Multi-Robot Frequency-Based Patrolling

Yehuda Elmaliach
Department of Computer Science

Ph.D. Thesis

Submitted to the Senate of Bar-Ilan University
Ramat Gan, Israel
January 2009

This work was carried out under the supervision of Prof. Gal A. Kaminka,
Department of Computer Science, Bar-Ilan University.

Acknowledgments

I would like to give special thanks to: Noa Agmon for helping me in many useful discussions and bringing some of the ideas to publication; Avi Rosenfeld for useful comments early on; Ruti Schechter-Glick for discussing early ideas for fence patrolling. I thank Hagai Cohen and Asaf Shiloni for carrying out significant programming tasks in Part II. I would like to thank the other members of the MAVERICK lab for the friendly atmosphere.

I would like to thank Sagit, my wife, for supporting me throughout the thesis and taking care of the family needs in order to give me free time to focus on my thesis and to my parents, for supporting me and sharing their life experience with me, thank you for all their clever ideas.

Last but not least I would like to thank my advisor Prof. Gal Kaminka, to whom I owe extraordinary thanks for his professional advice, for help over and beyond his duty, for creating a wonderful feeling which made it a pleasure to work in his MAVERICK lab, and for being more than an advisor—for being my best friend.

This work was supported in part by ISF Grant #1357/07.

Abstract

Mobile robots can save human lives and financial costs by replacing humans in mundane, or dangerous tasks. Patrolling is one such task: It is often mundane, is inherently repetitive, and may involve risk to human patrollers. One important type of patrolling is *frequency-based patrolling*, which involves visiting all points within a target work area at a fixed frequency (as much as possible).

Our focus in this dissertation is on frequency-based patrolling by teams of multiple, cooperating, robots. First, it formally defines the frequency-based patrolling problem, and distinguishes three different possible optimization criteria by which to evaluate solutions. Second, it presents solutions that are provably robust to robot death, guaranteeing that patrolling will continue as long as at least one robot is functioning. It also addresses event handling in patrolling, exploring algorithms that optimally select which robot should break the patrolling motion to respond to an event at a given location, and how to distribute the event handling load among different robots.

The dissertation makes a clear distinction between patrolling areas (enclosed by polygons), and patrolling polylines. In particular, the dissertation shows that patrolling an open polyline (e.g., a two-ended fence) is inherently in conflict with the goal of achieving uniform point visit frequency. Different algorithms are therefore presented for patrolling areas and for patrolling polylines.

Finally, we allow each patrolling unit to include multiple robots that move in formation. Unfortunately, known algorithms for formation maintenance typically utilize the robots' sensors to carry out the formation, thus prohibiting their use for carrying out the surveillance task. We thus present a first step towards a novel formation-maintenance technique which multiplexes the use of sensors and communications, and allows the robots to utilize their sensors for monitoring their surroundings, while maintaining the formation.

Throughout the dissertation we utilize experiments with real and simulated robots to evaluate the various techniques and demonstrate their effectiveness. Where possible, we analytically provide hard guarantees on the optimality and capabilities of the different models.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Mobile robots can save human lives and financial costs by replacing humans in mundane, or dangerous tasks. For instance, robots may be used for cleaning (e.g., [16]), hazardous waste removal (e.g., [47]), warehouse automation (e.g., [65]), and humanitarian de-mining (e.g., [56]). Recent applications of interest involve the use of robots in patrolling, as part of surveillance tasks. Patrolling is often mundane, is inherently repetitive, and may involve risk to human patrollers.

Patrolling is defined as *"The act of walking or traveling around an area, at regular intervals, in order to protect or supervise it"* [1]. It is a standard surveillance task that is carried out by humans along borders and open-ended fences, around perimeters, and in areas requiring continuous monitoring. In all three settings, we typically distinguish between *frequency-based patrolling*, which involves visiting all target points at a fixed frequency (as much as possible), and *adversarial patrolling*, which involves monitoring target points to detect an intrusion by a mobile adversary.

Our focus in this dissertation is on frequency-based patrolling by teams of multiple, cooperating, robots. Patrolling involves repeatedly visiting all points within a target work area. If the entire terrain cannot be monitored at all times, each point p in the target area is monitored once every t_p time cycles. The frequency is, then, $f_p = \frac{1}{t_p}$. Increased availability of multiple robots raises new opportunities for patrol missions. First and foremost, patrolling can be made more time-efficient in the sense that the frequency is potentially higher, i.e., t is smaller. In addition, robustness can be attained in the sense that if at least one robot is active, the patrol mission can still be accomplished.

Previous work on frequency-based patrolling in the context of multi-robot systems, has left key challenges open. First, patrolling has been investigated largely in an ad-hoc fashion, without a formal analysis of the quality of the task in light of its principal visit frequency goals. Indeed, most previous work do not distinguish patrolling from repeatedly covering the target area, without any regard to frequency constraints. Second, the opportunity for increased robustness in the sense of overcoming robot failure has not been investigated theoretically. Third, previous work does not distinguish between target work areas based on their topology (e.g., closed perimeter versus an area enclosed by a simple polygon, versus an open-ended fence); however, the topology of the work area makes a very significant difference in the patrolling algorithm to be deployed by the robots. Chapter 2 discusses related previous work in detail.

This dissertation tackles these open challenges in multi-robot frequency-based patrolling. First, it formally defines the frequency-based patrolling problem, and distinguishes three different possible optimization criteria by which to evaluate solutions. Second, it presents solutions that are provably robust to robot death, guaranteeing that patrolling will continue as long as at least one robot is functioning. It also addresses event handling in patrolling, exploring algorithms that optimally select which robot should break the patrolling motion to respond to an event at a given location, and how to distribute the event handling load among different robots.

The dissertation makes a clear distinction between patrolling areas (enclosed by polygons), and patrolling polylines. In particular, the dissertation shows that patrolling an open polyline (e.g., a two-ended fence) is inherently in conflict with achievement of the various patrolling optimization criteria. Different algorithms are therefore presented for each different topology. Section 1.1 below provides an overview of the first part of dissertation, which explores patrolling areas. Section 1.2 provides an overview of patrolling open polylines.

1.1 Patrolling in Areas

The first part of the dissertation discusses the problem of patrolling a target work area, enclosed within a closed polygon. Patrolling involves repeatedly visiting all points within the work area, to assess environmental state, e.g., by deploying sensors in those points. If the entire terrain cannot be monitored at all times, each location in the target

area is monitored once every f time cycles. The frequency is, then, $1/f$. Increased availability of multiple robots raises new opportunities for patrol missions. First and foremost, patrolling can be made more time-efficient in the sense that the frequency is potentially higher, i.e., f is smaller. In addition, robustness can be attained in the sense that if at least one robot is active, the patrol mission can still be accomplished.

Previous work has offered several approaches to patrolling of areas [7, 8, 11, 13, 38, 52]. However key challenges in surveillance have been left open. First, patrolling has mostly been done in ad-hoc fashion, without a formal analysis of the quality of the task in light of its principal frequency-based goals. Second, the opportunity for increased robustness has not been investigated theoretically. Third, handling non-uniform terrains in terms of velocity and directional constraints has not been addressed.

Hence, this part deals with constructing patrol paths for a group of mobile robots that are required to patrol in a non-uniform continuous target area (divided into a grid). We base our solution on recent work in multi-robot coverage [2, 3, 44, 45], in which the authors suggest a family of algorithm for generating cyclic paths for covering a terrain *once*. The solution we present guarantees that every point will be attended at the same frequency, by creating one cyclic patrol path visiting all points in the target area (a Hamiltonian cycle in the grid), and instructing all robots to move along this cycle while maintaining equidistant relative positions.

Robots have velocity limitations, which depends on both the *terrain* in a given location, and *direction* in which they travel. For example, climbing a hill is typically done in a lower velocity compared to climbing down the same hill. Therefore a cost should be associated with each point (and direction) of the terrain, making the terrain grid directionally non-uniform.

We therefore consider directionally non-uniform terrains. We first provide an algorithm that finds the minimal cyclic path (minimal Hamiltonian cycle) given the terrain. We then find points along the path from which the patrol will start, and find an optimal assignment of robots to those locations in the sense that they will arrive at their starting points in minimal time. Finally, we evaluate our derived patrol algorithm using the frequency optimization criteria described in Section 3.1. By basing our solution on the choice of minimal Hamiltonian cycle, we guarantee maximal uniform frequency in the cycle. Similar to the robustness of the multi-robot coverage described in [44], our solution is robust, therefore guarantees maximal uniform frequency for one or more

non-faulty robots.

This part also deals with the case in which while patrolling along the area, the robots are required to handle events. An event is a transient addition to the area in a specific location, such that it requires special attention by the robots as they pass through it for a limited period of time. We assume that each event has an associated urgency of handling it. This is expressed by a limit on the time during which this event should be handled. For example, in security applications an event could be a detection of an unauthorized personnel inside the area, and in cleaning missions it could correlate to cleaning a broken glass in addition to the regular cleaning task.

We provide a set of algorithms that deal with a single event. The difference between the algorithms lies in the urgency of handling the event. If there is enough time to handle the event jointly by all robots, then it will be divided between the team of robots uniformly. This division can guarantee that the event will be handled along with minimizing the interference with the patrol path to other cells in the area. Specifically, we are interested in minimizing the time it takes the robots to regain their uniform distribution along the cyclic path. In addition, we would like to minimize the frequency disturbance to other cells during the period of time the event is handled. We show that the algorithms we provide guarantee these properties whenever possible.

This part is organized as follows. Chapter 3 formulates the area patrol problem from the point of view of point-visit frequency optimization. It presents the spanning-tree patrolling (STP) algorithm, and the initialization algorithm required to show optimal patrolling. Chapter 4 extends the original STP algorithm to account for events which cause delays in patrolling.

1.2 Patrolling of Open Polylines

A common thread through much of existing work is reliance on the underlying work-area to be enclosed by a polygon. A topologically-circular path is computed inside the work area, and is used as the basis for the patrol. The robots move in a coordinated fashion along the path, such that robots are equidistant in time from each other. This improves performance in terms of the frequency optimization criteria, as well as minimization of travel time to any single point (e.g., in response to an event taking place there).

In the second part of the dissertation we focus on multi-robot patrolling *along a polyline*, e.g., a two-ended fence. Point-visit frequency is difficult to optimize along a polyline, since inherently, towards the end-points of the polyline, the robot must backtrack and therefore re-visit the points that it just visited when going towards the endpoint. Thus maintaining uniformity of point visit frequency, for instance, is challenging. Moreover, as we show later, uncoordinated movements of the robots may result in point arrival times that are unnecessarily large.

We make four concrete contributions in this part:

1. First, we show that to maintain minimal arrival time at any given point on the polyline (e.g., responding to an alarm at a given point), robots must coordinate to maintain fixed spatio-temporal distance between them.

2. Second, we present a parametrized distributed patrolling algorithm that divides the fence into segments of equal motion time (i.e., the endpoints of segments are temporally equidistant). Each robot then patrols one or more segments, such that its associated segments may overlap with those of others (depending on the parameters). We show how to incorporate real-world constraints such as velocity errors and turning velocities into these algorithms, and analytically explore the performance of the algorithms with respect to the different performance criteria.

3. Third, we present procedures for handling events in polyline patrolling. Events are defined by their location along the polyline, their handling duration requirement, and their deadline. The procedures we develop allow distribution of event handling among the robots, in order to minimize the inherent negative impact on patrolling performance.

4. Finally, we demonstrate the use of the algorithms with laboratory robots, and show that the addition of these real-world constraints leads to model predictions that match actual results.

We introduce a coordinated multi-robot fence-patrol technique: The *synchronized* patrolling method relies on dividing the fence into segments of equal motion time (i.e., the endpoints of segments are temporally equidistant). We then assign each segment to a robot. Each robot repeatedly covers its own segment of the fence, while synchronizing its velocity to its peers, such that the all begin and end segments jointly. We

show that this method reduces the maximal (worst-case) response time to any event on the polyline, compared to a non-coordinated approach.

We then generalize the synchronized method, introducing the *synchronized-overlap* method. Here, each robot is associated with more than one segment, such that its associated segments overlap with those of others. The robot therefore covers its peers' segments, when they move out of them (see Figure 1.1 for illustration).

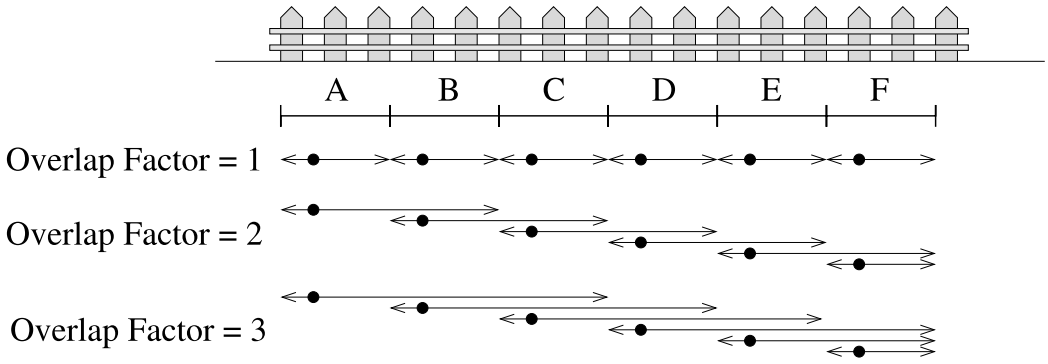


Figure 1.1: An illustration of the synchronized-overlap fence patrol technique. When the overlap factor is 1, the technique reduces to synchronized patrolling.

We analyze and contrast the different patrolling methods, with respect to two sets of performance criteria: Response time (arrival time to any single point of interest), and frequency criteria described in this dissertation: Uniformity, maximal average frequency, and maximal minimum frequency (under-bounding frequency). We examine the synchronized and synchronized-overlap methods with respect to the three frequency optimization criteria, using naïve and realistic motion models (which include, for instance, explicit treatment of robot turning durations, and velocity errors). We show that in many cases, the use of an overlap leads to improved results in frequency uniformity, without sacrificing average frequency or under-bounding frequency. However, this comes at a cost of worse performance in the extremities of the fence. We provide the analytical tools that would allow selection of the patrolling algorithm most suitable to given settings, and a detailed discussion of the trade-offs involved.

This part is organized as follows. Chapter 5 argues for the benefits of coordination in polyline frequency-based patrolling, presents the general patrolling algorithm, and discusses how to analytically choose optimal parameters for it, for given performance

criteria. Chapter 6 revisits the analytical model given more realistic robot motion models. Chapter 7 discusses procedures for event handling within the context of polyline multi-robot patrolling. Chapter 8 provides the results of patrolling experiments with real robots. These experiments validate the predictions of the analysis, given the more realistic motion models.

1.3 Additional Patrolling Challenges

All existing investigations of multi-robot patrolling discuss the task under the assumption that each of the patrolling robot is independently controlled, and monitors its assigned trajectory by itself, i.e., a patrolling unit consists of a single robot. However, in large-scale realistic applications of patrolling, it may be desired to have multiple robots in each patrolling unit. This allows improved sensing capabilities and survivability of the patrol unit.

One way to bridge the gap between patrolling algorithms—which treat every patrolling unit as a single controlled robot—and multi-robot patrolling units is by allowing patrolling units to move in formation. Here, a single robot selected as the leader of the robot formation follows the trajectory set by the patrolling algorithm. A group of follower robots—the remainder of the patrolling unit—track its movements to maintain a geometric shape for the purposes of the patrol. Unfortunately, known algorithms for formation maintenance typically utilize the robots' sensors to carry out the formation, thus prohibiting their use for detecting obstacles and carrying out the surveillance task.

We thus present, in Chapter 9, a first step towards a novel formation-maintenance technique which multiplexes the use of sensors and communications, and allows the robots to utilize their sensors for monitoring their surroundings, while maintaining the formation. Experiments with real and simulated robots explore this technique and demonstrate its effectiveness.

1.4 Thesis Overview

This dissertation is constructed of 10 chapters, organized in three main parts (see Figure 1.2). This chapter constitutes the introduction to this thesis. The next chapter sur-

veys the related work. Chapters 3–4 constitute Part 1 of the dissertation, which deals with patrolling areas. Chapters 5–8 constitute Part 2, which deals with patrolling polylines. In Part 3, Chapter 9 addresses formation maintenance. Chapter 10 concludes and discusses future work.

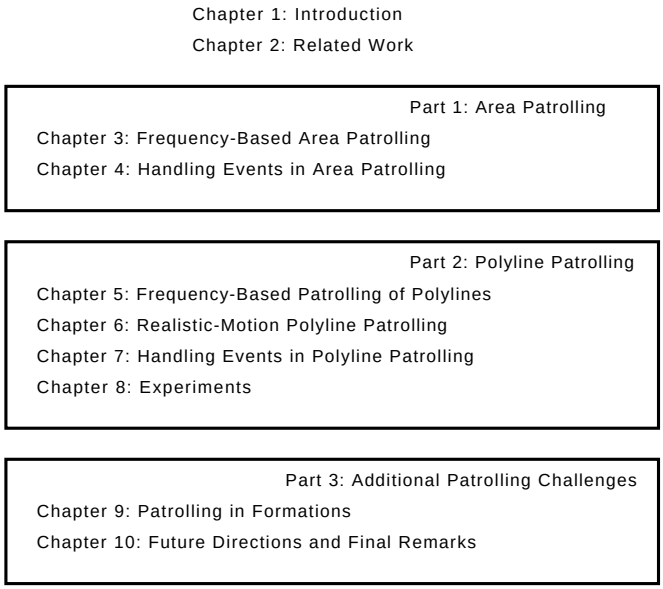


Figure 1.2: Thesis Structure.

1.5 Publications

Subsets of the results that appear in this dissertation were published in the proceedings of the following refereed journals, conferences, books and workshops:

- Yehuda Elmaliach, Noa Agmon and Gal A. Kaminka. Multi-Robot Area Patrol under Frequency Constraints. Annals of Mathematics and Artificial Intelligence, 2009 [27].

- Yehuda Elmaliach, Asaf Shiloni, and Gal A. Kaminka. A Realistic Model of Frequency-Based Multi-Robot Fence Patrolling. In Proceedings of the Seventh International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-08), 2008 [30].

- Yehuda Elmaliach and Gal A. Kaminka. Robust Multi-Robot Formations under Human Supervision and Control. Journal of Physical Agents, 2(1):31–52, 2008 [28].

- Yehuda Elmaliach, Asaf Shiloni, and Gal A. Kaminka. Frequency-Based Multi-Robot Fence Patrolling. Technical Report MAVERICK 2008/01, Bar Ilan University, Computer Science Department, MAVERICK Group, 2008 [29].

- Yehuda Elmaliach, Noa Agmon, and Gal A. Kaminka. Multi-Robot Area Patrol under Frequency Constraints. In Proceedings of IEEE International Conference on Robotics and Automation (ICRA-07), 2007 [26].

Videos showing actual runs in which these techniques were used, as well as videos of related techniques, are available at http://www.cs.biu.ac.il/~maverick/Movies/ [53].

Chapter 2

Related Work

In this chapter, we present a detailed discussion on related work. As our work in this dissertation is related to a number of different areas of research, we discuss each separately. Section 2.1 discusses related work on patrolling areas, perimeters, and polylines. Section 2.2 discusses related work on multi-robot formation maintenance. Section 2.3 discusses other miscellaneous related investigations.

2.1 Patrolling

The patrolling task, sometimes referred to as *repetitive sweeping* or *repetitive coverage*, is relatively a recent challenge area for multi-agent and multi-robot researchers. In general, three approaches can be found in the literature:

Cyclic Paths. A principled approach to patrolling, in which the algorithms plan cyclic paths through the work area. By nature, points on the cyclic path are visited repeatedly as long as the robots continue to move along the path.

Randomized Movement. A different approach to patrolling relies on continuously executing randomized movement in the work area. The nature of the randomization results in uniform (given sufficient time) distribution of visits to each point in the work area.

Hierarchical area division (partitioning). Not strictly a patrolling approach, but one used often in connection with multi-robot patrolling. In this approach, the work

area is (recursively) partitioned into sub-areas. Each sub-area is allocated, using some task-allocation mechanism, to different robots. Each robot then patrols the sub-area using a single-robot patrol algorithm (Randomized Movement or Cyclic Path). Robots may switch or take over areas as needed.

Mechado et al. in [52] describe the patrolling problem in terms of movement in a general graph, and introduce a measure of patrolling quality, called *idleness*. Idleness (of a graph) measures the average number of time steps between visits to all node. They then utilize idleness in a number of heuristic patrolling algorithms for multiple robots, based on local or global path-planning, which they compare empirically in simulations. These are intended to balance average and worst-case idleness over time. In contrast to their work, we provide in this thesis a formal treatment of patrolling and idleness, which we translate to point-visit frequency. We introduce three different optimization criteria that build on idleness (average frequency, uniformity of frequency, and worst-case frequency) and analytically show that these criteria can all converge to optimal in the case of area patrolling, using the algorithms we develop. We also argue that specifically for polyline patrolling, uniformity of frequency is inherently impossible, as cyclic paths cannot be generated. Thus in contrast to Mechado et al., we distinguish between graphs of different topologies.

A survey by Almeida et al. [8] brings a discussion of different patrolling approaches, with respect to the idleness criteria. They compare paths based on machine learning, agents using negotiation mechanisms, heuristic algorithms (based on local idleness criteria), and an approach based on approximating the solution to the Traveling Salesman Problem (TSP), i.e., finding a cyclic path through the graph. They empirically demonstrate significant advantages to the TSP-based approach in average idleness. Our work on area patrolling addresses patrolling formally and analytically, in contrast to this work. We focus on a cyclic-path approach, and analytically show its optimality. However, rather than examining general graphs, we focus on patrolling in two-dimensional work-areas, in which we use approximate cellular decomposition. Moreover, we address patrolling in polylines, where cyclic paths cannot be generated.

Chevaleyre [13] and Chevaleyre et al. [14] offer a theoretical analysis of the patrol problem. They discuss two approaches to multi-robot patrolling in general graphs: One in which the problem is treated as finding a cyclic path through all nodes (i.e., the Traveling Salesman Problem TSP); and another in which the general graph is par-

titioned into k sub-graphs (where there are k robots), and each partition is patrolled independently. They examine the conditions under which a cyclic approach would be preferable to a partition-based approach, in terms of the worst idleness criteria. While we examine the optimality of other criteria as well (e.g., uniformity of point-visit frequency), we specialize our techniques to specific graphs, grids that form approximate cell decompositions of two-dimensional work areas. However, we also provide guarantees on robustness and efficiency of multi-robot solutions. In the case of polyline target areas, as cyclic paths are not possible, we develop a graph-partitioning solution. However, in contrast to [13, 14], we demonstrate the usefulness of maintaining right coordination between the robots in different segments.

Empirical investigations of patrolling have often utilized a partitioning approach, but often disregarded point-visit frequency, in favor of coordination and robustness concerns. For instance, Guo et al divide the patrolling area between robots [42, 43]. This study focused on the robots' localization and sensorial capabilities. This thesis' goals are complementary, in the sense that we assume perfect sensors and localization, but provide guaranteed frequency optimization.

Ahmadi and Stone [7] describe a negotiation-based approach for dividing the area between the robots, dealing with events such as addition and removal of robots from the system. We instead provide analytical treatment of robot removal and addition, and provide algorithms that guarantee optimal patrolling frequency, as well as a procedure to minimize the time for adjusting the patrolling to the removal or addition of robots.

Jung and Sukhatme describe in [48] a region based approach for tracking targets in a system with multiple robots and stationary sensors. They explicitly discuss patrolling frequency. We do not utilize stationary sensors in this work, and show that there are several different frequency criteria possible.

It is also possible, in principle, to carry out patrolling by repeated movements within the work area. Many swarm or ant-based coverage algorithms, when executed indefinitely, may in practice result in uniform distribution of point visits, though the frequency of their visits might not be easily guaranteed. One work that stands out among these is work by Yanovski et al. [66] who have shown that an ant-like exploration in a general graph, by multiple agents using simulated markings in the vertices, can result in point visit frequency which is uniform up to a factor of two (i.e., the number of visits to the most visited edge is no more than twice the number of visits to the

least visited edge).

Ryale et al. [59] and Girard et al. [40] describe architectures for multiple robot patrolling, using unmanned aerial vehicles. These systems focus on allowing a single operator to operate and command multiple robots. In contrast to these investigations, our work focuses on automatic optimization of frequency-based performance criteria, in polylines. We take into account velocity constraints along the path, turning velocities, etc.

Correll and Martinoli [18, 19] describe a distributed coverage for swarm robots, which combines elements of both area and boundary coverage. They implement their idea on swarm robots to cover turbine elements aligned on a surface area. Thus the robots have to cover the work area completely, yet each detected element is to be circumscribed by the robots. The main idea is to combine probabilistic and deterministic models in order to achieve better real-world performance. However, this work does not address frequency of repeated coverage.

We emphasize that patrolling, as studied in this thesis, is investigated from the point of view of optimizing point-visit frequency. There are alternative optimization criteria for patrolling. For instance, Agmon et al. [5, 6] study *adversarial patrolling* of perimeters (closed polylines) where the objective is to detect an adversary that is trying to evade the robots and penetrate a closed area. This work has recently been extended to adversarial patrolling in open polylines [4]. Paruchuri et al. [58] study the placement of checkpoints in adversarial environments, in which the robots' goal is to maximize their rewards. These rewards are received if the robots manage to observe an evading adversary.

2.2 Formations

Maintaining formation while moving requires the robots to locate themselves according to reference points. Balch and Arkin [9] examine three fundamental techniques for formation maintenance, in experiments with up to four (4) homogeneous robots:

1. *Unit-center-referenced* is a technique where the robots place themselves according to X, Y coordinates, relative to their peers in the formation, and subject to the geometric shape to be maintained. This technique relies on the ability of robots

to sense the locations of all others.

2. To address this requirement, the *leader-referenced* technique instead allows robots to position themselves relative to the position of only a single robot, which acts as a leader. However, all robots must orient themselves with respect to the same leader.

3. Finally, the *neighbor-reference* technique relaxes this requirement further. Here, each robot positions and orients itself with respecting to a single robot—called the *target*—but different robots can choose different targets.

It was shown that the last two categories in Balch and Arkin's work (*Leader-Referenced* and *Neighbor-Referenced*) are both related to a general method for formation maintenance, called *Separation-Bearing Control* (SBC) [20, 21, 31, 33]. In SBC, a single robot is chosen as the leader of the formation. Each robot (but the leader) must maintain connectivity—a given distance (separation) and angle (bearing)—with respect to an assigned target. There must be a path of such connected robots from every robot in the team to the leader. It was shown that SBC controllers are sufficient to maintain stable formations.

SBC is arguably the most practical formation-maintenance technique today for real-world settings. This is likely due to its simple requirements of sensing (monitoring) only one other robot, and to the wealth of opportunities it presents for optimizing sensor usage [49, 55] and robot role assignment [51, 54].

There have been several works addressing the robustness of SBC-based formations. Fredslund and Matarić [33] describe an algorithm for generating SBC monitoring rules for robots in a given formation. The robots are assumed to have supporting sensing capabilities, and the position of the leader is given. The monitoring rules are supplemented by communications for robustness against robot death.

Kaminka et al. [49] describe an algorithm that generates SBC monitoring rules based on the sensor configuration of the robots, and dynamically adjusts these rules to overcome sensor failures. They show that this leads to significantly improved robustness, as long as alternatives exist to prevent a robot from becoming completely disconnected. Their approach is susceptible to latencies of the communication protocol used to switch between different monitoring rules.

Our approach relies on fusing SBC control with open-loop, communication-based control of the formation, which relies on the localization of the robots and their ability to accurately estimate their own movements. In this, we complement the techniques outlined above, rather than compete with them.

