Placement and Motion Planning Algorithms for Robotic Sensing Systems

A DISSERTATION
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

Pratap Rajkumar Tokekar

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
Doctor of Philosophy

Prof. İbrahim Volkan İşler

October, 2014

Acknowledgements

First and foremost, I would like to express my sincere gratitude towards my adviser, Prof. Volkan Isler. Thank you for believing in me, for your constant support and encouragement, for all the efforts you have put in me, and for being a true role-model and inspiration. I could not have asked for a better adviser. I am very proud to have been your student and hope to keep learning from you. Special thanks to Dr. Hilal Isler for your kind words of encouragement and support over the years.

I am grateful for the guidance and invaluable feedback from my committee members Prof. Ravi Janardan, Prof. Nikolaos Papanikolopoulos, and Prof. Sonia Martinez. I would also like to thank Prof. Stergios Roumeliotis, who although could not be on my final defense committee, has been very helpful and from whom I've learned a lot.

Thanks to all my collaborators and co-authors for their contributions to this dissertation. I am indebted to the financial support from National Science Foundation grants #1317788, #1111638, #0917676 and #0936710. Special thanks to Dr. Antonio Franchi for hosting me at the Max Planck Institute in May 2013, and for your kind help and support since then. The work presented in Chapter 5 is a direct result of this collaboration (and many Skype conversations). I hope we can continue to collaborate in the future.

I started at Minnesota as a Masters in Electrical Engineering student with no previous exposure to algorithms. The excellent courses I took at Minnesota have played a big role in making this dissertation possible. I would especially like to thank Prof. Janardan (Advanced Algorithms and Computational Geometry), Prof. Roumeliotis (Sensing and Estimation in Robotics), and Prof. Isler (Special Topics: Robotic Sensor Networks) for the most enjoyable and enriching courses I've taken. I would also like to thank the members of the Robotic Sensor Networks lab for all the fruitful, sometimes frustrating, but ultimately rewarding discussions in the journal club.

I am grateful for the excellent office staff of the Computer Science department. In particular, thanks to Georganne Tolaas, Peggy Stewart, and Sara Grothe for your help and patience, and for making all my administrative tasks hassle-free and very enjoyable.

The Robotic Sensor Networks lab has been a home for the past six years thanks to the wonderful people who made it so. Thank you Onur Tekdas, Nikhil Karnad, Deepak Bhadauria, Patrick Plonski, Deniz Ozsoyeller, Joshua Vander Hook, Narges Noori, Elliot Branson, Alessandro Renzaglia, Gabriel Oliveira, Ubaldo Ruiz, Roman Ripp, Andy Erickson, Nikolaos Stefas and Pravakar Roy. Thanks for all the lively discussions, for all the help, and for all the Village Wok dinners after (fun and occasionally frustrating) field experiments. I am proud of the research we do and the culture in the lab, and it has been a privilege working with all of you.

Thanks to all my friends in Minnesota for the fun times (with a sprinkling of long discussions about the ups and downs of life and research): Vineet Bhatawadekar, Vinit Padhye, Anup Holey, Shreyas Bhaban, Neha Kulkarni, Mikhil Masli, Amruta Bamnikar, Vineeth Mekkat, Ram Varma, Sayan Ghosal, Raamesh Deshpande, Shruti Patil, Pushkar Nandkar, and Nachiket Gokhale. Thanks to friends who were never more than a phone-call away: Amey Deshpande, Shriram Devi, Amey Bhagwat, Chetan Tonde, Prasanna Kulkarni, Mayur Tailor, Rohit Sawkar, Sonal Joshi, Mandar Jadhav, Rakesh Khoje, Mithila Burute, and all members of the *Global Katta* and the *Pune Katta*. Thanks for being there!

Special thanks to Starbucks for being a constant companion and for much more.

Finally, I cannot fully express my gratitude towards my family. Their confidence in me has kept me going these past six years and more. They have always encouraged me and given me the freedom and opportunities to pursue my goals. It is impossible to overstate how important their endless support and love has been in making this dissertation possible. Thank you to my parents, Rajkumar and Sheela Tokekar, my two lovely sisters, Urmila Tokekar and Ashwini Prabhu, and special thanks to my grandmother (Aai) who couldn't be there to see me complete my Ph.D. but has been just as instrumental in making it possible.

Dedication

To Akka, Tai, Dad and Maa.

Abstract

Recent technological advances are making it possible to build teams of sensors and robots that can sense data from hard-to-reach places at unprecedented spatio-temporal scales. Robotic sensing systems hold the potential to revolutionize a diverse collection of applications such as agriculture, environmental monitoring, climate studies, security and surveillance in the near future. In order to make full use of this technology, it is crucial to complement it with efficient algorithms that plan for the sensing in these systems. In this dissertation, we develop new sensor planning algorithms and present prototype robotic sensing systems.

In the first part of this dissertation, we study two problems on placing stationary sensors to cover an environment. Our objective is to place the fewest number of sensors required to ensure that every point in the environment is covered. In the first problem, we say a point is covered if it is seen by sensors from all orientations. The environment is represented as a polygon and the sensors are modeled as omnidirectional cameras. Our formulation, which builds on the well-known art gallery problem, is motivated by practical applications such as visual inspection and video-conferencing where seeing objects from all sides is crucial. In the second problem, we study how to deploy bearing sensors in order to localize a target in the environment. The sensors measure noisy bearings towards the target which can be combined to localize the target. The uncertainty in localization is a function of the placement of the sensors relative to the target. For both problems we present (i) lower bounds on the number of sensors required for an optimal algorithm, and (ii) algorithms to place at most a constant times the optimal number of sensors.

In the second part of this dissertation, we study motion planning problems for mobile sensors. We start by investigating how to plan the motion of a team of aerial robots tasked with tracking targets that are moving on the ground. We then study various coverage problems that arise in two environmental monitoring applications: using robotic boats to monitor radio-tagged invasive fish in lakes, and using ground and aerial robots for data collection in precision agriculture. We formulate the coverage problems based on constraints observed in practice. We also present the design of prototype robotic systems for these applications. In the final problem, we investigate how to optimize the low-level motion of the robots to minimize their energy consumption and extend the system lifetime.

This dissertation makes progress towards building robotic sensing systems along two directions. We present algorithms with strong theoretical performance guarantees, often by proving that our algorithms are optimal or that their costs are at most a constant factor away from the optimal values. We also demonstrate the feasibility and applicability of our results through system implementation and with results from simulations and extensive field experiments.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Today assembly lines throughout the world employ robots to carry out a variety of complex tasks with high speed and precision. The industrial robot has been the most widespread and successful accomplishment for robotics researchers and engineers. Replicating this success outside of the controlled, industrial settings on the other hand, has remained elusive. Fortunately, this situation is changing. We are now starting to witness early success of robots deployed "in the wild." An autonomous underwater vehicle from Bluefin robotics was used to search over 850 sq. km. of the ocean floor to help locate a missing airplane [4]. Google's self-driving cars have already logged over 700,000 autonomous miles [5]. Telepresence robots are available for purchase from a number of companies [6]. This early success is built on the technological advances in sensing, computing, and actuation and fundamental robotics research on estimation, perception, and motion planning. We are building robots that have the potential to carry out complex tasks with minimal human intervention. In order to realize this potential, what we need are new algorithms that make efficient use of the robot's capabilities and enable a robot to carry out complex tasks. In this dissertation, we make progress towards this goal by studying the sensing capability of robots and developing planning algorithms for robotic sensor systems.

Sensing becomes critical when a robot has to carry out a task in an uncontrolled or dynamic environment. Sensing the environment and reacting to this information are defining capabilities of a robot. The capability of a robot to execute a task is limited by the on-board sensors [7]. Consequently it is crucial to understand the performance limits for sensing and develop algorithms that can achieve this limit.

There are two broad categories of sensing: proprioceptive and exteroceptive. Proprioceptive sensing is primarily used for determining the state of the robot in a suitable reference frame. Accelerometers, rate gyros, compass, odometers are examples of proprioceptive sensors. Exteroceptive sensing, on the other hand, refers to sensing the environment exterior to the robot. Cameras, radio antennas, laser range-finders are examples of exteroceptive sensors. We focus on planning for exteroceptive sensing in this dissertation.

Exteroceptive sensors typically have a strong underlying geometric structure. With a better understanding of this structure, efficient robotic sensing systems can be designed. We can answer fundamental questions such as: How many sensors are required to complete a given task? Where should the sensors be located? How should the robots plan their movements in order to improve the sensing quality? In this dissertation, we study these questions for a number of sensing problems. Before we introduce the specific problems, we will discuss two prominent tasks that arise in many robotic sensing applications and the associated challenges in completing these tasks.

1.1 Applications, Typical Tasks and Challenges

Robotic sensing systems can be broadly categorized with respect to the main tasks they are designed to carry out. Of these, two tasks, coverage and target tracking, are among the most common. In the former, we want to ensure every point in the environment is sensed (i.e., covered) by the sensors. In contrast, in the latter task we want to ensure a smaller subset of the environment, typically corresponding to regions likely occupied by one or more targets-of-interest, is covered by the sensors. The sensors may have to reconfigure in order to track the moving targets, where as a static placement may suffice for a coverage task. We discuss each of the task in more details next.

1.1.1 Coverage

For coverage, we start with a layout of the environment given in some suitable representation such as a building floor-plan. Typically, we also know the relevant characteristics of the sensors (e.g., sensing range and failure rates) and the robots (e.g., maximum speed and battery lifetime). The coverage task requires that every point in the environment or in a pre-determined region-of-interest within the environment is sensed by one or more sensors.

Installing cameras for security in supermarkets or for video conferencing are common examples of coverage applications. Coverage is also frequently seen in environmental monitoring applications. For example, Carter et al. [8] covered an area of over 1500 hectares with motion detection cameras placed at 76 locations in order to monitor the spatiotemporal changes in tiger habitat. Coverage will also play a big role in emerging applications such as indoor maps and location-aware services, which use a combination of stationary sensors (e.g., Apple's iBeacon) and mobile devices (e.g., cellphones).

The coverage requirement can vary depending on the application. Some applications, such as surveillance, may require all points in the environment to always be covered. Other applications, such as robotic vacuum cleaning, may tolerate the environment to be covered over time or covered periodically. Former scenarios are suitable to be solved by installing a network of stationary sensors whereas the latter are better solved using fewer but mobile sensors. A combination of the two is also possible. For example, Tekdas et al. [9] showed how to use a mobile robot to gather data from stationary sensors placed in the environment.

The standard coverage problem can be formulated as an optimization problem: *Given the layout of an environment, find a trajectory for a robot to sense every point in the environment in the least amount of time (or distance or energy).* Equivalently, for placing stationary sensors we can formulate the coverage problem as: *Given the layout of an environment, find a placement for the fewest number of stationary sensors to sense every point in the environment.*

In this formulation we treat coverage as a constraint and time spent or number of sensors as resources to be optimized. Depending on the task at hand, we can formulate the dual version in which we are given a time budget or a fixed number of sensors and we wish to maximize the number of points covered. In this dissertation, we study problems for both formulations.

Solving coverage problems are challenging for a number of reasons. For an example, consider Figure 1.1 which shows regions that would be covered if an omnidirectional camera were to be placed at either location x or y. Although the locations are very

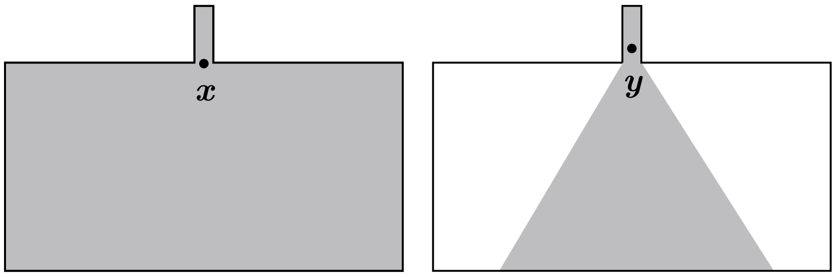


Figure 1.1: Visibility regions of nearby points can be significantly different.

close to each other, the regions that they cover differ vastly. Thus, naively applying general discretization or optimization techniques may not be suited for solving coverage problems. Although, brute-force approaches may work in restricted scenarios, they do not scale as environments become more complex and number of sensors increase leading to difficult combinatorial problems. In fact, as we will see in Chapter 2 many coverage problems are NP-hard. Nevertheless, we show how to devise efficient approximation algorithms for many coverage problems.

1.1.2 Active Target Tracking

Achieving a complete coverage of the environment may be unnecessary when we are interested in sensing a small number of targets in the environment. We can restrict our sensing to only those regions of interest which are likely occupied by the targets at any time. These regions of interest will change as the targets move and the sensors may have to reconfigure themselves correspondingly. We refer to this scenario as the target tracking task.

Target tracking is an important sensing task that appears in a diverse number of applications. For example, fish biologists study the movement and aggregation of invasive fish by tracking radio-tagged fish [10]. Researchers have been studying algorithms for tracking and capturing tumbling satellites using robotic manipulators in space [11]. Target tracking also features as a prominent task in emerging applications such as robotic assembly of furniture [12].

The typical objective for target tracking is to accurately estimate the position of one or more mobile targets and/or maintain targets within some robot's sensing range for as long as possible. There are two complimentary aspects to the problem. First, we have algorithms that are used to process and combine the sensor measurements in order to accurately estimate the location of the targets. Second, we have algorithms that decide where the robot should obtain the measurements from. In this dissertation, we will focus on the latter aspect, distinguished as *active* target tracking. In the special case when the targets are stationary at an unknown location, the problem is referred to as active target localization.

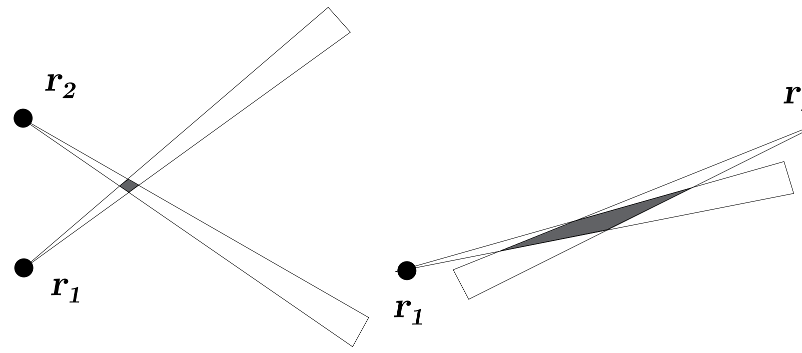


Figure 1.2: Bearing measurements from robots r_1 and r_2 are intersected to obtain an estimate of the target's position. The robot's position relative to the target determines the area of intersection which is a measure of the uncertainty in the estimate.

Actively planning while tracking is crucial since often the accuracy in estimating the target's location is a function of the sensors' locations relative to that of the target. For example, Figure 1.2 shows two instances in which a pair of robots, r_1, r_2, obtain measurements from different positions relative to the target. Each measurement is a cone which is guaranteed to contain the true target location. Intersecting the cones gives an estimate of the target's location. The area of intersection varies greatly for different relative positions. Since the true location of the target is not known a priori, active tracking algorithms will have to plan in an online fashion. These algorithms must be robust against noisy measurements. The motion limitations on the robots (e.g., speeds relative to the targets) also affects their ability to track the targets, possibly making it infeasible. These factors make developing active target tracking problems a challenging prospect.

In this dissertation we study a number of coverage and tracking problems which address many of the challenges listed above. Our approach is to formulate specific sensing problems, often motivated by practical applications, and provide algorithms with provable performance guarantees for each problem. We list these problems and our contributions next.

1.2 Contributions

Towards building robotic sensing systems, this dissertation makes the following contributions: In the first part, we present placement algorithms for coverage using stationary sensors. In the second part, we present coverage and tracking algorithms for mobile sensors. We demonstrate the efficiency of our algorithms by giving theoretical performance guarantees. We prove that our algorithms are optimal or that their costs are guaranteed to be within some factor of the optimal values. A minimization algorithm whose solution has a value at most α times the optimal value in the worst-case is known as an α–approximation algorithm.

To ground our work in practical constraints, we study some of the problems in the context of two real-world applications: autonomously monitoring fish in lakes, and using robots in agriculture (Figure 1.3). We present the design and architecture of prototype systems developed along with results from large-scale field experiments. A major bottleneck for practical deployments is the limited on-board energy for the robots. In the last part, we will study the problem of optimizing paths and velocity profiles of the robots in order to minimize their energy consumption.

A brief overview of each problem and our contributions is given next.

