5 different roadmap seeds; each data point is the average of 6 runs. As before, we use 10 robots and 6 waypoints.

The results are shown in Figure 5.3. We make the following observations:

- In most cases, increasing the reach distance slightly improves the success rate initially, probably due to more and better observations of leader.
- In general, as reach distance increases, the success rate steadily drops because the chance of colliding with obstacles increases.
- The roadmap seed greatly influences how much and how gradually the performance degrades. For example, with seeds 1 and 5, the degradation is more

step-wise.

5.2.3 Scalability and Redundancy Experiment

In this experiment, we characterize the scalability of the approach to an increasing number of robots. We also examine how the ordering of the robots' initial positions affects the ability of our method to exploit redundancy to recover from failures of individual robots.

We use the larger office-like environment, with robots starting as a chain in the bottom portion of the environment. The environment is divided into 4 overlapping regions. Robots are assigned to different regions such that the distribution of robots to regions is equal.

We explore the effect of increasing the number of robots on the success rate. In particular, we are interested in whether the success rate can be maintained as the number of robots increases. We run the experiment with increasing numbers of robots (in increments of 4), ensuring that the distribution of regions to robots is always equal.

We are also interested in whether the initial ordering of the robots affects the success rate. We therefore examine two schemes to distribute regions among robots. In the first scheme, all the robots assigned to the first region occupy the first few positions of the chain, followed by all the robots assigned to the second region, and so on. We refer to this as the Sorted Order. In the second scheme, the first robot is assigned to the first region, the second robot is assigned to the second region, etc. When the last region has been assigned, the next robot is assigned the first region. We refer to this as the Interleaved Order.

The results for this experiment are shown in Figure 5.4. A number of conclusions can be drawn from these results:

- For both Interleaved and Sorted orderings, the success rate declines noticeably but not steeply as the number of robots increases. This suggests that any benefit to the group of having more candidates during the leader election phase is offset by increased chances for failure in other phases. Two possible culprits are:
  - In the Leader Switching step, a newly elected leader may need to travel a longer a distance to reach its designated leader position when the group is larger.
  - In the Leader Following step, if a robot fails when following, all the robots behind it also fail. The effect of these failures on the success rate is higher when there are more robots.
- The two orderings selected for this experiment are not significantly different from one another in their effect on scalability. This suggests that the effect of robot ordering on the success rate is minimal compared to other factors, at least for the chosen environment. More work is needed to conclusively establish the effect of ordering, if any, on the success rate.

5.2.4 Evaluation on Physical Robots

In this experiment, we qualitatively characterize the robustness of the physical robot implementation. We are particularly interested in noting the causes of failure when using different numbers of robots.

We run our experiment in the physical environment described in Section 5.1.1, using 5 waypoints. In each case, every robot is assigned a region in the environment

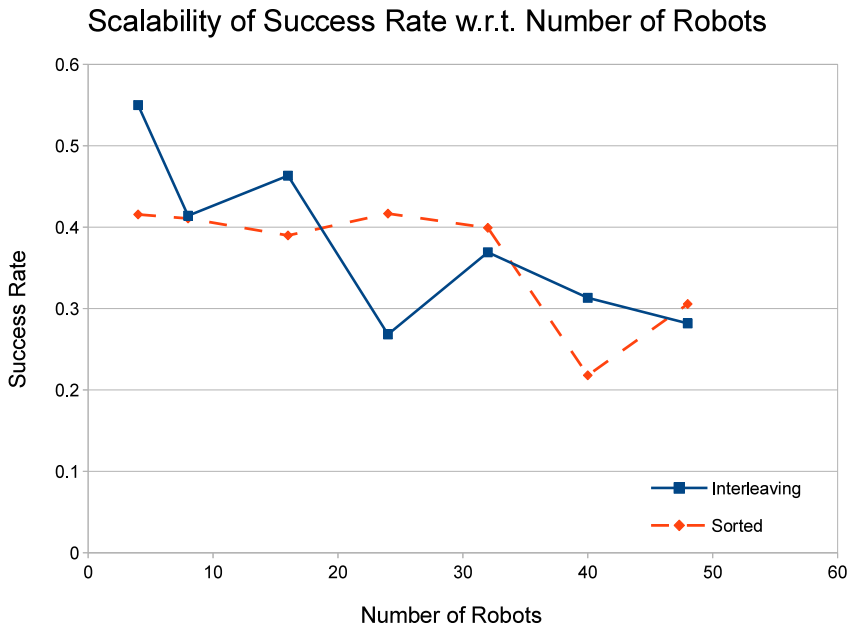


Figure 5.4: Scalability of success rate with respect to increasing numbers of robots, for two different orderings (Sorted and Interleaved).