Mourikis and Roumeliotis [55] discuss optimal sensor scheduling policies for formations, in which sensor use for localization within the formation is optimally balanced between resource consumption (e.g., energy) and localization accuracy. The sensors themselves are assumed to be fixed in configuration, but the frequency in which they are used is determined by the policies. Our work focuses on general sensor use, not only for localization. However, we do not address load- and energy- balancing.

Most previous work on formation maintenance in the presence of obstacles has assumed that obstacles are detectable in some unspecified fashion. Using the techniques presented below, robots can use their sensors to detect obstacles, to a greater extent than they do when they have to utilize their sensors to maintain the formation. In this, we facilitate the use of techniques which rely on the use of obstacle-detection sensors, and that are difficult to use in sensor-impoverished robots that utilize their robots for formation maintenance.

For example, Chen and Li [12] propose a technique where obstacles are recognized by the leader robot, which builds a path for the formation to avoid the obstacles. Thus the leader is responsible for detecting any obstacles. Our approach complements this technique, by allowing other robots to also detect obstacles.

Similarly, Ogren and Leonard [57] describe an approach for allowing a group of robots moving in formation to avoid known obstacles. They show how to calculate a path for each robot that best maintains the formation while avoiding obstacles. Our work is complementary: The multiplexing technique we present is focused on detection of unknown obstacles; but we do not provide a method for calculating obstacle-avoiding paths.

Balch and Hybinette [10] use social potential fields which use attraction and repulsion to position robots within their relative positions in a defined formation. This technique is robust to obstacles in the path of the robots, in the sense that the geometric shape maintained by the formation is dynamically stretched to account for obstacles. However, the techniques assumes that robots know of the positions of obstacles. The technique we present in this thesis frees up the robots' sensors for this purpose.

Dougherty et al. [24] and Balch and Arkin [9] address formation-maintenance, and discuss it in the presence of obstacles. However, the question of how obstacles are detected is left open. The techniques we present in this work can be useful for this task, and thus complement their work.

2.3 Coverage and Other Related Works

The patrol problem is closely related to the area coverage problem, in the sense that both require the robot or group of robots to visit all points in the given terrain. However, while coverage seeks to minimize the number of visits to each point (ideally, visiting it only once), patrolling seeks to maximize it (while still visiting all points). Therefore solutions that are used for the coverage problem might be used as basis for patrolling.

For example, Yanovski et al.'s work on patrolling [66] builds on their earlier work on ant-robot coverage [61–63]. Similarly, our own approach builds on earlier work in coverage; specifically, on Spanning Tree Coverage (STC), first introduced by Gabriely and Rimon [36] for single robots, and then extended to the multi-robot case by Hazon and Kaminka [44–46] and by Agmon et al. [2, 3]. The key idea in this family of algorithms is to approximate a two-dimensional work area using a grid, such that a Hamiltonian cycle is guaranteed to exist through the grid, which can be found by generating a spanning tree in the grid graph. This Hamiltonian cyclic path is used as the basis for patrolling, as the next section shows. Although we build on MSTC, our work here differs from it in three important ways. First, we allow modeling non-uniform terrains, in the sense of velocity and direction constraints imposed on different locations within the area. Second, we provide an algorithm for placing the robots such that their patrolling can commence as quickly as possible, a stage only worthwhile in patrolling. Finally, we address here also events that cause delay in patrolling, whereas previous work ignored such events.

Williams and Burdick [64] report on a related coverage investigation. They investigate a boundary coverage technique, in which a path is planned that allows a single robot to inspect (cover) the faces of multiple polygons located in a plane. The technique they develop allows for changes to the path, to allow for robot death failures or changes in the environment. The generated path is circular (robots go back to to their

initial locations), and thus in principle can be used as a basis for repeated boundary coverage, i.e., a type of patrolling path. However, the point visit frequency characteristics of such an application are unknown.

There are additional related investigations. A key difference between our work and all works discussed above is that we address the allocation of patrolling robots to handle events. This allocation is an instance of the general task allocation problem for multiple robots. There are many approaches used for solving this problem in various domains; however, recently, market-based approaches seem to be of particular interest: Smith created a system called *Contract Net*, a distributed problem solving in which the nodes in the system negotiate and send their bid to the manager which allocates the task for the lowest cost bid [60]. Dias et al. [22, 23] first used the concept of market based for multiple robots that cooperate for achieving a common goal. Golfarelli et al. [41] proposed a negotiation protocol in environment that the only possible contract could be swapping the task between the agents.

In contrast to these general techniques, we take advantage of the cooperative nature of the patrolling task (as described in this thesis), and specialize the task allocation mechanism such that the load of handling events is divided between the robots equally. Each event is handled by all robots using no negotiation, or other tools for determining which robot will handle which part of the event. Our solution is, therefore, easy to solve and is determined quickly.

Part I

Patrolling Areas (Closed Polygons)

The challenge of area patrolling involves generating patrol paths for a team of mobile robots inside a designated target work area, such that every point in the area is repeatedly covered. In this part of the dissertation, we address optimal multi-robot patrolling algorithms for areas enclosed by polygons.

Chapter 3 begins by presenting formal frequency optimization criteria used for evaluation of patrol algorithms. Then, we present a patrol algorithm that guarantees maximal uniform frequency, i.e., each point in the target area is covered at the same optimal frequency. This solution, called Spanning-Tree Patrolling (STP), is based on finding a circular path that visits all points in the area, while taking into account terrain directionality and velocity constraints. Robots are positioned uniformly along this path in minimal time, using a second algorithm. Moreover, the solution is guaranteed to be robust in the sense that uniform frequency of the patrol is achieved as long as at least one robot works properly.

In Chapter 4, we present a set of algorithms for handling events along the patrol path, which require the robots to stop at the event location to carry out some event-handling activity. Events are thus described by their location, the total handling time they require, and a deadline for handling them. Handling events is always in conflict with optimal frequency-based patrolling, as it always requires robots to stop, rather than continue their patrolling motions. The algorithms we develop differ in the way they handle the event, as a function of the time constraints for handling them. The advantage of these algorithms is that they handle the events as well as maintaining the patrol path and minimizing the disturbance to the system as much as the time constraint on the event permits.

Chapter 3

Frequency-Based Area Patrolling

We introduce our approach for patrolling of a two-dimensional work area. Section 3.1 introduces the area patrol problem, and the criteria used to assess patrolling quality. Section 3.2 then introduces the basic ideas underlying spanning-tree patrolling, the key idea in the algorithms we present. It also shows how to account for velocity and directionality constraints in generating the patrolling paths, to guarantee optimal patrolling frequency. Section 3.3 addresses the bootstrapping stage in which robots find their initial positions along the patrolling path. Section 3.4 ties the generation of the patrol paths, and the assignment of initial positions to robots, to discuss the optimality and robustness guarantees on the algorithm.

3.1 The Area Patrol Problem

We are given a simple polygon enclosing a continuous work area. We are given k robots, who are to *repeatedly* visit every point within the area. The work area may contain obstacles, through-which robots cannot move. We denote the size of the area that is not an obstacle by N. If $k \geq N$, then all points can be visited at all times by the team simply by assigning a robot to each point. Formally, the interval between visits is 0, since each point is visited with every passing time unit, by at least one robot. In the more common case, $k \ll N$. Necessarily, at least some of the points a_i, $1 \leq i \leq N$ have a non-zero interval between visits.

Thus the frequency in which points are visited are the focal point for the area patrol problem. There are several possible point visit frequency criteria according to which

patrol algorithms can be evaluated:

Uniform frequency: The goal is to decrease the variance between the frequencies in which each point is visited, i.e., all targets should ideally be visited with uniform frequency f.

Average frequency: The goal is to increase the average frequency f in which targets are visited. Note that this is independent of achieving uniform frequency.

Under-bounded frequency: The goal is to increase the minimal frequency in which any target is visited, such that every target is visited with frequency of at least f. In other words, all points should be visited at least once every $1/f$ cycles.

Given the goals above, the area patrol problem is to generate trajectories (velocity along paths) for each of the k robots, such that these achieve one or more of the goals above (or maximize some given trade-off between them). As in work on multi-robot coverage, different variations exist [15]: *Offline* (map of area and obstacles given apriori) or *online* (work area unknown), using *approximate* coverage (area is patrolled only up to some ε, typically due to division into a grid), or *exact* (all points in the area visited). Patrolling may take into account directionality and velocity constraints (e.g., the velocity in one direction may be different than in another), priorities in patrolling, etc.

3.2 Spanning-Tree Patrolling

This thesis addresses the area patrol problem as defined above, focusing on offline trajectory planning and approximate cell decomposition (utilizing a grid). We also take into account directional movement constraints, allowing the algorithms to model different robot velocity constraints in different directions. The algorithms achieve all three goals stated above: Uniformity of patrolling frequency, maximal average patrolling frequency, and maximal minimum frequency. We do this by generating a cyclic path visiting all target areas (a Hamiltonian cycle) and then place the robots uniformly along the cyclic path. If all robots move at the same direction then clearly each cell is visited at the same frequency (uniform frequency). Moreover, in uniform terrains each cell is visited at least once every $\lceil \frac{cycle\ length}{num\ robots} \rceil$ number of cycles, where cycle length is simply the number of nodes plus one.

We base our work on Multirobot Spanning Tree Coverage (MSTC) [3, 44]. A single robot is assumed to be equipped with a square-shaped tool of size D: Any point within the tool's area is taken to be visited. The robot moves by sliding the tool over the area in any of the four basic directions (North, South, East, West). The work area is approximately divided into a grid with cells of size $D \times D$. The grid is then made coarse, such that each new cell is of size $2D \times 2D$. The center-points of all such cells are connected to those of their neighbors in the four basic directions (North,South,East,West), to form a graph. A spanning tree is then induced from the graph. The robots follow the tree around in a clockwise or counterclockwise direction, creating a Hamiltonian cycle visiting all cells of the original grid (see example in Figure 3.1).

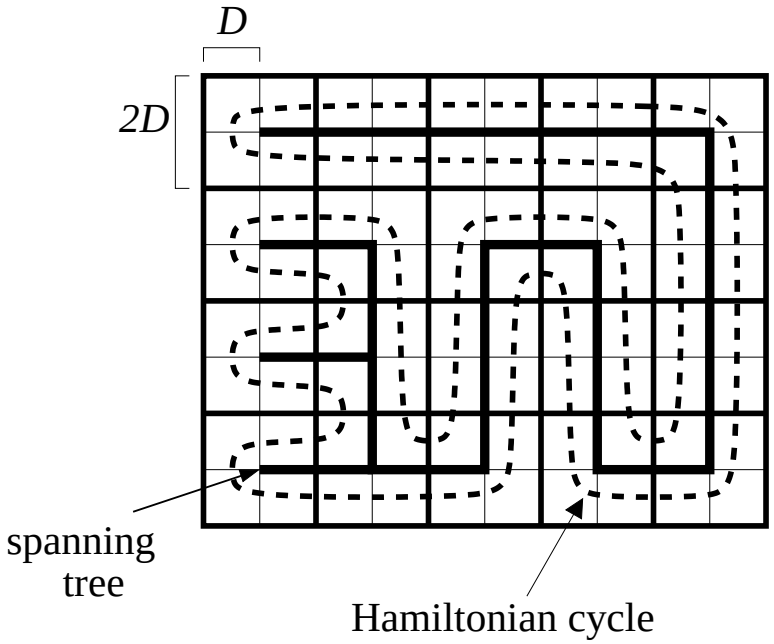


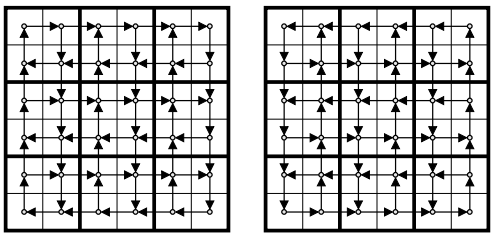
Figure 3.1: An example of spanning tree based coverage. Coarse grid is in bold, and the spanning tree connects all coarse grid cells. The Hamiltonian cycle over the fine grid is the dotted line along the spanning tree.

The key idea in Spanning-Tree Patrolling (STP) is to utilize the cyclic path for repeated patrolling. By placing robots in equidistant positions along the cycle induced by the spanning tree, synchronized movement by the robots will provide uniform, maximal frequency of visits to all points along the path. However, in non-uniform

terrains, movement in the four different directions can occur in different velocities. Moreover, terrain directionality constraints may mean that clockwise movement in any given location may have different velocity than in the counterclockwise direction.

We assign a cost—signifying velocity—to a movement between any two adjacent cells in the fine grid. Different costs may be assigned to two edges connecting the vertices in opposite directions. Our objective is now to convert the directed edges of the fine grid to undirected edges in the coarse grid while preserving the properties of the edges such that a minimal spanning tree on the coarse grid will yield a minimal Hamiltonian cycle on the fine grid. Then, placing the robots in equidistant *spatio-temporal* positions along the cycle will guarantee uniform maximal frequency.

Since the patrol tour can be conducted in either clockwise (CW) or counterclockwise (CCW) directions along the spanning tree path, we divide our world to CW and CCW. In general, there are four directed edges entering and four leaving each cell in the fine grid. Since we follow some spanning tree path, the options decrease and each cell have up to two incoming and two outgoing edges in each world (CW and CCW), as described in Figure 3.2. We find a minimal spanning tree in each of the worlds separately, and choose the minimal between both as base for the patrol path.



a. *b.*

Figure 3.2: Division of the area to clockwise (a.) and counterclockwise (b.) directions. The graphs are built such that the movement is suitable for traveling along a spanning tree. Union of the two graphs provide all possible movement options from each cell: up, down, right and left.

Note that Figure 3.2 clearly illustrates that the CW and CCW worlds are complementary in the following sense. First, the intersection between the worlds is empty, i.e., $\forall e \in E$, $e \in \mathsf{CW}(G)$ xor $e \in \mathsf{CCW}(G)$. Second, together they provide all connections between adjacent edges in four possible directions: North, South, East, and West.

In the following, we describe a principled assignment of weights (Assignment Assign_Opt) to the undirected edges of the coarse grid based on the weights of the directed edges of the fine grid. We then argue that using this assignment, finding a minimal spanning tree on the coarse grid representation guarantees determining a minimal Hamiltonian cycle on the fine grid. In order to do that, we first prove that in our scenario, a Hamiltonian cycle is created by each spanning tree and vice versa, i.e., each Hamiltonian cycle in the fine grid is translated to a spanning tree in the coarse grid. Based on that, we then prove, in Lemma 3, that Assignment Assign_Opt yields the minimality property we seek[1].

Assignment Assign_Opt**:** The cost assigned to the undirected edge (u,v) in the coarse grid (see Figure 3.3) is the sum of the directed edges in the fine grid, parallel to (u,v) from its two sides minus the sum of the directed edges perpendicular to (u,v) and intersecting it, or: $(a+b)-(c+d)$. Note that this can generate edges with negative cost. In this case we shift the cost of all edges by the minimal negative value, and use Kruskal's algorithm ([17]) for finding an MST.

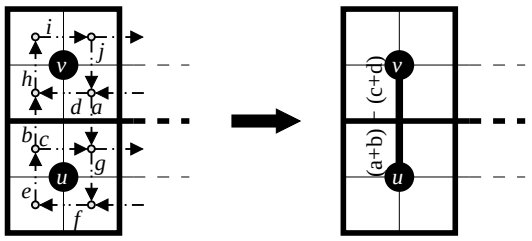


Figure 3.3: The assignment of weights to the undirected edges of the coarse grid based on the directed edges of the fine grid (here in the CW direction).

Lemma 1. *Every spanning tree on the coarse grid can be translated to a Hamiltonian cycle on the fine grid and vice versa, i.e., every Hamiltonian cycle on the fine grid can be translated to a spanning tree on the coarse grid.*

[1]Gabriely and Rimon discuss edge weights in single-robot STC [37]. In this work, each edge in the coarse grid was given a different weight in order to favor movement in certain directions, and a *minimal* spanning tree (MST) was found. However, they do not discuss the correspondence of these weights to the fine grid, and minimality of the Hamiltonian cycle was not proven.

Proof. The first part of the Lemma is shown in the initial algorithm of Gabriely and Rimon [36]. According to their algorithm, a Hamiltonian cycle is generated simply by moving along the spanning tree path in the fine grid.

In order to prove the second direction, we will first show that the existence of Hamiltonian cycle in the fine grid guarantees that only full edges are picked in the coarse grid. We choose, without loss of generality, the *CW* case. Figure 3.4 illustrates two adjacent vertices in the coarse grid, u and v, and their corresponding vertices in the fine grid. Denote the edge (u_2, v_1) by a, (v_3, v_4) by b, (u_2, u_4) by d and (v_3, v_1) by c. We must show that choosing edge a guarantees choosing also b and excludes c, d, and vice versa, i.e., choosing c forces choosing d and excludes a, b. Assume, towards contradiction, that a Hamiltonian cycle exists in the grid, but it uses only edge a and not b. Therefore in order to visit vertex u_4 d is chosen, contradicting the fact that they are all part in a Hamiltonian cycle, as u_2 cannot have two outgoing edges. The fact that c and d cannot be chosen along with a and b is proven similarly.

It is left to show that the Hamiltonian cycle on the fine grid creates a spanning tree on the coarse grid. Assume, towards contradiction, that there exists a Hamiltonian cycle that does not translate into a spanning tree on the coarse grid. This means that there exists a vertex in the coarse grid that is not covered by the spanning tree. This can happen only if not all fine vertices are visited, contradicting the fact that we have a Hamiltonian cycle. □

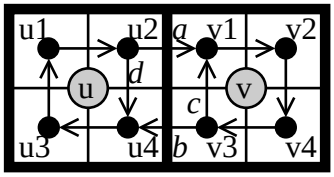


Figure 3.4: Illustration of Lemma 1.

Corollary 2. *A Hamiltonian cycle on the fine grid can include edges from either the* CW *world or the* CCW *world, but not from both.*

Figure 3.5 shows an example illustrating the corollary. A closed path is a Hamiltonian cycle if it covers all vertices of the graph once, meaning that the in-degree and out-degree of vertices in this path is 1. In Figure 3.5, all edges in the path are from

the CCW world except for edge (v_1, v_2) (dashed), which is from the CW world. The resulting path is *not* a Hamiltonian cycle, as v_1 has in-degree 2 and v_2 has out-degree 2.

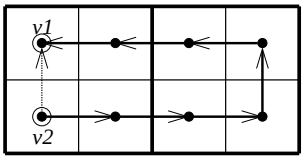


Figure 3.5: Illustration of Corollary 2, demonstrating the problem of combining edges from the CW and the CCW world.

Lemma 3. *Using Assignment* Assign_Opt*, an* MST *on the coarse grid representation yields a minimal Hamiltonian cycle (*HC_Min*) on the fine grid.*

Proof. Assume, towards contradiction, that there exists a Hamiltonian cycle, HC$'$ with total weight smaller than HC_Min. This can happen in one of two scenarios.

Case 1. has lower cost than the MST. This contradicts the minimality of the MST, hence this case is impossible.

Case 2. The spanning tree ST$'$ (corresponding to HC$'$) has higher total weight than the MST's weight and still HC$' <$ HC_Min. Consider the case in which the trees differ by one edge, $e \in$ MST, $e \notin$ ST$'$ and $e' \in$ ST$', e' \notin$ MST. Denote the directed edges forming e by a,b,c,d and the directed edges forming e' by a',b',c',d' (as described in Assign_Opt). Since ST$' >$ MST and based on Lemma 1, it follows that that weight$(e') >$ weight(e). Therefore, according to Assign_Opt, $a' + b' - (c' + d') > a + b - (c + d) \Rightarrow a' + b' - (a + b) > c' + d' - (c + d)$. Since we assume that HC$' <$ HC_Min and they differ only by e and e', if follows by the inclusion of e in HC and exclusion in HC_Min that $a' + b' + c + d < a + b + c' + d'$ $\Rightarrow a' + b' - (a+b) < c' + d' - (c+d)$, leading to a contradiction. It can be shown similarly for every spanning tree greater than the MST that this case is impossible. □

As a corollary of Lemmas 1 and 3, algorithm Generate_Cycle(Algorithm 1) finds the minimal Hamiltonian cycle over the work area. If robots move along this cycle such that their spatio-temporal positions are equidistant, they will visit all points along

the path with uniform, maximal frequency. The next section examines an algorithm for moving the robots from their initial positions into such equidistant positions as quickly as possible.

Algorithm 1 Generate_Cycle

1: Divide the area into two CW and CCW scenarios.
2: **for** each scenario (CW and CCW) **do**
3: create a graph on the coarse grid by assigning weights to edges as described in Translation A
4: Find a minimal spanning tree in the coarse grid using Kruskal's algorithm.
5: Calculate the total length of the Hamiltonian cycle generated by the minimal spanning tree.
6: Report scenario (CW or CCW) and cycle with shorter total length.

Note that the construction of minimal Hamiltonian cycle for patrolling applies to the coverage problem as well. The minimal cycle as found here for non-uniform terrains can be used also for single- or multi- robot coverage, achieving minimal coverage time with respect to the constraints on the terrain, as long as the robots move in either CCW or in CW direction (as implied by the output).

3.3 Allocating Robots to Initial Positions

After establishing the minimal cyclic path for the patrol mission by the group of mobile robots, it is left to determine the position of the robots along the cycle from which they begin their patrol. Clearly, in order to achieve uniform frequency it is sufficient to spread the robots uniformly along the cyclic path. The distance between every two robots along the cyclic path should be the total weight of the cycle divided by the number of robots, yielding an equal distance between every two consecutive robots along the patrol path. Since there is more than one possible assignment of the robots to such positions, we want to find the assignment that requires minimal change from current positions of the robots. Therefore we describe herein the algorithm Initialization, which finds the locations from which the robots should start patrolling, while minimizing the maximal distance a robot should travel in order to arrive at its location. As the robots move simultaneously from their initial positions to their positions along the

cycle, this corresponds to minimizing the time it takes all robots to be positioned and ready for the patrol mission.

We define the *Minimal Path Matching* (Min_Path_Match) problem as follows. Given a weighted graph $G = (V, E, W)$, a Hamiltonian cycle visiting all vertices in the graph, and a set of initial positions of k robots on vertices of G. Find an assignment of each robot to a position in the graph such that the following are fulfilled.

1. The distance between every two consecutive robots along the Hamiltonian cycle is equal.

2. The maximal distance traveled by a robot from its initial position to the assigned location is minimized.

We suggest the initialization algorithm Initialization (Algorithm 2) for solving the Min_Path_Match problem. The input to the algorithm includes: (1) The fine graph G; (2) the minimal Hamiltonian cycle HC found by Generate_Cycle; (3) the set of initial locations of the robots on the graph RI; and (4) BW, the smallest unit of time by which we measure the length of an edge, and which allows integer division of all edge lengths[2]. We define the length of a Hamiltonian cycle by len(HC).

The algorithm works as follows. First, it generates HC′ by separating the edges of the cyclic path into sub-edges, each of size BW (see Figure 3.6). Each vertex in HC' represents an optional starting point. It then sets the initial positions of the robots along the path such that the distance between them is $\frac{\text{len}(\mathsf{HC}')}{k}$. Then, it finds the assignment of robots to these locations such that the maximal distance traveled by a robot from its initial position to the assigned location is minimized. It does that by using procedure PMPM, and a subsequent check of the minimal maximal distance of all rotations of the positions along the cycle. It returns the positions yielding minimal maximal distance. We discuss these steps in detail below.

In step 3, algorithm Initialization creates optional starting points along the spanning tree path. In step 4 it arbitrarily selects a set of k starting points, one for each robot, with equal weights from one to the next. In steps 5–6, it computes the lengths of the shortest paths from each robot's current location to all other vertices in HC′. These lengths are used later on to weight the bi-partite graph used to compute the minimal time for robots to take their place.

[2]This allows measurement at low or high temporal resolutions. However, generally, BW equals 1.

Algorithm 2 Initialization(G,HC,RI,BW)

1: $L \leftarrow \emptyset$ {output optimal match}
2: $min \leftarrow \infty$ {minimal match weight}
3: $\mathsf{HC}' \leftarrow$ separation of HC by BW.
4: $VL \leftarrow k$ vertices from HC' with distance of $\frac{\text{len}(\mathsf{HC}')}{k}$ between consecutive vertices along HC'
5: **for all** robots r **do**
6: Compute shortest path from r's position to all vertices in HC'
7: **for** $i \leftarrow 1$ to $\frac{\text{len}(\mathsf{HC}')}{k}$ **do**
8: Let BG be a full bipartite graph of the two sets RI and VL {the weights will be based on the above Dijkstra calculation}
9: $\langle ML, \text{MatchValue} \rangle \leftarrow \mathsf{PMPM}(BG)$
10: **if** MatchValue $<$ min **then**
11: $L \leftarrow ML$
12: $min \leftarrow$ MatchValue
13: $VL \leftarrow VL \oplus 1$ {update VL, see discussion}
14: return L

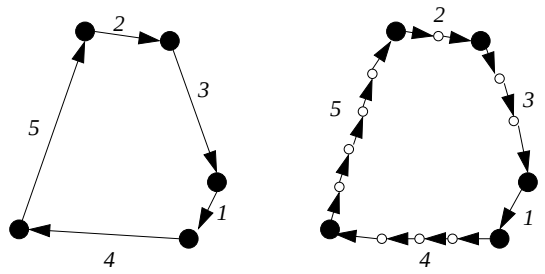


Figure 3.6: On the left: The basic Hamiltonian cycle HC. On the right: The separated Hamiltonian cycle HC' (in this example $BW = 1$)

In steps 7–13, algorithm Initialization goes through all $\frac{\text{len}(\mathsf{HC}')}{k}$ possible configurations of k starting points that are evenly spaced in HC'. For each configuration (beginning with the first configuration set in step 4), the algorithm creates a bipartite graph with k nodes on one partition (signifying the current locations of the robots in RI, and the k possible target points in VL in the other. The edges connecting the two partitions are weighted based on the lengths of the shortest paths computed in steps 5–

6. The algorithm calls algorithm PMPM (Algorithm 3), which receives the weighted bipartite graph BG. Procedure PMPM returns a tuple: A set ML of optimal possible match considering BG, and the value the match, MatchValue. An optimal match is one in which the maximal weight of an edge in the bipartite graph is minimal, over all possible permutations of maximal edges. MatchValue is the weight of this maximal edge. The result is checked to see if it improves the current best match (steps 10–12). Then the next configuration of k points is selected in step 13. The notation $\oplus+1$ is used here to denote the operation of updating the set VL such that each original vertex v in VL is replaced by the new vertex u, where the edge (v,u) exists in HC$'$. Note that because HC' is a circle, exactly one such edge must necessarily exist.

Procedure PMPM (Algorithm 3) uses the *Hungarian algorithm* [50] which finds a match in bipartite graphs with minimal sum of edges. As illustrated in Figure 3.7, the Hungarian algorithm finds a match between r_1 and d_1 and between r_2 and d_2 since this is the minimal match sum. But in our application we would like to find the minimal largest edge from all the possible permutations. In this example, we want to match r_1 to d_2 and r_2 to d_1. In this match the maximal edge weight is 9 while in the previous it is 10. Steps $5-7$ in the PMPM algorithm construct BG' from BG. The BG' graph, by construction, causes the *Hungarian algorithm* call (in step 8) to prefer the permutation in which the maximal edge is minimal.