1.2.1 Sensor Placement for Visibility-Based Coverage with Orientation

Cameras are one of the most commonly used sensors for coverage tasks. The capability of seeing every point in the environment is useful in a large number of applications. The problem of placing cameras for environment coverage is a classical one, broadly known as the *art gallery problem*. In the standard formulation, the environment is represented by an n sided 2D polygon and the sensors, also called guards, are modeled as points with omnidirectional vision. A guard is said to cover a point in the environment if

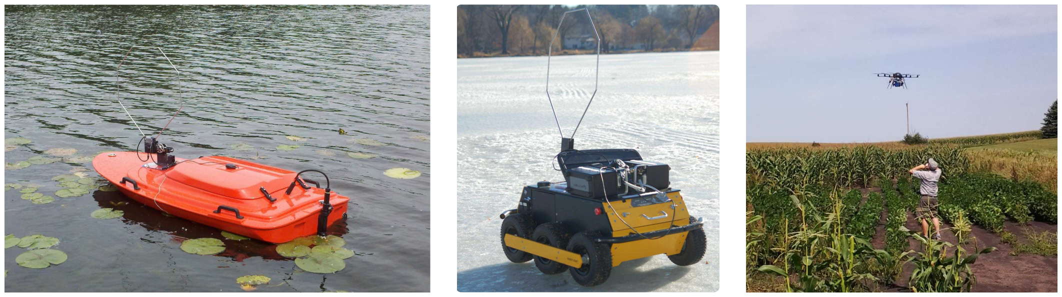


Figure 1.3: (Left) Autonomous boat developed for monitoring radio-tagged invasive fish during field experiments in lake Keller, Maplewood, MN. (Middle) During winters, we use a mobile robot on frozen lake. (Right) UAV obtaining multispectral images to form a nitrogen map of a corn plot in Janesville, MN. We study mobile sensing problems motivated by these applications.

the line segment joining them lies completely within the environment. The art gallery problem asks for the minimum number of guards sufficient to cover all points in the environment [1]. A summary of related research on art gallery problems is presented in the next chapter, in Section 2.2.

The standard formulation of the art gallery problem does not consider self-occlusions. Even when each point in the environment is covered, if some person starts moving in the environment, his/her back may occlude the front view. Obtaining a good view from all orientations is seen as an important requirement for many applications such as surveillance and video-conferencing.

Motivated by such applications, we study the coverage problem by imposing a new constraint termed $\triangle$-guarding. The $\triangle$-guarding constraint, introduced by Smith and Evans [13], states a point is covered if it is visible from two or more guards and it lies in the convex hull of the visible guards. If all points in the environment satisfy the $\triangle$-guarding constraint, then even if any convex object is introduced anywhere in the environment all points on its perimeter will always be visible, in spite of self-occlusion.

The $\triangle$-guarding problem is to place the fewest number of guards such all points in a given input polygon satisfy the $\triangle$-guarding constraint. Smith and Evans [13] proved the problem is NP-Hard. Efrat et al. [14] presented a randomized algorithm achieving

$\mathcal{O}(\log c_{\text{opt}})$–approximation for polygons without holes,[1] where c_{opt} is the optimal number of guards.

Our contributions are as follows: First, we prove a lower bound on the number of guards required for $\triangle$-guarding any input polygon. We show that any $\triangle$-guarding set uses at least $\Omega(\sqrt{n})$ guards for any n-sided simple polygon. Second, we use this lower bound to present an $\mathcal{O}(\log c_{opt})$ approximation algorithm for polygon with and without holes, when the guards are restricted to vertices of the polygon. Since c_{opt} itself can be very large in practice, we restrict the input to a set of chords in the polygon. These chords can represent, for example, paths a target is likely to take in the environment. Our goal is to $\triangle$-guard at least one point per chord. We present an approximation algorithm that is guaranteed to use at most 12 times the optimal number of guards. This result is one of the few constant-factor approximations for visibility-based coverage problems.

The results on this problem were first presented at ICRA 2014 [15] and a journal version is currently under review.

1.2.2 Bearing Sensor Placement for Target Localization

While a single camera can detect a target, information from multiple cameras might be necessary to precisely localize it. Cameras give bearing measurements towards targets in their field-of-view. Measurements from multiple sensors can be combined to estimate the target's position. The uncertainty in estimation decreases as more sensors are used. Furthermore, the uncertainty is a function of the relative position of the target and the sensors.

In such a scenario, it is no longer sufficient to say a point is covered if it is sensed by one or more sensors. Instead, a richer notion of coverage is required. We will require each point to be sensed with multiple sensors placed in such a way so as to guarantee some upper bound on the estimation uncertainty. Thus, the coverage requirement is to guarantee that if the target were at any point in the environment, measurements from all sensors can be combined to yield an estimate with sufficiently good quality.

We model the sensors as measuring a bearing towards the target corrupted by an

[1]A polygon with holes is a polygon which contains one or more non-overlapping polygons within it. See Section 2.2 for more related terminology.

unknown but bounded amount of noise. We seek worst-case guarantees for our placement: Given the true location of the target, imagine an adversary choosing the noise values for sensors. The true target location can be anywhere in the intersection and we would like this set to be "small" no matter where the target is. One way of solving this problem would be to place sensors everywhere in the environment. There is a trade-off between the number of sensors and the guarantee on resulting uncertainty. We study the bi-criteria optimization problem of minimizing the number of sensors used and the resulting uncertainty achieved. We consider a simple square environment. Even in this basic setting, devising a sensor placement scheme and analyzing its performance turns out to be challenging, as we will see in Chapter 4.

Our first contribution is to present a lower bound on the number of sensors required for any placement algorithm as a function of the desired uncertainty. Our main result shows that by placing sensors on a triangular grid-like placement, 9 times as many sensors as an optimal algorithm are sufficient to guarantee 6 times the desired uncertainty when the maximum sensing noise is less than $\frac{\pi}{4}$. We also show that in the triangular grid placement, only a constant number of sensors need to be activated to achieve the desired uncertainty, a property that can be used for designing energy/bandwidth efficient sensor selection schemes.

The results on this problem were first presented at ICRA 2013 [16] and a journal version is currently under review.

1.2.3 Multi-Target Visual Tracking with Teams of Aerial Robots

If we are given a large number of robots, sufficient to cover an environment at any time, we may not need to design active tracking algorithms. Instead we can treat the robots as stationary sensors. On the other hand if the number of robots is too small, it may make it infeasible to track all the targets. Thus, it is important to understand the effect of the relative number of robots and targets on the feasibility of tracking. In order to study this effect, we consider the following target tracking scenario. A collection of k targets are moving on the ground. A team of n aerial robots are tasked with tracking all the targets. Each robot carries a camera that can detect targets within its footprint.

A robot can potentially view more targets by flying to a higher altitude, thus increasing its camera footprint. However, this may reduce the quality of the view due to the increased distance between the cameras and the targets. There is a trade-off between the number of targets tracked and the corresponding quality of tracking. We investigate this trade-off and show that $k \geq 3$ robots may not be able to track $n > k$ targets while maintaining a constant factor approximation of the optimal quality of tracking at all times.

The infeasibility result requires constructing specific adversarial target trajectories. However, for other non-adversarial trajectories it may be possible to track more than n targets with some guaranteed quality of tracking. Alternatively, it may be possible to maximize the total quality of tracking, for example, by tracking fewer targets each with higher quality. We study the problem of how to choose robot trajectories to maximize either the number of targets tracked or the quality of tracking. We formulate this problem as the weighted version of a combinatorial optimization problem known as Maximum Group Coverage (MGC). We show that a greedy algorithm yields a 1/2 approximation for weighted MGC. Finally, we evaluate the algorithm and the sensing model through simulations and preliminary experiments.

The results on this problem were first presented at IROS 2014 [17] and a journal version is in preparation.

1.2.4 Sampling Algorithms with Aerial and Ground Robots (Precision Agriculture)

Coverage with mobile robots is suited for scenarios where instantaneous measurements are not necessary, and some delay in coverage is tolerated. Nevertheless, we would still like to minimize the time taken by the robots to cover the environment. Typically, the coverage time is defined as the traveling time and formulated as an instance of the Traveling Salesperson Problem (TSP) or its variant, TSP with Neighborhoods (TSPN).[2] However, for many sensors the time taken to obtain a measurement cannot be neglected, especially when the robot has to stop to obtain a measurement. In such cases, the robot can spend less total time by sampling in locations where a larger area is covered and thus requiring fewer samples, even if it comes at the expense of additional travel time.

Motivated by these practical constraints, we introduce a new coverage problem, termed SamplingTSPN. The input in SamplingTSPN consists of a set of possibly

[2]See Section 2.3 for a review of TSP and TSPN problems.

overlapping disks lying in the plane. The disks may have varying radii. When the robot is sensing a field with spatial correlation, we can get sufficient information without having to sample at a point. Instead we can sample anywhere within a neighborhood of the point. We use disks to model this property. SamplingTSPN asks for a sampling location in each disk, and a tour to visit the set of sampling locations. The objective is to minimize the sum of travel time and measurement time.

There is a trade-off between number of samples and travel time since it is possible to reduce the measurement time by combining sampling locations of overlapping disks, possibly at the expense of travel time (as opposed to standard TSPN, where the objective is only travel time). Our main contribution is a $\mathcal{O}\left(\frac{r_{\max}}{r_{\min}}\right)$ approximation algorithm for SamplingTSPN, where $r_{\min}$ and $r_{\max}$ are the minimum and maximum radii of input disks.

A practical application of SamplingTSPN is in the emerging application of precision agriculture. Precision agriculture is a data-driven technique to precisely estimate the status of crops and prescribe targeted fertilizers applications [18]. Soil measurements obtained by an Unmanned Ground Vehicle (UGV) can be combined with aerial images obtained with an Unmanned Aerial Vehicle (UAV) to estimate the status of crops in an agricultural plot. We show how to apply our SamplingTSPN algorithm to the problem of obtaining ground measurements with the UGV for estimating nitrogen levels in a plot.

We also study the aerial coverage problem for the UAV. Since small UAVs have limited battery lifetime, we focus on the problem of maximizing the number of aerial measurements subject to an energy budget. The problem of finding a tour to maximize the number of points visited (i.e., rewards) subject to a budget is known as the *orienteering problem.* The novelty of our formulation is the capability of the UGV to mule the UAV to deployment points. Instead flying between all points, the UAV may land on the UGV and piggy-back on the UGV before taking off at the next deployment location. Thus, the UAV can conserve energy, although it still spends some energy taking-off and landing. This leads to the question of when the UAV should use the UGV. We show how to formulate this capability as complete, metric graph which allows us to apply a 4-approximation [19] algorithm for orienteering.

Along with theoretical results, we present results from simulations conducted with real data collected from the field. Finally, we present preliminary field experiments conducted with the UAV in a corn plot.

The results on this problem were first presented at IROS 2013 [20] and a journal version is currently under review.

1.2.5 Coverage and Tracking with Autonomous Boats (Monitoring Invasive Fish)

The second application we study in this dissertation is that of autonomously monitoring radio-tagged Common Carp, an invasive species of fish in Minnesota lakes, in order to enable biologists to study their behavior. We present the design of a robotic system consisting of autonomous boats (summer) and wheeled robots (winter) that carry radio antennas in order to search for the radio tags. We also study coverage and tracking problems motivated by practical considerations arising in our system.

The problem of searching for radio-tagged carp can be formulated as a coverage task under the assumption that carp loiter in their home ranges for long periods of time [21]. However, instead of covering the entire lake, fish biologists can often provide a set of regions within the lake that are likely to contain the fish. We study the problem of designing a tour that covers a set of regions scattered in the lake in minimum time. While this problem can be solved by discretizing and formulating a TSP instance, we show approximation algorithms for coverage and TSPN can combined to obtain computationally efficient solutions. We prove that the resulting algorithm is a constant-factor approximation. In particular, we obtain a 3-approximation when the regions are rectangles touching the boundary of a simply-connected polygonal lake.

Once the radio-tagged fish are detected in the coverage phase, we switch to the active tracking to precisely localize them. For this, we use the directional properties of the radio antenna to obtain bearing measurement towards static radio tags. Such bearing measurements are noisy and take appreciable time to obtain (about 1 min). A good active target localization algorithm must be able to robustly estimate the target location using only a few measurements. We propose three active localization strategies and evaluate their performance through simulations and experiments. Finally, we report results from deployments where the robot executed both coverage and active localization algorithms, covering more than 5 kms in over an hour of autonomous operation.

The results on this problem were first presented at ICRA 2010 [22] and IROS 2011 [23]. The journal versions appeared in the Journal of Field Robotics [24] and IEEE Robotics & Automation Magazine [25].

1.2.6 Energy-Optimal Trajectory Planning

A major bottleneck for practical deployments of robotic sensing systems is the limited on-board battery capacity of the robots. Driving motors are a major source of power consumption for most mobile robots. The power consumption due to motion can be minimized by optimizing the velocity and the acceleration profiles of the robots. Motivated by this, we study the problem of finding minimum energy paths and velocity profiles for mobile robots with car-like steering.

Often, in many applications, the path to be followed by the robot is given by high-level planners such as the coverage and tracking algorithms described previously. However, the velocity and acceleration profiles are either set arbitrarily or left to some low-level controller. We study the problem of optimizing the velocity profiles to minimize energy consumption when the path to be followed by the robot is given. The path for a car-like robot may be composed of multiple straight line segments or curves of various radii. The maximum safe velocity of the robot along each path is a function of the turning radius. We present a closed-form solution for the minimum-energy velocity profile under these constraints.

Obtaining both minimum-energy paths and velocity profiles in closed form is difficult. However, instead of solving the problem with naive discretization, we show how to use minimum-energy velocity profiles as a subroutine to compute minimum energy paths. We contrast these paths with the minimum time Dubins' paths. Finally, we present a calibration procedure to obtain the energy model parameters and experimentally validate the velocity profiles.

The results on this problem were first presented at ICRA 2011 [26] and the journal version appeared in Autonomous Robots [27].

1.3 Organization of the Dissertation

This dissertation is organized in 8 chapters, following this chapter.

In Chapter 2, we present a review of the fundamental problems of set cover, art gallery problem, and the traveling salesperson problem. We refer to these problems in later chapters, however, each chapter is written to be self-contained.

In Chapters 3 and 4 we study sensor placement problems for the coverage task. Chapter 3 is on visibility-based coverage with the $\triangle$-guarding constraint. In Chapter 4, we study the problem of placing sensors in order to localize targets using bearing measurements with bounded uncertainty.

Chapters 5-8 present planning algorithms and system design for robotic sensing systems. We study the problem of active target tracking with a team of aerial robots in Chapter 5. In Chapters 6 and 7, we present coverage and tracking algorithms for problems motivated by two practical applications: precision agriculture and autonomous monitoring of radio-tagged fish. We also present the design of prototype systems for both applications which were developed as part of this dissertation, along with results from field experiments. In Chapter 8 we show how to compute energy optimal velocity profiles and paths in order to minimize the energy consumption and extend deployments for robotic sensing systems.

We conclude the dissertation with an overview of our contributions and highlight avenues of future research. Some additional proofs are presented in the appendix. Videos and software corresponding to the work in this dissertation are available online at `http://pratap.tokekar.com/thesis/`.

Chapter 2

Preliminaries

In this chapter we review the fundamental problems in combinatorial optimization and computational geometry that are related to the work in this dissertation. Many of the sensing tasks we study can be reduced to one of the following problems. However, we can often get better results by considering the additional structure present in the sensing tasks, as we demonstrate in subsequent chapters. Additional related work specific to each topic is presented in the corresponding chapter.

2.1 Set Cover and Maximum k–coverage Problems

The sensing coverage problem can be reduced to a *set cover* instance defined as follows: *Let $(X, \mathcal{R})$ be a set system, where X is a set of n elements and $\mathcal{R} = \bigcup R_j$ is a set of subsets of X ($R_j \subseteq X, \forall j$). The set cover problem is to find the smallest subset of $\mathcal{R}$ such that its union is X.*

Set cover is NP-hard [28] (the decision version is NP-complete). Johnson [29] showed that a simple greedy algorithm (iteratively, choose sets in $\mathcal{R}$ that contain most uncovered elements) yields an $H(n) \leq \ln n + 1$ approximation algorithm, where $H(n)$ is the n^{th} harmonic number. In *weighted set cover*, we are given a positive integer c_j for each R_j representing the cost of choosing R_j. The objective is to pick a subset of $\mathcal{R}$ with minimum total cost such that its union is X. Chvátal [30] showed that the greedy algorithm (iteratively, select a set R_j maximizing the number of uncovered elements contained in R_j divided by c_j) also yields an $H(n) \leq \ln n + 1$ approximation. Feige [28] showed that a $(1-\epsilon)\ln n$ approximation is not possible, unless NP is contained in the class that has deterministic algorithms with $n^{\mathcal{O}(\log\log n)}$ running time.