for which it can plan. No robot is able to plan for the whole environment. This forces the robots to cooperate through caravanning to visit all the waypoints. Each robot uses MAPRM sampling to generate a roadmap with 90 nodes. We repeat the experiment for increasing numbers of robots, starting from 2, collecting 10 runs for each. For each run, we record the number of leader switches that occur, the number of waypoints each robot reaches, as well as the causes of failure for each robot.

Our results for 2 robots and 3 robots are shown in Tables 5.1 and 5.2, respectively. Runs in which a robot reaches the final waypoint are in bold.

A comparison of the two tables reveals that the number of runs in which one or more robots reached the final waypoint is about the same for both the 2-robot and 3-robot cases. In both cases, the main cause of failure is collision during the leader-

Table 5.1: Results of 10 physical robot trials with 2 robots, with failure event (if any) and last waypoint reached

Run #	Robot 1	Robot 2
1	**Reached final waypoint (5)**	**Reached final waypoint (5)**
2	**Reached final waypoint (5)**	**Reached final waypoint(5)**
3	**Reached final waypoint (5)**	**Reached final waypoint(5)**
4	**Reached final waypoint (5)**	**Reached final waypoint (5)**
5	Lost sight of target (3)	No valid path to next waypoint (4)
6	No valid path to next waypoint (3)	Failed because target (robot 1) failed (3)
7	**Reached final waypoint (5)**	**Reached final waypoint (5)**
8	**Reached final waypoint (5)**	**Reached final waypoint (5)**
9	Collided with other robot while following (4)	Collided with other robot while being followed (4)
10	Collided with other robot while switching (3)	Collided with other robot while switching (3)

Table 5.2: Results of 10 physical robot trials with 3 robots, with failure event (if any) and last waypoint reached. © 2013 IEEE.

Run #	Robot1	Robot2	Robot3
1	**Reached final waypoint** (5)	Lost sight of target (3)	Lost sight of target (2)
2	Collided with wall (3)	Failed because target (robot 1) failed (3)	Lost sight of target (2)
3	**Reached final waypoint** (5)	**Reached final waypoint** (5)	**Reached final waypoint** (5)
4	Collided with other robots while switching (3)	Collided with leader while switching (3)	Collided with leader while switching 3)
5	Collided with other robots while switching (4)	Collided with leader while switching (4)	Collided with leader while switching (4)
6	**Reached final waypoint** (5)	**Reached final waypoint** (5)	**Reached final waypoint** (5)
7	**Reached final waypoint** (5)	**Reached final waypoint** (5)	**Reached final waypoint** (5)
8	**Reached final waypoint** (5)	**Reached final waypoint** (5)	**Reached final waypoint** (5)
9	**Reached final waypoint** (5)	**Reached final waypoint** (5)	**Reached final waypoint** (5)
10	**Reached final waypoint** (5)	Lost sight of target (3)	Target failed (3)

switching phase, whereby a robot that has become the leader collides with another robot while following the path it has planned to its designated leader position. In addition, there is at least one instance in which a robot lost its target during the leader-following phase, leading to failure. Both of these failures can ultimately be attributed to errors in observation.

These results demonstrate that the proposed method can be reasonably applied to physical robots but is not completely free from failure.

6. CONCLUSION

In this thesis, we present a novel solution to the problem of multi-robot caravanning, which is the problem of ensuring that a group of robots visits a sequence of locations in an environment while staying together as a group, which we refer to as a caravan.

Our solution employs leader election to select a robot that will find and follow a path between a given pair of waypoints, and leader following to ensure that other robots follow the leader until the next waypoint. Our novel application of leader election allows robots to exploit redundancy in information and to tolerate incomplete information among individual robots.

We evaluate an implementation of our approach experimentally, both in simulation and on a physical platform. We demonstrate the feasibility of the approach and explore the effect of various parameters on the success rate.