Algorithm 3 Procedure PMPM(BG)

1: $ML \leftarrow \emptyset$
2: Let V be BG vertices
3: Sort the edges in BG
4: $w \leftarrow 1$
5: **for** each edge e in BG by nondecreasing weight **do**
6: add it to BG' with weight w
7: $w \leftarrow w * \frac{|V|}{2}$ (see Figure 3.7)
8: Run the Hungarian Algorithm(BG', ML)
9: Let m be the largest edge weight from the match ML in the corresponding BG edges
10: Return $\langle ML, m \rangle$

Lemma 4. *The construction of BG' in step 3 of* PMPM *assures that the Hungarian*

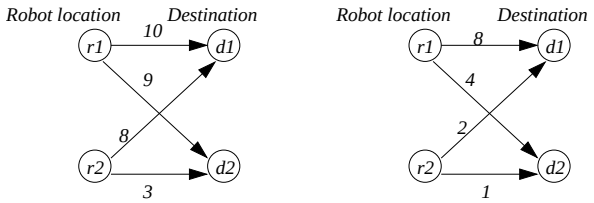


Figure 3.7: Basic bipartite graph, and bipartite graph after conversion.

algorithm returns a match with minimal maximal edge in BG.

Proof. First, we prove that the construction assures the selection of a match with minimal maximal edge in BG'. For that, assume, towards contradiction, that the Hungarian algorithm returns a minimal match with sum of edges M and maximal edge m but there exists another match with sum of edges M' that has a maximal edge m' such that $m > m'$. By the construction of BG' in step 3 of PMPM it follows that any edge in BG' is greater than the sum of all edges smaller than it. Specifically, by our assumption that $m > m'$ it follows that $m > M' \Rightarrow M > M'$ contradicting the minimality of M returned by the Hungarian algorithm.

It is left to show that the match found on BG' yields a match on BG with minimal maximal edge as well. This follows directly from the fact that the order of edges remains through construction, hence minimal maximal edge in BG transforms to the minimal maximal edge in BG', and back. □

The time complexity of Procedure PMPM is as the Hungarian algorithm [50] k^3 and Algorithm Initialization runs it $\frac{|V|}{k}$ times. It also run Dijkstra's shortest path algorithm k times. Thus the overall run-time complexity of Initialization is $O(KV(K+\log V)$.

3.4 Performance of STP

In this section we evaluate the performance of the patrol algorithm that is based on procedures Generate_Cycle (Algorithm 1) and Initialization (Algorithm 2). We first examine their combined performance according to the frequency optimization criteria described in Section 3.1, and then discuss their robustness to robot death failures.

3.4.1 Point-Visit Frequency

We prove that the combination of Procedures Generate_Cycle and Initialization guarantees optimality in *all* three point-visit frequency criteria.

Theorem 5. *The patrol algorithm which is derived by the combination of Procedures* Generate_Cycle *and* Initialization *guarantees:* a. *Perfectly uniform frequency* b. *Maximal average frequency* c. *Optimal under-bounded frequency.*

Proof. a. The first part of the Min_Path_Match problem requires the robots to be placed initially, i.e., before they begin their patrol, in uniform distance along the cyclic path. This requirement is clearly fulfilled by step 4 in Procedure Initialization, where the only positions considered are the ones where all robots are equidistant along the cyclic path. Since the robots are homogeneous and all target areas are covered by the cyclic path, their movement along the cyclic path yields a uniform frequency of $\frac{1}{\text{len}(\mathsf{HC})/k}$, where k is the number of robots and $\text{len}(\mathsf{HC})$ is the total length of the minimal HC found by Generate_Cycle. Since all points are visited in this same frequency, the standard deviation is 0, and the point-visit frequency is perfectly uniform.

b. and c. The cyclic path found by Generate_Cycle was proven by Lemma 3 to be minimal. Therefore one robot traveling along this cycle has maximal frequency of $\frac{1}{\text{len}(\mathsf{HC})}$, hence the maximal possible frequency by k robots is $k \times \frac{1}{\text{len}(\mathsf{HC})}$, which is exactly the frequency guaranteed by our algorithm. All targets are monitored, then, exactly once every $\frac{\text{len}(\mathsf{HC})}{k}$ cycles, and by that optimal under-bounded frequency is guaranteed. Since we have proven uniform and under-bounded frequency, maximal average frequency is straightforward. □

3.4.2 Guaranteeing robustness

We use the circular path not only for assuring uniform frequency while patrolling, but for robustness as well. Specifically, we refer to robustness in the sense that as long as at least one robot remains intact, the patrol mission is guaranteed to be performed successfully (with all three optimality criteria maintained). This notion of robustness, and the basis for its existence in circular paths, has been discussed also in multi-robot coverage work [44].

Robustness is clearly guaranteed: If one robot fails the other robots simply divide the circular path again between them by re-running Procedure Initialization. Theorem 5 is, then, guaranteed for the new number of robots. In this statement we have a hidden assumption that the system is stable in the sense that the uniform, maximal-average, optimal under-bounded frequency is guaranteed as long as the system performs properly, and if a failure occurs it again guarantees the above properties after a short reorganization time. This reorganization time is the period of time necessary for the robots to execute the algorithm and arrive at their new initial positions. If all robots are to move along the cyclic path following the current direction, then this period of time will not exceed $\frac{\text{len}(\mathsf{HC})}{6}$ (see Lemma 6 for the proof). If the system is unstable, i.e., robots fail one after the other, then Theorem 5 is guaranteed for the final number of robots after stabilization.

Lemma 6. *The reorganization time required when decreasing the number of robots from k to $k-1$ is at worst the time required to travel the distance $\frac{\text{len}(\mathsf{HC})}{6}$ if robots follow the circular path on their way towards their new initial positions.*

Proof. Consider the case in which there are three robots, and one fails. The length of the HC is divided by the three robots prior to the failure, and is divided by two after the failure. Therefore, if only one robot travels along the path, it has to travel from $\text{len}(\mathsf{HC})/3$ to $\text{len}(\mathsf{HC})/2$, which is exactly $\text{len}(\mathsf{HC})/6$. For any other k, the distance traveled is smaller: $\frac{\text{len}(\mathsf{HC})}{k-1} - \frac{\text{len}(\mathsf{HC})}{k} < \frac{\text{len}(\mathsf{HC})}{2} - \frac{\text{len}(\mathsf{HC})}{3}$ for any $k > 2$. Clearly, for $k = 2$ the remaining robot has no reorganization phase, as it simply patrols along the circular path alone. □

Chapter 4

Handling Events in Area Patrolling

In environments in which a team of robots is required to continuously visit target locations for various missions, it is likely that the team will need to handle events. We define an event as a location in the environment that requires special treatment by the robots for a limited period of time. For example, if the robots patrol in an area in order to clean it, then an event could be cleaning a broken glass (in addition to the regular cleaning duties). These events are transient additions to the cost of traveling through one or more a cells in the grid. Because of their transient nature, it is highly important that they will not effect the system in the following sense. (i), the frequency criteria should be maintained as much as possible for the entire area during the time the event is handled; and (ii), the *recovery time*, i.e., the time it takes to stabilize the system to the situation in which the robots are dispersed with equivalent distance (in time) around the HC should be minimal.

Definition: An *event* is a tuple $\langle l_i, t_i, T_i \rangle$, where l_i is the location in which the event occurs; t_i is the time that a robot needs to attend and handle the event (duration of handling), and T_i is the deadline for handling the event. We assume that an event can occur only in one cell, otherwise it is handled separately for each cell.

The key idea in the algorithm is to divide the time it takes to handle event e_i (t_i) uniformly between all the robots. Therefore the total time invested by each robot in handling the event is $\frac{t_i}{k}$. However, if each robot will spend $\frac{t_i}{k}$ the first time it reaches l_i, then it will starve the other cells along the cycle HC. Therefore, handling the event could be done along a number of rounds, depending on the overall time allocated for

handling the event (T_i).

When an event occurs, the first step is to check if the system can handle it. Let $\mathsf{Feasible}(e_i)$ be a Boolean variable that represents the possibility of the system to handle event e_i, and let d_{min} be the shortest distance (in time) between any robot from the team to the event's location (l_i). Therefore,

$$\mathsf{Feasible}(e_i) = \begin{cases} \mathsf{True} & \text{if } d_{min} + t_i \leq T_i \\ \mathsf{False} & \text{otherwise} \end{cases} \tag{4.1}$$

In other words, if the minimal time it takes the robot closest to l_i to arrive to the event plus the time it takes to handle the event is greater than the time restriction on this event, then the system will fail to handle it.

As mentioned above, the main goal of our algorithm is to keep handling the frequency criteria of other points in the area by dividing the event between all the robots, and adding some extra cost to the cost of traveling along the HC in that location. However, there are cases in which the structure of the robots that travel along the HC is broken, i.e., one of the robots has to leave its course and travel directly to l_i. Denote the minimal distance (in time) between the location of the event (l_i) and the robot next to arrive at l_i *on its path* along the HC by d_{Nmin}. Note that always $d_{min} \leq d_{Nmin}$. Therefore we define the Boolean variable $\mathsf{NoBreak}$ that represents the possibility of handling the event while maintaining the movement of all robots along the original patrol path (the HC). $\mathsf{NoBreak}$ is defined as follows.

$$\mathsf{NoBreak}(e_i) = \begin{cases} \mathsf{True} & \text{if } d_{Nmin} + t_i \leq T_i \\ \mathsf{False} & \text{otherwise} \end{cases} \tag{4.2}$$

The algorithm $\mathsf{CooperateEvent}$ (Algorithm 4) deals with an event e_i in case $\mathsf{NoBreak}(e_i) = \mathsf{True}$. The algorithm $\mathsf{IsolatedEvent}$ (Algorithm 6) deals with cases in which $\mathsf{NoBreak}(e_i) = \mathsf{False}$, yet $\mathsf{Feasible}(e_i) = \mathsf{True}$.

We define the success criteria for handling an event e_i as follows (priority in descending order):

a. t_i time units were invested in handling the event jointly by the team members within T_i time units, i.e., the event is handled on time.

b. Recovery time is minimized.

c. Frequency criteria is maintained throughout the patrolled area.

4.1 Handling events along the patrol path

In this section, we describe Procedures CooperateEvent and SingleRoundEvent that handles the simplest events from the robot team's perspective. These events do not require any of the robots to divert from their patrol path, and can be handled co-operatively between the robots (all of them or only part of them). This is possible for all events e_i where T_i is large enough (the event should not be handled urgently), i.e., if $\mathsf{NoBreak}(e_i) = \mathsf{True}$.

We divide this case into two sub-cases. In the first, T_i is large enough to allow t_i to be divided between the k robots. Here, Procedures CooperateEvent is executed. If this is not the case, then Procedure SingleRoundEvent is called. We first describe Procedure CooperateEvent and prove how it achieves the successful event handling criteria, and then turn to discuss Procedure SingleRoundEvent and its characteristics.

4.1.1 Procedure CooperateEvent

The key idea in algorithm CooperateEvent (Algorithm 4) is to divide the handling time of an event e_i (t_i) between all robots. First, the procedure will calculate the number of cycles along the HC that the robots will patrol while handling e_i, denoted by r_i. Then, it will find the amount of time each robot should attend the event during each cycle, denoted by x_i. Denote the case in which the event cannot be divided between all the robots by the Boolean variable NoDivision. The following formula describes the condition in which this case exists.

$$\mathsf{NoDivision}(e_i) = \begin{cases} \mathsf{True} & \text{if } \mathsf{HC}(1-\frac{1}{k}) + d_{Nmin} + t_i > T_i \\ \mathsf{False} & \text{otherwise} \end{cases} \tag{4.3}$$

$\mathsf{NoDivision}(e_i)$ is true in case e_i cannot be divided between *all* the robots, and there is less than a single round to handle the event. In this case procedure SingleRoundEvent will be called.

For a given event e_i, if $\mathsf{NoDivision}(e_i)$ is false, then Procedure CooperateEvent will divide the handling time of the event (t_i) between all the robots. The procedure gives all the robots the same amount of time x_i to handle the event for each round along all r_i rounds.

Since r_i represents the number of rounds, it should be an integer. Therefore if the optimal solution requires a fraction of a round, then the number is rounded to the first integer below, and the number of time units invested by each robot during the cycles increases. This is done in order to maintain a fair division of t_i between the robots, which also leads to a recovery time of zero from handling e_i.

In case $x_i > \frac{\mathsf{HC}}{k}$, i.e., the time that each robot should invest in handling the event is greater than the distance between consecutive robots along the cycle, then a neighbor robot will arrive to the event location that currently handled by a robot. In this case, the period of time the robots will handle the event per round will be $\frac{\mathsf{HC}}{k}$ and not as calculated at the beginning of the procedure. The number of rounds will also change (see procedure CooperateEvent).

Algorithm 4 CooperateEvent$(e_i < l_i, t_i, T_i >, k, HC)$

1: $r_i \leftarrow \lfloor \frac{T_i - t_i}{\mathsf{HC}} \rfloor$
2: $x_i \leftarrow \frac{t_i}{r_i k}$
3: **if** $x_i \leq \frac{\mathsf{HC}}{k}$ **then**
4: **for** $j \leftarrow 1$ to r_i, for each robot R_i **do**
5: Move along HC until arrive at location l_i.
6: Handle event for x_i time units.
7: **else**
8: $visitCounter \leftarrow r_i x_i$
9: **while** $visitCounter > 0$, for each robot R_i **do**
10: Move along HC until arriving at location l_i.
11: $handle \leftarrow \min(\frac{\mathsf{HC}}{k}, visitCounter)$
12: Handle event for $handle$ time frame.
13: $visitCounter \leftarrow visitCounter - handle$

Procedure CooperateEvent first finds the value of r_i and x_i. Then (in line 6) it checks whether x_i is less (or equal) than the time it takes to neighbor robot (along the HC) to arrive the event location. If so, then the robot handles the event for x_i time units and proceeds its area patrol along the HC. This happens iteratively for r_i rounds. When the time x_i of handling event is greater than the time it takes a neighbor robot to arrive the event location, then at each round a robot will handle the event without interfering it neighbor robot, i.e., $\frac{\mathsf{HC}}{k}$ time units (the distance in time between two

robots along the HC path). Now, the number of rounds will change to $r_i = \frac{t_i}{\mathsf{HC}_k}$.

Lemma 7. *For an event e_i, if* NoDivision(e_i) = False*, then* CooperateEvent *(Algorithm 4) guarantees that e_i will be handled within T_i time units, i.e., on time.*

Proof. Since NoDivision(e_i) is false, then the robots can share the event and handle it during at least one round. If $x_i \leq \frac{\mathsf{HC}}{k}$ (the distance between two consecutive robots along the patrol path), then according the assignment to x_i by CooperateEvent , the robots will finish handling the event on time. If $x_i > \frac{\mathsf{HC}}{k}$ the algorithm changes x_i value to be exactly $\frac{\mathsf{HC}}{k}$. Now, a robot handles the event for a duration of $\frac{\mathsf{HC}}{k}$ per round, and the time that the event is handled per round (by all the robots) is HC: A robot handles the event, then its neighbor replaces it, and handles the event and so on. That means that the event is always being handled, for a total duration of $t_i + d_{Nmin}$. Where d_{Nmin} is the time it took to the closest robot (along the HC path) to arrive the event location and t_i is the time that should invest the event. Since NoDivision(e_i) is false, we can be sure that $t_i + d_{Nmin} \leq T_i$ which says that the robot will finish handling the event on time. □

Recall that we defined the *recovery time* from handling an event e_i as the time it takes to stabilize the system to the situation in which the robots are dispersed with equivalent distance (in time) around the HC after handling e_i. The following Lemma 8 ensures that using procedure CooperateEvent, the recovery time of the system will be zero.

Lemma 8. *The recovery time from handling event e_i using Procedure* CooperateEvent *is zero.*

Proof. Since t_i is divided equally between all the robots, each robot invests the same period of time on handling e_i. Moreover, each robot handles the event for exactly the same amount of time during each round. This ensures that all robots will finish handling the event during the same round. Thus the robots will be again with the same distance (in time) between them after the last robot finishes handling the event. □

Lemma 9. *Procedure* CooperateEvent *guarantees minimal interference to the frequency of visits to other cells other than l_i, under the restriction that the recovery time is zero.*

Proof. Assume, towards contradiction, that there exists a Procedure $\mathscr{B}$ different than CooperateEvent that handles the event on time and with recovery time zero, yet the robots visit a cell $c \neq l_i$ with higher frequency during the handling time of e_i. Denote by x_i^B the number of time units $\mathscr{B}$ instructs the robots to invest in handling the event, in each round. If $x_i^B > x_i$, then necessarily the frequency of visiting c decreases. Therefore $x_i^B \leq x_i$. Denote the number of rounds during which e_i is handled according to $\mathscr{B}$ by r_i^B. If $x_i^B < x_i$, then necessarily $r_i^B > r_i$. However, $r_i \leftarrow \lfloor \frac{T_i - t_i}{\mathsf{HC}} \rfloor$, which gives the maximal number of rounds possible to complete handling e_i within T_i time units if we assume all robots handle e_i uniformly. Hence if $x_i^B < x_i$, then handling e_i by $\mathscr{B}$ exceeds T_i, contradicting the assumption that $\mathscr{B}$ handles e_i on time. If $\mathscr{B}$ instructs some robots to work more than the others (for example in the last round), then the recovery time is not zero, again leading to a contradiction. Therefore $x_i = x_i^B$, i.e., $\mathscr{B} =$ CooperateEvent, leading again to a contradiction. □

By combining Lemmas 8, 7 and 9, it follows that Procedure CooperateEvent handles event e_i successfully. Therefore we turn to consider cases in which Procedure CooperateEvent cannot be used, i.e., if T_i is to small to allow cooperation between the robots.

4.1.2 Procedure SingleRoundEvent

The best possible case for the team of robots is if T_i is large enough to permit the division of t_i uniformly between the robots. However, this is not always possible. Therefore if $\mathsf{NoBreak}(e_i) = \mathsf{True}$ yet $\mathsf{Feasible} = \mathsf{True}$, then procedure SingleRoundEvent will be activated. Since $\mathsf{NoBreak}(e_i) = \mathsf{True}$, at least one robot that patrols along the HC can handle the event. If $\mathsf{NoDivision}(e_i)$ is true, then not all robots can share the event as described in CooperateEvent procedure. Procedure SingleRoundEvent handles this situation.

$$d_{Nmin} + \frac{\mathsf{HC}}{k} n_i \leq T_i - t_i \tag{4.4}$$

$$n_i = \lfloor \frac{k(T_i - t_i - d_{Nmin})}{\mathsf{HC}} \rfloor \tag{4.5}$$

In order to achieve the good frequency performance, procedure

SingleRoundEvent (Algorithm 5) needs to share the event handling by maximal number of robots. Let n_i be the number of neighbor robots that will help the closest robot to the event location along the HC path to handle the event. Since NoBreak(e_i) is true, then also $d_{Nmin} \leq T_i - t_i$ (where $T_i - t_i$ is the slack time available), allowing the event can be abandoned from the moment it occurs until its deadline. d_{Nmin} is the time it takes the first robot (the closet one along the HC) to arrive the event location - while traveling along the cycle. We would like to find the number of neighbor robots n_i that can assist the event handling. This is formulated in Equation 4.4 and the goal is to find the maximal integer n_i number which is shown in Equation 4.5. Now, the number of robots that will share the event will be $n_i + 1$ (the closest robot plus the n_i neighbors). The amount of time that each of the $n_i + 1$ robots will devote to the event will be $\frac{t_i}{n_i+1}$. If this calculated time is greater than $\frac{HC}{k}$ then it will be adjusted to be exactly $\frac{HC}{k}$ and the number of robot that will handle the event will grow to be $\frac{t_i k}{HC}$ as shown in procedure SingleRoundEvent

Algorithm 5 SingleRoundEvent$(e_i < l_i, t_i, T_i >, k, HC)$

1: $n_i \leftarrow \lfloor \frac{k(T_i - t_i - d_{Nmin})}{HC} \rfloor$
2: $r_i \leftarrow n_i + 1$
3: $x_i \leftarrow \frac{t_i}{r_i}$
4: **if** $x_i > \frac{HC}{k}$ **then**
5: $\quad x_i \leftarrow \frac{HC}{k}$
6: $\quad r_i \leftarrow \frac{t_i k}{HC}$
7: **for all** r_i closest robots along HC (on the selected direction) **do**
8: $\quad$ patrol along HC until arriving at location l_i.
9: $\quad$ handle event for the duration of x_i.

Algorithm SingleRoundEvent ensures that the robots will not bump into each other, since the robots that handle the event will do so only in their path along the HC. They will not bump into their neighbor since the distance (in time) between neighbor is $\frac{HC}{k}$ which is also the limit time to handle an event for a robot. The procedure also ensures that each point in the area will be patrolled optimally as before the event occurred, but with maximal delay of $\frac{HC}{k}$ from the time they finish handling the event, as proven in the following lemma.

Lemma 10. *The maximal recovery time from event e_i using Procedure* SingleRoundEvent *is* $\frac{\mathsf{HC}}{k}$.

Proof. Let x_i be the time that a robot handles an event e_i using procedure SingleRoundEvent. The robots that share the event are in distance (in time) x_i from the place they should be if the event did not occur. In order to recover from the event (when it is finished) all the robots that share the event should move x_i while the other robots should stop their movement before they could patrol again. In procedure SingleRoundEvent the maximal time that robot can handle event is $\frac{\mathsf{HC}}{k}$ thus the maximal recovery time is $\frac{\mathsf{HC}}{k}$. □

4.2 Handling events outside of the patrol path

Procedures SingleRoundEvent and CooperateEvent will be executed when the robots can attend the event while still patrolling along the HC path. In case where this situation is not possible, i.e., NoBreak = False, yet Feasible = True, then a robot can handle the event by breaking out of the HC path. In this case algorithm IsolatedEvent (Algorithm 6) is executed.

Algorithm IsolatedEvent finds the preferred robot to send to handle the event. This robot will move to event location, handle it and proceed the patrol when its neighbor along the HC will arrive. The neighbor will now handle the event until its neighbor arrives and so on. This procedure finishes when the event was handled for t_i time units, i.e., finished completely.

IsolatedEvent finds the preferred robot that should handle the event by calculating the minimal time it takes a robot to arrive at the event location by crossing the HC. If there are few robots with the same minimal time to arrive at the event location, the chosen robot will be the one whose shortest path is as much as possible along its original path of the HC.

The procedure IsolatedEvent ensures that the robots will not bump into each other when they are attending to the event. Where they are moving along the HC they cannot meet since they are in different locations. The only way that two robots can meet is when one of the robots crosses the HC to the event's location, but as shown in Lemma 11, this situation is impossible.

Algorithm 6 Procedure IsolatedEvent$(e_i < l_i, t_i, T_i >, k, HC)$

1: $S \leftarrow \emptyset$
2: **for all** robots r **do**
3: $d \leftarrow$ the shortest path from the current robot location to event location l_i
4: $S \leftarrow S \cup \{\langle d, r \rangle\}$ {add the distance and associated robot to S}
5: $D \leftarrow \min_d(S)$ {all tuples with minimum distance}
6: **if** $\mid D \mid = 1$ **then**
7: The corresponding robot r will move on its (calculated) shorted path to l_i
8: **else**
9: Choose from D the corresponding robot that has the longest path to move on its HC path.
10: This robot will move to l_i
11: **repeat**
12: The robot in l_i handles the event until replaced or t_i reached.
13: **until** event was handled t_i

Lemma 11. *The robots will not bump into each other when the selected robot crosses the* HC *path to attend an event e_i using Procedure* IsolatedEvent

Proof. Procedure IsolatedEvent computes for each robot the shortest path to the event location. Let R_a be the robot chosen to first handle the event by Procedure IsolatedEvent. We first assume that R_a has the minimal path to l_i (and shares this minimal value with no other robot). Assume, towards contradiction, that there is a robot R_b that will bump into R_a on its way to l_i, i.e., R_a and R_b lie at the same location along the HC at the same time. Therefore at that time the distance between R_a and l_i is equal to the distance between R_b and l_i, which leads to a contradiction (since R_a has minimal distance to l_i).

Assume now that there are several robots $R_a, R_b, \ldots \in D$ with the same shortest path length to l_i. If we choose arbitrarily some $R_a \in D$, then it could indeed possibly bump into some other member of D on its way to l_i. Thus, procedure IsolatedEvent selects the robot whose shortest path (with respect to the other robots in D) is mostly a subset of its regular path along the HC. This ensures that this robot will never bump any robot while it crosses the HC. □

Lemma 11 and algorithm IsolatedEvent deal with the robot that is sent to handle the event. However, a question remains as to what the robots do once the event is handled, to stabilize the system.

Algorithm IsolatedRecovery (Algorithm 7) describes the recovery stage, executed once the event handling procedure IsolatedEvent is triggered. Note that once a robot moves outside of the HC then necessarily, a segment of length $\frac{\mathsf{HC}}{k}$ is left without an associated robot (Figure 4.1). We call this segment the *empty segment*. The recovery procedure moves some of the robots along the HC to re-fill this empty segment such that again one robot is assigned to each of the k segments.

The algorithm works as follows. First, it computes the distance d between the event location (and the robot currently handling it, r_0), to its nearest neighbour r_1 along the path HC. If this distance d is greater than $\frac{\mathsf{HC}}{k}$, then the empty segment is between r_0 and r_1. In this case, r_0 needs to move until it is exactly at a distance of $\frac{\mathsf{HC}}{k}$ from r_1; it needs to move $d - \frac{\mathsf{HC}}{k}$. However, if d is smaller than or equal to $\frac{\mathsf{HC}}{k}$, then this means that there are more robots between r_0 and the empty segment. All of these robots (except for r_0 should move exactly one segment (i.e., $\frac{\mathsf{HC}}{k}$), and then r_0 should move a distance of d forward along HC, to reposition itself. Then the recovery is complete.

Algorithm 7 Procedure IsolatedRecovery$(e_i < l_i, t_i, T_i >, k, HC)$

1: Let r_0 be the robot at the event location.
2: Let r_1 be r_0's nearest neighbour along the path HC.
3: Set d to be the distance between r_0 and r_1.
4: **if** $d > \frac{\mathsf{HC}}{k}$ **then**
5: move r_0 a distance of $d - \frac{\mathsf{HC}}{k}$.
6: **else**
7: Let R be the set of robots along the HC, starting with r_0, such that the distance between them is equal or smaller than $\frac{\mathsf{HC}}{k}$.
8: **for all** $r \in R$ **do**
9: **if** $r = r_0$ **then**
10: move r a distance of d.
11: **else**
12: move r a distance of $\frac{\mathsf{HC}}{k}$.

The algorithm's operation is illustrated in Figures 4.1, 4.2 and 4.3. Figure 4.1

shows the state of the system after a robot D crosses the HC path and handles the event, a decision made by IsolatedEvent. Suppose, first, some time passes, the robots move clockwise, and now handling the event is taken over by A, which completes handling the event, i.e., now recovery can commence (Figure 4.2). Since the distance d here (between A and D) is smaller than $\frac{\mathsf{HC}}{k}$, A, D, B, C are in the set R (line 7). Robots D, B, and C should move ahead a distance of $\frac{\mathsf{HC}}{k}$, while robot A should move the distance of d ending up where D used to be. Then the system is stabilized and can patrol as usual.

Now suppose that A did not finish handling the event. In this case, the empty segment continues to shift around HC, until we reach the situation in Figure 4.3. If E completes handling the event, then the distance d (between E and F) will be greater than $\frac{\mathsf{HC}}{k}$, which means that only E needs to move forward, a distance of $d - \frac{\mathsf{HC}}{k}$, to stabilize the system.

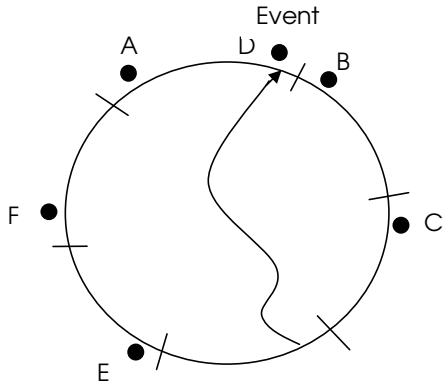


Figure 4.1: The robots patrol along the HC. An event occurred, and one robot (D) crosses the HC and handles the event.

Given an event handled by algorithms IsolatedEvent and IsolatedRecovery, we can bound the recovery times for the system, following the decision to handle an event.