Maximum k–coverage is a closely related problem: *Let $(X, \mathcal{R})$ be a set system and k be some integer. The maximum coverage problem is to find at most k sets in $\mathcal{R}$ such that the size of their union is maximized.* In the weighted formulation each element in X has an associated weight. The objective is to pick at most k sets in $\mathcal{R}$ such that the sum of the weights of elements they cover is maximized.

Hochbaum and Pathria [31] showed the greedy algorithm that iteratively picks sets in $\mathcal{R}$ containing highest total weight of uncovered elements, yields a $1-1/e$ approximation algorithm. Feige [28] showed that a $(1-1/e+\epsilon)$ approximation is not possible in polynomial time, unless NP is contained in the class that has deterministic algorithms with $n^{\mathcal{O}(\log\log n)}$ running time.

Many sensing coverage problems can be formulated as set cover or maximum k–coverage instances as follows. The universal set X is the set of all points in the workspace which must be sensed. Every set in $\mathcal{R}$ corresponds to a candidate location, say x_j, where a sensor may be placed. $R_j \in \mathcal{R}$ is defined as the points in the workspace (subset of X) that can be sensed from x_j. If the objective is to sense every point in the workspace using fewest sensors, we have a set cover instance. If the objective is to maximize the number of points in the workspace that are sensed using a limited number of sensors, we have a maximum k–coverage instance. The cost for R_j can represent, for example, the time or energy required to obtain a sensor measurement from x_j. Alternatively, the weights for elements in X can represent, for example, their importance or profits.

Without any other information, we cannot do better than the inapproximability results given above. However, the geometric structure of sensors often allows us to devise algorithms with stronger performance guarantees. The art gallery problem, described next, is one such example with better approximation algorithms than general set cover.

2.2 The Art Gallery Problem

A *simple* polygon is one whose edges do not intersect each other. We only consider simple polygons in this dissertation. A *polygon with holes* is a polygon P along with a number of non-overlapping polygons all of which are contained within P. A polygon with holes is also called as a *multiply-connected polygon*, whereas a polygon without any holes is called *simply-connected*.

The *art gallery problem* asks the following question: *What is the minimum number of guards required to see all points in an n–sided simply-connected 2D polygon?* A guard is modeled as a point with omnidirectional vision. We say a guard at point p sees a point q if the line segment $\overline{pq}$ contains no point exterior to the polygon. A polygon is said to be guarded or covered by a set of guards if all points within the polygon are seen from at least one guard.

Over the years the art gallery problem has evolved into an important branch of research on visibility-based sensing. There are broadly two types of results in this area: (i) bounds on the minimum number of guards necessary and/or sufficient to guard a given class of polygons, and (ii) algorithms to place the minimum number of guards (or some bounded deviation from the minimum number) for a specific input polygon. Books by O'Rourke [1] and Urrutia [32] and a recent survey by Ghosh [33] contain some of the important results established over the years.

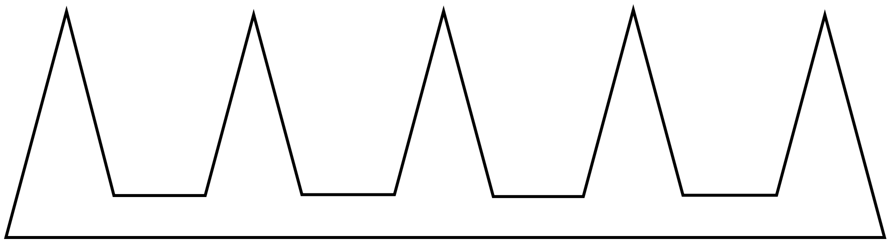


Figure 2.1: Polygon with $n = 15$ vertices for which $\lfloor n/3 \rfloor = 15$ guards are both necessary (one per prong) and sufficient (figure adapted from [1]).

Chvátal [34] was the first to prove that $\lfloor n/3 \rfloor$ guards are sometimes necessary and always sufficient to cover an n–sided polygon containing no holes. Figure 2.1 shows an example of a polygon for which $n/3$ guards are necessary (and sufficient). For polygons with holes, let n be the total number of vertices. That is, n is the sum of the number of vertices on the outer boundary and all hole boundaries. Bjorling-Sachs and Souvaine [35] and Hoffmann et al. [36] proved that $\lfloor (n+h)/3 \rfloor$ are sufficient, where h is the number of holes. Earlier, Shermer [1] had proved that there are polygons for which $\lfloor (n+h)/3 \rfloor$ guards are necessary (see Figure 2.2).

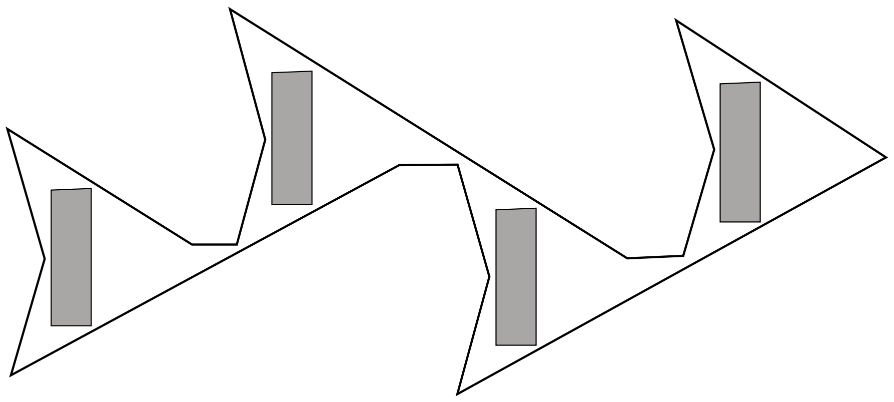


Figure 2.2: Polygon with $n = 32$ total vertices and $h = 4$ holes for which $\lfloor (n+h)/3 \rfloor = 12$ guards are both necessary and sufficient (figure adapted from [1]).

When guards can be placed anywhere within the polygon, they are termed as *point guards*. When they are restricted to the vertices of the polygon, they are termed as *vertex guards*. Chvátal's proof [34] (and a later proof by Fisk [37]) show that $\lfloor n/3 \rfloor$ vertex guards are always sufficient for polygons without holes. For polygons with holes, $\lfloor (n+2h)/3 \rfloor$ vertex guards are always sufficient. The necessity of $\lfloor n/3 \rfloor$ and $\lfloor (n+h)/3 \rfloor$ vertex guards comes from the same examples as point guards (Figures 2.1 and 2.2, respectively). The gap between the necessity of $\lfloor (n+h)/3 \rfloor$ vertex guards and sufficiency of $\lfloor (n + 2h)/3 \rfloor$ is still open for $h > 1$.

The optimization version of the problem is to guard a given input polygon by finding a placement using the fewest number of point guards. O'Rourke and Supowit [38] showed that this problem is NP-hard for polygons with holes. Lee and Lin [39] showed that the problem remains NP-hard even when the polygon does not contain any holes. Eidenbenz et al. [40] gave further inapproximability results for this problem. They proved there exists a $\delta > 0$ such that no polynomial time algorithm can achieve a $1+\delta$ approximation for polygons without holes. For polygons with holes, the inapproximability bound rises to $((1 - \epsilon)/12) \ln n$. These bounds also hold for vertex guards.

Ghosh [33] presented a deterministic $\mathcal{O}(\log n)$ approximation algorithms for vertex guards in polygons with or without holes. The algorithm shows how to formulate the vertex guarding problem as a set cover instance. The $\mathcal{O}(\log n)$ approximation comes from the greedy set cover algorithm. The approximation ratio can be improved to

$\mathcal{O}(\log c_{\text{opt}})$, where c_{opt} is the optimal number of guards, using techniques presented by Brönnimann and Goodrich [41] for solving set cover instances for set systems with finite VC-dimension. The VC-dimension for visibility is at most 14 [42]. Recently, King and Kirkpatrick [43] improved the best known approximation for vertex guards to $\mathcal{O}(\log \log c_{\text{opt}})$.

For point guards, Efrat and Har-Peled [44] presented a $\mathcal{O}(\log c_{\text{opt}})$ approximation when given an arbitrarily dense grid where guards may be placed. If the polygon contains h holes, the approximation ratio for the same algorithm becomes $\mathcal{O}(\log h \log(c_{\text{opt}} \log h))$. Here, c_{opt} is the size of the optimal guarding set restricted to the grid. Deshpande et al. [45] presented an algorithm which constructs a grid such that the optimal guarding set on the grid has size at most three times that of an optimal guarding set not restricted to any grid. Their algorithm constructs such a grid in time polynomial in n and the ratio of the largest and smallest pairwise distances between the vertices of the polygon. In the worst-case, the running time is exponential in the input size.

Stronger approximation guarantees are known for special classes of polygons. For example, a constant factor approximation algorithm was presented by Krohn and Nilsson [46] for guarding the interior of monotone polygons with point guards. However, no constant factor approximation algorithm for either vertex or point guards is known for the general case, which remains the main open problem in this area.

2.3 The Traveling Salesperson Problem

In the *Traveling Salesperson Problem* (TSP) we are given a graph with weights on all edges. The objective is to find a tour that visits each vertex exactly once while minimizing the sum of weights of edges along the tour. *Metric TSP* is an instance where the vertices lie in a metric space (edges satisfy triangle inequality). *Euclidean TSP* is the special case when the vertices are points on a plane and the weights are Euclidean distances between the two points.

TSP, even when restricted to the Euclidean version, is NP-hard [47]. Christofides [48] presented a $3/2$ approximation algorithm for metric TSP which is currently the best known performance guarantee for metric TSP [49]. Euclidean TSP, on the other hand, can be approximated arbitrarily close to the optimal. Arora [50] and Mitchell [51] presented Polynomial Time Approximation Schemes (PTAS) for Euclidean TSP which produce a $1 + \epsilon$ approximation for any $\epsilon > 0$ in $\mathcal{O}(n^{\frac{1}{\mathcal{O}(\epsilon)}})$ time.

Of the many variations of TSP in existence, most relevant to our work is *TSP with Neighborhoods* (TSPN). In TSPN, instead of points in the plane, we are given a collection of neighborhoods (e.g., disks). The objective is to find a tour of minimum length that visits at least one point in each neighborhood.

TSPN was introduced by Arkin and Hassin [52] where they presented constant-factor approximations for some classes of neighborhoods. Dumitrescu and Mitchell [2] presented constant factor approximations when the neighborhoods are unit radius disks. In particular, they presented a PTAS when all the disks are disjoint and an 11.15 approximation when some disks overlap. Recently, Dumitrescu and Tóth [53] improved the approximation ratio for the overlapping disks case to 6.75. They also presented constant factor approximation algorithms for neighborhoods that are lines, balls or planes in $\mathbb{R}^3$.

For the case of general neighborhoods in a plane, Mitchell [54] presented a constant factor approximation if the neighborhoods are disjoint. If the neighborhoods are both disjoint and fat, there is a PTAS given by Mitchell [55]. A region is called *fat* if it contains a disk whose radius is within a constant factor of the diameter of the region containing all the neighborhoods. The existence of a constant factor approximation for the case of overlapping and arbitrarily sized neighborhoods is still an open problem.

Chapter 3

Sensor Placement for Visibility-Based Coverage with Orientation

In this chapter, we study the problem of placing a minimum number of cameras to cover a polygonal environment. The novelty of our formulation is in the notion of covering a point where simply seeing an object is not sufficient but getting a good view is also important. Obtaining a good view is an important requirement for many applications such as surveillance, visual inspection and video-conferencing. We formulate a new coverage problem which formalizes this requirement and builds on the classical art gallery problem.

Art gallery problems are a class of visibility problems that deal with placing omni-directional cameras, also called as guards, in polygonal environments. The original art gallery problem asked for the fewest number of guards sufficient to see every point in an n-sided 2D polygon with no holes. Chvátal [34] answered this question in 1975 by showing that $\lfloor n/3 \rfloor$ guards are always sufficient and sometimes necessary. Since then, a number of bounds have been established for various classes of polygons. We review some of the important results in Section 2.2.

Art gallery problems are important for robotics since cameras (or more generally visibility-based sensors) are commonly used in many tasks such as coverage [56], surveillance and inspection [57], and simultaneous localization and mapping [58]. Solutions to art gallery problems can be leveraged as subroutines for robotic planning problems. For example, Danner and Kavraki [57] used locations given by solutions to art gallery problems to construct 2D and 3D inspection tours for mobile robots. González-Baños and Latombe [59] presented an algorithm to acquire 3D views of an environment using a robot equipped with a 3D laser scanner by solving a variant of the art gallery problem which models some practical sensing constraints. Blaer and Allen [60] extended this work to include more practical constraints and presented experimental results where their robot reconstructed 3D models of two sites in New York. Ganguli et al. [56] studied a distributed art gallery deployment problem and presented control strategies for robots with line-of-sight communication to achieve visual coverage in unknown environments. Thus, efficient algorithms for art gallery problems can be useful for a number of robotic tasks.

The classical art gallery problem only requires each point in the environment to be visible from at least one camera. However, for many applications a binary notion of visibility is not sufficient. Obtaining a good view is equally important. For example, consider a video conferencing system where a person can move within a room. If the room is convex, then a single camera is sufficient to guarantee visibility (Figure 3.1). However, if the person stands with his or her back to the only camera, no good view of the person will be available. Our goal will be to place cameras such that any person or object will be seen from all orientations, in spite of self-occlusion.

We use this as motivation to study the problem of placing the minimum number of cameras in order to see all faces of any convex object moving in the environment. Smith and Evans [13] introduced this problem, and formalized it with the following $\triangle$-guarding condition:

Definition 1. *A point p is said to be* $\triangle$-guarded *by a set of guards G, if p is visible from a non-empty set of guards $G' \subseteq G$ and p lies in the convex hull of G'. A simple polygon P is said to be* $\triangle$-guarded *by G, if every point $p \in P$ is $\triangle$-guarded by G.*

Based on this definition, if a polygon is $\triangle$-guarded then the perimeter of any convex object located anywhere in the polygon will always be visible from the set of guards.

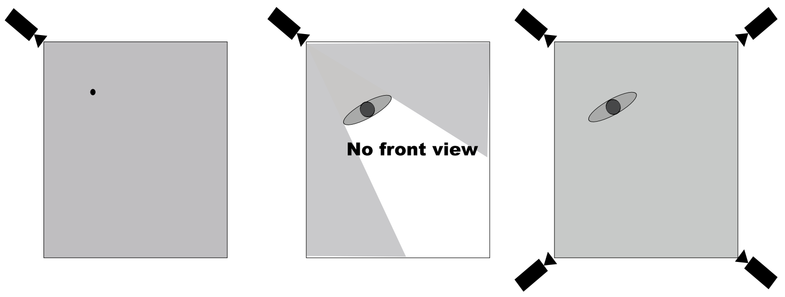


Figure 3.1: The standard art gallery problem ensures that every point in the environment is seen from at least one guard (left). However, due to self-occlusions, some part of a person may not be visible (middle). We study the art gallery problem in the presence of self-occlusions (right).

Thus, the $\triangle$-guarding constraint models our requirement of getting a good view of an object despite possible self-occlusion. Note that the guards themselves need not be visible from each other.

Smith and Evans [13] proved that deciding if k vertex guards can $\triangle$-guard a simple polygon is NP-hard. Efrat et al. [14] presented a randomized algorithm based on [41] that when applied to the $\triangle$-guarding problem yields a $\mathcal{O}(\log c_{\text{opt}})$–approximation for polygons without holes (c_{opt} is the optimal number of guards). Since the $\triangle$-guarding constraint generalizes the simple visibility requirement for the art gallery problem, we expect to place more guards. The first problem aims to find how large c_{opt} can be.

Problem 1. *How many guards are necessary to $\triangle$-guard every point in any n-sided 2D simple polygon?*

We show that $\Omega(\sqrt{n})$ guards are always necessary to $\triangle$-guard any simple polygon. Contrast this with the standard formulation without $\triangle$-guarding, where there are polygons, namely, star-shaped polygons, where a single guard is necessary and sufficient.

The $\Omega(\sqrt{n})$ lower bound applies to any n–sided polygon. The optimal number of guards for a specific input polygon may be higher. Next, we study the algorithmic problem of placing guards in order to $\triangle$-guard a given input polygon. We consider the case when guards can only be placed on the vertices of the polygon, termed vertex guards.