REFERENCES

[1] A. Agha-Mohammadi, S. Chakravorty, and N. M. Amato. FIRM: Feedback controller-based information-state roadmap - a framework for motion planning under uncertainty. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 4284–4291, 2011.

[2] N. M. Amato, O. B. Bayazit, L. K. Dale, C. Jones, and D. Vallejo. OBPRM: an obstacle-based PRM for 3D workspaces. In *Proceedings of the Third Workshop on the Algorithmic Foundations of Robotics: the Algorithmic Perspective*, WAFR '98, pages 155–168, Natick, MA, USA, 1998. A. K. Peters, Ltd.

[3] N. Basilico, N. Gatti, and F. Amigoni. Leader-follower strategies for robotic patrolling in environments with arbitrary topologies. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems*, volume 1 of *AAMAS '09*, pages 57–64, Richland, SC, 2009. International Foundation for Autonomous Agents and Multiagent Systems.

[4] O. B. Bayazit, J.-M. Lien, and N. M. Amato. Roadmap-based flocking for complex environments. In *Proc. Pacific Graphics*, pages 104–113, Oct. 2002.

[5] K. E. Bekris, K. Tsianos, and L. E. Kavraki. A decentralized planner that guarantees the safety of communicating vehicles with complex dynamics that replan online. In *Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS)*, 2007.

[6] D. Canepa and M. G. Potop-Butucaru. Stabilizing flocking via leader election in robot networks. In *Proceedings of the 9th International Conference on Stabilization, Safety, and Security of Distributed Systems*, SSS'07, pages 52–66, Berlin, Heidelberg, 2007. Springer-Verlag.

[7] S. Carpin and L. Parker. Cooperative leader following in a distributed multi-robot system. In *Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on*, volume 3, pages 2994–3001, 2002.

[8] S. Carpin and L. E. Parker. Cooperative motion coordination amidst dynamic obstacles. In *Distributed Autonomous Robotic Systems*, pages 145–154, 2002.

[9] H. Choset. Coverage of known spaces: the boustrophedon cellular decomposition. *Autonomous Robots*, 9(3):247–253, 2000.

[10] H. Choset. Coverage for robotics–a survey of recent results. *Annals of Mathematics and Artificial Intelligence*, 31(1-4):113–126, 2001.

[11] J. Denny, A. Giese, A. Mahadevan, A. Marfaing, R. Glockenmeier, C. Revia, S. Rodriguez, and N. M. Amato. Multi-robot caravanning. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, Nov. 2013. To appear.

[12] J. Desai, J. Ostrowski, and V. Kumar. Controlling formations of multiple mobile robots. In *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, volume 4, pages 2864–2869, May 1998.

[13] K. Easton and J. Burdick. A coverage algorithm for multi-robot boundary inspection. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 727–734. IEEE, 2005.

[14] H. Garcia-Molina. Elections in a distributed computing system. *Computers, IEEE Transactions on*, C-31(1):48–59, Jan. 1982.

[15] R. Geraerts. *Sampling-based Motion Planning: Analysis and Path Quality*. Ph.D. Thesis, Utrecht University, 2006.

[16] B. P. Gerkey, R. T. Vaughan, and A. Howard. The Player/Stage Project: Tools for multi-robot and distributed sensor systems. In *In Proceedings of the 11th International Conference on Advanced Robotics*, pages 317–323, 2003.

[17] Y. Guo and L. Parker. A distributed and optimal motion planning approach for multiple mobile robots. In *Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on*, volume 3, pages 2612–2619, 2002.

[18] J. Huang, S. Farritor, A. Qadi, and S. Goddard. Localization and follow-the-leader control of a heterogeneous group of mobile robots. *Mechatronics, IEEE/ASME Transactions on*, 11(2):205–215, Apr. 2006.

[19] J. Jennings, G. Whelan, and W. Evans. Cooperative search and rescue with a team of mobile robots. In *Advanced Robotics, 1997. ICAR '97. Proceedings., 8th International Conference on*, pages 193–200, 1997.

[20] S. Karaman and E. Frazzoli. Sampling-based algorithms for optimal motion planning. *International Journal of Robotics Research (IJRR)*, 30:846–894, 2011.

[21] L. E. Kavraki, P. Švestka, J. C. Latombe, and M. H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans. Robot. Automat.*, 12(4):566–580, Aug 1996.