Lemma 12. *The maximal recovery time from procedure* IsolatedEvent *is* $\frac{\mathsf{HC}}{k}$.

Proof. When robot attend event and cross the HC there could maximum one empty segment (as shown above). The robot that attend event is far from its neighbor (along the HC) maximal $\frac{\mathsf{HC}}{k}$ or the empty segment is the neighbor. If the empty segment is the neighbor it take maximal $\frac{\mathsf{HC}}{k}$ to arrive the relative location of the other robots in the other segments as shown above. If there is neighbor robot in the next segment, the

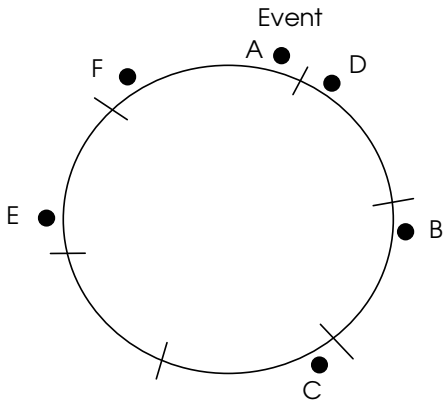


Figure 4.2: The robots' location after $\frac{\mathsf{HC}}{k}$ time units. In order to recover from this event, only robots A, D, B and C will need to move, while the others wait.

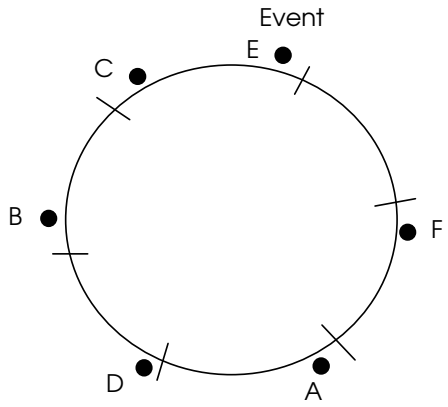


Figure 4.3: The robots' location after three additional cycles, each of duration $\frac{\mathsf{HC}}{k}$. The empty segment follows the robot that handles the event. In order to recover from the event, robot E will only move to the empty space.

maximal distance is $\frac{\mathsf{HC}}{k}$. Since the robot that behind the robot that handle the event location should keep the same $\frac{\mathsf{HC}}{k}$ distance from the robot located in the neighbor segment. Which insure that the maximal distance for recovery is $\frac{\mathsf{HC}}{k}$. □

4.3 Summary

Algorithm PatrolEvent (Algorithm 8) ties all the different conditions together, to fully address a new event and stabilize the system following its handling. The algorithm checks the values of $\mathsf{Feasible}(e_i)$ and $\mathsf{NoDivision}(e_i)$, and decides which procedure

to execute for a new event. Be aware that after running procedures IsolatedEvent and SingleRoundEvent the robots needed to be ordered again along the HC, before patrolling can be resumed completely.

Algorithm 8 PatrolEvent$(e_i < l_i, t_i, T_i >, k, HC)$

1: $D \leftarrow \emptyset$
2: **for all** robots **do**
3: Compute Dijkstra's shortest path where the source is the current robot location and the destination is the event location l_i
4: Add the distance to D
5: **if** $\min(D) + t_i > T_i$ **then**
6: return "cannot handle event"
7: **else if** $d_{Nmin} + t_i > T_i$ **then**
8: execute IsolatedEvent$(e_i < l_i, t_i, T_i >, k, HC)$
9: **else if** HC$(1 - \frac{1}{k}) + d_{Nmin} + t_i > T_i$ **then**
10: execute SingleRoundEvent$(e_i < l_i, t_i, T_i >, k, HC)$
11: **else**
12: execute CooperateEvent$(e_i < l_i, t_i, T_i >, k, HC)$

Part II

Patrolling an Open Polyline

Frequency-based patrolling—sometimes referred to as repeated coverage—is a task where points in a target work-area are repeatedly visited by robots. Frequency-based patrolling algorithms optimize various performance criteria, such as frequency of point visits, minimization of travel time to arbitrary points of interest, etc. A polyline (e.g., a two-ended fence) inherently poses challenges for existing algorithms, since no circular paths exists, and thus some points necessarily are visited more often than others. In this part of the dissertation, we propose a general technique for frequency-based patrolling in polylines.

First, in Chapter 5 we show that in general, a coordinated approach to multi-robot patrolling, where robots are equidistant from each other in travel time, outperforms uncoordinated methods. We then present FOP (Frequency-based Overlapping Patrol), a general coordinated patrolling algorithm, in which robots move back and forth along the polyline, in an synchronized manner, such that they are assigned overlapping areas of movement, in a parametrized fashion. We analyze the performance of this coordinated method in depth, with respect to different performance goals, and investigate key trade-offs.

In Chapter 6 we extend the analysis of the FOP algorithm to account for more realistic settings. In particular, we extend the analysis to account for turn durations, and errors in the robot motion velocity. These extensions change the predictions of the optimal overlap settings to be used in applications, based on the measureable physical performance of the robots in practice.

In Chapter 7 we explore the effects of events on polyline patrolling. We provide algorithms for determining which robots are to be assigned to handling events, and under what conditions.

Finally, in Chapter 8, we use the developed models to predict the independently-programmed patrolling movements of physical robots, in extensive experiments conducted in our laboratory. We show that the extended models predicts the behavior of the robots accurately, supporting the compatibility of the analytical model with actual robot performance.

Chapter 5

Frequency-Based Patrolling of Polylines

We have earlier introduced three frequency-based performance criteria [26, 27]:

- **Uniformity.** The goal is to decrease the variance between the frequencies in which each target is visited, i.e., all targets should ideally be visited with uniform frequency f.
- **Maximal average.** The goal is to increase the average frequency f in which targets are visited.
- **Maximal minimum frequency (under-bounded frequency).** The goal is to increase the minimal frequency f with which any target point is visited, such that every target is visited with frequency of at least f. In other words, all targets are monitored at least once every $1/f$ cycles.

For a single robot, perfect uniformity of point visit frequency is impossible to achieve in polyline patrolling. The fact that the polyline is not circular prevents the robot's trajectory from being continuous; at some point (at the very least, at the outer-most edge point) the robot needs to change direction. The direction change forces the robot to immediately backtrack over points in the path that it has visited only moments before, and therefore the visit frequency is non-uniform along the path.

From a more formal perspective, the argument is as follows. The basic motion for a single robot along a polyline is necessarily a monotonic movement from left to right

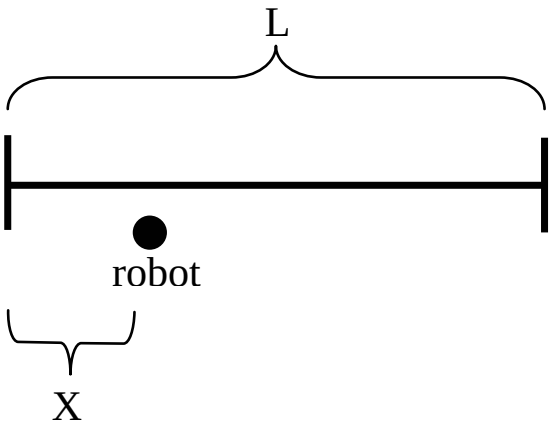


Figure 5.1: Single robot polyline patrol.

and vice versa. Figure 5.1 shows a robot at a distance X from the left edge of the polyline (the length of the polyline is L). Even assuming turning does not take any time, the times in which the point is visited form the series $2t(X), 2t(L-X), 2t(X), 2t(L-X)\ldots$ (where $t(a)$ is used here to denote the time for traveling distance a). The frequency will be uniform only in the midpoint of the polyline (when $X = L/2$), while any point towards the endpoints of the polyline will have a large frequency variance.

The addition of robots can, in principle, improve the point-visit frequency. For instance, The variance in patrolling frequency is tied to the length of the segment assigned to each robot. With more robots, there are more segments possible, and the length of each segment—and the respective time to traverse it—is shortened.

The following sections explore the use of multiple robots to improve the performance of patrolling. Section 5.1 argues for uniform spatio-temporal distribution of the robots along the polyline, and synchronizing their motions to maintain them equidistant from each other. Section 5.2 presents the generalized synchronized patrolling algorithm (FOP). Section 5.3 analytically examines this model and shows that it can improve patrolling frequency uniformity in middle segments, at the expense of edge segments, without reducing from other performance criteria.

5.1 Synchronizing Motions to Reduce Response Time

The first performance criterion that we consider in polyline patrolling is the time it takes for a robot to arrive at an arbitrary point of interest. We refer to the arrival of a robot at a target point on the polyline as *response to an event*.

For a single robot, the worst time it takes to respond to an event is the time it takes the robot to traverse the entire length of the polyline. This occurs when the robot is at one end of the polyline and an event happens at the other end.

With multiple robots, it is possible to rely on others to assist in responding to events. A simple uniform spatio-temporal distribution of the robots along the polyline would of course ensure that any event can be responded to in minimal time. If the only optimization criterion were in fact response time to events, then a fixed placement of the robots, such that they remain stationary until an event occurs, is easily computable. However, other performance criteria (e.g., maximization of point-visit frequency) exist, and thus the robots must actually visit all points along the polyline, i.e., the robots must move.

We begin by making some basic observations about the underlying characteristics of multi-robot polyline patrolling. Under the assumption that robots have a physical mass that is moving on the line segments composing the polyline, robots cannot occupy the same point on the polyline at the same time. Thus they cannot overtake each other going in the same direction, or pass by each other when going in opposite directions. However, under the assumption that the robots are homogeneous, this does not at all limit the behavior of the robots, as they can exchange roles when they meet, one simply taking over the trajectory of the other from the point of meeting.

Since the spatio-temporal trajectories of robots cannot actually intersect, assigning trajectories to different robots is a matter of spatio-temporal division of the polyline into r sub-trajectories. These are assigned to the r robots, such that the union of the paths underlying the trajectories is equal to the polyline. The challenge is in planning the sub-trajectories to optimize patrolling performance.

Theorem 13. *For r robots patrolling a polyline, the worst-case arrival time, at an arbitrary point on the polyline is minimized when the polyline is divided into r segments that have an equal traversal time t_s.*

Proof. Let T the maximal time to traverse the polyline by single robot (i.e., endpoint to

endpoint). The time to traverse the entire polyline by r robots which are distributed—equidistant in travel time—along the polyline is $\frac{T}{r} = t_s$ (each robot traverses one of r segments along the polyline once). Thus the worst time to attend an event is t_s (one endpoint of a segment to the other). Assume towards contradiction that a different distribution of the robots along the polyline would have resulted in a maximal arrival time at an arbitrary point of $t_C < t_s$. Then the total polyline traversal time for a single robot would be, at worst, $t_C \times r < t_s \times r$, i.e., $t_c \times r < T$. But this contradicts T being the time for a single robot traversal. □

We now examine two underlying approaches to patrol trajectory-planning, where the r robots are allocated segments whose traversal time is fixed, t_s. In the *non-synchronized patrol*, the robots' movements are independent of each other, and each robot patrols inside a different segment without temporally coordinating its movement with the other robots. In contrast, in the *synchronized patrol*, the temporal behavior of all robots is coordinated, such that all robots move to left or right together, maintaining fixed relative distances. We examine the worst-case behavior of these two approaches. To do this, we distinguish between *middle segments* (which have other segments on either side of them), and *edge segments*, which have segments only to one side of them. By definition, for $r > 1$, there are two edge segments, and $r-2$ middle segments.

Non-synchronized patrol suffers from high response time to events. The robots are not synchronized in their movements and each robot patrols along its assigned segment. We denote the time it takes a robot to travel along its segment by t_s. Therefore the worst time to respond to an event is t_s . This worst time occurs, for instance, when two adjacent robots are at the opposite ends of their respectively assigned segments, and the event takes place at the point which adjoins their segments. This worst case is portrayed in Figure 5.2(a). Here, Robot B is at the left end of its segment and the event occurs at the right end. Assistance from robot C will not be useful since it is the same distance from the event point. Note that the worst-case time t_s is true regardless of which segments are involved. In particular, the worst-case time is true of both middle and edge segments.

Theorem 14. *Non-synchronized patrolling has, $r+1$ points in which the arrival time is, in the worst case, t_s, where t_s is the time for traversing a segment.*

Proof. The polyline is broken into r segments, of equal traversal time. Each segment

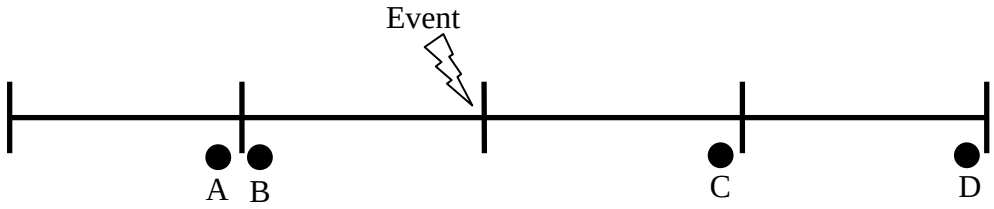
has two endpoints, left and right (where left and right are arbitrarily set). Each robot can be at most t_s time units away (the traversal time for for a whole segment) from an endpoint in its segment; this is when it is at the other endpoints. There are r segments, and therefore r right endpoints, which are t_s time units away from their corresponding segments' left endpoint. In addition, there exists one left endpoint which is *not* a right endpoint for a neighboring robot. This is the leftmost endpoint, i.e., an endpoint for the polyline. This endpoint is t_s time units away from its own right endpoint. Thus in total there are $r+1$ points whose arrival time in the worst case is t_s. □

To improve the worst-case response time of t_s, we introduce the *synchronized* patrol technique in which the robots are temporally synchronized in their movements. The polyline is divided into equal-time segments, i.e., the traversal of each segment takes the same amount of time. Each robot is then assigned a single segment, which it patrols while synchronizing its velocity with that of its peers. All robots thus move in the same direction (i.e. left to right, right to left) and maintain uniform distance (that of a segment) between a robot and its left and right neighbors. Since the distance between two robots is always the length of a segment, then the worst time it takes to arrive to an event point between two robots is $\frac{t_s}{2}$ in the *middle segments*. For the edge segments, the worst case time of t_s still applies. However, only two such segments exist.

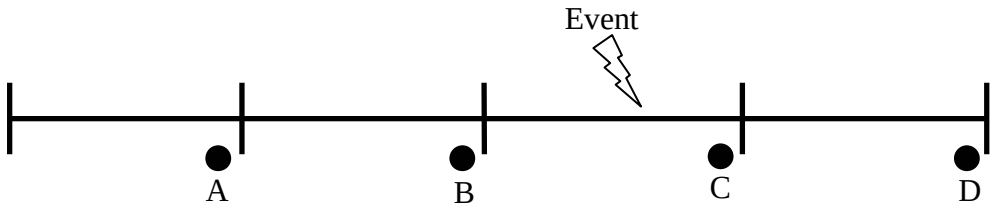
Theorem 15. *Synchronized patrolling has only two points in which the worst case arrival time is t_s, where t_s is the time for traversing a segment.*

Proof. Since all the robots keep the same traversal time t_s between their neighbors (from right and left), the time to attend a point along the polyline which is between two neighbor robots is bounded by $\frac{t_s}{2}$. The only points along the polyline which have no neighbor from right or left are the endpoints of the polyline, i.e., the leftmost endpoint and the rightmost endpoint. The time to arrive at these points in the worst case is t_s. Thus there are only two points that have a t_s worst case attending time when the robots are synchronized. □

Figure 5.2(b) shows an example of the synchronized patrol where an event occurred at a point between robots B and C. The middle segment's worst case occurs when the event point is exactly in the middle, and thus it takes $\frac{t_s}{2}$ for one robot to arrive. If the event is close to the left edge then robot B can reach it in time less than $\frac{t_s}{2}$. Similarly,



(a) Non-synchronized patrolling. Events in middle segments may be a segment-length away from the nearest robots.



(b) Synchronized patrolling. Due to the synchronization, no event in the middle segments is ever more than half a segment away. The edge segments have the same worst-case response time as in the non-synchronized method.

Figure 5.2: Worst-case robot positions with respect to an event in an middle segment, in the non-synchronized and synchronized multi-robot patrolling methods. Robots B and C are both maximally away from the event.

if the event is closer to the right edge, then robot C will handle the event (in time less than $\frac{t_s}{2}$). Note that if the event would have occurred at the outermost point of one of the edge segments, the response time would have been t_s.

5.2 Overlapping Synchronized Polyline Patrolling

Let us consider the visit time-series for such an edge point p, a point on the edge of a segment assigned to a robot B in the synchronized patrolling algorithm described earlier. Assume the time for traversing the segment is t_s. Then the point visit time series for p is $2t_s, 2t_s, \ldots$.

Imagine now that we allow the adjacent robot A to venture out into B's segment, such that instead of stopping short of point p, A will continue its movement infinitesimally, such that it also includes the point p. Now, in the synchronized patrolling method, A will visit p when B is at its farthest from it; and B will visit p when A is at its farthest. p's visit time-series is now $t_s, t_s, \ldots$.

Inspired by this observation, we propose the *synchronized-overlap* patrol method. Synchronized-overlap patrolling generalizes over the simple synchronized method introduced earlier. Here, the robots' movements are synchronized as before (all robots move to the right/left together), however more than one segment is assigned to each robot, such that the segments assigned to the robots overlap (intersect) in space. Each robot enters its neighbors' segments, depending on the size of overlap. This is a generalization of the simple synchronized patrolling method, which can be seen as a special case where there is no overlap between the segments.

For instance, Figure 1.1 shows the trajectories assigned to a fixed set of robots, for a given open polyline, using different overlap factors. The simple synchronized patrolling method corresponds to an overlap factor of 1. The others are generalized overlapping forms.

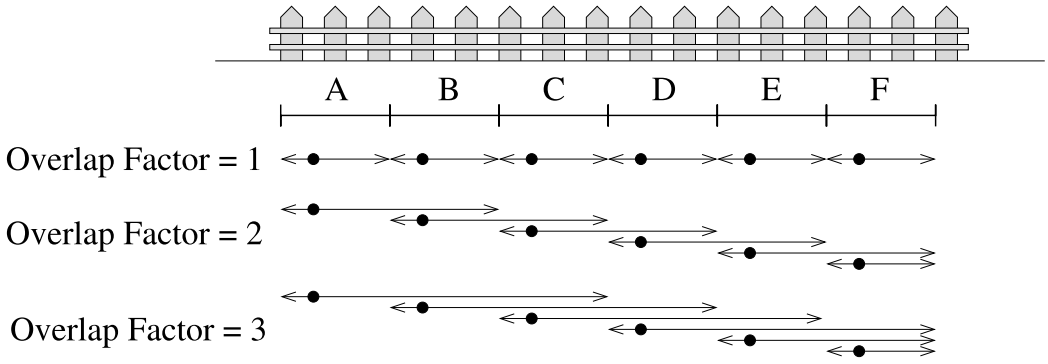


Figure 5.3: An illustration of the effect of overlap factor on patrolling behavior.

The frequency-based overlapping patrol (FOP) procedure is shown in Algorithm 9. All robots execute this algorithm in a distributed fashion, utilizing communications to synchronize their decision. The key idea in FOP is that each robot patrols more than a single segment. Each robot begins by moving along its own segment, but then, depending on the *overlap factor*, may move into adjacent segment (while the robot in this segment is moving into the next segment, etc.). Thus the robots trajectories overlap in space, but not in time.

FOP (Algorithm 9) controls the patrol movement for a robot in a fence where the overlap factor is o, robot i is initially located in segment i, and the number of segments is r (equal to the number of robots). Each robot i (of the r robots that participate in

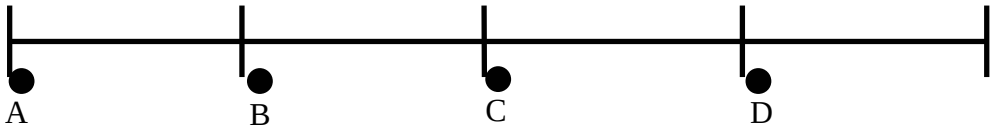
Algorithm 9 FOP(overlap factor o, robot id i, number of robots r)

1: Move min(o,$r-i+1$) segments, using the velocity constraints of each segment.
2: Turn in place and synchronize with others
3: **if** you are in the right edge segment **then**
4: wait until your left robot neighbor is one segment farther
5: Move to your base segment
6: Turn in place and synchronize with others
7: Return to step 1.

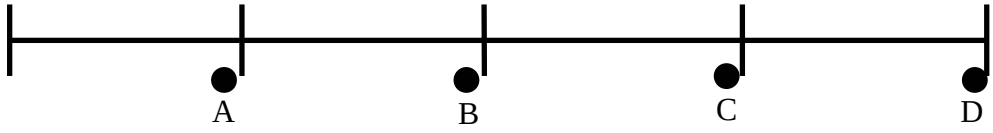
the patrol) runs this algorithm in a distributed fashion. The algorithm assumes that the open polyline has already been divided into r equal-time segments, and that all robots start at the beginning of their assigned segments, facing towards the direction of movement. Also, the algorithm assumes perfect communications (to allow the robots to synchronize their turns) and localization along the fence.

The behavior of the robots in FOP is dictated by a single parameter, the overlapping factor o. The first step of the algorithm moves the robot o segments (by the overlap factor). In case where the robot arrive to the right last segment, the $r-i+1$ value ensure that the robot will not move beyond the fence boundary (see robots C and D in Figure 5.4(b)). The second step ensures that the robots will synchronized and wait until the entire robots arrive to their destination. The third step occurs only for the robots that collect at the fence endpoint. They must wait until the other robots have left the segment. In the fourth step each robot returns to the segment that it started from. Finally, all robots turn in place, synchronized again, and repeat the process.

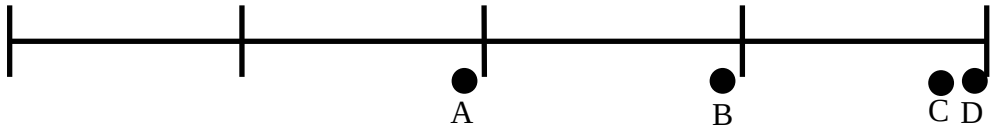
Figure 5.4 shows an illustration of the robots movements in a synchronized-overlap patrol, with an overlap factor of 3 (each robot visits 3 segments) segments. In Figure 5.4(a), the robots are in their initial locations. In 5.4(b) they have moved to the edge of their first assigned segment. In 5.4(c), the robots continued their movement into their second segment, overlapping with one other member. In 5.4(d), they reach the last of their segments, in an area overlapping with two of their neighbors'.



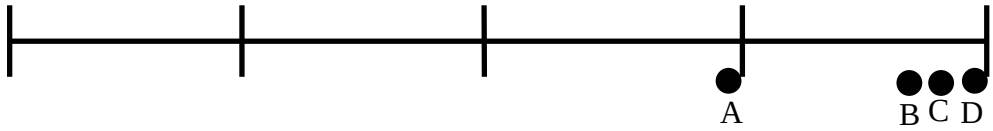
(a) Initial location.



(b) Edge of first segment.



(c) Edge of second segment, overlapping with the first segments.



(d) Edge of third segment, overlapping with the first and second segments.

Figure 5.4: An illustration of FOP running with $o = 3$ (four robots patrolling such that three are assigned for each segment).

5.3 An Analysis of Point Visit Frequency

We now turn to analyzing the synchronized-overlap patrol under a naïve motion model, in which turning—changing direction at the edges of segments—is done instantaneously. The only time influencing the frequency is therefore time spent traversing the segments. In analyzing the synchronized-overlap patrol technique under the frequency criteria, we utilize the following notations and definitions.

Definition 1 (Overlap Factor)**.** *When using the synchronized-overlap patrol technique, then the* overlap factor *is the number of segments visited by each robot. We denote this factor by o. Note that in the synchronized model, where no overlap occurs, $o = 1$. In all of the analysis, we assume the overlap factor o is a positive integer $o \in \mathbb{N}$.*

Let l be the polyline length, r the number of robots (hence the number of segments), p a point on a segment, defined by a fraction of the length of the segment, $p \in [0,1)$,

where for the left-most point, $p = 0$. We use v to denote the robots' velocity (we assume homogeneous robots, and thus the same velocity to all), and s_i the number of robots that visit a specific segment i.

Definition 2 (Edge and Middle Segments)**.** *A segment i is called an* edge segment *if $o \neq s_i$, or* middle segment*, otherwise.*

Note that edge and middle segments are *not* proper generalizations of our earlier informal definitions of the same, as an edge segment may have other segments to both sides of it. For instance, In Figure 5.3, for $o = 1$, all segments are middle segments. For $o = 2$ (second set of trajectories), segment A is covered by a single robot and is an edge segment, while the others are middle segments. For $o = 3$, segments A, B are edge segments.

For the purpose of the analytical discussion, we allow robots to occupy the same space at the endpoints of the polyline. For instance, we treat the positions of robots B, C, D in Figure 5.4 as being superimposed.

The function $time_p(l, r, p, v, o, s, n)$ calculates the time that passed between the $(n-1)$th visit to the point p, and nth visit. By minimizing this function we can improve the frequency of visits to the point p; by minimizing its variance over time, we can improve its uniformity. The function is defined as follows.

$$time_p(l, r, p, v, o, s, n) = \begin{cases} 2\frac{l}{r}\frac{p}{v} & \text{if } n \bmod 2s = 1 \\ 2(1-p)\frac{l}{rv} + 2(o-s)\frac{l}{rv} & \text{if } n \bmod 2s = s+1 \text{ or } s = 1 \\ \frac{l}{rv} & \text{otherwise} \end{cases} \tag{5.1}$$

The first condition in the formula is satisfied when the robots change direction at the left edge of a segment ($\frac{l}{r}$ is a length of a segment). The second condition is satisfied when the robots change direction at the right edge of the segment, and the third condition is satisfied in the overlapping regions.

We use the function $time_p$ to develop the time-series of visits to any point p. Based on the series, we will be able to analyze the behavior of the synchronized-overlap patrol. We do this separately for the *middle segments* (Section 5.3.1) and the *edge segments* (Section 5.3.2).

5.3.1 Middle Segments

The following cyclic series represents the time iteration of visiting a point p on a middle segment when using the synchronized-overlap patrol technique:

$$\underbrace{\frac{l}{vr},\cdots,\frac{l}{vr}}_{\text{o}-1},\frac{2l(1-p)}{vr},\underbrace{\frac{l}{vr},\cdots,\frac{l}{vr}}_{\text{o}-1},\frac{2lp}{vr},\cdots \tag{5.2}$$

In the synchronized technique $o=1$, since each robot patrols only in one segment and the robots do not overlap. Thus the time-series of visiting a point p by the synchronized technique is given in the following cyclic series, which is a special case of the above:

$$\frac{2l(1-p)}{vr},\frac{2lp}{vr},\cdots \tag{5.3}$$

Based on this series, we can analyze the theoretical behavior of the patrolling algorithms according to different point-visit frequency optimization criteria. We begin by discussing the maximum minimal frequency criteria (under-bounding frequency).

Lemma 16. *Suppose we are given a middle segment M. For all $o \in \mathbb{N}$, where $\mathbb{N}$ denotes the set of positive integers, the maximal minimum point visit frequency in M, $f_{maxmin}(o)$, is equal to $f_{maxmin}(1)$. In other words, $\forall o \in \mathbb{N}, f_{maxmin}(o) = f_{maxmin}(1)$.*

Proof. Let p be a point in M, defined as a fraction of the length of M, $p \in [0,1)$. If $0 < p \leq \frac{1}{2}$ then the longest period of time the point is left unvisited when $o=1$ and when $o>1$ is $\frac{2l(1-p)}{vr}$. If $\frac{1}{2} < p < 1$ then the longest period of time p is left unvisited when using $o=1$ and when $o>1$ is $\frac{2lp}{vr}$. Thus in both cases, the longest period of time in which p is unvisited is equal for all $o \in \mathbb{N}$. □

In other words, for a middle segment, there is no difference between the standard synchronized patrolling method, and any of its generalizations using integer overlaps. This of course pertains to the *under-bounding frequency* criterion.