Problem 2. *Given a simple polygon P, find the minimum number of vertex guards, and their placement, sufficient to $\triangle$-guard every point in the interior of P.*

We present a $\mathcal{O}(\log c_{\text{opt}})$ approximation algorithm for this problem. Our main insight is to show how to convert the problem of $\triangle$-guarding every point in the interior of P to $\triangle$-guarding only a finite number of points which can be solved using a greedy set cover algorithm.

In many applications such as surveillance or mobile video conferencing, we may not need to $\triangle$-guard the entire polygon. Instead, $\triangle$-guarding may be required only for a set of paths a person or object of interest is likely to take within the environment. With this as motivation, we study the problem of placing the fewest number of guards to $\triangle$-guard a set of line segments between visible points on the boundary of a polygon. Such line segments are termed as *chords*. For example, the points can correspond to entry and exit points in the environment, the line segments being paths likely to be taken by a person. Our goal is to $\triangle$-guard at least one point on each line segment, thus guaranteeing that independent of the orientation, all sides of the person will be seen at some point along the path.

Problem 3. *Let C be a set of chords in a simply-connected polygon P. Find the minimum number of guards, and their placement, in order to $\triangle$-guard at least one point on each chord in C.*

In this problem, the guards may be placed anywhere within P and not necessarily on the vertices of P. We present a constant factor approximation for this problem.

The rest of the chapter is organized as follows: We prove the lower bound on the number of guards for $\triangle$-guarding in Section 3.1. The log approximation for Problem 2 is given in Section 3.2. The constant factor approximation for Problem 3 is presented in Section 3.3. We conclude in Section 3.4.

3.1 Lower Bound on the Number of Guards

In this section, we prove a lower bound on the number of guards necessary to $\triangle$-guard any simple polygon P. For establishing the lower bound, we will prove necessary conditions on where the guards must be placed.

We first define an *edge extension* as follows. Extend an edge of P from either endpoint until it reaches the boundary of the polygon. Each of the (closed) line segments lying on either side of the edge is termed as an *edge extension*. An edge introduces as many edge extensions as the number of its reflex endpoints. As a matter of convention, we will refer to a vertex on a hole as a convex vertex if the angle formed by the two adjacent sides containing the interior of the polygon is smaller than $\frac{\pi}{2}$. Else, we refer to the vertex as a reflex vertex.

Lemma 1. *Let G be a set of guards that $\triangle$-guards a simple polygon P. If v is a convex vertex in P (lying on the exterior or hole boundary), then $v \in G$. If e is any edge extension in P, then there exists a guard in G that lies on e.*

The proof is presented in Appendix A.1. Using Lemma 1, we can prove the lower bound on the number of guards of any $\triangle$-guarding set of P.

Theorem 1 (Lower Bound)**.** *Let G be a set of guards placed in an n-sided simple polygon P. If G $\triangle$-guards P, then $|G| = \Omega(\sqrt{n})$.*

Proof. Let the total number of convex and reflex vertices in P be n_c and n_r, respectively. We have two cases, $n_c \geq n/4$ or $n_c < n/4$. First consider, $n_c \geq n/4$. From Lemma 1 we know $|G| \geq n_c$. Hence, $|G| \geq n/4$ and consequently $|G| = \Omega(\sqrt{n})$.

Now consider, $n_c < n/4$. That is, $n_r \geq 3n/4$. Each edge in P may introduce up to two unique edge extensions. Consider the set of edge extensions due to edges whose endpoints are both reflex vertices. Let m be the total number of such edge extensions. We know, $m \geq 2(n_r - n_c) \geq n$.

From Lemma 1, we know each of these m extensions must have a guard placed on them. The optimal algorithm may be able to use the same guard if two or more extensions intersect at a point. Let k be the maximum number of extensions that intersect in one point. To cover m extensions, any algorithm will require at least m/k guards. Hence, $|G| \geq m/k$.

Now consider the polygon edges that contributed to the k extensions which intersect at a point. Since we are focusing only on edges with reflex vertices on both ends, each such edge must have introduced another extension, contributing another k extensions. Since the two extensions resulting from a polygon edge are colinear, any guarding set will be forced to use a separate guard for covering each of the other k extensions. Hence, $|G| \geq k$.

Multiplying the two lower bounds, we get $|G|^2 \geq m$ or $|G| \geq \sqrt{m}$. Since $m \geq n$, the theorem statement follows. $\square$

The bound is tight for polygon with holes. Figure 3.2 shows an instance where the $\triangle$-guarding has size $\mathcal{O}(\sqrt{n})$. The bound may not be tight for polygons without holes.

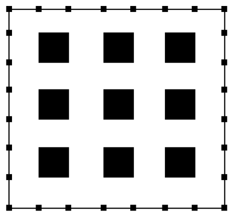


Figure 3.2: Polygon P consists of $k \times k$ holes aligned along a grid. The outer boundary of the polygon forms a square. The number of vertices of P are $n = 4k^2+4$. $\mathcal{O}(k) = \mathcal{O}(\sqrt{n})$ guards (marked by small squares) are sufficient for $\triangle$-guarding P.

3.2 $\mathcal{O}(\log c_{\mathbf{opt}})$–approximation with Vertex Guards

In this section, we present a deterministic algorithm that yields a $\mathcal{O}(\log c_{\text{opt}})$–approximation for $\triangle$-guarding polygons with and without holes when the guards are restricted to be placed only on the vertices of P (Problem 2). This improves upon the randomized algorithm presented by Efrat et al. [14] which would yield a $\mathcal{O}(\log c_{\text{opt}} \log(c_{\text{opt}} \log c_{\text{opt}}))$–approximation for polygons with holes. Our main result in this section is as follows.

Theorem 2 (Vertex Guards)**.** *There exists a deterministic algorithm which finds a set of vertex guards G that $\triangle$-guards any simple polygon P such that $|G| = \mathcal{O}(c_{opt} \log c_{opt})$, where c_{opt} is the minimum number of vertex guards required to $\triangle$-guard P.*

Before we describe our algorithm, we will present a more convenient definition (equivalent to Definition 1) for $\triangle$-guarding a point.

Proposition 1. *Let p be any point in a polygon, l be any line passing through p, and H be any of the two closed half-planes defined by l. p is $\triangle$-guarded if and only if H contains a guard visible from p.*

We represent a half-plane by drawing a vector which starts at p and is perpendicular to the line l (Figure 3.3). Let θ be the orientation of this vector with respect to some globally defined axis. By Proposition 1, in order to $\triangle$-guard p, we must ensure half-planes corresponding to every orientation $\theta \in [0, 2\pi)$ must contain a guard.

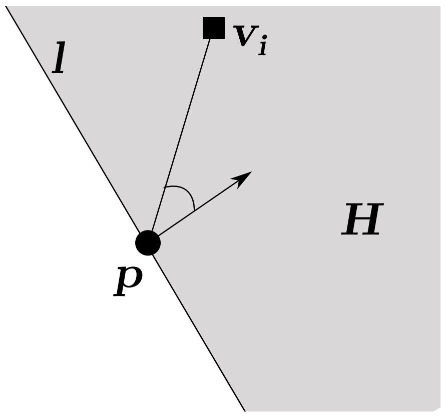


Figure 3.3: H is a closed half-plane defined by some line l passing through p. According to Proposition 1, p is $\triangle$-guarded only if half-planes of all possible orientations through p contain a guard. A guard v_i that sees p is contained in only those half-planes whose normal vectors are between $-\pi/2$ and $\pi/2$ of the segment $\overline{pv_i}$.

If a guard v_i sees p, then v_i will be contained in all half-planes whose vectors are between $-\pi/2$ and $\pi/2$ of the segment $\overline{pv_i}$. Hence, the point p is $\triangle$-guarded by a set of guards if and only if for any θ, the pair (p, θ) is covered by the set of guards. $\triangle$-guarding the interior of P thus is equivalent to covering (p, θ) for all points $p \in P$ and all orientations θ at p. Unfortunately, there are infinitely many such (p, θ) pairs in P. Nevertheless, we will show that there exists only finitely many points and finitely many orientations at each point that need to be considered in order to $\triangle$-guard a polygon. Using this, we construct a set system (X, R) with $|X| = \mathcal{O}(n^6)$. We can then apply a simple greedy set cover algorithm which gives a $O(\log |X|)$ approximation. Together with our lower-bound given in Theorem 1, Theorem 2 follows. We start by describing what these finitely many points are.

Create a visibility arrangement of the set of vertices in P as follows: If two vertices are visible from each other, draw a line segment joining them, extending out on both sides till you reach the boundary of P. The set of all such line segments yields the visibility arrangement A. The arrangement A partitions the interior of P into a set of cells, each of which is convex [33]. The vertices of each cell are the points of intersection of two or more segments. There are $\mathcal{O}(n^2)$ line segments and $\mathcal{O}(n^4)$ cells.

All points in the same cell are visible from the same set of vertices (see e.g., Lemma 2.1 in [33]). The following lemma shows that we can convert the problem of $\triangle$-guarding the entire interior of P into the problem of $\triangle$-guarding only the set of vertices in the visibility arrangement.

Lemma 2. *Let A_i be any cell in the visibility arrangement of all vertices of a simple polygon. Let p_i be any point inside A_i and $V(i)$ be the vertices of the polygon visible from p. If all vertices of A_j are $\triangle$-guarded by $V(i)$, then p_i is $\triangle$-guarded by V.*

Proof. Suppose not. Then, along with Proposition 1 this implies there exists a line passing through p_i, say l and a corresponding half-plane, say H, which does not contain any guard visible from p_i. Let a_i be a vertex of cell A_i that lies in H (a_i exists since the cell A_i is convex). We draw a line parallel to l passing through a_i which forms a half-plane, say H'. We know a_i is $\triangle$-guarded by vertices $V(i)$. Hence, by Proposition 1 H' contains a vertex, say $v_i \in V(i)$ of P visible from a_i. v_i is also visible from p_i. Hence, v_i lies in H and visible from p_i which is a contradiction. □

We can thus restrict the problem of $\triangle$-guarding the interior to the problem of $\triangle$-guarding only the finite set of vertices in the visibility arrangement. We will now show that there are only finitely many orientations that we need to consider at each such vertex.

Consider a vertex a_i of some cell A_i. Let $V(i)$ be the set of polygon vertices visible from any point in A_i. For every $v_i \in V(i)$ draw a line perpendicular to the segment $\overline{v_i a_i}$ and passing through a_i (Figure 3.4). These set of lines create $\mathcal{O}(|V(i)|)$ angular sectors about a_i. If θ_1 and θ_2 are any two orientations lying within the same sector, then any polygon vertex that covers (a_i, θ_1) also covers (a_i, θ_2) and vice versa. Thus, we need to consider only $\mathcal{O}(|V(i)|)$ orientations per vertex a_i.

We now create a finite set system (X, R) as follows: For every cell vertex a_i create $\mathcal{O}(|V(i)|)$ elements in X, one corresponding to each angular sector θ_i. R is a collection of n subsets of X, each corresponding to a polygon vertex v_i. The subset corresponding

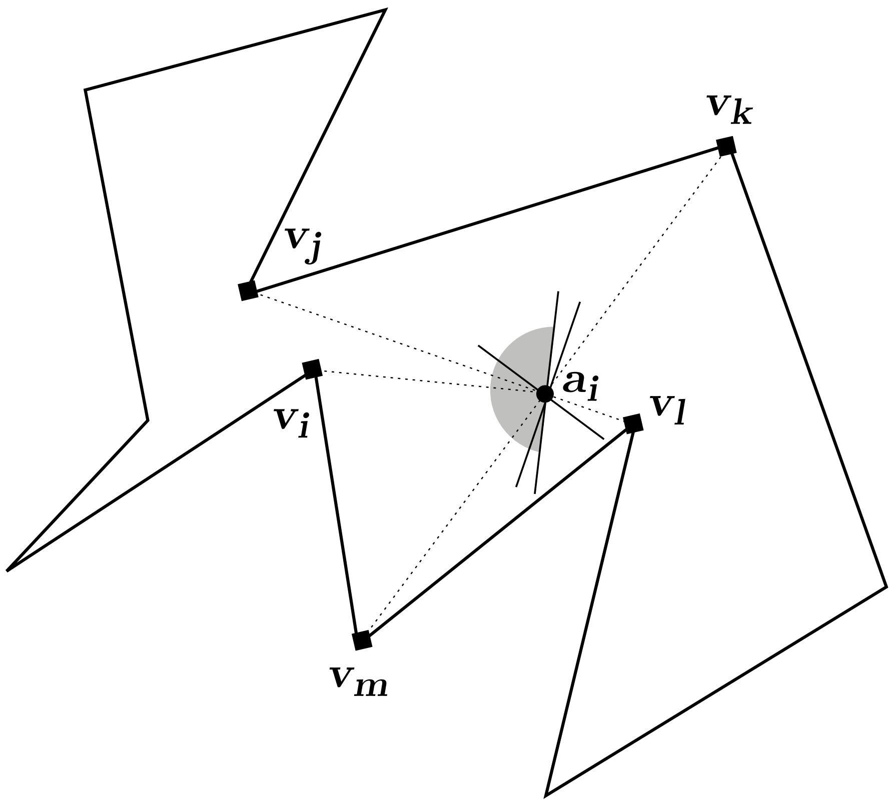


Figure 3.4: A vertex v_i is said to cover any orientation at point a_i if it is at most $\pi/2$ away from the line $\overline{v_i a_i}$. All such orientations covered by v_i are marked shaded.

to v_i contains all pairs (a_i, θ_i) that are covered by v_i. There are $\mathcal{O}(n^4)$ cells with $\mathcal{O}(n)$ vertices per cell and $\mathcal{O}(|V(i)|) = \mathcal{O}(n)$ sectors per vertex. Thus $|X|$ is at most $\mathcal{O}(n^6)$. A greedy set cover algorithm yields a $\log |X| = \mathcal{O}(\log n) = \mathcal{O}(\log c_{\text{opt}})$ approximation. This proves Theorem 2.

Nevertheless, c_{opt} itself is subject to the $\Omega(\sqrt{n})$ lower bound. The large lower bound results from having to guard each convex vertex and edge extension, which may not be important for many applications. Instead, we will restrict our attention to $\triangle$-guarding only regions of interest within the polygon, specifically, line segments joining points on the boundary of a simply-connected polygon.

3.3 $\triangle$-guarding Chords

In this section, we present a constant factor approximation for $\triangle$-guarding a set of chords in a polygon (Problem 3). A *chord* in a simple polygon P is any line segment which joins two mutually visible points that lie on the boundary of P. A diagonal is special type of chord where both points are vertices of P.

Problem 3 asks for $\triangle$-guarding at least one point per chord. For the problem of $\triangle$-guarding *every* point on the chord, one can construct an instance where the set of input chords fill the entire polygon. Thus, the problem becomes at least as hard as $\triangle$-guarding the entire polygon. Hence, we need $\Omega(\sqrt{n})$ guards in the worst-case. The algorithm from the previous section can be applied to obtain a log factor approximation for $\triangle$-guarding every point on a set of chords with vertex guards. We focus on $\triangle$-guarding at least one point per chord, and present a constant factor approximation algorithm.

Our main result for this problem is as follows.

Theorem 3 (Chord Guarding)**.** *Given a set of chords C in a simply-connected polygon P, there exists an algorithm which finds a set of guards G $\triangle$-guarding C, such that $|G| \leq 12c_{opt}$ where c_{opt} is the minimum number of guards required to $\triangle$-guard C.*

3.3.1 Notation

We label the points on the boundary of P in the clockwise order, starting from an arbitrarily chosen vertex. If a point p on the boundary appears before point q in the clockwise ordering, then we denote this by $p \prec q$. For each chord C_i, we term the endpoint that appears first in the clockwise ordering along the boundary as its *start point* (s_i) and the other endpoint as the *terminal point* (t_i). Thus, $s_i \prec t_i$.

We map all s_i and t_i to a circle maintaining their clock-wise ordering (Figure 3.5). The part of the boundary of P from s_i to t_i along the clockwise order maps to an arc on the circle; we term this as the *induced arc* (A_i). The chord also divides the polygon into two subpolygons. We term the subpolygon corresponding to the induced arc as the *induced subpolygon*, denoted by P_i. P_i is made up of the boundary of P between s_i and t_i and the edge t_is_i.