[22] A. Kolling and S. Carpin. Multi-robot surveillance: An improved algorithm for the graph-clear problem. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 2360–2365, May 2008.

[23] J.-C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Boston, MA, 1991.

[24] S. M. Lavalle. Rapidly-exploring random trees: A new tool for path planning. Technical report, 1998.

[25] S. M. LaValle and J. J. Kuffner. Randomized kinodynamic planning. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 473–479, 1999.

[26] J.-M. Lien. Hybrid motion planning using Minkowski sums. In *Proceedings of the Robotics: Science and Systems Confrence (RSS)*, June 2008.

[27] J.-M. Lien, S. Thomas, and N. Amato. A general framework for sampling on the medial axis of the free space. In *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*, volume 3, pages 4439–4444, Sep. 2003.

[28] S. R. Lindeman and S. M. Lavalle. Current issues in sampling-based motion planning. In *The International Symposium on Robotics Research (ISRR)*, 2004.

[29] T. Lozano-Perez. Spatial planning: A configuration space approach. *Computers, IEEE Transactions on*, 100(2):108–120, 1983.

[30] M. Mataric, M. Nilsson, and K. Simsarin. Cooperative multi-robot box-pushing. In *Intelligent Robots and Systems 95. 'Human Robot Interaction and Cooperative Robots', Proceedings. 1995 IEEE/RSJ International Conference on*, volume 3, pages 556–561 vol.3, 1995.

[31] M. Mazo Jr, A. Speranzon, K. H. Johansson, and X. Hu. Multi-robot tracking of a moving object using directional sensors. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 2, pages 1103–1108. IEEE, 2004.

[32] T. McMahon, S. Jacobs, B. Boyd, L. Tapia, and N. Amato. Evaluation of the k-closest neighbor selection strategy for PRM construction. Technical Report TR12-002, Texas A&M, College Station TX., 2011.

[33] R. Munoz-Salinas. Aruco, 2012. http://sourceforge.net/projects/aruco/.

[34] R. R. Murphy, J. Casper, M. Micire, and J. Hyams. Mixed-initiative control of multiple heterogeneous robots for urban search and rescue, 2000.

[35] G. A. Pereira, A. K. Das, R. V. Kumar, and M. F. Campos. Decentralized motion planning for multiple robots subject to sensing and communication constraints. 2003.

[36] J. Reif and M. Sharir. Motion planning in the presence of moving obstacles. *J. ACM*, 41(4):764–790, Jul. 1994.

[37] C. W. Reynolds. Flocks, herds, and schools: A distributed behavioral model. In *Computer Graphics*, pages 25–34, 1987.

[38] G. Sanchez and J.-C. Latombe. Using a prm planner to compare centralized and decoupled planning for multi-robot systems. In *Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on*, volume 2, pages 2112–2119, 2002.

[39] D. A. Shell and M. J. Mataric. On foraging strategies for large-scale multi-robot systems. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 2717–2723. IEEE, 2006.

[40] M. T. J. Spaan and F. C. A. Groen. Team coordination among robotic soccer players. In *In Proceedings of RoboCup International Symposium*, 2002.

[41] A. Stranieri, E. Ferrante, A. E. Turgut, V. Trianni, C. Pinciroli, M. Birattari, and M. Dorigo. Self-organized flocking with an heterogeneous mobile robot swarm. In T. Lenaerts, M. Giacobini, H. Bersini, P. Bourgine, M. Dorigo, and R. Doursat, editors, *Advances in Artificial Life. Proceedings of the 11th European Conference on Artificial Life (ECAL 2011)*, pages 789–796. MIT Press, Cambridge, MA, 2011.

[42] L. Tapia, S. Thomas, B. Boyd, and N. M. Amato. An unsupervised adaptive strategy for constructing probabilistic roadmaps. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 4037–4044, May 2009.

[43] K. Williams and J. Burdick. Multi-robot boundary coverage with plan revision. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 1716 –1723, may 2006.

[44] S. A. Wilmarth, N. M. Amato, and P. F. Stiller. MAPRM: A probabilistic roadmap planner with sampling on the medial axis of the free space. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, volume 2, pages 1024–1031, 1999.

[45] D. Y. Yeung and G. Bekey. A decentralized approach to the motion planning problem for multiple mobile robots. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 4, pages 1779–1784, Mar 1987.