We now turn to the average frequency criteria. Lemma 17 shows that a similar result holds.

Lemma 17. *Suppose we are given a middle segment M. For all $o \in \mathbb{N}$, the* average *visit frequency in M, $f_{avg}(o)$, is equal to $f_{avg}(1)$. In other words, $\forall o \in \mathbb{N}, f_{avg}(o) = f_{avg}(1)$.*

Proof. Let p be a point in M, defined as a fraction of the length of M, $p \in [0,1)$. The average frequency of visiting p using $o = 1$ is

$$(\frac{2l(1-p)}{vr} + \frac{2lp}{vr})/2 = \frac{l}{vr}.$$

The average frequency when using $o > 1$ is

$$(\frac{2l(o-1)}{vr} + \frac{2l(1-p)}{vr} + \frac{2lp}{vr})/2o = (\frac{2lo-2l+2l-2lp+2lp}{vr})/2o = \frac{l}{vr}$$

and therefore

$$f_{avg}(o) = f_{avg}(1) = \frac{l}{vr}. \tag{5.4}$$

□

Finally, we examine the visit frequency uniformity criterion. Here, we find that the use of an overlap improves the uniformity of frequency.

Lemma 18. *Suppose we are given a middle segment M. Given an overlap factor $o \in \mathbb{N}$, the standard deviation of the visit intervals to any point $p \in M$, $\sigma_p(o)$, decreases as o increases.*

Proof. Let p be a point in M, defined as a fraction of the length of M, $p \in [0,1)$. The average $\frac{l}{vr}$ is known from Eq. 5.4. The standard deviation of the point interval in p is given by:

$$\begin{aligned}
\sigma_p(o) &= \sqrt{\frac{1}{2o}(\underbrace{(\frac{l}{vr} - \frac{l}{vr})^2 + \cdots}_{\text{o}-1} + (\frac{2l(1-p)}{vr} - \frac{l}{vr})^2 + (\frac{2lp}{vr} - \frac{l}{vr})^2 + \underbrace{(\frac{l}{vr} - \frac{l}{vr})^2 + \cdots)}_{\text{o}-1}} \\
&= \sqrt{\frac{1}{2o}((\frac{2l-2lp-l}{vr})^2 + (\frac{2lp-l}{vr})^2)} && (5.5) \\
&= \sqrt{\frac{1}{2o}((\frac{l-2lp}{vr})^2 + (\frac{l-2lp}{vr})^2)} && (5.6) \\
&= (\sqrt{\frac{1}{o}})|\frac{l(1-2p)}{vr}| && (5.7)
\end{aligned}$$

Note the transition from step (5) to step (6). The transition from $(\frac{2lp-l}{vr})^2$ to $(\frac{l-2lp}{vr})^2$ relies on $(a)^2 = (-a)^2$.

Clearly, increasing o decreases $\sigma_p(o)$. □

Now let us examine the mean standard deviation along the segment M, for a given overlap factor o. It is given by:

$$\overline{\sigma_p}(o) = \frac{1}{l}\int_p \sigma_p(o).$$

Corollary 19. *In the middle segments, as the overlap between the robots increases, the frequency becomes more uniform, i.e., the mean standard deviation decreases.*

Thus for the naive motion model (instantaneous turns), as the overlap factor increases, patrolling becomes better from the perspective of visit frequency. However, this does not come without a cost, which we will see below for the edge segments.

5.3.2 Edge Segments

We will analyze the quality of the synchronized-overlap patrol approach along the edge segments. First, we clarify that as the overlap factor increases in the synchronized-overlap approach, there will be more edge segments. As we already mentioned, an edge segment is a segment where $s \neq o$. The number of edge segments is $o-1$. The synchronized approach is a private case of the synchronized-overlap approach in which $s = o = 1$, i.e., there are no edge segments in it.

Series 5.8 below shows the times of visiting a point p in the edge segments.

$$\underbrace{\frac{l}{vr},\cdots,\frac{l}{vr}}_{\text{s}-1},\frac{2l(o-s)+2l(1-p)}{vr},\underbrace{\frac{l}{vr},\cdots,\frac{l}{vr}}_{\text{s}-1},\frac{2lp}{vr},\cdots \tag{5.8}$$

Lemma 20. *Suppose we are given an edge segment E. For all $o \in \mathbb{N}$, where $o > 1$, the maximal minimum point visit frequency in E, $f_{maxmin}(o)$, is smaller than $f_{maxmin}(1)$. In other words, $\forall o > 1 \in \mathbb{N}, f_{maxmin}(o) < f_{maxmin}(1)$.*

Proof. Let p be a point in E, defined as a fraction of the length of E, $p \in [0,1)$. Based on the Series 5.8 above, we show that the longest interval for which synchronized-overlap technique ($o > 1$) neglects p is

$$\frac{2l(o-s)+2l(1-p)}{vr}$$

.

First, since $o \geq s+1$, and $1 > p \geq 0$, then

$$\frac{2l(o-s)+2l(1-p)}{vr} > \frac{2l+2l(1-p)}{vr} = \frac{4l-2lp}{vr} > \frac{2l}{vr}$$

and we know

$$\frac{2l}{vr} > \frac{l}{vr}, \frac{2l}{vr} > \frac{2lp}{vr}$$

Thus $\frac{2l(o-s)+2l(1-p)}{vr}$ is the longest interval for which p is neglected when $o > 1$.

In the case when $o = 1$, however, the worst case for the synchronized technique depends on the point p. It is

$$\frac{2l(1-p)}{vr}, \text{ if } 0 \leq p \leq \frac{1}{2}$$

and

$$\frac{2lp}{vr}, \text{ if } \frac{1}{2} < p < 1.$$

Since $s+1 \leq o$, and $1 > p \geq 0$ the following inequalities hold:

$$\begin{aligned} \frac{2lp}{vr} &< \frac{2l(o-s)+2l(1-p)}{vr} \\ \frac{2l(1-p)}{vr} &< \frac{2l(o-s)+2l(1-p)}{vr}. \end{aligned} \tag{5.9}$$

Thus the maximal duration between visits when $o > 1$ is greater than in $o = 1$, or $f_{maxmin}(o) < f_{maxmin}(1)$ for $o > 1$. □

A similar result holds for the average visit frequency. The average interval between visits is shorter when $o = 1$ than when $o > 1$.

Lemma 21. *Suppose we are given an edge segment E. For all $o \in \mathbb{N}$, where $o > 1$, the average point visit frequency in E, $f_{avg}(o)$, is smaller than $f_{avg}(1)$. In other words, $\forall o > 1 \in \mathbb{N}, f_{avg}(o) < f_{avg}(1)$.*

Proof. Based on the Series 5.8 above, we know the average time of visiting a point in the edge segment E, when $o > 1$, is

$$\frac{o}{s}\frac{l}{vr}$$

The average for the synchronized technique ($o = 1$) is

$$\frac{l}{vr}$$

Since $s+1 \leq o$ the inequality $\frac{o}{s}\frac{l}{vr} > \frac{l}{vr}$ holds. Thus the point visit interval in edge segments when $o = 1$ is shorter, and as a result, $f_{avg}(o) < f_{avg}(1)$ when $o > 1$. □

Finally, we show that in edge segments, frequency uniformity is better (the standard deviation decreases) when using the synchronized technique ($o = 1$), compared to the synchronized-overlap technique ($o > 1$).

Lemma 22. *Suppose we are given an edge segment E. For all $o \in \mathbb{N}$, where $o > 1$, the standard deviation of point visit frequency in E, $\sigma(1)$, is smaller than $\sigma(0)$, where $o > 1$. In other words, $\forall o > 1 \in \mathbb{N}, \sigma(o) > \sigma(1)$.*

Proof. Let p be a point in E, defined as a fraction of the length of E, $p \in [0,1)$. Because The standard deviation of visiting p when $o > 1$ is:

$$\sigma_{edge} = \sqrt{\frac{1}{s}}\frac{l}{vr}\sqrt{2(o-s)^2 + (1-2p)^2 + (3-4p)(o-s) + o(1-\frac{1}{s})}.$$

Since $0 \leq p < 1$ and $o \geq s+1$, it follows that

$$\sigma_{edge} > \sqrt{\frac{1}{s}} \mid \frac{l(1-2p)}{vr} \mid$$

To see this, we show that

$$2(o-s)^2 + (3-4p)(o-s) + o(1-\frac{1}{s}) > 0.$$

First, since o and s are positive integer numbers,

$$o(1-\frac{1}{s}) \geq 0.$$

And since $p \in [0,1)$, it follows that

$$2(o-s)^2 + (3-4p)(o-s) \geq 2(o-s)^2 - (o-s).$$

Finally, since $o \geq s+1$,

$$2(o-s)^2 - (o-s) > 0.$$

Thus $2(o-s)^2 + (3-4p)(o-s) > 0$.

But, since $o \geq s+1$, it follows that

$$\sqrt{\frac{1}{s}} \mid \frac{l(1-2p)}{vr} \mid > \sqrt{\frac{1}{o}} \mid \frac{l(1-2p)}{vr} \mid \Longrightarrow \sigma_{edge} > \sigma_{mid}$$

where σ_{mid} is the standard deviation of the synchronized technique with $o = 1$ in both edge and middle segments. □

Corollary 23. *As overlap increases, the frequency criteria become worse in the edge segments. The under-bounding frequency, average frequency, and frequency uniformity all decrease.*

By Corollary 19 we can see that there is advantage on increasing the overlap in the patrol. The advantage affected only in middle segments. However, Corollary 23 shows that there is a disadvantage of increasing the overlap in edge segments. For relatively long fences, where the number of middle segments is significantly higher than edge segments, this trade-off might be beneficial. This may especially be true if the edge segments (who are positioned at the extremities of the polyline) can be patrolled or monitored via some other means.

Chapter 6

Realistic-Motion Polyline Patrolling

Most realistic robots cannot turn instantaneously. Turning around requires time. In this section, we consider a more realistic motion model, and analyze its impact on the frequency of the visited points. In particular, not only does turning time influence the visit frequency directly, but it also becomes an important factor. The greater the overlap, the less turns are taken. We address turns in Section 6.1. In addition, real robots have velocity errors, due to interruptions and small corrections, which result in different distances being traveled. We address this additional constraint in Section 6.2.

6.1 Handling Turning Durations

We extend the $time_p$ function (Equation 5.1) to support arbitrary turning times. We denote the time it takes the robots to turn as t. Note that we still assume homogeneous robots, and thus all robots turn at the same velocity. The revised formula is shown in Equation 6.1 below:

$$time_p(l,r,p,v,o,s,n,t) = \begin{cases} 2\frac{l}{r}\frac{p}{v} + t & \text{if } n \bmod 2s = 1 \\ 2(1-p)\frac{l}{rv} + 2(o-s)\frac{l}{rv} + t & \text{if } n \bmod 2s = s+1 \text{ or } s = 1 \\ \frac{l}{rv} & \text{otherwise} \end{cases} \tag{6.1}$$

We analyze the performance of the patrol techniques (synchronized and synchronized-overlap) using the more realistic motion model in Section 6.1.1 (middle segments) and Section 6.1.2 (edge segments).

6.1.1 Middle Segments

The cyclic series 6.2 shows the times of visiting a point p in a middle segment, as computed by formula 6.1, for the case of the synchronized-overlap technique ($o > 1$). As before, p is given as a fraction of the size of the segment, $0 \leq p < 1$. We can easily see that it identical to series 5.2 but t is added to account for the robots' turning times.

$$\underbrace{\frac{l}{vr}, \cdots, \frac{l}{vr}}_{\text{o}-1}, \frac{2l(1-p)}{vr} + t, \underbrace{\frac{l}{vr}, \cdots, \frac{l}{vr}}_{\text{o}-1}, \frac{2lp}{vr} + t, \ldots \tag{6.2}$$

The cyclic series 6.3 shows the point visit time-series generated by the synchronized technique. It similar to series 6.2, but the difference is in the overlap factor o. Since there is no overlap (in synchronized) and $o = 1$.

$$\frac{2l(1-p)}{vr} + t, \frac{2lp}{vr} + t, \ldots \tag{6.3}$$

Lemma 24. *There is no difference between the synchronized and synchronized-overlap patrol techniques in terms of the under-bounding frequency criterion in the middle segments.*

Proof. Let p be a point in a middle segment, expressed as a fraction of the segment's length $p \in [0,1)$. If $0 \leq p \leq \frac{1}{2}$ then the worst duration of neglecting p in the synchronized and synchronized-overlap techniques is $\frac{2l(1-p)}{vr} + t$. If $1 > p > \frac{1}{2}$ then the worst time to abandon point by the synchronized and synchronized-overlap techniques is $\frac{2lp}{vr} + t$. Thus in both cases the maximal minimum frequency is the same. □

Lemma 25. *The synchronized-overlap patrol techniques get better results (the average time iteration is lower) than the synchronized technique in the average frequency criterion in middle segments.*

Proof. The average frequency of visiting a point using the synchronized technique is $\frac{l}{vr} + t$ (by averaging the series 6.3). The average of the point visit frequency using synchronized-overlap is $\frac{l}{vr} + \frac{t}{o}$ (by averaging series 6.2). Since in synchronized-overlap technique $o > 1 \Longrightarrow \frac{l}{vr} + t > \frac{l}{vr} + \frac{t}{o}$ □

Lemma 26. *Let p be a point in a middle segment, expressed as a fraction of the segment's length $p \in [0,1)$. The frequency of the synchronized-overlap patrol technique is more uniform than the synchronized patrol when $t < (\sqrt{o})|\frac{l(1-2p)}{vr}|$.*

Proof. The standard deviation of the frequency in a middle segment is

$$\sqrt{(o-1)(\frac{t}{o})^2+\frac{1}{o}(\frac{l-2lp}{vr})^2}.$$

Since in the synchronized technique $o=1$, then its standard deviation is $|\ \frac{l(1-2p)}{vr}\ |$ (it is the same as in the 0-time turning model). In order to achieve the following inequality $\sqrt{(o-1)(\frac{t}{o})^2+\frac{1}{o}(\frac{l-2lp}{vr})^2}<|\ \frac{l(1-2p)}{vr}\ |$, the following must hold $t<\sqrt{o}\ |\ \frac{l(1-2p)}{vr}\ |$. □

Corollary 27. *By using the synchronized-overlap technique we increase the average frequency of a point, and also the overall average frequency in the middle segments. By increasing the overlap factor o we can achieve more uniform patrol in the synchronized-overlap technique. However, this depends also on t (the time it takes the robot to turn). Given t and o, we can determine which patrolling technique is more uniform (synchronized-overlap or synchronized).*

6.1.2 Edge Segments

We analyze the different patrolling algorithms in the edge segments, when turning durations are taken into account. As we already mentioned (in Section 5.3.2), the number of edge segments in synchronized-overlap technique is $o-1$ (the synchronized technique does not contain edge segments since $o=1$). s_i is thus possibly different in each of the edge segments i. Series 6.4 shows the times of point visits in the edge segments, based on Equation 6.1.

$$\underbrace{\frac{l}{vr},\cdots,\frac{l}{vr}}_{\text{s}-1},\frac{2l(o-s)+2l(1-p)}{vr}+t,\underbrace{\frac{l}{vr},\cdots,\frac{l}{vr}}_{\text{s}-1},\frac{2lp}{vr}+t,\ldots \tag{6.4}$$

Lemma 28. *In edge segments, the synchronized-overlap shows worse results in the under-bounding frequency criterion, compared to the synchronized technique.*

Proof. Let p be a point in an edge segment, expressed as a fraction of the segment's length $p\in[0,1)$. The longest duration for which the synchronized-overlap technique neglects p is given by (see Series 6.4)

$$\frac{2l(o-s)+2l(1-p)}{vr}+t.$$

The worst-case for the synchronized technique depends on the point p, $\frac{2l(1-p)}{vr}+t$ if $0 \leq p \leq \frac{1}{2}$, or $\frac{2lp}{vr}+t$ if $1 > p > \frac{1}{2}$. Since $s+1 \leq o$ the inequalities in (Equation 6.5 below) always hold.

$$\begin{aligned} \frac{2l(1-p)}{vr}+t &< \frac{2l(o-s)+2l(1-p)}{vr}+t \\ \frac{2lp}{vr}+t &< \frac{2l(o-s)+2l(1-p)}{vr}+t. \end{aligned} \tag{6.5}$$

□

We have shown earlier that the average frequency of a point on edge segment is always better in synchronized patrolling than in the synchronized-overlap patrolling. However, in the realistic model, the time it takes the robot to turn (t) changes this conclusion in several cases, favoring the synchronized-overlap patrolling.

Lemma 29. *When $s > 1$ and $t > \frac{l(o-s)}{(s-1)vr}$ the average frequency of a point in edge segments is lower in the synchronized-overlap rather than synchronized technique.*

Proof. The average frequency of a point by the synchronized technique is $\frac{l}{vr}+t$. The average frequency in the synchronized-overlap technique is $\frac{o}{s}\frac{l}{vr}+\frac{1}{s}t$. In order to achieve the inequality in Equation 6.6 below the following must hold $t > \frac{l(o-s)}{(s-1)vr}$ for $s > 1$.

$$\frac{o}{s}\frac{l}{vr}+\frac{1}{s}t < \frac{l}{vr}+t. \tag{6.6}$$

□

Corollary 30. *There is only one edge segment in which the synchronized technique will always achieve better results in terms of average frequency: The leftmost segment. In the other segments, the other edge segments average criteria depends on o, s, t values.*

In conclusion, we have seen that the synchronized technique, and especially its generalization using an overlap factor o has interesting properties with respect to the different frequency-optimization criteria. In particular, the performance of the algorithms depends significantly on the segments chosen: Edge segments result in qualitatively different performance, compared to middle segments. We have shown that in many cases in the middle segments, the increase in overlap results in improved uniformity, without hurting the average and under-bounding frequencies. However, this

comes at a cost of greater neglect (and thus worse performance) in the edge segments. Moreover, the greater the overlap, the greater the neglect of points in edge segments, such that in those segments, performance is also reduced significantly in terms of the response time.

6.2 Handling Motion Errors

The selection of an optimal overlap factor critically depends on the model for the robot motion characteristics. In reality, robots not only take time to turn, but also have motion errors, which cause their actual velocity to diverge from the planned velocity v.

We add to the above discussion the function $d(x)$ to account for the accumulation of errors in robot motion. Let $T(x)$ denote the time it would take a robot to pass a distance x. Under assumption of no errors, $T = \frac{x}{v}$ where v is the robot's constant velocity. However, in realistic settings, due to acceleration changes and accumulating errors in motion, the actual travel time is going to be different: $T = \frac{x}{v} + d(x)$. By choosing to represent the error in travel time directly in terms of time, we bypass modeling the different factors accounting for delays, and focus on the symptoms. Note that we assume the $d(x)$ is non-decreasing function. Although in principle it is possible that a robot will travel too fast due to errors, in reality, this is rarely the case. For instance, a common source of velocity errors in laboratory robots is battery decay. This causes slowed motion, rather than acceleration.

We now analyze the visit frequency of a given point p using FOP, given the robots' motion characteristics t and d. The function $time_p(l,r,p,v,o,s,n,t,d)$ calculates the time that passes between two subsequent visits $(n-1,n)$ to the point p in a given segment i. By minimizing this function we improve the frequency of visits to the point p. The function $time_p(l,r,p,v,o,s_i,n,t,d)$ is defined as follows.

$$\begin{cases} 2\frac{l}{r}\frac{p}{v}+t+d(o\frac{l}{r})-d((o-p)\frac{l}{r}) & \\ \qquad +d(p\frac{l}{r}) & \text{if } n \bmod 2s_i = 1 \\ & \\ 2(1-p)\frac{l}{rv}+2(o-s_i)\frac{l}{rv}+t & \\ \quad +d(o\frac{l}{r})-d((s_i+p-1)\frac{l}{r}) & \text{if } n \bmod 2s_i = s_i+1 \\ \qquad +d((1-p+o-s_i)\frac{l}{r}) & \text{or } s_i = 1 \\ & \\ \frac{l}{rv}+d(\frac{l}{r}([(n-1) \bmod s_i]+p)) & \\ \quad -d(\frac{l}{r}([(n-2) \bmod s_i]+p)) & \text{otherwise} \end{cases} \tag{6.7}$$

The first condition in Eq. 6.7 is satisfied when the robots change direction at the left edge of a segment ($\frac{l}{r}$ is a length of a segment). The second condition is satisfied when the robots change direction at the right edge of the segment, and the third condition is satisfied in the overlapping regions.

Using Eq. 6.7, we can now construct the cyclic series which describes the times at which a point p is visited. The series is given in Eq. 6.8 below.

The first element is:

$$2\frac{l}{r}\frac{p}{v}+t+d(o\frac{l}{r})-d((o-p)\frac{l}{r})+d(p\frac{l}{r})$$

Then the next s_i-1 elements are of the form:

$$s_i-1\begin{cases} \frac{l}{rv}+d((1+p)\frac{l}{r})-d(p\frac{l}{r}) \\ \frac{l}{rv}+d((2+p)\frac{l}{r})-d((1+p)\frac{l}{r}) \\ \vdots \\ \frac{l}{rv}+d((s_i-1+p)\frac{l}{r})-d((s_i-2+p)\frac{l}{r} \end{cases}$$

Then, one element:

$$\begin{aligned} &2(1-p)\tfrac{l}{rv}+2(o-s_i)\tfrac{l}{rv}+t+d(o\tfrac{l}{r}) \\ &\quad -d((s_i+p-1)\tfrac{l}{r})+d((1-p+o-s_i)\tfrac{l}{r}) \end{aligned}$$

And finally, $s_i - 1$ elements of the form:

$$s_i - 1 \begin{cases} \frac{l}{rv} + d((1+p)\frac{l}{r}) - d(p\frac{l}{r}) \\ \frac{l}{rv} + d((2+p)\frac{l}{r}) - d((1+p)\frac{l}{r}) \\ \vdots \\ \frac{l}{rv} + d((s_i - 1 + p)\frac{l}{r}) - d((s_i - 2 + p)\frac{l}{r} \end{cases} \tag{6.8}$$

The first element of the series is the result of the first condition in Eq. 6.7. It matches the situation where a robot returns to its base segment and turns back until it meets the point p again: $2\frac{l}{r}\frac{p}{v} + t + d(o\frac{l}{r}) - d((o-p)\frac{l}{r}) + d(p\frac{l}{r})$. It is constructed from three components: (i) The time to arrive at p, based on the distance and robot velocity, the time it takes the robot to turn and the time error function, d; (ii) the time t it takes the robot to turn around; and (iii) the error (in travel time) due to the robot motion. The first component is given by $2\frac{l}{r}\frac{p}{v}$ where $\frac{l}{r}\frac{p}{v}$ is the time it takes the robot to arrive from point p to the left edge of the segment. We multiply it by 2 since the robot needs to return from this edge to point p. The third component is given by $d(o\frac{l}{r}) - d((o-p)\frac{l}{r}) + d(p\frac{l}{r})$, which is a reduction of $d(o\frac{l}{r}) - d((o-1)\frac{l}{r} + (1-p)\frac{l}{r}) + d(p\frac{l}{r})$. This value has two parts: Error in travel time from point p to the left edge and the error in travel time from the left edge back to point p. The error in traveling from point p to the left edge is equal to $d(o\frac{l}{r}) - d((o-1)\frac{l}{r} + (1-p)\frac{l}{r})$ which is the error of moving along all the (overlap) segments ($d(o\frac{l}{r})$) minus the uncertainty of moving along all the (overlap) segments until point p. The uncertainty of moving from the left edge to point p is equal to $d(p\frac{l}{r})$. Note that this model assumes that motion time errors are set to zero once a robot halts and turns.

The second element in the cyclic series (Eq. 6.8), corresponds to a neighboring robot that visits the point p, due to any overlap. The value is $\frac{l}{rv} + d((1+p)\frac{l}{r}) - d(p\frac{l}{r})$. It is composed of two factors: The time to arrive at the point (based on the distance and velocity), and the travel-time error function d. Since the robots are in $\frac{l}{r}$ distances from one another, the pure time it takes an adjacent robot to arrive the point p (after it has just been visited by another robot) is $\frac{l}{rv}$. The component $d((1+p)\frac{l}{r})$ comes from the adjacent robot's movement until it reaches point p. We then subtract from it $d(p\frac{l}{r})$, the error in time originating with the first robot movements as it leaves the point p behind it.

Overall, the segment i in question is visited by s_i robots, each twice (when moving left to right, and when moving right to left). The previous two paragraphs described the visit times due to the first two of these visits: The visit by original (first) robot, and a visit by an adjacent robot. Any other $s_i - 2$ robots follow the behavior of the adjacent robot, thus overall there are $s_i - 1$ elements due to adjacent robots in the first part of the series, as robots move left to right.

The other s_i elements in the cycle of series are due to the robots turning and repeating the movement, but from right to left. The next value (number $s_i + 1$) is $2(1-p)\frac{l}{rv} + 2(o-s)\frac{l}{rv} + t + d(o\frac{l}{r}) - d((s+p-1)\frac{l}{r}) + d((1-p+o-s)\frac{l}{r})$. This is a reduction of $2(1-p)\frac{l}{rv} + 2(o-s)\frac{l}{rv} + t + d(o\frac{l}{r}) - d((s-1)\frac{l}{r} + p)\frac{l}{r}) + d((1-p)\frac{l}{r} + (o-s)\frac{l}{r})$. This value is similar in form to the first value of the series.

6.2.1 Optimizing average frequency

The average time to visit a point p in the i'th segment is given the function avg_p, shown in Eq. 6.9. The function avg_p averages the previously shown series in Eq. 6.8, by relying on the function sum_p (Eq. 6.10) to sum the times between visits to point p in segment i.

$$avg_p(l,r,p,v,o,s_i,t,d) = \frac{1}{2s_i} sum_p(l,r,p,v,o,s_i,t,d) \tag{6.9}$$

$$\begin{aligned} sum_p(l,r,p,v,o,s_i,t,d) = \quad & \frac{2ol}{rv} + 2t + 2d(o\frac{l}{r}) - d(p\frac{l}{r}) - d((o-p)\frac{l}{r}) \\ & + d((s_i - 1 + p)\frac{l}{r}) + d((1-p+o-s_i)\frac{l}{r}) \end{aligned} \tag{6.10}$$

In order to find the optimal overlap factor o that minimizes the global average along all fence, we need to summarize all visit sequence of each point in all fence segment and minimizing this value respectively to o. We do this in two stages. First, we use an integral from 0 to 1 on function sum_p, with respect to p, to sum all of the visit intervals of all points in a specific segment. We then sum all such integrals in all segments, and divide by the length of the entire polyline.

Equations 6.11–6.12 show this process. The function *sum* in equation Eq. 6.11 sums all the visit intervals of all points in all segments. Here, S is the set of all s_i.

Then, the function *avg* (Eq. 6.12) divides the sum by l to get the average time between visits, over the entire length of the open polyline.

$$sum(l,r,p,v,o,S,t,d) = \sum_{i=1}^{r} \int_{0}^{1} sum_p(l,r,p,v,o,s_i,t,d) \text{ dp} \tag{6.11}$$

$$avg(l,r,p,v,o,S,t,d) = \frac{1}{l} sum(l,r,p,v,o,s,t,d). \tag{6.12}$$

To find the optimal o value that minimizes the *avg* function 6.12 by looking for a minimal value, e.g., by using the first and second derivative with respect to o to determine minimum points. Since in this article we did not place any restrictions on the structure of d, we refrain from doing so here. In practice, it should be done only once d is known.