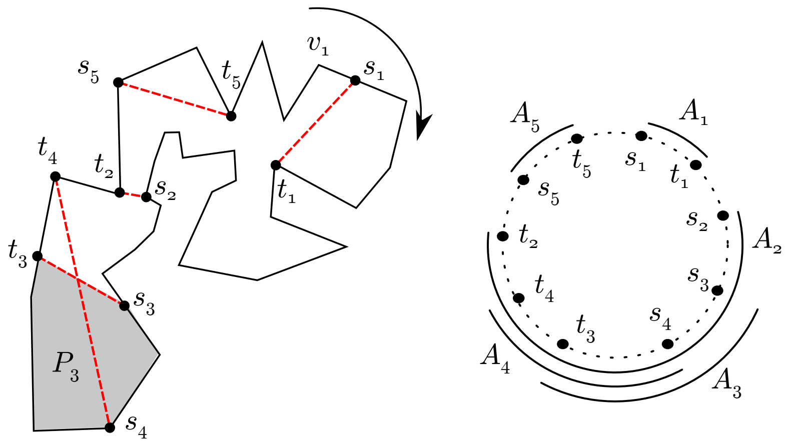


Figure 3.5: The endpoints of all chords map to a circle in clockwise order. The corresponding arc is termed as the induced arc A_i. P_i is the subpolygon induced by C_i.

The set of all arcs induced by C creates a circular-arc graph [61], with arcs as vertices, and an edge between two vertices if the corresponding arcs overlap. The maximum independent set (MIS) of this graph is the largest set of disjoint arcs. Masuda and Nakajima [61] presented an optimal algorithm for finding the MIS of circular-arc graphs.

We use the following distinction for non-disjoint arcs: A_i and A_j with $A_i \cap A_j \neq \emptyset$ are termed *cutting arcs*, if $A_i \not\subseteq A_j$ and $A_j \not\subseteq A_i$. A_i and A_j are said to cut each other.

We will refer to a chord, its induced arc, and the corresponding vertex in the circular-arc graph, interchangeably. Next, we present a high level discussion of our strategy for placing guards.

3.3.2 Overview

Given the MIS of the circular-arc graph, we classify each chord in C into four types. A chord C_i is of

- Type I if A_i is in the MIS,
- Type II if A_i cuts some arc in the MIS,
- Type III if A_i contains some arc in the MIS,

- Type IV if A_i is contained in some arc in the MIS.

First in Section 3.3.3, we describe the placement of a guard set $\triangle$-guarding chords of Types I & II. In Section 3.3.4, we will $\triangle$-guard a subset of Type III guards. Finally, in Section 3.3.5 we describe an algorithm for $\triangle$-guarding the remaining set of guards of Type III and Type IV chords.

We will show that the total number of guards placed by our algorithm is at most a constant times that of an optimal algorithm. We will use the following two useful properties that will allow us to obtain a constant factor approximation.

Lemma 3. *Two chords C_i and C_j intersect if and only if their corresponding arcs A_i and A_j cut each other.*

The proof, which verifies the ordering of s_i, s_j, t_i, t_j for both directions, is presented in Appendix A.2.

Lemma 4. *If chord C_i is $\triangle$-guarded by a set of guards G, then at least one guard in G must lie in its induced subpolygon P_i.*

Proof. Let p be a point on C_i that is $\triangle$-guarded by G. Consider the line containing chord C_i which passes through p. This line creates two closed half-planes one of which contains all points from P_i visible from p. From Proposition 1, we know this closed half-plane must contain a guard visible from p. Since no point in this half-plane outside of P_i lies within the polygon, this guard must be contained in P_i. $\square$

We term such a guard as the *cardinal guard* of C_i. We will charge a constant number of guards in our placement to a cardinal guard in the optimal placement. We first establish a lower bound on the minimum number of guards necessary to $\triangle$-guard C using the MIS of the circular arc graph.

3.3.3 Guarding Type I and II chords

Lemma 5. *If M is the MIS of disjoint arcs in the circular-arc graph, then $|M| \leq c_{opt}$, where c_{opt} is minimum number of guards for $\triangle$-guarding C.*

Proof. Since all arcs in the MIS are disjoint, their induced subpolygons are disjoint. That is, for any two arcs $A_i, A_j \in M$ we have $P_i \cap P_j = \emptyset$. From Lemma 4, we know each chord must have at least one guard in its induced subpolygons. Since the subpolygons for all chords in the MIS are disjoint, no two chords may share a cardinal guard. Hence, there are at least as many cardinal guards as the number of disjoint subpolygons. Therefore, $|M| \geq c_{\text{opt}}$. □

We now describe set S_1 for guarding chords of Types I & II.

Lemma 6. *If S_1 is the set of endpoints of chords in M, then S_1 $\triangle$-guards all chords of Types I & II, and $|S_1| \leq 2c_{opt}$.*

Proof. First consider Type I chords. Since we place a guard at both endpoints of each such chord, *all points* lying on a Type I chord are $\triangle$-guarded. Let C_i by a Type II chord whose arc cuts an arc of C_j, a Type I chord. According to Lemma 3, C_i and C_j must intersect in a point. Since all points on C_j are $\triangle$-guarded, C_i is $\triangle$-guarded. Hence, all Type II chords are $\triangle$-guarded. □

3.3.4 Guarding a subset of Type III chords

Consider chords of Type III. We call the portion of the circle between two consecutive arcs in the MIS *gaps*. Type III chords have both endpoints in a gap, and the start and terminal endpoints must lie in different gaps. Each gap may contain multiple start and terminal points. Since there are as many gaps as arcs in the MIS, from Lemma 5, we may place a constant number of guards per gap and perform comparable to an optimal algorithm.

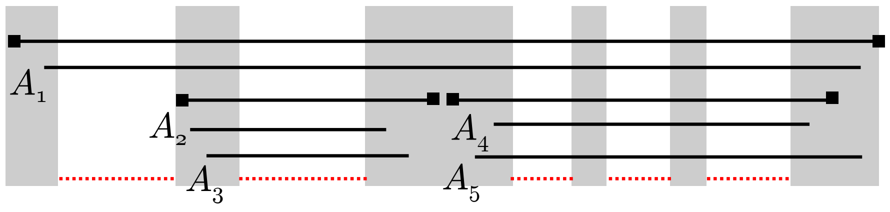


Figure 3.6: Type III chords. The arcs in MIS are shown dotted, gaps are marked shaded. In each gap, we place guards (marked square) on the endpoints of chords with earliest start point or latest terminal point. Chords with arcs $A_1, \ldots, A_4$ may not be $\triangle$-guarded by this set of guards, where as A_5 is.

We will place at most four guards per gap in a guard set S_2 as follows (Figure 3.6):

- on the two endpoints of the Type III chord with the first start point within each gap (if any), and
- on the two endpoints of the Type III chord with the last terminal point within each gap (if any).

Lemma 7. *If C_i and C_j are any two Type III chords not $\triangle$-guarded by S_2, then either A_i and A_j are non-cutting arcs or both chords start from the same gap and end in the same gap. $|S_2| \leq 4c_{opt}$, where c_{opt} is the optimal number of guards for $\triangle$-guarding C.*

Proof. There are as many gaps as the number of arcs in the MIS. We place at most four guards per gap. Using Lemma 5, $|S_2| \leq 4c_{\text{opt}}$.

We will prove the contrapositive of the statement of the lemma. If A_i and A_j are cutting arcs with either their start or terminal points in different gaps, then C_i and C_j are $\triangle$-guarded by S_2. We will prove the case when their start points lie in different gaps. The case for the terminal points of C_i and C_j lying in different gaps is symmetric.

Without loss of generality, let $s_i \prec s_j$. For contradiction, assume that C_i and C_j are not $\triangle$-guarded by S_2. Consider the gap containing s_j. We know this gap contains at least one start point of a Type III chord, i.e., s_j. If s_j is the earliest start point in this gap, then S_2 contains two guards placed on either endpoints of C_j and hence, C_j must be $\triangle$-guarded, which is a contradiction. Thus, there exists some other start point in the same gap before s_j, say s_k corresponding to a Type III chord C_k.

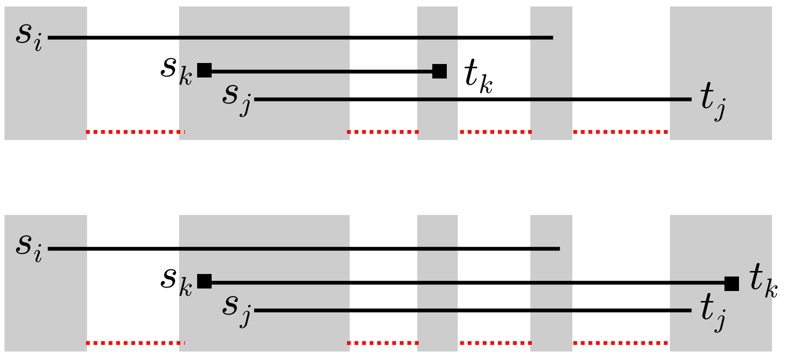


Figure 3.7: Illustration of the proof for Lemma 7. C_i and C_j start in different gaps. At least one of C_i or C_j cuts a chord with guards placed on two endpoints, C_k.

For the terminal point of C_k, we have two possibilities (See Figure 3.7)

1. $t_k \prec t_j$. We know $s_k \prec s_j$. t_k and t_j do not lie in the same gap as s_k and s_j respectively. Thus we get, $s_k \prec s_j \prec t_k \prec t_j$. Therefore, A_k cuts A_j. From Lemma 3, C_k must intersect with C_j. Since we have guards placed on both endpoints of C_k, all points on C_k are $\triangle$-guarded including C_j's point of intersection with C_k. Hence, C_j is $\triangle$-guarded, which is a contradiction.

2. $t_j \prec t_k$. Since C_i and C_j are cutting arcs and $s_i \prec s_j$, we get $t_i \prec t_j$. Therefore $t_i \prec t_k$. Since s_i lies in a gap before the one that contains s_j and s_k, we get $s_i \prec s_k \prec t_i \prec t_k$. Hence, the arcs of C_i and C_k cut each other. Following a similar argument as above, C_i must be $\triangle$-guarded, which is a contradiction.

This completes the proof. □

Lemmas 6 and 7 present a guard set of size at most $6c_{\text{opt}}$ covering all Type I, II and a subset of III chords in C. We describe the placement of another guard set to $\triangle$-guard all remaining chords in C.

3.3.5 Guarding remaining Type III and IV chords

Let $C' \subset C$ be the set of chords not $\triangle$-guarded by guard sets S_1 and S_2 described in Section 3.3.3. C' consists of a subset of Type III chords given by Lemma 7 and all Type IV guards. Lemma 7 states that if $C_i, C_j \in C'$ cut each other, then they must start and terminate in the same gap. We will define an equivalence class of all Type III chords that start and terminate in the same gap. Similarly, we will define another equivalence class of Type IV chords that are contained in the same arc in the MIS. We term each such class as a *group*. Thus two chords in C' lie in the same group if they start and terminate in the same gap, or if they are contained within the same arc in the MIS.

While the chords within each group may cut each other, we show that chords in distinct groups do not.

Lemma 8. *If $C_m \in G^i$ and $C_n \in G^j$ are two chords in distinct groups, then A_m and A_n do not cut each other.*

The full proof, presented in Appendix A.3, verifies all the cases and shows that the arcs cannot cut each other. Hence, two groups are either disjoint or one completely contains the other. This gives a partial ordering on all groups based on inclusion. We use this to create a tree of chords $\mathcal{T}$:

1. Re-index all chords in $\mathcal{T}$, such that for any C_i and C_j if $s_i \prec s_j$ then $i < j$. That is, if a chord starts before another, then it has a lower index than the other.
2. The circumference of the circle forms the root.
3. Create a tree of groups. Iteratively add all groups as nodes in the tree using the rule: group G^j is an ancestor of G^i if and only if the induced arc of G^i is completely contained in G^j.
4. Replace each group node G^i with a chain of chord nodes, one node per chord in the group. The chord with a lower index is at a lower depth in this chain. The subtree rooted at G^i is attached to the chord node with the highest index, and the parent of G^i is attached to the chord node with the lowest index.

In the following lemmas, we will prove useful properties of $\mathcal{T}$ which will form the basis of our guard placement algorithm. Denote the shortest path from any node C_k towards the root by $\Pi(C_k)$. We show the start points of chords lying on the same path follow in order of the path. Furthermore, no chord which is an ancestor of C_k in $\Pi(C_k)$ terminates before C_k starts.

Lemma 9. *If C_m is the ancestor of C_n then $s_m \preceq s_n$ and $s_n \preceq t_m$.*

Proof. First let C_m and C_n belong to the same group. By construction, $s_m \preceq s_n$. Furthermore, if both are Type III chords, then s_m and s_n must lie in the same gap which comes before the gap containing t_m and t_n. Therefore, $s_n \prec t_m$. Similarly, if both are Type IV chords, then if $t_m \prec s_n$ then A_m and A_n are disjoint leading to a contradiction about them being contained in the same arc in the MIS. Hence, if C_m and C_n belong to the same group then the lemma follows.

Next, let C_m and C_n belong to different groups. Since C_m is an ancestor of C_n, we know that the group containing C_m completely contains the group containing C_n (Steps (3) and (4) of the construction of $\mathcal{T}$). Therefore, A_m completely contains A_n implying $s_m \prec s_n \prec t_n \prec t_m$. □

We will place guards to $\triangle$-guard chords in the ordered tree $\mathcal{T}$. By construction, all leaf nodes in $\mathcal{T}$ have disjoint induced subpolygons. Furthermore, only guards along the same path to the root *may* share a cardinal guard. Hence, any guard set must contain at least as many cardinal guards as the number of paths from leaf nodes to the root. However, this lower bound is not sufficient to obtain a constant factor approximation directly. There are instances where the number of guards necessary to $\triangle$-guard a path can vary from as few as two to as many as the number of chords along the path. In addition, two or more paths may merge and thus be able to share guards. Nevertheless, we show that the greedy approach in Algorithm 1 correctly $\triangle$-guards all chords in $\mathcal{T}$ using at most a constant times the number of guards in an optimal guard set (Lemma 12).

The algorithm uses the ordering property presented in Lemma 9. Initially all chords are marked as not being $\triangle$-guarded. At the start of each iteration (Step 4), we pick a chord C_k with the highest depth not yet marked $\triangle$-guarded. All descendants of C_k have been $\triangle$-guarded in previous iterations. We will place a cardinal guard $x \in P_k$ for C_k. We will choose its location to be such that it sees a point on the chord with the lowest depth which lies on C_k's path to the root. All intermediate chords are marked $\triangle$-guarded using at most six guards as given in Step 6. The following lemma proves the correctness of this intermediate step.

Algorithm 1: TreeGuarding

```
  Input: $\mathcal{T}$ Ordered tree of chords in $C'$
  Output: $S_3$ guard set $\triangle$-guarding $C'$
1 $S_3 \leftarrow \emptyset$
2 mark all chords in $\mathcal{T}$ as not $\triangle$-guarded
3 while $\exists$ a chord in $\mathcal{T}$ is not marked $\triangle$-guarded do
4     $k \leftarrow$ largest index such that $C_k$ is not $\triangle$-guarded
5     $i \leftarrow$ smallest index such that some point $y \in C_i \in \Pi(C_k)$ is visible from a
      point $x \in P_k$
6     $S_3 \leftarrow S_3 \cup \{x, y, s_k, t_k, s_i, t_i\}$
7     mark all $C_j \in \Pi(C_k)$ with $i \leq j \leq k$ as $\triangle$-guarded
8 end
9 return guarding set $S_3$
```

Lemma 10. *If a point $x \in P_k$ sees a point $y \in C_i$ such that C_i is the ancestor of C_k, then $\{x, y, s_k, t_k, s_i, t_i\}$ $\triangle$-guard all chords on the path from C_k to C_i.*

Proof. First observe that C_i and C_k are $\triangle$-guarded by guards on their endpoints. Let C_j be any chord on the path from C_k to C_i. If either endpoint of C_j is shared with that of C_i or C_k, then C_j is $\triangle$-guarded. Otherwise, we have C_j lying on the path from C_k to C_i, $i < l < k$. By the ordering property (Lemma 9), $s_i \prec s_j \prec s_k$. We have two cases:

(1) $t_i \preceq t_k$. From Lemma 9, we get the ordering $s_i \prec s_j \prec s_k \preceq t_i \preceq t_k$. Also from Lemma 9, C_j cannot terminate before s_k since C_k is a descendant of C_j. Therefore, C_j must intersect at least one of C_i and C_k and thus be $\triangle$-guarded by the guards placed on the endpoints of C_i and C_k.