6.2.2 Maximal minimum frequency

For the under-bounded frequency criteria it is easy to determine that the segment i for which $o - s_i$ is greatest, has the lowest frequency of visits to segment points. The edge where this occurs is known in advance—it is the leftmost edge segment ($i = 1$), which is left alone for long periods of time when the robot responsible for it is busy in the overlapping portions of its trajectory, and no other robots visit it. As we move right, and more robots patrol the segments, the better this measure becomes.

The worst time to visit a point in the leftmost segment is (as appears in Eq. 6.8):

$$2(1-p)\frac{l}{rv} + 2(o-s_1)\frac{l}{rv} + t + d(o\frac{l}{r}) - d((s_1+p-1)\frac{l}{r}) + d((1-p+o-s_1)\frac{l}{r})$$

Thus if the maximal minimum criteria is important at the polyline level (i.e., across all segments), then no overlap should be used. setting $o = 1$ maximizes the minimal frequency in this case.

However, suppose we are instead seeking to examine the maximal minimum frequency per segment. In middle segments (where $o = s_i$) the lowest frequency (the greatest time between visits) of visiting a point will be the maximal value from the following options:

1. $2\frac{l}{r}\frac{p}{v} + t + d(o\frac{l}{r}) - d((o-p)\frac{l}{r}) + d(p\frac{l}{r})$
2. $max_{i=1}^{o-1}(\frac{l}{rv} + d((i+p)\frac{l}{r}) - d((i-1+p)\frac{l}{r}))$
3. $2(1-p)\frac{l}{rv} + t + d(o\frac{l}{r}) - d((o-1+p)\frac{l}{r}) + d((1-p)\frac{l}{r})$

The maximal value depends on the form of d function and the point p.

6.2.3 Maximal frequency uniformity

As before, we measure the uniformity of visit frequency by the standard deviation of frequency values. Lower values indicate improved uniformity, as it means that the frequency values for different points p are clustered more closely around the average frequency.

The standard deviation of visiting a point along the polyline is shown in Eq. 6.13. In order to find the o value that minimizes the function.

$$\sigma(o) = \sqrt{\frac{\sum_{i=1}^{r}(\int_{0}^{1}(\sum_{j=1}^{2s_i}(time_p(l,r,p,v,o,s_i,j,t,d) - avg(l,r,p,o,s_i,t,d))^2)\ \mathrm{dp})}{l\sum_{i=1}^{r}2s_i}} \quad (6.13)$$

6.3 Summary

The analysis in this section has shown how to extend the FPO algorithm beyond a naïve motion model, to account for robot turning durations, and velocity errors. The result of this analysis, however, shows that there is no longer a clear-cut trade-off between edge and middle segments, and the overlap factor is increased. Rather, the optimization depends critically on the actual error function $d(x)$ and the turn duration t. Indeed, in Section 8, we will demonstrate that both parameters are necessary to predict the behavior of real robots.

In principle it is possible to optimize the selection of the overlapping factor o to maximize performance across three frequency-based criteria. Thus given environment parameters (e.g., length of fence, number of available robots) and robot motion characteristics (turning duration t, the error function d), it is possible to determine the optimal

o. Naturally, by manipulating the analysis, it might be possible to instead determine the optimal number of robots for a given overlapping factor, or the optimal turn duration for a given number of robots and overlapping factor, etc. We leave this for future work.

We note that all the models described above assume that the robot size is insignificant relative to the fence size, which is not true in typical laboratory conditions (see our own experiments below). This assumption is, however, true in many target patrolling applications, where the patrolling unmanned ground vehicles (UGVs) are a few meters long, and patrol a fence that is at least a few kilometers in length.

Chapter 7

Handling Events in Polyline Patrolling

In the fence patrol task, we expect the system to execute nothing but the patrolling motion for most of the time. However, the purpose of the patrol is to be able to respond to events. We follow earlier work [26] in accounting for event handling duration within the algorithms. On the other hand, we would like to keep an optimal frequency of visiting in most of the fence locations. This requires robots to spend as little time as possible handling event, so as to interrupt the patrolling motion as little as possible. On the other hand, We want to make sure events are handled by specific deadlines, and are not ignored.

Definition 3 (Polyline event)**.** *An event e_j is tuple $\langle i, p, t_j, T_j \rangle$, where i is the segment number where the event occurs; p is a point in the segment where the event takes place, given as a fraction of the length of the segment, $p \in [0,1)$; t_j is the time it takes to handle the event and T_j is the time-limit to finish handling the event from the moment it occurred. $T_j - t_j$ is thus the duration for which the event point can be neglected from the moment the event occurred until it is fully handled.*

The key to optimal event handling is to share the event handling between the robots that are already patrolling along the segment in which the event occurs. This will allow the robots to stay in their original patrol path and stay synchronized in their movement most of the time the event is handled.

The first step of the event handling procedure is to check if the event location can be feasibly reached, i.e., the event is *feasible*. We define t_f to be the time it takes the closest robot to the event point to arrive the event. Let $\mathsf{Feasible}(e_j)$ be a Boolean

variable that represents the feasibility of the event. If the $T_j - t_j$ is less then t_f the closest robot (to the event) will not succeed to arrive the event and handle it in less than T_j. In such situation the event cannot be handled without changing the velocity of the robots or some other external intervention.

$$\mathsf{Feasible}(e_j) = \begin{cases} \mathsf{False} & \text{if } T_j - t_j < t_f \\ \mathsf{True} & \text{otherwise} \end{cases} \tag{7.1}$$

The robot motion model must be taken into account when deciding on how to best allocate robots to handling an event. In the discussion below, we assume a naïve motion model where robot turns are instantaneous, and no velocity errors occur. The extension to the more realistic models is technically straightforward.

We use the robots that are assigned to patrol (in overlap, if any) the specific segment in which the event occurs, to handle the event. The number of robots that could handle the event by our algorithm is s_i if the event occurs in segment i. In order to minimize the effect on the patrolling trajectories, we divide the event handling over many patrolling passes as possible, as this necessarily means that we deviate minimally from the current patrol schedule. We define *rounds* to be the number of visits the robots make to the event locations, either left to right or vice versa. The time series below (Series 7.2) shows the point visit times at point p on segment i, which compose a complete set of visits (a round) by all the robots that patrol and overlap within the segment. Our goal is therefore to maximize *rounds*.

$$2\frac{l}{r}\frac{p}{v}, \underbrace{\frac{l}{rv}, \cdots, \frac{l}{rv}}_{\mathrm{s_i}-1}, 2(1-p)\frac{l}{rv} + 2(o-s_i)\frac{l}{rv}, \underbrace{\frac{l}{rv}, \cdots, \frac{l}{rv}}_{\mathrm{s_i}-1}, \cdots \tag{7.2}$$

Let *series* be the total duration of a round (the sum a cycle in Series 7.2). It is always $2o\frac{l}{rv}$ for any point p. This is the time it takes the robots to pass the event location from right to left and vice versa. The number of such *rounds* is

$$\frac{T_j}{series}.$$

Therefore at each round each of the robots should attend the event for period of

$$\frac{t_j}{s_i(rounds)}$$

which is divided into two halves; one in passing left to right, and one in passing right to left. Thus in each pass, regardless of direction, each robot should spend a time of

$$\frac{t_j}{2s_i(rounds)}$$

on handling the event. The number of complete rounds the robots will do so will be $\lfloor rounds \rfloor$. Later on in this section, we will explain how the event is handled at the last, incomplete, round.

During the $\lfloor rounds \rfloor$ complete rounds, the robots that attend the event should move the same period of time to one direction (right to left and left to right) as before the event occurred. This, to ensure that all the robots (and not only those that attend the event) continue moving in a synchronized manner as before. Specifically, each of robot moves to one direction for period of $o\frac{l}{rv}$ time. When a robot attends the event, it must move for a time of only $o\frac{l}{rv} - \frac{t_j}{2s_i(rounds)}$. The rest of the time ($\frac{t_j}{2s_i(rounds)}$) it will attend the event. This will keep the robot patrol synchronized.

It might be that there is a need for a final, incomplete round, where only some of the robots should attend the event (the time is not sufficient for all to participate). Algorithm REMAINDEREVENT (Algorithm 10) calculates the maximal time that the event can be attended to by the passing robots in the last incomplete round. Each robot here moves for a time of $o\frac{l}{rv}$ in each direction, as before the event occurred.

Let $\langle t_p, intervals, startIndex, T_j, t_j \rangle$ be the input of algorithm REMAINDEREVENT. t_p is the time it takes the closest robot to arrive the event point without breaking its regular patrol path. $intervals$ is an interval array of visiting the event point p, as shown in Series 7.2. $startIndex$ is the index of the visiting interval of the event point after the closest robot attends to it. T_j is as described above, but we update it to be the residual from the previous rounds. So T_j is updated to be: $T_j \leftarrow T_j - \lfloor rounds \rfloor \times series$. t_j is as described above but also decrease with the time that the robots handle the event at the previous rounds. So t_j is updated to be $t_j \leftarrow t_j - \lfloor rounds \rfloor \frac{t_j}{rounds}$

The algorithm calculates the number of visits needed to handle the event during the last (incomplete) round. The time to attend to the event, by each robot, is easy to calculate. It is t_j divided by the number of visits, i.e., $t_j / NumberOfVisits$.

Line 2 of the algorithm initializes the variable *sum* to be the time it takes the closest robot (in its regular pass) to arrive at the event location. The variable *sum* represent the time that the event location is neglected. The while loop (lines 3–15) iteratively

Algorithm 10 REMAINDEREVENT(t_p,$intervals$,$startIndex$, T_j, t_j)

1: $NumberOfVisits \leftarrow 1$.
2: $sum \leftarrow t_p$.
3: **while** $T_j - t_j \geq sum$ **do**
4: $sum \leftarrow t_p$.
5: $i \leftarrow startIndex$
6: $NumberOfVisits \leftarrow NumberOfVisits + 1$
7: $HandleTime \leftarrow \frac{t_j}{NumberOfVisits}$
8: **for** $visits \leftarrow 1$ to $NumberOfVisitis - 1$ **do**
9: $i \leftarrow i$ mod $intervals.length$
10: **if** $i = 0$ or $i = \frac{intervals.length}{2}$ **then**
11: $sum \leftarrow sum + intervals[i]$
12: **else**
13: $sum \leftarrow sum - HandleTime$
14: $i \leftarrow i + 1$
15: $visits \leftarrow visits + 1$
16: $NumberOfVisits \leftarrow NumberOfVisits - 1$
17: return $NumberOfVisits$

increase the number of times the event handled (line 6), updates the influence on the system (lines 8–13) and makes sure that we do not go over the time limit (line 3).

Event handling will always decrease the patrol performance since the robots need to dedicate some of their patrol time to handling events. Given the above algorithm, the segments that could suffer from such decreasing frequency (but will not abandon) will be, at most, the o neighboring segments to the segment in which the event occurred. This because the robots, while they attend to the event, need to move to one side (left to right or right to left) for $o\frac{l}{rv}$ time, exactly as before the event occur. Now, the robots that handle the event could not succeed to pass all the o segments they should pass during this time, but only part of them.

The above leaves one possible case unexplored. It might be that an event is Feasible, but the time to arrive the event by the regular patrol path t_p is not sufficient to arrive at the event location (plus handling time). In other words, $T_j - t_j < t_p$. In this situation the closest robot to the event location must break the patrol trajectory

and travel to the event location. There, it will handle the event until its neighbor robot will arrive and take over. From that point, the regular event handling procedure can continue.

Chapter 8

Experiments

To evaluate the usage of the patrolling algorithm FOP (Algorithm 9, we conducted a series of experiments on physical robots, during which the robots are performing patrols of $o = 1$ and $o = 2$, using the algorithm. The purpose of the experiments was to evaluate the predictions of the different models as to the actual patrolling frequencies of the robots. For instance, the realistic-motion model discussed in Section 6.2 predicted that an overlap of one ($o = 1$) would be better for the given length of the polyline used in the experiments, while the original, more abstract model predicted that an overlap of two ($o = 2$) would work better.

To carry out these experiments, the robots were programmed by students, without knowledge of the motion models developed. The experiment settings are discussed in Section 8.1. We recorded the point visit frequencies in extensive trials, and conduct post-hoc analysis, in which we compare the predictions of the different models to the actual behavior of the robots (Section 8.2).

8.1 Experiment Settings

Our experiments utilized a team of three robots, patrolling a mock fence, using Friendly Robotics' RV-400 [35] vacuum cleaning robots (Figure 8.1). Each commercial robot was modified to be controlled by a small Linux-running computer, sitting on top of it. A generic interface driver for the RV-400 robot was built in the Player robotics API [39], and a client program was built to control it, implementing FOP (Algorithm 9). The robots have 8 short-range sonar sensors, pointing forward and

sideways, which we utilize for maintaining distance to the mock fence. The robot interface also provides rudimentary odometry readings (coordinates and heading), which are unfortunately fairly inaccurate.

Figure 8.1: The RV-400 vacuum cleaner robot, with our lab's computer overriding its commercial control software.

The experiment settings consisted of a carton-box mock fence, 5.40 meters in length. The fence was divided into three equal-length segments (180cm each). Figure 8.2 shows a birds-eye view of the mock fence. The three robots are equidistant from each other, though it might be difficult to see because the picture was taken at an angle.

The FOP algorithm—in its abstract form—was described to students carrying out course projects in the laboratory. Thus their implementation of the FOP algorithm is untainted by our expectations, given our own knowledge of the motion characteristics model. This is a critical point in the design of the experiment.

Independent variables. We ran two sets of patrolling runs. In the first set of nine patrol runs, the robots had to complete one back and forth round with an overlapping factor of 1, i.e., no overlap at all ($o = 1$). In the second set of seven trials (originally, nine, but two were dismissed because a weak battery caused noticeable slowdown in their movements), the robots also had to complete one round trip, but this time with an

Figure 8.2: A snapshot from experiments. The three robots are positioned at the initial points for the three segments.

overlapping of 2 ($o = 2$), in which the leftmost and middle robot are also responsible for the segment to their right.

Dependent variables. To measure frequency, we recorded movies of the robots patrolling the fence, and later analyzed the video recordings to determine the duration between subsequent visits to points within each segment. Four sampling points p were selected at a distance of 45cm, 225cm, 255cm, and 285cm from the left edge of the fence. The points were selected for their visibility in the videos. The 45cm point is on the leftmost edge segment, at $p = \frac{45}{180} = \frac{1}{4}$. The other three points are respectively in the middle segment. Each patrolling run consisted of a number of visits to each point, the results (below) are averaged over approximately 30 data-points (when $o = 2$), or 20 data-points ($o = 1$).

Each set of results from the robots' patrols was contrasted with the predictions of two different analytical models: The realistic model described in Section 6.1 which

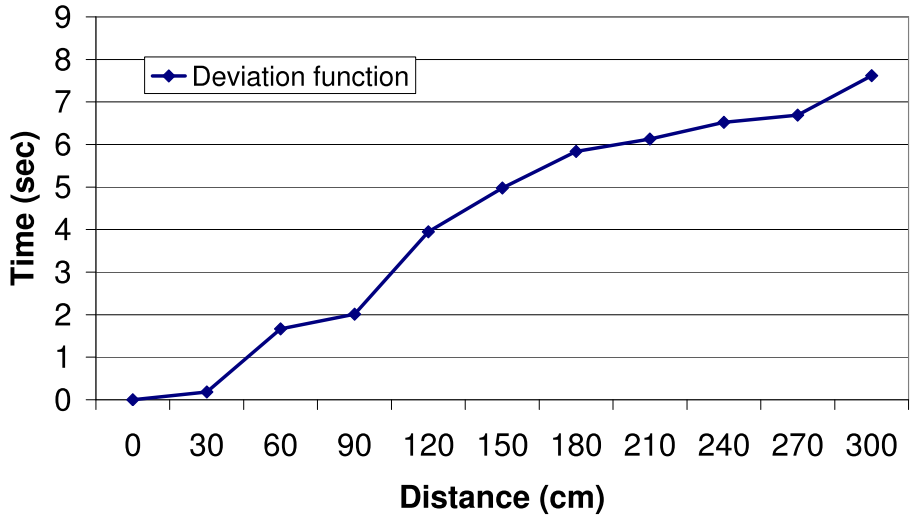


Figure 8.3: The deviation function $d(x)$ which measures the extra time that the robot delays when moving a distance x, as a result of uncertainty in movement and accumulating errors.

accounts for turning durations, and the more abstract model described in Section 6.2 which adds also explicit accounting for motion errors affecting segment travel times.

To compute the predictions of the models, we estimated the t and d parameters for the robots, from a small subset of the recorded videos. We determined that the turning duration t for the robots was approximately 6 seconds. The travel time error function $d(x)$ is shown in Figure 8.3. It was estimated based on measurements of 10 different distances, in a subset of the experiments. Here, the x-axis shows the distance x, and the y-axis measures the *error* in travel time, compared to the predictions based on the robot velocity alone. Note the monotonically increasing error function: We remind the reader that this error function was not intentionally built in, but was discovered post-hoc, providing empirical support for the assumptions we made earlier on regarding the monotonically-increasing nature of d.

8.2 Experiment Results

We calculated the predicted intervals of visitation according to two models:

New Model. The full realistic motion model (which includes turning durations and motion uncertainty, as described in Section 6.2).

Old Model. The motion model allowing only for turning durations (i.e., as described in Section 6.1).

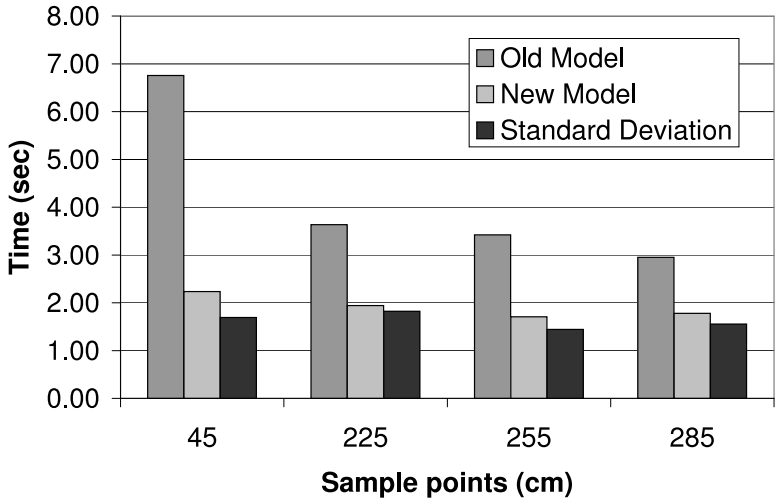
We contrasted the predictions of both models with the empirical results observed in practice. In this comparison, adjustments were made to the predictions of both models, to account for the size of the robot compared to the length of its segment (essentially, the robot size was deducted from the distance traveled).

8.2.1 Point-level predictions

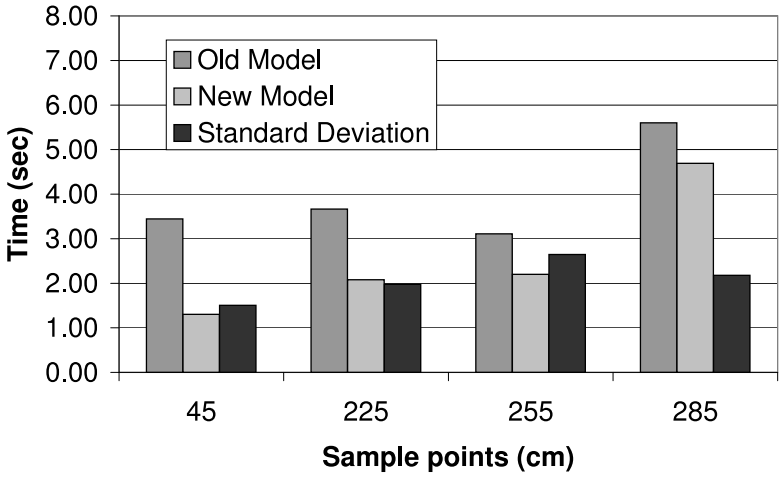
We begin by a direct comparison of the two models, by specifically focusing on the errors in their predictions for the sampling points we chose earlier. In Figure 8.4 we present the average errors of each model with respect to the observed results, and the standard deviation for the observed results. In both sub-figures, the x-axis shows the four sampled points along the fence, denoted by their distance in centimeters from the left edge of the fence. In the y-axis, we present the error in seconds. The left bar shows the average error in the prediction of the old model. The middle bar shows the error in prediction for the new model, described in Section 6.2. The right bar shows the sample standard deviation for the observed results, for comparison.

Several conclusions can be reached based on Figure 8.4. First, for both overlapping factor settings, the new model is clearly more accurate than the simpler old model; both Figure 8.4(a) and Figure8.4(b) clearly show a substantial reduction in prediction error in the new model (middle bar), compared to the old model (left bar).

Second, in most cases, the average errors of the new model are approximately equal to the standard deviation of the observed results. This suggests that the new model is not just *relatively accurate* (being superior to the other model), but also *absolutely accurate*, in that the error is indistinguishable from the normal observed varying measurements. We will revisit this point later, when discussing the statistical significance of differences.



(a) $o = 2$



(b) $o = 1$

Figure 8.4: Errors in predictions, and the sample standard deviation. Lower values are better. The new model's results are always better than those of the old model.

8.2.2 Segment-level predictions

We now abstract away from specific points along the fence, and turn to examine the predictions of the model at the segment level. Figures 8.5–8.7 contrasts the segment

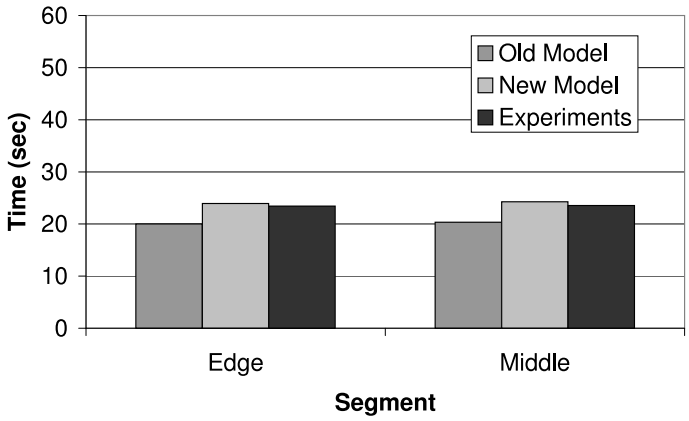
level average frequency (Figure 8.5), maximal minimum frequency (Figure 8.6), and uniformity (as measured by the frequency standard deviation) for two segments: The leftmost segment (which, when $o = 2$ is an edge segment), and the middle segment. Each sub-figure shows two groups of bars. The left group of bars shows the results for the leftmost (edge) segment. The right group of bars shows the results for the middle segment. Within each group, the leftmost bar shows the predictions of the old model, the middle bar shows the predictions of the new model, and the final bar shows the actual results (averaged over the different runs). Sub-figures should be contrasted vertically: The top sub-figures show the frequency criteria measurements for $o = 1$, and the bottom sub-figures show the measurements for the same criteria, for $o = 2$.

Multiple issues are raised by the comparison. First, by contrasting predictions of the old and new models with the actual results (i.e., within each group of bars), we again see that the new model, developed in this thesis, accurately predicts the observed behavior of the robots in practice, regardless of the overlap factor. To see this, we look at the statistical significance of the differences between the predictions of the old model and the experiment results, versus the significance of the difference between the predictions of the new model and the results. These results are shown in Table 8.1.

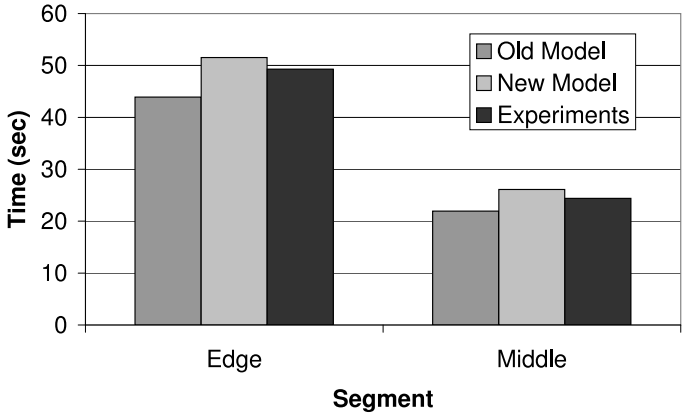
Performance	Overlap	Old Model		New Model	
Criteria	Factor	Edge	Middle	Edge	Middle
Uniformity	$o = 1$	0.00008	0.032	0.9991	0.4674
	$o = 2$	$< 1 \times 10^{-14}$	0.000001	0.7001	0.0041
Average	$o = 1$	0.0047	0.00765	0.5586	0.701
	$o = 2$	0.3215	0.00361	0.9994	0.969
Maximal	$o = 1$	$< 1 \times 10^{-14}$	0.570	0.9987	0.9999
	$o = 2$	$< 1 \times 10^{-14}$	0.675	0.9973	0.9999

Table 8.1: Significance of comparison of the experiment results to the old and new models. Each cell holds the results of a two-tailed z-test p value for the corresponding segment (edge or middle), overlap factor o (1 or 2), for the different performance criteria.

We find in Table 8.1 that generally, in terms of average frequency and uniformity of frequency, the predictions for the *old* model are statistically significantly different than



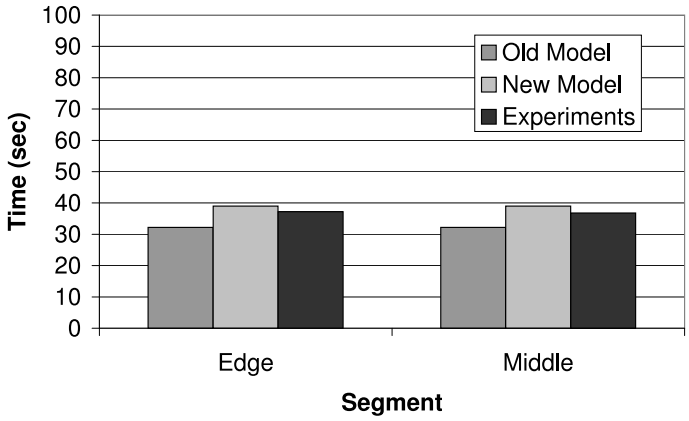
(a) Average time between visits in segments, $o = 1$.



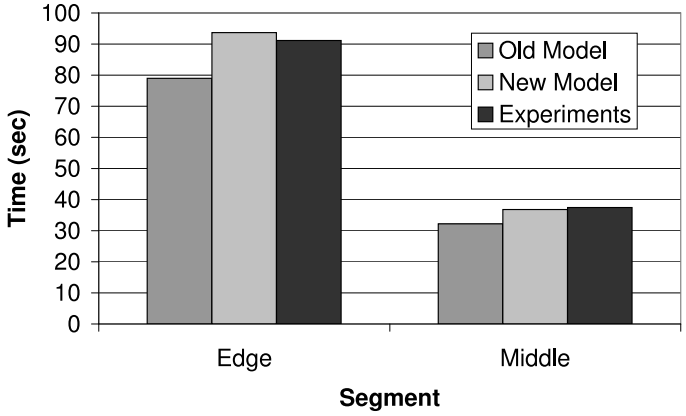
(b) Average time between visits in segments, $o = 2$.

Figure 8.5: Average time between visits, for different overlap factors. A lower result is better on an absolute scale.

the experiment results (two-tailed Z-test, $p < 0.05$). This implies that the old model does not accurately predict the experiment results. However, there is in general no statistically significance difference between the predictions of the new model and and the results. While this does not prove they are the same (i.e., that the new model is accurate), it does lend support to this hypothesis. This is especially true given the high values of the null hypothesis probability (p) values. For the maximal minimal-frequency



(a) Maximal time between visits, $o = 1$.

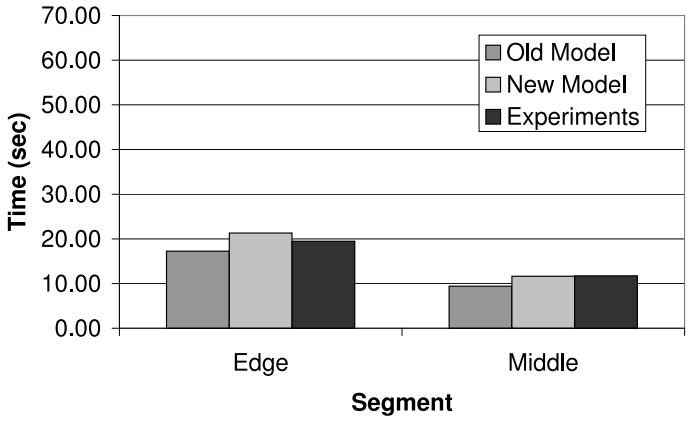


(b) Maximal time between visits, $o = 2$.

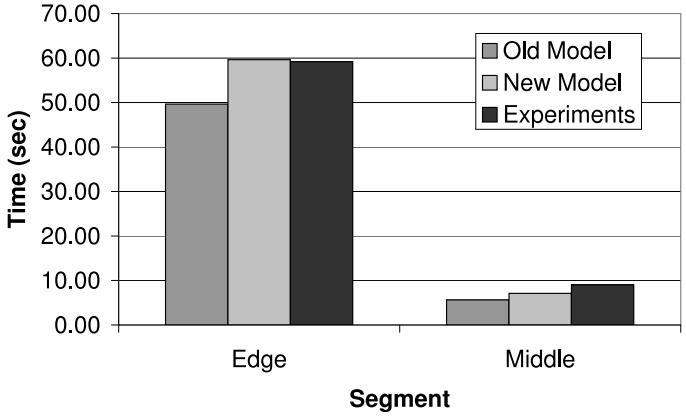
Figure 8.6: Maximal time between visits, for different overlap factors. A lower result is better on an absolute scale.

criteria, both the old and new model's predictions are not significantly different than the robot results.

A second issue is raised when examining the left and right groups of bars within each sub-figure, and contrasting them vertically. We see a clear qualitative difference between the middle and edge segments (when $o = 2$), which is not present in the case $o = 1$. This distinction between the results of the FOP algorithm in middle and



(a) Sample standard deviation of segments, $o = 1$.



(b) Sample standard deviation of segments, $o = 2$.

Figure 8.7: Uniformity of patrolling frequency—measured by the standard deviation of the results—for different overlap factors. A lower result is better on an absolute scale.

edge segments, for $o > 1$, is theoretically predicted by the analytical models developed earlier. In this respect, selecting $o = 1$ may be a better choice, if one does not allow variability in patrolling frequency at the segment level.

Lastly, by contrasting individual groups of bars vertically, we can begin to see where there may be an advantage to the overlapping group ($o = 2$). For the middle

segments, the results of the average and maximum minimum frequency are essentially the same for $o = 2$ as for $o = 1$. However, we can also see that the uniformity of the middle segments in the case of $o = 2$ is much improved compared to the case of $o = 1$ (Figures 8.7(a) and 8.7(b)).

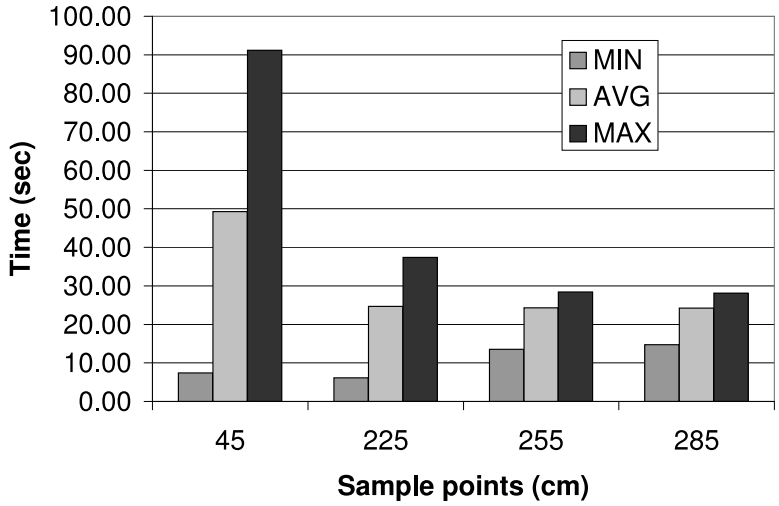
8.2.3 Polyline-level predictions

Finally, we discuss the results of the experiments in terms of the entire length of the polyline, encompassing all three segments. Based on Equation 6.13 above, the model predicts that for the given length (5.40cm), number of robots (3), and their motion parameters, an overlap of 1 ($o = 1$) is preferable. Figures 8.8(a) and 8.8(b) show the minimum, average, and maximum time between every 2 consecutive visits when $o = 2$, and $o = 1$, respectively. In the x-axis we present the 4 points sampled. In the y-axis we present the time in seconds between every 2 consecutive visitations.

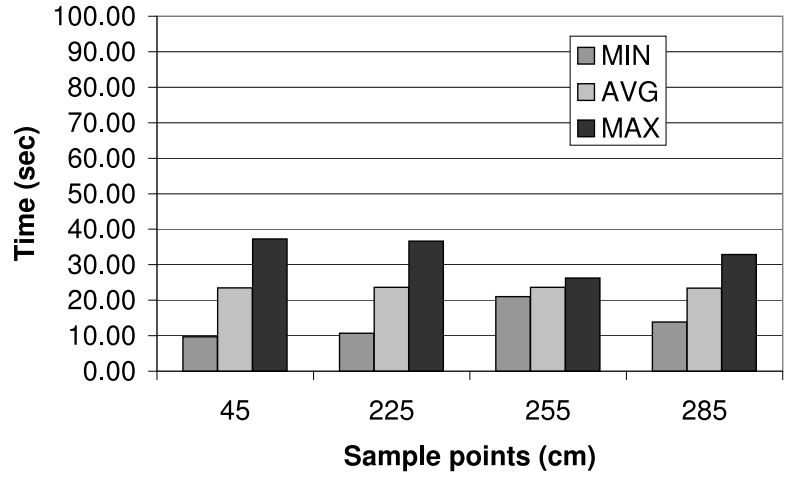
Here, the disadvantage in using an overlap greater than 1 is evident, where the maximum between 2 visitations in the 45cm point is almost 3 times all other points sampled, in Figure 8.8(a) (where $o = 2$). In contrast, the results are much more uniform in all criteria, when $o = 1$ (Figure 8.8(b)).

Indeed, the results of the experiments with real robots show that the standard deviation of visits in the case of $o = 1$ is 10.945, while in the case of $o = 2$, the standard deviation is 19.745. Thus as predicted, an overlap factor one is preferable in terms of uniformity of point-visit frequency.

Note that these results are for the specified parameters. For instance, if there were 5 robots, an overlap of two ($o = 2$) would have been better. And had there been a longer polyline, the overlap could have changed as well: The longer the polyline—the greater the number of middle segments—the more advantageous it would likely be to use $o > 1$: The effects we are seeing here for the middle segment would be the same for all middle segments, however many; while there will only be a few edge segments regardless of the length of the polyline.



(a) $o = 2$.



(b) $o = 1$.

Figure 8.8: Minimum, maximum and average visit frequencies with different overlap factors.

Part III

Additional Patrolling Challenges

Chapter 9

Patrolling in Formations

We depart from the standard assumption that each patrolling unit is made of a single robot. Instead, we now allow multiple robots that move coherently, in formation, in each patrolling unit. Previous work on formation-maintenace assumes that the robots have enough sensors to maintain the formation and to avoid obstacles or carry out their surveillance goals. A challenge arises when robots do not have sufficient sensors to both track their peers and their environment at the same time. Our work explore a controller-*multiplexing* technique in which the robots alternate between open- and closed- loop control. The work also explores a *fusion* technique in which the robots merge the open- and closed- loop controllers. This allows the robots to maintain formation and avoid obstacles when their sensors are limited and their odometry is noisy. We evaluate those techniques with physical robot (Sony AIBO ERS-7) experiments and also in simulation. We show that *multiplexing* and *fusion* techniques reduce the number of obstacles that the robots hit and also maintains the formation more robustly than the constituent controllers by themselves.

9.1 Introduction to Patrolling in Formations

There is significant interest in formation-maintenance tasks, where robots move together (maintaining a geometric shape), while avoiding obstacles. For instance, Dougherty et al. [25] and Matarić and Fredeslund [32, 34] describe algorithms which allow robots to move in formation and to avoid obstacles. They do so under the assumption that the robots have sufficient sensing capabilities to maintain the formation

via closed-loop control (tracking their peers) and also recognize obstacles or otherwise sense the area around the robot.

A challenge arises when robots do not have sufficient sensors to both track their peers and their environment at the same time. This could be, for example, if the limited sensors are kept busy providing input to another method that needs them. For instance, on the Sony AIBO ERS robots, the sensors used for formation maintenance are on a single pan-tilt component (the head). The robot cannot follow a leader (at some fixed angle) and simultaneously scan for obstacles.

One obvious alternative is to utilize open-loop control in formation maintenance, to free up robots' sensors for other uses such as obstacle avoidance. While operating in open-loop control, the leader of the formation transmits its movements, while its followers translate these into corresponding movements of their own, without relying on sensors. However, this relies on odometry reading in both the leader and the follower. In principle, translating the movements of the leader into each follower's actions, via communications, is sufficient. However, in practice, accumulating odometry errors prohibit this technique from being used exclusively.

We therefore explores ways to *multiplex* and/or *fuse* closed-loop and open-loop formation control. Multiplexing is done in time, giving the alternative methods different periods of time in which they control each robot. Switching between the different methods utilized the following principle: Each robot relies on visual tracking (closed-loop control) until it is within tolerance levels of its position in the formation. When this occurs, the robot switches to *communication-based* open-loop control, and uses its sensors to scan for obstacles, while communicating with the leader. To verify its position and inhibit accumulating errors, the robot switches back to visual tracking after a fixed period of time. We also explore combining controllers by fusion, by merging the output commands of each open-loop and close-loop controllers.

To evaluate the contribution of those approaches, we compare the *multiplexing* and *fusing* methods with their closed-loop and open-loop components, by themselves. The experiments are carried out using physical (Sony AIBO) and simulated robots. We carried out two separate repeated-trials experiments. The first experiment evaluated the benefit offered by the combining methods, by showing the formation's ability to detect objects surrounding the formation (e.g., obstacles). A second experiment explores the hypothesized costs of this ability: The stability and accuracy of the maintained

formation, in the four methods.

The results of the experiments show that indeed, the combination techniques (multiplexing and fusing) are able to significantly reduce the number of undetected obstacles, and thus increase the robustness of the formation to obstacles, even with limited sensing. Moreover, the combination technique also works to more robustly *reduce* the error in the positions of robots in the formation, in contrast to the hypothesis. The results also show that using multiplexing is preferable to fusing when the robot odometry is more accurate, and/or when the formation path includes sharp turns. The advantage of the fusing technique over the multiplexing technique is shown clearly when uncertainty grows in the odometry.

9.2 Maintaining Robust Formations

The operator of a formation is inherently limited by the capabilities of the robots to sense their surroundings, and provide information about potential failures. A challenge arises when robots do not have sufficient sensors to both track their peers and their environment at the same time. This could be, for example, if the limited sensors are kept busy providing input to the closed-loop controller that is used to maintain the formation. This section addresses this challenge.

We first differentiate sensor-based closed-loop formation maintenance from communication-based open-loop formation maintenance (Section 9.2.1). We then (Section 9.2.2) discuss the two key methods used to combine open- and closed-loop maintenance (*multiplexing* and *fusing* controllers). Finally, we report on experiments evaluating the different methods (Section 9.3).

9.2.1 Open-Loop and Closed-Loop Formation Maintenance

In the *sensor-based* formation-maintenance algorithm, each follower but the leader is to maintain a specific distance and angle to another robot (called the *anchor*). This is called Separation-Bearing formation-maintenance control, and is proven to be stable [31]. A problem arises when the robot's sensors are limited and the robot also needs to detect obstacles: If the follower does not scan for obstacles, it may fail to discover them. And if it scans for obstacles, it may lose sight of its anchor, and thus lose its

place in the formation.

For instance, on the Sony AIBO ERS robots, the sensors used for formation maintenance are on a single pan-tilt component (the head). The robot cannot follow a leader (at some fixed angle) and simultaneously scan for obstacles.

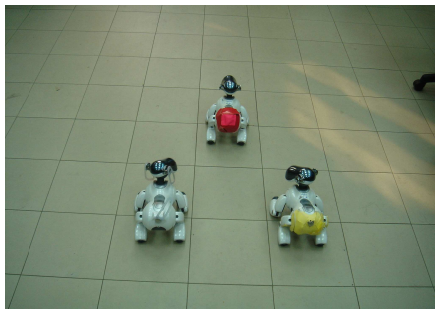
One obvious alternative is to utilize open-loop control in formation maintenance, to free up robots' sensors for other uses such as obstacle avoidance. While operating in *communication-based* open-loop control, the leader of the formation broadcasts its movement vector (velocity and heading changes). Based on this communication, and their predefined ideal positions in the formation, all other robots calculate their own relative movements, without relying on sensors. However, this relies on odometry reading in both the leader and the follower. In principle, translating the movements of the leader into each follower's actions, via communications, is sufficient. In practice, accumulating odometry errors prohibit this technique from being used exclusively.

This is indeed an open-loop controller for the formation: Messages cannot in practice be sent continuously, and thus a projection is made as to the anticipated position of the leader (and by implication, the follower), using *affine transformations* [67]. Once the anticipated position is known, the follower can set it as a goal position, and use simple motion planning to generate a movement vector of its own. This movement vector is maintained until a new broadcast from the leader initiates this calculation once again.

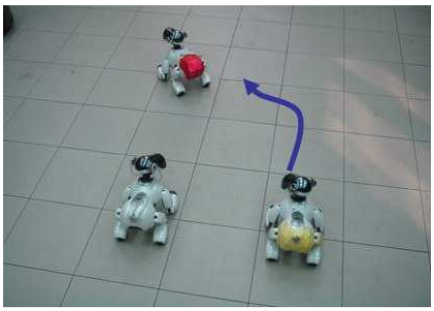
The translation of target position to movement vector has two factors. The first is handled by the affine transformation. The second requires additional corrective actions by the follower robots. We describe these factors below.

The first factor is the effect of the leader's heading on the path chosen by its follower. Figure 9.1-b,c show cases where the position of the leader is identical, but its heading is different. As a result, the target location for the follower, and the path to it (both indicated by the arrow in the figures) are radically different. This also implies that the affine transformations are sensitive to errors in their inputs, as even small deviations in the heading may result in large difference in the computed movement vectors.

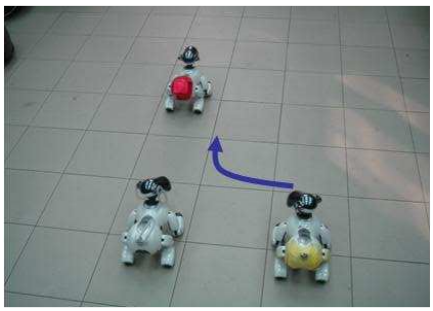
The second factor in correctly computing the movement vector is tied to the difference in the body orientations of the leader and follower robots, after the latter reach their target positions. Ideally, the orientation of the leader and followers should be



(a) Ideal positions of the robots.



(b) Leader changed heading.



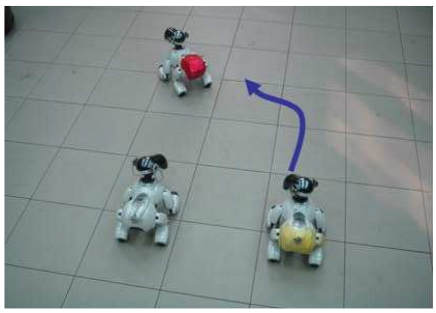
(c) Leader did not change heading.

Figure 9.1: A triangle formation of three Sony AIBO robots. Figure (a) shows the ideal poses of the robots. Figures (b) and (c) illustrate the sensitivity to heading; the leader is in the same x, y location in both figures, but its heading is different, implying a radically different target position for the right follower robot.

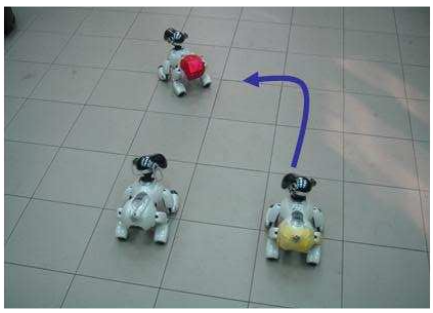
equal at that point. However, depending on the path taken by the follower, the orientation of its body might be different from that of the leader (see Figure 9.2).

To maintain the orientation error in the followers as small as possible, we recommend explicitly tracking the difference in orientation between the leader and the follower, and correcting it in each time step. However, this approach can result in jerky movement on the part of the robots, when they attempt to correct a large error within a single time-step. To address this, the controller should limit itself to corrections that are only of a limited range, and instead apply them over multiple time-steps, if necessary.

The advantage of the communication-based controller is that it can free up some of the robot's sensors. Instead, the follower robot maintains the formation only by



(a) Follower orientation maintained at end of path.



(b) Follower orientation not maintained at end of path.

Figure 9.2: In (a) and (b) the x, y location of the follower is the same, as the target position. However, the path taken by the right follower to the target greatly affects the final orientation of its body with respect to that of the leader.

communication. The disadvantage of this technique is that it requires perfect odometry, a requirement that cannot be fulfilled in realistic settings. If the anticipated position of the leader and the follower are computed based on imperfect, noisy odometry, the errors quickly accumulate. Moreover, as we have seen, slight differences in values of the heading can imply radically different movement vectors.

9.2.2 Combining Controllers

To allow limited-sensor robots to maintain formation while still recognizing obstacles, we propose to combine the two controllers described above, in settings where the robots' sensors are limited, but communication between robots is possible. In such settings we propose to combine two formation controllers types: A closed-loop formation-maintenance algorithm using sensors, and an open-loop algorithm using internal navigation (odometry) and communications.

We compare between two combination approaches: *multiplexing* the controllers (using one at a time), and *fusing* them (using both in parallel). The idea in combining the controllers is to offset their disadvantages, and gain from their complementary advantages.

The *multiplexing* technique works as follows. Each follower robot relies on the sensor-based algorithm until it arrives to its predefining position in the formation. We

therefore explore ways to *multiplex* and/or *fuse* closed-loop and open-loop formation control. Multiplexing is done in time, giving the alternative methods different periods of time in which they control each robot. Switching between the different methods utilized the following principle: Each robot relies on visual tracking (closed-loop control) until it is within tolerance levels of its position in the formation. When this occurs, the robot switches to *communication-based* open-loop control, and uses its sensors to scan for obstacles, while communicating with the leader. To verify its position and inhibit accumulating errors, the robot switches back to visual tracking after a fixed period of time. We also explore combining controllers by fusion, by merging the output commands of each open-loop and close-loop controllers.on (within some tolerance radius, to allow for uncertainty in sensing). When this occurs, the robot switches to the communication-based formation-maintenance behavior. Now, the robot's sensors are free and the robot can search for obstacles. The follower robot moves in this mode for a fixed period of time (which we vary in the experiments, see Section 9.3). It then switches back to the sensor-based algorithm, and the cycle repeats.

In the *fusing* technique, the robots multiplex between the open- and closed- loop controllers (otherwise, they cannot hope to detect obstacles). However, during the time when both sensor-based and communication-based controllers are active, the output commands of the controllers are fused: The average of the two controllers is taken as the output.

There are competing goals in using the open-loop controller, with both combination techniques. On one hand, the more the robots rely on open-loop control, the more they can scan for obstacles, and provide improved performance. On the other hand, the longer they remain in open-loop control, the more errors in position are accumulated (in relative positions of the robots, with respect to their teammates), and thus the formation degrades.

Thus the timeout period, which limits the amount of time robots remain under communication-based open-loop control must be determined. We take an empirical approach to determining this value. We note that it might be possible to set theoretical bounds on this value, depending on expected obstacle density. We leave this direction of research to future work.

9.3 Combined-Control Experiments

To evaluate the contribution of these approaches, we compare the *multiplexing* and *fusing* methods with their closed-loop and open-loop components, by themselves. The experiments are carried out using physical (Sony AIBO) and simulated robots.

We carried out two separate repeated-trials experiments. The evaluation has two facets. First, in Section 9.3.1, we evaluate the impact of the combination techniques on the ability of sensor-limited AIBO robots to detect obstacles, the motivation for the techniques. Second, in Section 9.3.2, we evaluate the hypothesized costs of combination, i.e., the hypothesized decrease in precision.

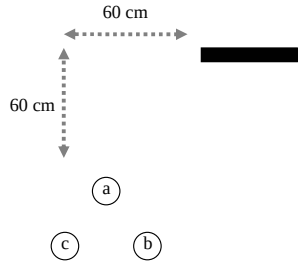
9.3.1 Detecting Obstacles

This section report on experiments carried out with physical Sony AIBO robots moving in formation. The goal of the experiment is to evaluate to what degree does controller combinations (e.g., multiplexing) allow robots to detect obstacles that may otherwise be undetectable.

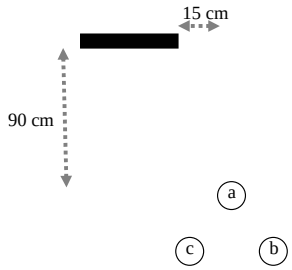
Here, three Sony AIBO ERS-7 robots were arranged in a triangular formation (Fig. 9.1-a). While operating in sensor-based separation-bearing control mode, the two followers in the rear monitor the leader using their head-mounted camera and infra-red range sensors. The robots utilize the color patch on the rear of the leader for identification, and maintain the distance and angle to it [49]. Otherwise (when using communications) they scan for obstacles and maintain the formation by communication.

The leader actively scans for obstacles. On detection, it finds a path around them that considers its own physical body, rather than the entire team (as proposed in [12]). Such a path cannot be considered safe for the followers, and indeed we intentionally place obstacles such that such a path would put them in the way of the followers. This is done so as to examine the followers' ability to detect obstacles.

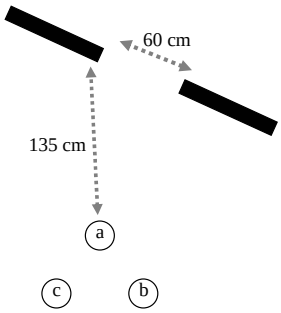
We use three different obstacle courses for this experiment (see Fig. 9.3). In the *Right* obstacle course the robots walk in a straight line; the right follower robot needs to recognize the obstacle blocking its path. The *Left* obstacle course poses the same challenge to the left follower. Finally, in the *Diagonal* course the right follower needs to recognize the right obstacle and the left follower needs to recognize the left obstacle (the leader will try to pass between the obstacles).



(a) Right.



(b) Left.



(c) Diagonal.

Figure 9.3: Three obstacle courses used in experiments with the AIBO robots.

In each of the obstacle courses, the formation was run five times, in both the visual sensing control mode, and the *multiplexing* mode, for a total of 30 runs (10 in each course). We did not experiment with the open-loop control in these experiments; as it essentially frees up all the robots sensors to focus only on the task of detecting obstacles, it serves as a theoretical upper limit. We therefore assumed that with pure communication-based control, all obstacles are detected. We note also that there are no separate results for the fusion method, because it is identical to the multiplex-

ing method in terms of time available for detecting obstacles (since then only one controller is generating output). The distinction between them is explored in Section 9.3.2.

Fig. 9.4 shows the result of the comparison between the multiplexing technique and the sensor-based formation maintenance. The X axis shows the obstacle course. The Y axis shows the fraction of the undetected obstacles over all trials, thus a lower value indicates improved performance. We can see that the multiplexing technique performs better than the sensor-based algorithm used earlier, though statistical testing shows that the difference is only moderately significant (one-tailed t-test, $p = 0.07$).

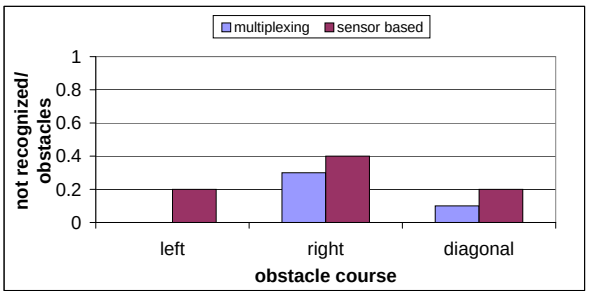


Figure 9.4: Fraction of undetected obstacles over multiple runs of each technique, in different obstacle courses.

A one-tailed t-test significance test of the experiments with the robots (above) showed that multiplexing was only moderately significantly better ($p = 0.07$). We believe this is due to the relatively small number of experiments. We thus ran additional experiments with *simulated* AIBO robots, using the player/stage environment [39], where many more trials could be run. Figure 9.5 describes the obstacle course used in the simulated environment. Each of the techniques (multiplexing, sensor-based) was run 25 times.

Figure 9.6 shows the fraction of unrecognized obstacles (Y axis) in this experiment, for each of the techniques, over 25 runs. The Y axis shows the fraction. We again see that the multiplexing approach significantly decreases the fraction of undetected obstacles. The results are significant at a level of $p = 0.00000000164$ (one-tailed t-test).

Thus both in simulation and in experiments in the real world, we see that the multiplexing approach decreases the number of undetected obstacles, though it does not

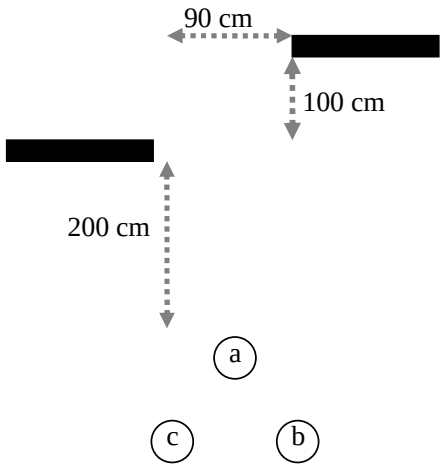


Figure 9.5: Obstacle course in the simulation experiments. The leader robot moves in straight line, but its followers must detect the obstacles on their left and right.

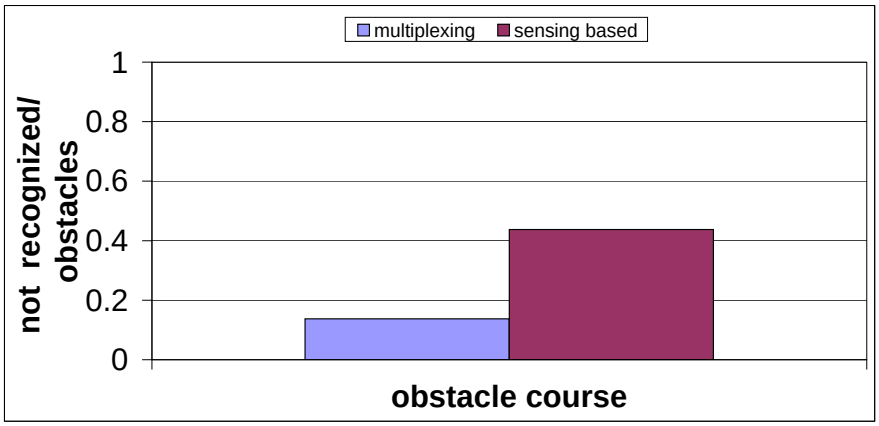


Figure 9.6: Fraction of undetected obstacles over 25 runs for each technique.

perform as the theoretical best (i.e., with perfect open-loop control and perfect knowledge of obstacles). This happens because the multiplexing technique, while giving more opportunity to the followers to detect obstacles, occasionally switches back to sensor-based closed-loop control, for correcting the accumulating odometry errors. In such cases, the follower robots cannot use its sensors to detect obstacles.