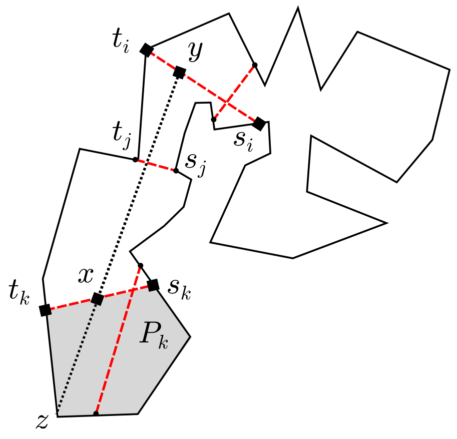


Figure 3.8: One iteration of Algorithm 1 (Steps 4–7). The guards are placed at locations marked by a square. Any chord with a starting vertex lying in between s_i and s_k is $\triangle$-guarded.

(2) $t_k \prec t_i$. We have three cases: (a) $t_k \prec t_j \prec t_i$, (b) $t_j \prec t_k$, or (c) $t_i \prec t_j$. Recall that $s_i \prec s_j \prec s_k$. Hence for (b) and (c), C_j intersects with either C_k or C_i, respectively. Hence, C_j will be $\triangle$-guarded by the guards on the endpoints of C_k and C_i.

Consider case (a) (Figure 3.8). We have $P_k \subset P_j \subset P_i$. $x \in P_k$ sees a point $y \in C_i$. Extend the segment from y to x till it hits the boundary of P_k at point z. Segment zy is a chord in P_i. Since $z \in P_j$, let y' be the point of intersection of segment zy (other than z) with the boundary of P_j. y' may either lie on the edge C_j of P_j or on the part of the boundary of P from s_j to t_j. However, the latter is also a part of the boundary of P_i – in fact, the part of the boundary of P_i which does not contain the edge C_i. This leads to the contradiction that a chord zy intersects the boundary of P_i at three distinct points, z, y and y'. Hence, y' must lie on C_j which implies y' is visible from the guards at x and z. Thus, C_j is $\triangle$-guarded. □

The correctness of the algorithm follows from the correctness of the intermediate step.

Corollary 1. *All chords in $\mathcal{T}$ are $\triangle$-guarded by Algorithm 1.*

We show that the size of S_3 is only a constant times that of any optimal guarding set. Consider an optimal guard set G_{opt} covering C'. For each guard in G_{opt}, we create a new set containing all chords for which the guard acts as a cardinal guard. That is, for any $g \in G_{\text{opt}}$ we create the set $\{C_i | C_i \in C', g \in P_i\}$. Denote this collection of sets by $\mathcal{C}_{\text{opt}}$.

We create another collection of sets, denoted $\mathcal{C}$, for Algorithm 1. For each iteration of the algorithm, we create a new set that contains all chords marked $\triangle$-guarded in Step 7. That is, create the set $\mathcal{C}_k = \{C_j | i \leq j \leq k\}$ and add it to $\mathcal{C}$. The largest index of chords contained in this set corresponds to the largest unmarked index (i.e. k) found in Step 4.

Lemma 11. *If k and k' are the largest indices in distinct sets $\mathcal{C}_k$ and $\mathcal{C}_{k'}$ in $\mathcal{C}$ respectively, then $k \neq k'$ and no set in $\mathcal{C}_{opt}$ contains both C_k and $C_{k'}$.*

Proof. Consider any iteration of Algorithm 1 and the corresponding set in $\mathcal{C}$. If k was the largest unmarked index in Step 4, then it is not included in the sets in $\mathcal{C}$ from previous iterations. Furthermore, all descendants of k are marked $\triangle$-guarded. All chords in the current iteration marked $\triangle$-guarded have indices smaller than k. Hence, if k and k' are the largest indices in two distinct sets of $\mathcal{C}$ then $k \neq k'$.

Now we show that C_k and $C_{k'}$ cannot appear in the same set in $\mathcal{C}_{\text{opt}}$. Suppose they do. We have two possibilities: C_k and $C_{k'}$ lie on the same or different paths to the root. If C_k and $C_{k'}$ lie on different paths to the root, then their induced subpolygons P_k and $P_{k'}$ are disjoint. Hence, their cardinal guards cannot be the same, implying $C_{k'}$ and $C_{k'}$ cannot be in the same set in $\mathcal{C}_{\text{opt}}$.

Then $C_{k'}$ and $C_{k'}$ must lie on the same path. Assume without loss of generality, $k < k'$. Since k and k' lie in the same set in $\mathcal{C}_{\text{opt}}$, they must share the same cardinal guard, say $g \in P_{k'}$. Furthermore, g also sees a point on C_k. Therefore, C_k will be marked $\triangle$-guarded and included in $\mathcal{C}_{k'}$ according to Step 7. However, C_k cannot be included in some other set $\mathcal{C}_{k'} \in \mathcal{C}$, which gives a contradiction. $\square$

Lemma 12. *If S_3 is the guarding set obtained in Algorithm 1, and c_{opt} is the optimal number of guards for $\triangle$-guarding C', then $|S_3| \leq 6c_{opt}$.*

Proof. Since we place at most six guards per iteration, $|S_3| \leq 6|\mathcal{C}|$. We know $|\mathcal{C}_{\text{opt}}| = c_{\text{opt}}$. If we show $|\mathcal{C}| \leq |\mathcal{C}_{\text{opt}}|$, we are done. Suppose $|\mathcal{C}| > |\mathcal{C}_{\text{opt}}|$. Using Lemma 11 this implies there is some chord C_i not contained in any set in $\mathcal{C}_{\text{opt}}$ such that i is the largest index of some set in $\mathcal{C}$. This implies no guard in the optimal guard set acts as the cardinal guard for C_i. From Lemma 4 this implies C_i is not $\triangle$-guarded, which is a contradiction. Thus, $|\mathcal{C}| \leq |\mathcal{C}_{\text{opt}}|$, which proves the statement of the lemma. $\square$

From Lemmas 6, 7, and 12, the guard sets S_1, S_2 and S_3 $\triangle$-guard all input chords using at most 12 times as many guards as an optimal algorithm, thus proving Theorem 3.

3.4 Conclusion

In this chapter, we studied the problem of guarding a polygon under the $\triangle$-guarding constraint [13]. The $\triangle$-guarding constraint is motivated by practical surveillance scenarios where the goal is to see all sides of a person despite self-occlusion. We showed that $\Omega(\sqrt{n})$ guards are always necessary to $\triangle$-guard any simple n–sided polygon. We also presented a $\mathcal{O}(\log c_{\text{opt}})$ approximation algorithm for $\triangle$-guarding the interior using vertex guards. Since the required number of guards to cover the complete interior is large, we turned our attention to a scenario in which we are given entry and exit points to the environment connected by straight-line paths, i.e., chords. The goal is to $\triangle$-guard at least one point on each chord. We presented an approximation algorithm for simply-connected polygons which uses at most 12 times the optimal number of guards. In addition to solving a practical problem, our result is of theoretical interest because this is one of the few instances where a constant factor approximation algorithm for an art gallery problem is known.

Chapter 4

Bearing Sensor Placement for Target Localization

In this chapter, we study a coverage problem where the goal is to accurately locate targets that may be anywhere in the environment. This coverage task arises in many applications, e.g., locating parts in warehouses, intruder detection in surveillance and location-aware services. In fact, indoor positioning systems have been identified as key technologies for the advancement of robotics and automation [62]. A good sensor placement scheme can significantly aid indoor positioning systems.

We focus on the problem of placing bearing sensors for target localization. Bearing sensors are used often in robotics. Monocular cameras, microphone/acoustic arrays, directional radio antennas, passive infrared receivers, etc., all measure bearing towards a target. We consider sensors whose bearing measurements are corrupted by unknown but bounded noise. Bounded noise models provide a useful alternative to probabilistic models especially when a precise device model is not available (perhaps due to the difficulty of calibration or changing device parameters). Such models have long been used for state estimation [63] and for sensor fusion [64].

As an example, consider an application where sensors are deployed to be used as beacons for localizing a person navigating in an indoor setting where no GPS is available (Figure 4.1). At each time instant, the person can query the sensors for their bearing measurements. In the bounded uncertainty model, the true bearing is guaranteed to be in a 2D wedge which is centered at the measured bearing and has an apex angle equal to the maximum sensing noise. Measurements from multiple sensors are combined by intersecting the corresponding wedges. The uncertainty in location is usually taken to be the diameter or area of the intersection. We seek worst-case quality guarantees for our estimate. Irrespective of where the target is in the environment and what measurements the sensors receive (subject to the maximum noise), we would like the uncertainty in locating the target to be below a desired level.

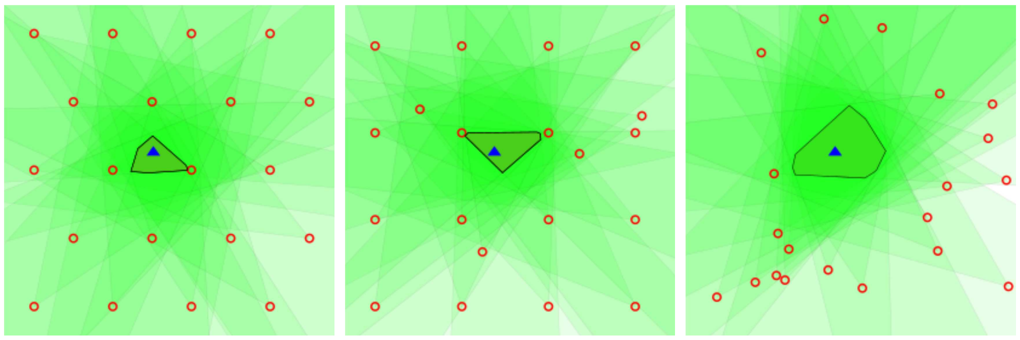


Figure 4.1: The areas of intersection for a square grid[2](middle) and random placement (right) are 1.32 and 3.27 times that of the triangular placement (left). The true location of the target is marked by a triangle.

We consider a simple workspace for the problem: The target can lie anywhere within a square environment without any obstacles or visibility constraints. It is intuitively clear that the optimal placement should be some kind of a uniform grid. However it is not clear if the grid should be square, triangular or some other shape. Further, optimizing parameters of the grid (e.g. resolution) is not straightforward because as illustrated in Figure 4.1, the estimate is obtained by combining measurements from *all* sensors. This makes it difficult to express its area or diameter in closed-form in order to optimize grid parameters. While there have been attempts to find the optimal solution [65], the problem of optimal placement for bearing sensors remains open.

In this dissertation, we make progress towards solving this fundamental problem. We focus on a triangular grid placement and derive the relationship between uncertainty

[2]We randomly place additional sensors to the square grid, so that it has the same number of sensors as the triangular grid.

and grid resolution. We prove that the number of sensors required to achieve a desired uncertainty is only a constant times that of an optimal algorithm. Furthermore for a triangular grid, only a constant number of sensors can be queried to obtain performance comparable to querying *all* sensors. This implies for our motivating example, the person may query only a fixed number of nearby sensors to localize itself without losing much estimation quality.

The rest of the chapter is organized as follows: We begin by presenting the related work in Section 4.1. We describe the sensing model and formalize the problem in Section 4.2. The analysis for lower bounds for an optimal placement, and upper bounds for a triangular grid placement are presented in Sections 4.3 and 4.4. We conclude with a discussion of our results in Section 4.5. Proofs for the main results are given in the chapter, whereas those for some technical lemmas and corollaries are presented in Appendix B.

4.1 Related Work

The problem of optimizing the placement of sensor nodes has received significant attention in the past decade [66]. A large amount of research has focused on self-localization of networks, for example in the case of mobile, reconfigurable sensor networks [67] and for stationary sensor networks with reference anchor nodes [68]. In our present work, we assume that the locations of the sensors themselves are accurately known and focus on the complementary problem of placing sensors so as to localize targets.

For bearing sensors, the uncertainty in target's estimate depends on the relative position of the sensors and the target. Motivated by this, Efrat et al. [14] studied the problem of minimizing the number of sensors to be placed in a polygon, such that each point in the polygon is visible from at least two sensors and their relative angle lies within a desired interval. They presented a log factor approximation subject to a fine discretization.

In addition to the relative angles, the uncertainty is also affected by the distance between the sensors and the target. Geometric Dilution of Precision (GDOP) is one measure relating the uncertainty with distance and relative angles. Tekdas and Isler [69] presented a placement scheme which guarantees that for any target location there are always two sensors whose GDOP is a constant factor of the GDOP achieved by any two sensors from an optimal placement. We do not restrict the estimator to use only two sensors. Instead we allow combining measurements from all sensors.

Ercan et al. [70] studied the problem of placing horizontal scan-line cameras only along the boundary of a circular room to minimize least-squares localization error for a target with a given prior. Their placement result shows that a uniform placement along the boundary is optimal. We allow sensors to be placed anywhere within a square workspace, without assuming any prior for the target's location.

Sensor selection is closely related to the sensor placement problem. Isler and Magdon-Ismail [71] considered the problem of selecting a small subset of sensors from a given placement. Each sensor's output is a convex subset of the plane. They proved that irrespective of the total number of sensors, there is always a subset of four measurements that can be selected, which when combined yield an intersection area at most twice of that obtained by intersecting all measurements. In their problem, the placement of the sensors and the actual sensor measurements are already given. For the same placement of sensors, this subset would change if the measurement changes. This poses an interesting question of whether there is some placement of sensors for which the same subset can be used to approximate the uncertainty region for different (but perhaps "nearby") measurements. In this chapter, we present a result in this direction for bearing measurements with bounded noise.

We begin by defining the bounded noise sensing and uncertainty models in the following section.

4.2 Problem Formulation

We first describe the notation, then define the sensing and estimation models, and use them to formulate the problem studied in this chapter.

4.2.1 Notation and Sensing Model

The workspace $\mathcal{A}$ is a $d \times d$ square. The target's true location x can be anywhere within $\mathcal{A}$. Consider a sensor placement $S = \{s_1, \ldots, s_n\}$ where each $s_i \in \mathcal{A}$ denotes the sensor location. Each sensor measures the bearing towards the target as $\theta_i^m = \theta_i^t + n_i$, where

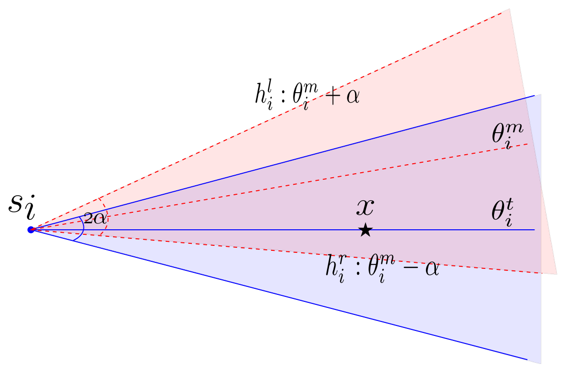


Figure 4.2: The actual measurement θ_i^m lies anywhere between $\theta_i^t \pm \alpha$. θ_i^t is the true bearing. The wedge for a given measurement is guaranteed to contain the true target location x.

$\theta_i^t \in [0, 2\pi)$ is the true bearing (Figure 4.2). $n_i \in [-\alpha, +\alpha]$ is the bounded sensor noise. α is the bound on the absolute noise in the sensor. The pre-image of a measurement θ_i^m is a 2D wedge (denoted by $W(s_i, \theta_i^m)$) as shown in Figure 4.2. This wedge is not the same as a fixed field-of-view sensor; for the same target location, the sensor can receive any sensing wedge of angular width 2α so long as it contains the true target location.

The target estimate obtained by combining a set of measurements $\theta^m = [\theta_1^m, \ldots, \theta_n^m]^T$ from n sensors, is defined as the intersection of the n sensing wedges $W(s_i, \theta_i^m)$. That is, $\hat{P}(S, \theta^m) \triangleq \bigcap_{i=1}^n W(s_i, \theta_i^m)$. Here $\hat{P}$ is a convex polygonal region which can possibly be unbounded.

4.2.2 Adversarial Formulation of Uncertainty

The size of $\hat{P}$ depends on the actual measurements. Figure 4.3 shows two instances where the size of $\hat{P}$ differs significantly for different measurements obtained from the same placement of sensors. The actual measurements obtained by the sensors cannot be controlled by the user. However, we will show that by carefully placing the sensors one can guarantee there always exists a good set of valid measurements.

There are two approaches to model the situation: We can assume a distribution for the measurements and the target and optimize for the expected quality of estimate.