9.3.2 Formation Precision

An hypothesis underlying the combination approach is that the gains it offers (as the previous section demonstrates) will come at a price of decreased precision. The reliance on open-loop control, even if only for limited periods of time, should in principle cause some degradation in the ability of robots to position themselves in the formation. It might therefore be hypothesized that selecting fusion as the combination method may lead to improved results.

This section examines this hypothesis. We compare the quality of the formation maintenance with different formation techniques, under varying conditions of noise in movement. The quality of the formation maintenance is measured as the average absolute deviation of the follower robots from their ideal location in the formation. Our expectation is that combination would fare worse than its constituent techniques, especially with increased noise.

We compare the multiplexing and fusing combination techniques, presented earlier, to the two constituent techniques: Open-loop formation control (*communication-based maintenance*), and closed-loop formation control (*visual formation maintenance*). For the combination technique, we use a timeout of 8 seconds for the period in which the robot scans for obstacles, relying only on open-loop control. The timeout was determined empirically, but experimenting with different timeout values.

Precise positioning in formations is relatively easy when the formation moves in a straight line. It becomes more difficult to achieve in realistic settings, when formation (and robots) have to turn. We thus examine the precision resulting from each formation control technique, when the angle of the leader's turn is varied. In the following experiments, the leader robot moves in a straight line for 20 seconds and then turns in place and proceeds. We control the leader's turn angle ($0, 15, 30, 90$ degrees), and measure the resulting position errors in the followers once the turn is complete.

Given that we wanted to control the amount of uncertainty in the movements of the robots, we chose to run these experiments in simulation. We used a Gaussian to model the noise in the movements of the robots, at several qualitative levels of $0\%, 20\%$ and 40%. The percentages signify the uncertainty in terms of standard deviation, i.e., a level of 20% Gaussian noise means that the standard deviation of target value X will be 20% of X.

Figures 9.7, 9.8 and 9.9 show the results of these experiments, for noise levels 0%, 20% and 40%, respectively. In these figures, the X axis shows the sharpness of the leader's turn in degrees. The Y axis represents the average absolute deviation (error) of the follower robots from their ideal position in the formation. The line marked *visual* shows the results of the closed-loop sensor-based visual maintenance. The line marked *communication* corresponds to the open-loop communication-based maintenance. The lines marked *multiplexing* and *fusing* correspond to the multiplexing and fusing (the combination approaches). Each one of the points is an average over 40 data points (20 runs, two follower robots).

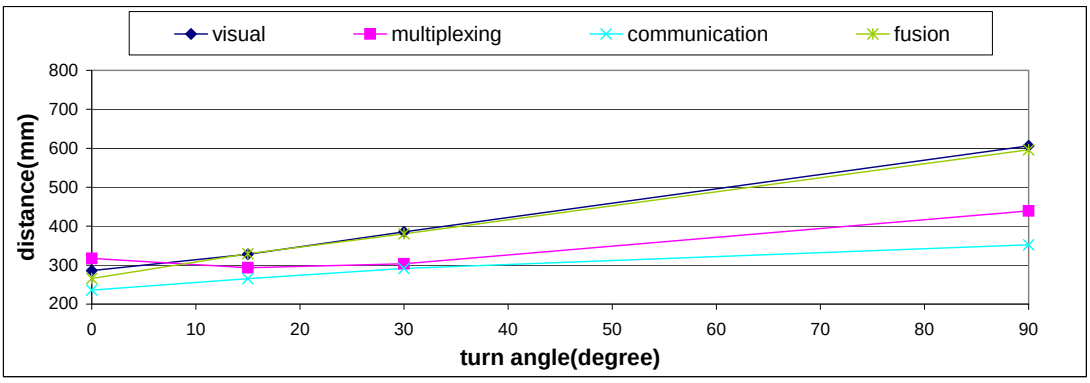


Figure 9.7: Deviation from the ideal position in formation vs. the turn angle, with no uncertainty in movement/odometry.

The results in Figure 9.7 show that the worst results are achieved by the visual formation maintenance (closed-loop control, by itself), and by the fusing technique. As the sharpness of the turn increases, use of these techniques lead to increasing errors in the positioning of the follower robots. In contrast, the multiplexing and communication-based maintenance are quite similar in most cases and they have the best results. This happens since in a world where there are fewer odometry errors, a technique that is based on mathematical calculations can calculate the exact location where the follower robot should be and with accurate odometry (i.e. lack of noises) can lead the follower robot to its ideal position in the formation.

However, as odometry noise levels increase, we can see that visual formation maintenance achieves good performance, except for the sharpest (90-degree) turn. Similarly, the fusing technique improves as well, and achieves good performances even in

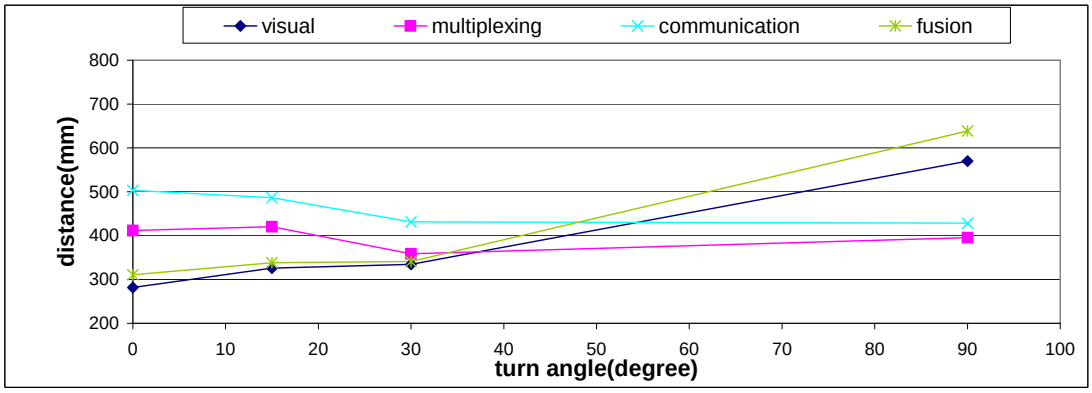


Figure 9.8: Deviation from the ideal position in formation vs. the turn angle, with uncertainty levels set at 20%.

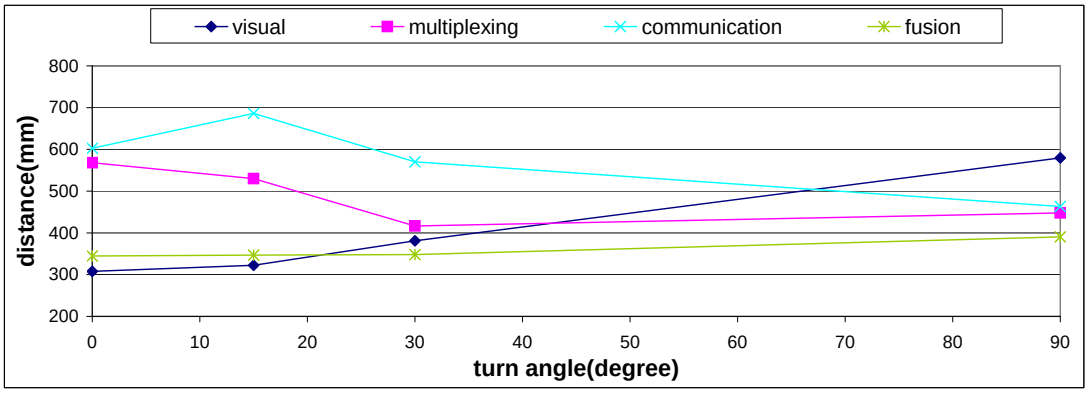


Figure 9.9: Deviation from the ideal position in formation vs. the turn angle, with uncertainty levels set at 40%.

sharp turns. Indeed, the gap between visual maintenance and communication-based maintenance increases (see Figures 9.8, 9.9).

Thus one conclusion of these experiments is that the two constituent controllers work well, but not for the same settings. In sharp turns, open-loop control is best (even at higher noise settings). But for robustness to noise, closed-loop control is preferable.

We remind the reader that our hypothesis was that the combination variants would result in decreased precision compared to their constituents. The intuition was that as the combination methods gain the ability to detect obstacles, they sacrifice precision.

The results show that instead, the multiplexing technique emerges as a good con-

troller when the odometry noise level decreases, and is robust in sharp turns as well (a benefit compared to the communication-based controller). It is indeed never the best performer, but it is also never the worst. In fact, this technique seems to be robust both to the noise settings (like its visual maintenance constituent) and to the turn angle sharpness (like the communication-based maintenance constituent). These results thus provide evidence that for robustness, multiplexing controllers (alternating between them) may be a good strategy.

We additionally see that the fusing technique performs well and is robust when the odometry noise level increases. Thus if the robots can recognize its odometry noise level in the environment, it can switch from fusing to multiplexing, and vice versa (depending the noise) and behave ideally. This second level of multiplexing, however, is beyond the scope of this thesis.

9.4 Robust Formations: Conclusions

We introduce here a combination approach (involving either multiplexing or fusing of controllers) to formation maintenance. The approach combines two different formation maintenance controllers: One open-loop and one closed-loop. Our technique helps to maintain a formation and detect obstacles when the robots' sensors are limited, and therefore cannot easily detect obstacles and track their peers at the same time.

In experiments with real and simulated Sony AIBO robots, we have found that the combination approach decreases the number of undetected obstacles (compared to the closed-loop visual formation maintenance controller), and maintains the precision of the formation more robustly than either of its constituent controllers by itself.

We also conclude that the level of odometry errors influences the best performer between the multiplexing and fusing methods. In particular, we find that the multiplexing technique is better when the odometry error decreases, and that the fusing is preferable otherwise. Thus, we propose to switch between those two methods when the robots know their odometry error level (e.g., use multiplexing in flat surfaces, and fusing in rocky terrains. Both techniques allow the robots to use their sensors to detect obstacles and survey their surroundings, something not possible with their constituent methods.

Chapter 10

Future Directions and Final Remarks

We summarize the key contributions of this thesis in Section 10.1. We discuss future directions for this research in Section 10.2.

10.1 Summary of Key Contributions

Our first contribution in this work is a formalization of the multi-robot patrol problem and its frequency optimization goals. We have discussed point-visit frequency criteria according to which a patrol mission can be evaluated. We additionally examine an independent measure, of event response time.

In the first part of this dissertation, we focus on multi-robot patrolling in areas enclosed by polygons. We describe a spanning-tree patrolling (STP) approach for area patrolling. STPis based on finding a minimal Hamiltonian cyclic path in a non-uniform, directional, terrain. Based on this cyclic path, we analytically demonstrate that an algorithm that assigns locations to the robots along the path such that the time necessary to arrive to those locations is minimal, and patrolling from those locations create a uniform maximal-frequency patrol. Last, we show that this algorithm is robust in the sense that it guarantees patrol at uniform frequency as long as at least one robot works properly.

We then turn to discuss the problem of allocation robots to handling events along the patrol path. Given the projected duration of handling an event, and the deadline by which handling the event must complete, we investigate different conditions and different methods for allocating robots to events. We describe a set of algorithms for

dividing the time it takes to handle the event between the robots, depending on the time constraint for finishing handling the event.

In the second part of the dissertation we focus on patrolling along an polyline, e.g., a two-ended fence. Because cyclic paths are not possible in polyline patrolling, there are inherent challenges to maintain uniformity of point visit frequencies, and other performance criteria.

We first examine individual and coordinated patrolling algorithms. We show that in general, the synchronized approach to multi-robot patrolling outperforms the individual, unsynchronized methods in response-time minimization. We then introduce a generalized coordinated patrolling method, Frequency-based Overlapping Patrol (FOP) in which robots have patrolling trajectories that intersect in space, but not in time; their assigned segments overlap. We analyze the performance of FOP methods in depth, with respect to different performance goals, and investigate key trade-offs.

We develop a more realistic analytical treatment of the performance of the algorithm with multiple patrolling robots. We develop a realistic model of robot motion, that considers real-world uncertainties and accumulating motion errors. We mathematically analyze the model, and then use it to predict the empirically observed behavior of robots patrolling with different overlapping factors; robots that have not been developed with the model in mind. The results of extensive experiments show that the new model is not only more accurate relative to previous models, but is also accurate on an absolute scale.

We have shown that the selection of an optimal overlap factor depends significantly on the segments chosen, as well as the parameters of motion (e.g., turning time, velocity errors). In addition, edge segments result in qualitatively different performance, compared to middle segments. We have shown that in many cases in the middle segments, the increase in overlap results in improved uniformity, without hurting the average and under-bounding frequencies. However, this comes at a cost of greater neglect (and thus worse performance) in the edge segments.

Lastly, we consider the case where the patrolling unit is not a single robot, but a formation of multiple robots that move together, maintaining a fixed geometric shape. To free up sensors for surveillance and obstacle detection we introduce here a *multiplexing* formation control technique that combines sensor-based closed-loop formation-maintenance, and communication-based open-loop formation maintenance, by either

fusion or multiplexing in time.

Our technique helps to maintain a formation and detect obstacles when the robots' sensors are limited, and cannot easily detect obstacles and track their peers at the same time. In experiments with real and simulated Sony AIBO robots, we have found that the multiplexing approach decreases the number of undetected obstacles (compared to the closed-loop visual formation maintenance controller), and maintains the precision of the formation more robustly then either of its constituent controllers by itself. Moreover, we have shown that the power of the technique comes from the decision to switch (multiplex) controllers, rather than fuse their actions. The experiments show that this allows robots to better utilize their sensors for detecting and avoiding obstacles, while still maintaining their positions in the formation.

10.2 Future Directions

There are still several areas that are left open, and we plan to pursue them in future work.

Priorities in patrolling. An important requirement in applications of patrolling is being able to set priorities to different portions of the patrolling work area, be in an area enclosed by a polygon, or a polyline. These priorities are to be used by the patrolling algorithms such that the resulting under-bounded and average frequencies in different segments (or sub-areas) are proportional to the priorities.

Heterogeneous Robots. We have not considered, in this dissertation, the case of heterogeneous robots, e.g., in terms of their maximal velocities or sensorial capabilities. This would be another important consideration, especially in assigning robots to respond to events in patrolling.

Uncertainty in Event Handling Durations. Our events are defined as tuples composed of location, handling duration, and deadline. In reality, many applications would not be able to provide these values with certainty. Instead, a distribution over possible values would be available, at best. The event handling techniques would necessarily need to be changed to account for such uncertainty.

Velocity Ranges. In considering non-uniform terrains, we have modeled changes to the velocity in which robots travel in different segments in the polyline, or sub-areas in the polygon. However, it is more realistic to model different terrains as having different *maximal* velocities (i.e., having different velocity ranges). The algorithms would then be able to slow down (and possibly, speed up) robots as needed to maintain improved uniformity of point visit frequency, etc.

Physical Constraints. The physical properties of robots should be better taken into account. For instance, we assumed thus far that robots can turn anywhere on the polyline, which is not necessarily true. Moreover, we would like to take into consideration turns when we generate cyclic paths for area patrol, e.g., to minimize the number of turns.

Patrolling in Formations. In this dissertation, the values for the multiplexing interval were determined empirically. In the future, we hope to provide an analytical framework to help guide selection of this value. Also, the multiplexing technique complements existing work which computes obstacle-free paths for all team-members [12]. Multiplexing offers opportunity for increasing the effectiveness of such techniques, by providing them with more sensorial capabilities.

Bibliography

[1] F. R. Abate. *The Oxford Dictionary and Thesaurus: The Ultimate Language Reference for American Readers*. Oxford Univ. Press, 1996.

[2] N. Agmon, N. Hazon, and G. A. Kaminka. Constructing spanning trees for efficient multi-robot coverage. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA-06)*, 2006.

[3] N. Agmon, N. Hazon, and G. A. Kaminka. The giving tree: Constructing trees for efficient offline and online multi-robot coverage. *Annals of Math and Artificial Intelligence*, 2009.

[4] N. Agmon, G. A. Kaminka, and S. Kraus. Multi-robot fence patrol in adversarial domains. In *Proceedings of the Tenth Conference on Intelligent Autonomous Systems (IAS-10)*. IOS Press, 2008.

[5] N. Agmon, S. Kraus, and G. A. Kaminka. Multi-robot perimeter patrol in adversarial settings. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA-08)*, 2008.

[6] N. Agmon, V. Sadov, G. A. Kaminka, and S. Kraus. The impact of adversarial knowledge on adversarial planning in perimeter patrol. In *Proceedings of the Seventh International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-08)*, volume 1, pages 55–62, 2008.

[7] M. Ahmadi and P. Stone. A multi-robot system for continuous area sweeping tasks. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA-06)*, 2006.

[8] A. Almeida, G. L. Ramalho, H. P. Santana, P. Tedesco, T. R. Menezes, V. Corruble, and Y. Chevaleyre. Recent advances on multi-agent patrolling. In *Advances in Artificial Intelligence SBIA 2004: 17th Brazilian Symposium on Artificial Intelligence*, volume 3171 of *Lecture Notes in Computer Science*, pages 474–483. Springer-Verlag, 2004.

[9] T. Balch and R. Arkin. Behavior-based formation control for multi-robot teams. *IEEE Transactions on Robotics and Automation*, 14(6):926–939, 1998.

[10] T. Balch and M. Hybinette. Social potentials for scalable multirobot formations. In *Proceedings of IEEE International Conference on robotics and automation (ICRA-00)*, 2000.

[11] D. M. Carroll, C. Nguyen, H. R. Everett, and B. Frederick. Development and testing for physical security robots. In *SPIE*, Orlando, 2005.

[12] X. Chen and Y. Li. Smooth formation navigation of multiple mobile robots for avoiding moving obstacles. *International Journal of Control, Automation, and Systems*, 4(4):466–479, August 2006.

[13] Y. Chevaleyre. Theoretical analysis of the multi-agent patrolling problem. In *Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT)*, 2004.

[14] Y. Chevaleyre, F. Sempé, and G. L. Ramalho. A theoretical analysis of multi-agent patrolling strategies. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-04)*, 2004. Short Paper.

[15] H. Choset. Coverage for robotics—a survey of recent results. *Annals of Mathematics and Artificial Intelligence*, 31:113–126, 2001.

[16] J. Colegrave and A. Branch. A case study of autonomous household vacuum cleaner. In *AIAA/NASA CIRFFSS*, 1994.

[17] T. Corman, C. Leiserson, and R. Rivest. *Introduction to Algorithms*. MIT Press, 1990.

[18] N. Correll and A. Martinoli. Robust Distributed Coverage using a Swarm of Miniature Robots. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA-07)*, pages 379 – 384, 2007.

[19] N. Correll, S. Rutishauser, and A. Martinoli. Comparing Coordination Schemes for Miniature Robotic Swarms: A Case Study in Boundary Coverage of Regular Structures. In *The 10th International Symposium on Experimental Robotics (ISER)*, Springer Tracts in Advanced Robotics, 2006.

[20] J. P. Desai. A graph theoretic approach for modeling mobile robot team formations. *Journal of Robotic Systems*, 19(11):511–525, 2002.

[21] J. P. Desai, J. P. Ostrowski, and V. Kumar. Modeling and control of formations of nonholonomic mobile robots. *IEEE Transactions on Robotics and Automation*, 17(6):905–908, 2001.

[22] M. B. Dias and A. Stentz. A free market architecture for distributed control of a multirobot system. In *Proceedings of the Sixth Conference on Intelligent Autonomous Systems (IAS-6)*, pages 115–122, 2000.

[23] M. B. Dias, R. M. Zlot, N. Kalra, and A. Stentz. Market-based multirobot coordination: a survey and analysis. *Proceedings of the IEEE*, 94(7):1257–1270, 2006.

[24] R. Dougherty, V. Ochoa, Z. Randles, and C. Kitts. A behavioral control approach to formation-keeping through an obstacle field. In *Proc. of the IEEE Aerospace Conference*, volume 1, pages –175, March 2004.

[25] R. Dougherty, V. Ochoa, Z. Randles, and C. Kitts. A behavioral control approach to formation-keeping through an obstacle field. In *proc. of the 2004 IEEE Aerospace Conference*, 2004.

[26] Y. Elmaliach, N. Agmon, and G. A. Kaminka. Multi-robot area patrol under frequency constraints. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA-07)*, 2007.

[27] Y. Elmaliach, N. Agmon, and G. A. Kaminka. Multi-robot area patrol under frequency constraints. *Annals of Math and Artificial Intelligence*, 2008.

[28] Y. Elmaliach and G. A. Kaminka. Robust multi-robot formations under human supervision and control. *Journal of Physical Agents*, 2(1):31–52, 2008.

[29] Y. Elmaliach, A. Shiloni, and G. A. Kaminka. Frequency-based multi-robot fence patrolling. Technical Report MAVERICK 2008/01, Bar Ilan University, Computer Science Department, MAVERICK Group, 2008.

[30] Y. Elmaliach, A. Shiloni, and G. A. Kaminka. A realistic model of frequency-based multi-robot fence patrolling. In *Proceedings of the Seventh International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-08)*, volume 1, pages 63–70, 2008.

[31] R. Fierro, A. K. Das, V. Kumar, and J. P. Ostrowski. Hybrid control of formations of robots. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA-01)*, 2001.

[32] J. Fredslund and M. J. Matarić. A general algorithm for robot formations using local sensing and minimal communication. *In IEEE Transactions on Robotics and Automation, Special Issue on Multi Robot Systems*, 18(5):837–846, October 2002.

[33] J. Fredslund and M. J. Mataric. A general algorithm for robot formations using local sensing and minimal communications. *IEEE Transactions on Robotics and Automation*, 18(5):837–846, 2002.

[34] J. Fredslund and M. J. Matarić. Robots in formation using local information. In *proc. of the 7th International Conference on Intelligent Autonomous Systems (IAS-7)*, Marina del Rey, California, USA, March 25-27 2002.

[35] Friendly Robotics®, Ltd. Friendly robotics vacuum cleaner. http://www.friendlyrobotics.com/friendly_vac/.

[36] Y. Gabriely and E. Rimon. Spanning-tree based coverage of continuous areas by a mobile robot. *Annals of Mathematics and Artificial Intelligence*, 31:77–98, 2001.

[37] Y. Gabriely and E. Rimon. Competitive on-line coverage of grid environments by a mobile robot. *Comp. Geometry*, 24:197–224, 2003.

[38] D. W. Gage. Command control for many-robot systems. In *The Nineteenth Annual AUVS Technical Symposium (AUVS-92)*, 1992.

[39] B. P. Gerkey, R. T. Vaughan, and A. Howard. The player/stage project: Tools for multi-robot and distributed sensor systems. In *Proceedings of the International Conference on Advanced Robotics*, pages 317–323, Coimbra, Portugal, Jul 2003.

[40] A. Girard, A. Howell, and J. Hedrick. Border patrol and surveillance mission using multiple unmanned air vehicles. In *Proceedings of the 43rd IEEE Conference on Decision and Control*, pages 620–625, 2004.

[41] M. Golfarelli, D. Maio, and S. Rizzi. A task-swap negotiation protocol based on the contract net paradigm. Technical Report 005-97, CSITE, 1997.

[42] Y. Guo, L. Parker, and R. Madhavan. Towards collaborative robots for infrastructure security applications. In *Proceedings of the 2004 International Symposium on Collaborative Technologies and Systems (CTS-04)*, pages 235–240, 2004.

[43] Y. Guo and Z. Qu. Coverage control for a mobile robot patrolling a dynamic and uncertain environment. In *Proceedings of the Fifth World Congress on Intelligent Control and Automation (WCICA-04)*, volume 6, pages 4899–4903, 2004.

[44] N. Hazon and G. Kaminka. On redundancy, efficiency, and robustness in coverage for multiple robots. *Robotics and Autonomous Systems*, 2008.

[45] N. Hazon and G. A. Kaminka. Redundancy, efficiency, and robustness in multi-robot coverage. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA-05)*, 2005.

[46] N. Hazon, F. Mieli, and G. A. Kaminka. Towards robust on-line multi-robot coverage. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA-06)*, 2006.

[47] S. Hedberg. Robots cleaning up hazardous waste. *AI Expert*, pages 20–24, 1995.

[48] B. Jung and G. Sukhatme. Tracking targets using multiple robots: The effect of environment occlusion. *Autonomous Robots*, 13(3), 2002.

[49] G. A. Kaminka, R. Schechter-Glick, and V. Sadov. Using sensor morphology for multi-robot formations. *IEEE Transactions on Robotics*, pages 271–282, 2008.

[50] H. W. Kuhn. The hungarian method for the assignment problem. In *Naval Research Logistics Quarterly*, volume 2, pages 83–97, 1995.

[51] M. Lemay, F. Michaud, D. Létourneau, and J.-M. Valin. Autonomous initialization of robot formations. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA-04)*, 2004.

[52] A. Machado, G. Ramalho, J.-D. Zucker, and A. Drogoul. Multi-agent patrolling: An empirical analysis of alternative architectures. In *Third International Workshop on Multi-Agent Based Simulation (MABS-02)*, Lecture Notes in Computer Science, 2002.

[53] MAVERICK. The MAVERICK Group movies page, Computer Science department, Bar Ilan University; last checked: Feb 24, 2008. http://www.cs.biu.ac.il/~maverick/Movies/, 2005.

[54] F. Michaud, D. Létourneau, M. Gilbert, and J.-M. Valin. Dynamic robot formations using directional visual perception. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2002.

[55] A. I. Mourikis and S. I. Roumeliotis. Optimal sensor scheduling for resource constrained localization of mobile robot formations. *IEEE Transactions on Robotics*, 22(5):917–931, October 2006.

[56] J. Nicoud and M. Habib. The pemex autonomous demining robot: Perception and navigation strategies. In *IROS*, pages 419–424, 1995.

[57] P. Ogren and N. E. Leonard. Obstacle avoidance in formation. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, Taipei, Taiwan, 2003.

[58] P. Paruchuri, J. P. Pearce, M. Tambe, F. Ordonez, and S. Kraus. An efficient heuristic approach for security against multiple adversaries. In *Proceedings of the Sixth International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-07)*, 2007.

[59] A. Ryan, X. Xiao, S. Rathinam, J. Tisdale, M. Zennaro, D. Caveney, R. Sengupta, and K. Hedrick. A modular software infrastructure for distributed control of collaborating UAVs. In *Proceedings of the AIAA Conference on Guidance, Navigation, and Control*, Keystone, 2006.

[60] R. G. Smith. The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, C-29(12):1104–1113, 1981.

[61] I. A. Wagner, M. Lindenbaum, and A. M. Bruckstein. Efficiently searching a graph by a smell-oriented vertex process. *Annals of Mathem and Artificialatics Intelligence*, 24:211–223, 1998.

[62] I. A. Wagner, M. Lindenbaum, and A. M. Bruckstein. Distributed covering by ant-robots using evaporating traces. *IEEE Transactions on Robotics and Automation*, 15(5):918–933, 1999.

[63] I. A. Wagner, M. Lindenbaum, and A. M. Bruckstein. MAC vs. PC: Determinism and randomness as complementary approaches to robotic exploration of continuous unknown domains. *International Journal of Robotics Research*, 19(1):12–31, 2000.

[64] K. Williams and J. Burdick. Multi-robot boundary coverage with plan revision. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA-06)*, 2006.

[65] P. R. Wurman, R. D'Andrea, and M. Mountz. Coordinating hundreds of cooperative, autonomous vehicles in warehouses. *AI Magazine*, Spring, 2008.

[66] V. M. Yanovski, I. A. Wagner, and A. M. Bruckstein. A distributed ant algorithm for efficiently patrolling a network. *Algorithmica*, 37:165–186, 2003.

[67] D. Zwillinger, editor. *CRC Standard Mathematical Tables and Formulae*, chapter 4.3, pages 312–314. CRC press, 30th edition, 1995.