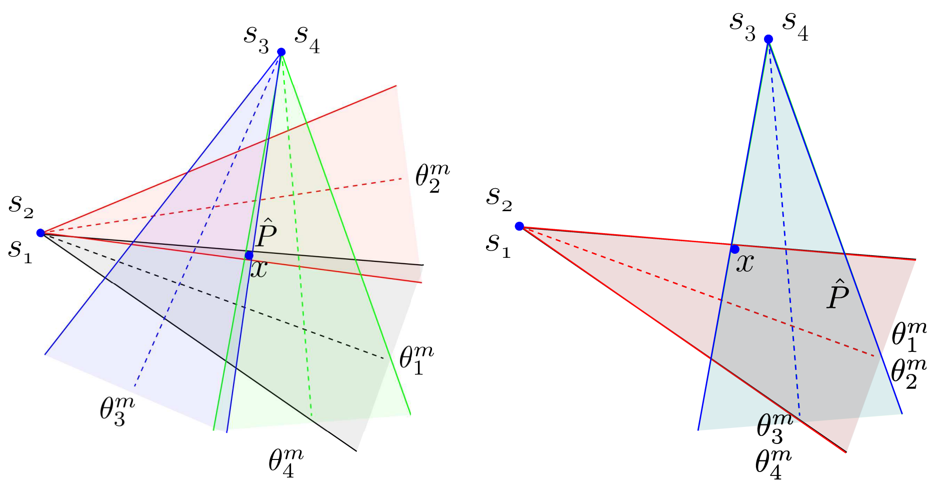


Figure 4.3: Two estimates for the same sensors and target location, but different measurements resulting in different uncertainty. We use worst-case intersection as the uncertainty measure.

Alternatively, we can model this as an adversarial process and guard against worst-case measurements and target locations. Adversarial models are appealing since no additional information about the sensors or targets is needed and since they provide guarantees over all possible scenarios, as opposed to only the average case scenario.

We choose to model the objective using an adversarial process: Given a placement of sensors, an adversary selects a target location within the square and a corresponding set of measurements to maximize the uncertainty in the target estimate. We use two measures (area and diameter[3] of $\hat{P}$) to define the uncertainty. The diameter uncertainty of a placement S is defined as:

$$U_D(S) \triangleq \max_{x \in \mathcal{A}} \max_{\theta^m \in \theta(x)} \text{diameter}(\hat{P}(S, \theta^m)), \tag{4.1}$$

where $\theta(x)$ is the set of valid measurements that can be obtained from S for a target location x. The area uncertainty can be similarly defined.

[3]The diameter of a polygon is the length of the largest segment contained completely within the polygon.

4.2.3 Objective

Broadly, there are two factors that affect the worst-case uncertainty: (i) the number of sensors, and (ii) the location of placed sensors. In this work, we take the approach that the user specifies a desired uncertainty and the objective is to minimize the number of sensors and find the corresponding placement to guarantee that the worst-case uncertainty is below the user-specified value. In particular, we address the following problem: *Find the minimum number of sensors required and the corresponding placement to achieve a desired diameter uncertainty U_D^* (or area uncertainty U_A^*).*

Our main result shows that a triangular placement scheme compares competitively with respect to the (unknown) optimal algorithm.

Theorem 4. *Let the maximum absolute noise for bearing sensors be $0 < \alpha \le \frac{\pi}{4}$. Let the desired diameter uncertainty for a $d \times d$ square environment be $U_D^* < \dfrac{d}{7\sin\alpha}$ (respectively, area uncertainty be $U_A^* < \dfrac{\pi\sin^2\alpha}{196}d^2$). If an optimal placement algorithm achieves U_D^* (respectively, U_A^*) with n^* sensors, then a triangular grid-like placement achieves at most $5.88U_D^*$ (respectively, at most $7.76U_A^*$) with at most $9n^*$ sensors.*

The analysis for Theorem 4 is based on covering a $d \times d$ square with equilateral triangles of sensors. When the desired uncertainty is higher than the restriction in Theorem 4 and comparable to the size of $\mathcal{A}$, an optimal placement may use very few sensors. Nevertheless, even for that case the total number of sensors for the grid-like placement is bounded (given by Lemma 17).

In the following sections, we analyze the number of sensors required for an optimal algorithm and for a triangular grid-like placement.

4.3 Lower Bounds for Optimal Placement

In this section, we first present lower bounds on the uncertainty achieved by any placement of sensors in the plane. We apply this to bound the number of sensors placed within $\mathcal{A}$ by an optimal algorithm.

First consider the case when the maximum sensing noise $\alpha \ge \frac{\pi}{2}$, i.e., the sensing wedges are at least half-planes. We show that the adversary can always choose a valid measurement set for any placement, such that the sensing wedges have an unbounded intersection.

Theorem 5. *For any placement S of n bearing sensors with maximum absolute noise $\alpha \geq \frac{\pi}{2}$, there exists a measurement set θ^m such that the intersection of the wedges $(\bigcap_{i=1}^{n} W(s_i, \theta_i^m))$ is unbounded.*

Proof. Consider the following construction for $\alpha = \frac{\pi}{2}$: Draw a line l passing through any sensor s_i and any point in $\mathcal{A}$ (denoted by x). For all sensors s_b such that vector product $\overrightarrow{xs_i} \times \overrightarrow{xs_b}$ is zero or negative, let θ^m be the direction obtained by rotating $\overrightarrow{s_bx}$ clockwise by $\frac{\pi}{2}$ and for all sensors s_u such that $\overrightarrow{xs_i} \times \overrightarrow{xs_u}$ is positive, let θ^m be obtained by rotating $\overrightarrow{s_ux}$ counter-clockwise by $\frac{\pi}{2}$. For all sensors, the sensing wedges are half-planes described by lines which pass through x. Further each sensing wedge contains the half-line starting from x and passing through s_i. Hence, the intersection of all sensing wedges is unbounded. For all $\alpha > \frac{\pi}{2}$, the sensing wedges are a superset of that obtained with $\alpha = \frac{\pi}{2}$. Hence, the proof holds. □

Theorem 5 implies that when $\alpha \geq \frac{\pi}{2}$ the uncertainty can be as large as $\mathcal{A}$, i.e., $U_A(S) = \Theta(d^2)$ and $U_D(S) = \Theta(d)$ for any placement of sensors, including the optimal. This is not surprising, since $\alpha \geq \frac{\pi}{2}$ corresponds to very high noise. In practice, bearing sensors are much more accurate. For the rest of this chapter, we only focus on the case when the maximum sensing noise $\alpha < \frac{\pi}{2}$.

In the following, we will lower bound the uncertainty for any placement parametrized by the distance of the target to the closest sensor. Recall from Equation 4.1, the uncertainty is defined as the max over all possible target locations, and all valid measurements. Hence, for a lower bound, it is sufficient to consider a particular target location and valid measurement set, as given next.

Lemma 13. *If there exists a circle C with radius r which doesn't contain any sensor from a placement S of n bearing sensors, then the diameter uncertainty is bounded as $U_D(S) \geq 2r \sin \alpha$ (respectively, $U_A(S) \geq \pi r^2 \sin^2 \alpha$).*

The proof for Lemma 13, given in the appendix, shows that when the target lies at the center of C and each sensor receives a measurement equal to the true bearing, a circle of radius $r \sin \alpha$ centered at the target lies completely within the intersection of all sensing wedges. This instance gives a lower bound for the worst-case uncertainty.

When a desired uncertainty is given, we can apply Lemma 13 to find the radius of the largest such circle lying in the workspace $\mathcal{A}$ and not containing any sensor. Such a radius, denoted by r^*, is the maximum distance for any point in $\mathcal{A}$ to the closer of the closest sensor in S^* or the closest point on the boundary of $\mathcal{A}$. We can now apply Lemma 13 to bound how large r^* can be, when a desired diameter or area uncertainty is given.

Corollary 2. *Let S^* be an optimal placement achieving a desired diameter uncertainty U_D^* (respectively, area uncertainty U_A^*) in a square workspace of side d. If r^* is the radius of the largest circle lying completely within $\mathcal{A}$ and not containing any sensor in its interior, then $r^* \leq \frac{U_D^*}{2 \sin \alpha}$ (respectively, $r^* \leq \sqrt{\frac{U_A^*}{\pi} \frac{1}{\sin \alpha}}$).*

Corollary 2 implies an upper bound on how far each point in $\mathcal{A}$ can be from any sensor or the boundary of $\mathcal{A}$. This allows us to bound the number of sensors required for an optimal algorithm as a function of r^*. Corollary 3 states that $\Omega\left(\frac{d^2}{r^2}\right)$ sensors are needed to guarantee coverage of a $d \times d$ area.

Corollary 3. *Let r^* be the radius of the largest circle within a square of side d, not containing any sensor from an optimal placement in its interior. If the desired diameter uncertainty is $U_D^* < d \sin \alpha$ (respectively, $U_A^* < d^2 \frac{\pi \sin^2 \alpha}{4}$) then the number of sensors for an optimal algorithm $n^* \geq \frac{(d - 2r^*)^2}{\pi r^{*2}}$.*

When $d \leq 2r^*$, the desired uncertainty is comparable to $\mathcal{A}$, and the optimal algorithm would place very few sensors, yielding a trivial lower bound. The bound on the uncertainty implies that $d > 2r^*$ is an interesting case: If $d > 2r^*$, then there is a smaller square within $\mathcal{A}$ where all points are more than r^* away from the boundary and hence require at least one sensor within r^*. We can show that the set of circles of radii r^* drawn about each sensor in the optimal placement, should form a cover of this smaller square, yielding the bound.

4.4 Performance Analysis for the Triangular Grid Placement

Next, we analyze the number of sensors required and the uncertainty for a triangular grid-like placement. While for lower bounds it sufficed to consider specific instances, upper bounds require considering all possible target locations and sets of measurements.

4.4.1 Uncertainty with Triangular Grid

Before the main analysis, first consider two special configurations of sensors: (i) three sensors placed on the vertices of an equilateral triangle $\triangle s_1 s_2 s_3$ with side r, when $0 < \alpha < \frac{\pi}{6}$, and (ii) six sensors placed on the vertices of a regular hexagon when $\frac{\pi}{6} \leq \alpha \leq \frac{\pi}{4}$. For case (i), the target may lie anywhere within $\triangle s_1 s_2 s_3$ (Figure 4.4(a)). We further divide the analysis into intervals based on α, given next.

Lemma 14. *Let $\triangle s_1 s_2 s_3$ be an equilateral triangle of side r with a bearing sensor placed at each vertex. If the target lies within $\triangle s_1 s_2 s_3$ and $S = \{s_1, s_2, s_3\}$ then*

$$U_D(S) \leq \begin{cases} 11.35 r \sin\alpha & 0 < \alpha < \frac{\pi}{18}, \\ 2.04 r & \frac{\pi}{18} \leq \alpha < \frac{\pi}{12}, \\ \left(1 + \frac{1}{\sqrt{3}}\right) r & \frac{\pi}{12} \leq \alpha < \frac{\pi}{6} \end{cases}$$

and,

$$U_A(S) \leq \begin{cases} 23.46 r^2 \sin^2\alpha & 0 < \alpha < \frac{\pi}{18}, \\ \frac{\sqrt{3} r^2}{4} + 10.1 (r \sin\alpha)^2 & \frac{\pi}{18} \leq \alpha < \frac{\pi}{12}, \\ \frac{3\sqrt{3} r^2}{4} & \frac{\pi}{12} \leq \alpha < \frac{\pi}{6}. \end{cases}$$

The proof, given in Appendix B.2.1, partitions the triangle into three regions, and assigns sensors for each region such that *any* valid set of measurements results in bounded intersection. The sensing wedges corresponding to each partition are approximated to bound their intersection.

When $\alpha \geq \frac{\pi}{6}$, the sensing wedges become too large to result in bounded intersection with just three sensors. Instead we use six sensors, placed on a regular hexagon with center o and side r, to bound their intersection. The target can lie anywhere within a circle of radius $\dfrac{r}{\sqrt{3}}$ centered at o. We find an upper bound to the uncertainty, by finding the intersection of the union of all sensing wedges for each sensor, corresponding to all target locations within the circle.

Lemma 15. *Let $s_1 \dots s_6$ be a regular hexagon of side r and center o with a bearing sensor placed at each vertex, and maximum absolute noise $\frac{\pi}{6} \leq \alpha \leq \frac{\pi}{4}$. If the target lies inside a circle of radius $\dfrac{r}{\sqrt{3}}$ centered at o then,*

$$U_D(\{s_1, \dots, s_6\}) \leq r\left(\sin^2\alpha + 3.76\sin\alpha + 1.232\right)$$

and, $U_A(\{s_1, \dots, s_6\}) \leq 1.5rU_D(\{s_1, \dots, s_6\}$.

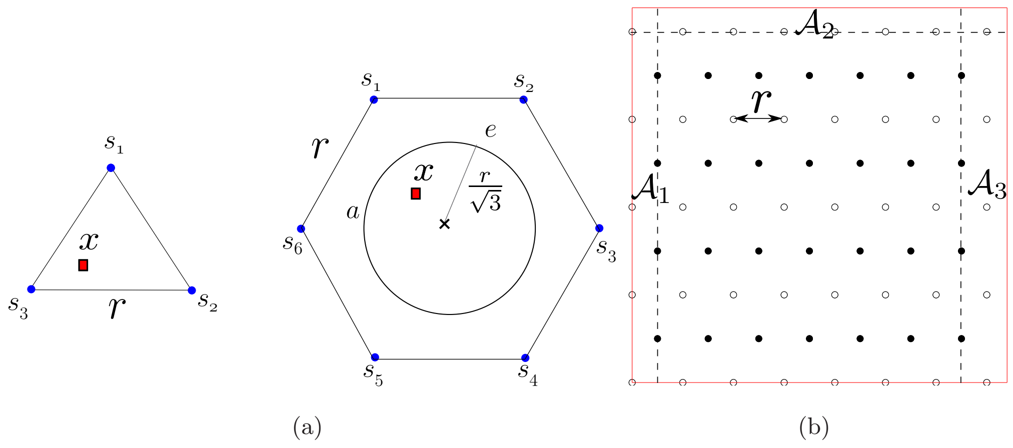


Figure 4.4: (a) Based on α, we upper bound the uncertainty when the target lies within an equilateral triangle or a circle contained within a regular hexagon of sensors. (b) We pad the three regions with additional sensors to ensure any point in $\mathcal{A}$ is enclosed by an equilateral triangle of sensors.

Lemma 15 bounds the intersection when the sensing wedges are at most a quadrant ($\alpha \leq \frac{\pi}{4}$), and the target lies within a circle of radius $\dfrac{r}{\sqrt{3}}$. We can extend the result in Lemma 15 for $\alpha = \frac{\pi}{2} - \epsilon$ with $0 < \epsilon$, to bound the number of sensors placed on a triangular grid, sufficient to guarantee that the intersection of all wedges is bounded.

Theorem 5 shows that when sensing wedges are at least a half-plane ($\alpha \geq \frac{\pi}{2}$), the resulting intersection can be unbounded in the worst-case.

Lemma 16. *Let the maximum absolute sensing noise be* $\alpha = \frac{\pi}{2} - \epsilon$ *with* $0 < \epsilon$. *If* $\sin^{-1}\left(\frac{1}{\sqrt{3}k}\right) + \frac{\pi}{6(k-1)} < 2\epsilon$ *then* $\mathcal{O}(k^2)$ *sensors placed on a triangular grid are sufficient for bounded intersection of sensing wedges when the target lies within a circle of radius* $\frac{r}{\sqrt{3}}$.

The proof is given in Appendix B.2.3.

4.4.2 Number of Sensors with Triangular Grid

Lemma 14 gives an upper bound on the uncertainty for a placement of sensors in $\mathcal{A}$, if there exists an equilateral triangle of sensors enclosing any point in $\mathcal{A}$. Since the sensors cannot be placed outside of $\mathcal{A}$, regions near the boundary of $\mathcal{A}$ may not have an enclosing equilateral triangle if sensors are placed only on a triangular grid. The three regions where this occurs are marked $\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3$ in Figure 4.4(b). We place additional sensors within these regions to ensure that any point in $\mathcal{A}$ is enclosed by an equilateral triangle of sensors. Lemma 17 states that $\mathcal{O}\left(\frac{d^2}{r^2}\right)$ sensors are sufficient to cover a square of area $d \times d$.

Lemma 17 (Upper Bound on Number of Sensors)**.** *If* $w_r = \lfloor d/r \rfloor + 1,$, $w_c = \lfloor d/\sqrt{3}r \rfloor + 1$, $b_r = \left\lfloor (d - \frac{r}{2})/r \right\rfloor$, $b_c = \left\lfloor \frac{d - \frac{\sqrt{3}r}{2}}{\sqrt{3}r} \right\rfloor + 1$ *are the number of sensors in odd and even rows and columns, respectively of a triangular grid with side* r *in a square of side* d, *then* $w_r w_c + b_r b_c + 3(2w_r + b_r) + 8$ *sensors are sufficient to cover the square with equilateral triangles of side* r.

The lower and upper bounds obtained can be applied to get the main result of this chapter. Recall from Corollary 2, that an optimal algorithm has to place a sensor within distance $r^* \leq \frac{U_D^*}{2\sin\alpha}$ (equivalently, $r^* \leq \sqrt{\frac{U_A^*}{\pi}\frac{1}{\sin\alpha}}$) of every point $\mathcal{A}$ to ensure the desired uncertainty. For the triangular grid placement, set the grid length as $r = \frac{U_D^*}{2\sin\alpha}$ (respectively, $r = \sqrt{\frac{U_A^*}{\pi}\frac{1}{\sin\alpha}}$). Hence, $r^* \leq r$. Corollary 3 gives a lower bound on the number of sensors required for an optimal algorithm in terms of r^*, and Lemma 17 gives an upper bound for the grid-like placement in terms of r. Lemmas 14 and 15 bound the uncertainty of the grid-like placement in terms of r. Using $r^* \leq r$ and substituting the value of r, the result in Theorem 4 can be obtained.

The upper bounds from Lemma 14 and 15 reveal that only a small number of sensors in our placement suffice to achieve uncertainty comparable to that obtained by combining all measurements. This is useful when there is prior knowledge about the target location (e.g. a subset of $\mathcal{A}$) and only those sensors corresponding to the enclosing triangle or hexagon need be queried for their measurements.

Corollary 4. *Given a target location x within a square and desired diameter uncertainty U_D^* (respectively, area uncertainty U_A^*), if sensors are placed on a triangular grid with side $r = \frac{U_D^*}{2\sin\alpha}$ (respectively, $r = \sqrt{\frac{U_A^*}{\pi}\frac{1}{\sin\alpha}}$), three sensors are sufficient when $0 < \alpha < \frac{\pi}{6}$ and six sensors are sufficient when $\frac{\pi}{6} \leq \alpha \leq \frac{\pi}{4}$ to ensure diameter uncertainty at most $5.88U_D^*$ (respectively, $7.76U_A^*$).*

Corollary 4 provides a sensor selection method which may be useful in sensor network applications with energy or bandwidth constraints that require activating only a small number of sensors.

4.5 Conclusion

In this chapter, we studied a sensor placement problem for covering an environment with bearing sensors. Bearing measurements from multiple sensors are used to precisely locate a target within the environment. We used a bounded uncertainty formulation which allowed us to represent each measurement as a wedge containing the target's location. The quality of the estimated target location was quantified by the diameter or the area of the intersection of wedges. In this setting, a fundamental question that arises is: What is the minimum number and placement of sensors that guarantees that no matter where the target is, or what the actual measurements are, the uncertainty in the estimate is below a desired level?

This basic question turned out to be surprisingly hard due to the fact that the quality of the estimation depends on the locations of all sensors as well as the actual measurements. Our results provided insights about the structure of this problem and yielded a placement scheme with constant-factor approximation guarantees. In particular, we showed that unless the sensor noise is too large, a placement of sensors on a triangular grid yields a good performance. Further, (excluding some extreme cases) we showed that for the triangular grid placement, if a rough estimate of the target location is available, one can obtain a good estimate by querying only a fixed number of sensors. This latter sensor selection scheme is particularly appealing for resource constrained sensor-network applications.

Chapter 5

Multi-Target Visual Tracking with Teams of Aerial Robots

In this chapter, we study the problem of visually tracking targets that are moving on the ground using a team of aerial robots. This task routinely appears in a number of applications. For example, Vermeulen et al. [72] flew a small unmannned aerial aircrafts at an altitude of 100 m to survey the population of elephants in a game ranch in Burkina Faso. Developing efficient target tracking algorithms, consequently, has important practical implications.

In previous chapters, we studied coverage problems where our emphasis was on ensuring every point in the environment was sensed using the least number of sensors. In this chapter we will consider the dual formulation where the number of sensors, i.e., the robots, is fixed and instead we would like to achieve the best possible sensing performance. The overall goal is to plan for the trajectories of the robots in order to track the most number of targets, and accurately estimate the target locations using the images. The two objectives can conflict since a robot may fly to a higher altitude and potentially cover a larger number of targets at the expense of accuracy.

We start by showing that it may not always be possible to track all targets while always maintaining the optimal quality of tracking (or any factor of the optimal quality), even if the targets' motion is fully known. Hence, we focus on the following two variants: maximize the number of targets tracked subject to a desired tracking quality per target, and maximize the sum of quality of tracking for all targets. We show how the two problems can be formulated as the unweighted and weighted versions of the Maximum Group Coverage Problem (MGC). A simple greedy approach provides a 1/2 approximation to unweighted MGC [73]. We show that the approximation guarantee also holds for the weighted case which allows a practical solution to the trajectory planning problem with provable performance guarantees. We evaluate the algorithm in simulations and preliminary experiments with an indoor platform using four aerial robots.

The rest of the chapter is organized as follows: We begin with a review of the related work in Section 5.1. The problem setup and a discussion of the sensing quality are presented in Section 5.2. The infeasibility of tracking all targets with a constant factor of the optimal quality is proven in Section 5.3. The tracking algorithm is presented in Section 5.4, and evaluated through simulations and preliminary experiments in Sections 5.5 and 5.6 respectively. Section 5.7 concludes the chapter.

5.1 Related Work

Active target tracking is an important problem for robotics, and has been widely studied under different settings. Spletzer and Taylor [74] considered the problem of tracking multiple mobile targets with multiple robots. They presented a general solution based on particle filtering in order to choose robot locations for the next time step that maximizes the quality of tracking. Frew [75] studied the problem of designing a robot trajectory, as opposed to just the next robot location, in order to maximize the quality of tracking a single moving target. LaValle et al. [76] studied the problem of maintaining the visibility of a single target from a robot for the maximum time. Gans et al. [77] presented a controller that can keep up to three targets in one robot's field-of-view.

When the motion of the targets is fully known, the tracking problem can be formulated as a kinetic facility location problem. The goal of the stationary version is to place k facilities (sensors) given the location of n sites (targets), so as to minimize the maximum distance between a facility and a site. In the kinetic version, the sites are mobile and the motion of the facilities is to be designed. Bespamyatnikh et al. [78]

and Durocher [79] presented approximation algorithms to control respectively one and two mobile facilities, when the trajectories for the sites are given. Recently, de Berg et al. [80] presented improved approximation algorithms with two mobile facilities when only an upper bound on the velocities of the sites is available. However, the general problem of kinetic facility location with k facilities is open.

On the other extreme, when no prior information of the targets is available, the multi-robot tracking problem can be formulated as a coverage problem [81]. Schwager et al. [82] presented strategies to control the position and orientation of overhead cameras mounted on aerial robots in order to achieve equal visual coverage of the ground plane.

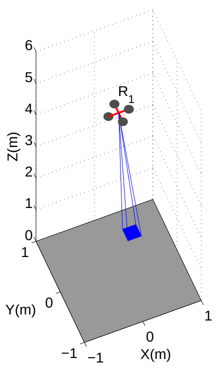
Unlike previous works, we study the trade-off between quality of tracking, and the number of targets tracked. We present an algorithm that chooses *trajectories* for each robot, instead of choosing just the next best location. This algorithm can be applied to the following two versions of the problem: tracking maximum number of targets, and maximizing the quality of tracking. We begin by formulating the problem and describing the sensing model.

5.2 Problem Formulation

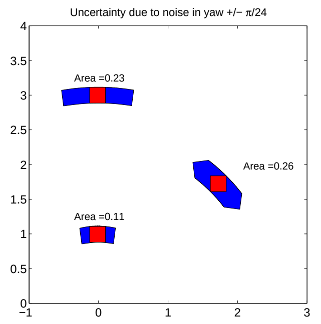
Let k denote the number of robots, and n denote the total number of targets in the environment. We assume that the robots can communicate with each other at all times. The position of any robot or target is specified by their 3D coordinates x, y, z. The position of the i^{th} robot at time τ is denoted by $r_i(\tau)$. Let $z_{\min}$ be the minimum flying altitude. All robots have a camera that faces downwards. Let ϕ represent the field-of-view angle for the cameras.

Let $t_i(\tau)$ denote the position of the i^{th} target. $t_i(\tau)$ is given by the position of a reference point that the robots can use to uniquely identify any target. For example, the reference point can be the centroid of a colored patch or a unique feature point on the object. All targets always move on the ground plane, i.e., $z = 0$ for all t_i.

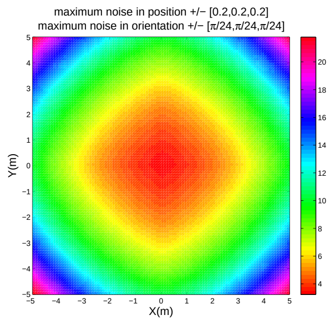
The reference point of any target t_i in the field-of-view of a robot projects to some pixel in the image. A pixel can be backprojected to a ray in the world frame. In general, with no other information, it is not possible to solve for the target's location along this ray with a single camera measurement. However, since we assume that all targets move



(a)



(b)



(c)

Figure 5.1: (a) Backprojection from a pixel yields a pyramid. (b) Uncertainty in target's estimate due to uncertain yaw angle of the robot. (c) Map showing the area of projection for the true target at $[x, y, 0]$ (best viewed in color). The camera pose is estimated to have position $[0, 0, 5]$ m and roll, pitch and yaw angles as 0 radians. Maximum image noise is ± 5 pixels.

on the ground plane, we can solve for the coordinates of t_i.

Ideally, we can exactly estimate t_i given an image measurement, the camera pose, and the projection matrix. In practice, however, the following factors lead to an uncertain estimate of t_i:

1. The backprojection of camera pixels, which have quantized, integer coordinates, is no longer single ray but a pyramid (Figure 5.1(a)).

2. Pixel measurements may be corrupted by noise. If the maximum noise is bounded by Δp pixels, we backproject the set of pixels $\pm\Delta p$ around the measured pixel. The true target location is contained within the larger backprojection.

3. The pose of the camera (or the robot) may not be accurately known. Typically, using exteroceptive sensors such as GPS and compass, we can bound the maximum uncertainty in estimating the robot pose. When the robot pose is known up to a bounded uncertain set, we can compute the backprojection for each pose within the set (Figure 5.1(b)).

In general, the quality of tracking under the three sources of errors, is a function of the relative distance and angle between the robot and the target, as seen in Figure 5.1(c). For a given true location of the target and an estimate of the robot pose, Figure 5.1(c) plots the maximum area of backprojection over all possible noisy measurements of the target, and all possible true robot poses.

Robots only have an estimate of the true target position while tracking. The uncertain estimate can be represented as a set of possible target locations on the ground plane. Given a motion model, the robots can propagate the set to obtain predicted target position, e.g., using particle filtering [83]. The maximum area of backprojection can be computed for each predicted target position as shown in Figure 5.1(c).

The quality of tracking for a given target and robot pair can be defined as some measure of the areas of backprojection found for a predicted target position. Let $q_i(r_j, \tau)$ denote the measure for target t_i and robot r_j at time τ. The quality of tracking t_i at τ, is given by the best quality of tracking amongst all robots tracking t_i, i.e., $q_i(\tau) = \max_j q_i(r_j, \tau)$. Finally, the total quality of tracking at τ is given by the sum of quality over all targets $Q(\tau) = \sum_{\forall i} q_i(\tau)$ over all targets. Alternatively, we may also consider the bottleneck quality over all targets $Q(\tau) = \min_i q_i(\tau)$.

5.3 Infeasibility of Tracking All Targets

In this section, we show the infeasibility of tracking all targets while maintaining any constant factor approximation of the optimal quality of tracking. We prove this by constructing an instance where the two goals, track all targets and maximize quality of tracking, conflict each other. We create a simple instance on a line where the quality of tracking is inversely proportional to the distance between the robot and the target: $q_i(r_j, \tau) = 1/d(t_i(\tau), r_j(\tau))$ if t_i is in the field-of-view of r_j, and $q_i(r_j, \tau) = 0$ otherwise. The overall quality of tracking will be given by the bottleneck quality $Q(\tau) = \min_i q_i(\tau)$.

We use the instantaneous optimal quality of tracking, $Q^*(\tau)$, as the baseline for comparison. $Q^*(\tau)$ is the quality of tracking at τ, if one were to optimally *place* all the cameras at any location for any τ, regardless of their locations before τ. The placement of k cameras achieving $Q^*(\tau)$ may be significantly different from the placement achieving $Q^*(\tau - \epsilon)$. There may or may not exist k continuous robot trajectories achieving $Q^*(\tau)$. Nevertheless, $Q^*(\tau)$ is an upper bound on the quality of tracking. This raises the question of whether we can at least maintain a constant-factor approximation of $Q^*(\tau)$ while tracking all targets. The theorem given next shows this is not possible, even when the motion of the targets is fully known.

Theorem 6. *Let $Q^*(\tau)$ be the instantaneous optimal quality of tracking at time τ. Let the maximum speed of all targets be v. For any $0 < \alpha \leq 1$ and $\beta > 0$, no algorithm can track $n > k$ targets with at least $\alpha Q^*(\tau)$ quality for all τ with $k \geq 3$ robots having a maximum speed of βv.*

Proof. Consider Figure 5.2. We have $k = 3$ robots and $n = 4$ targets on a line. The distance between t_3 and t_4 is 0 at time 0. Targets t_1, t_2 and t_3 remain stationary at all times, and t_4 moves with $v = 1$ to the right on the line. $z_{\min} = 1$ and $\phi = \pi/4$ denote the minimum flying altitude and field-of-view angles (Section 5.2).

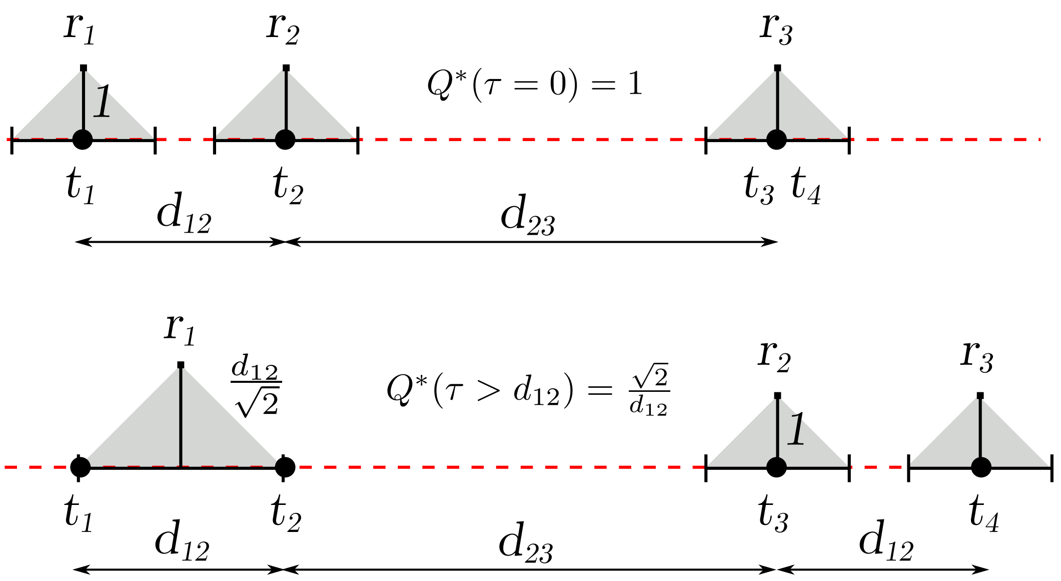


Figure 5.2: At $\tau = 0$, t_3 and t_4 are covered by the same robot to achieve $Q^*(0)$, where as for $\tau > d_{12}$, t_3 and t_4 are covered by separate robots.

If we have 4 targets and 3 robots, then there must exist a robot covering at least two targets at any given time. At $\tau = 0$, we can verify that the optimal algorithm uses separate robots to cover t_1 and t_2, and one robot to cover t_3 and t_4 (Figure 5.2). That is, $Q^*(0) = 1$. Similarly, for any time $\tau > d_{12}$, optimal uses separate robots to cover t_3 and t_4, and same the robot to cover t_1 and t_2 making $Q^*(\tau) = \frac{\sqrt{2}}{d_{12}}$.

Thus, in any optimal algorithm, of the two robots covering t_1 and t_2, one will switch to cover either t_3 or t_4, after $\tau = d_{12}$. *An approximation algorithm, on the other hand, does not necessarily have to make the same switch.* Nevertheless, by setting d_{12} appropriately, we will show that any approximation algorithm will be required to make a switch at some time. By making d_{23} sufficiently large, we will show that such a switch is infeasible with bounded velocity robots. The rest of the proof shows the existence of appropriate d_{12} and d_{23} values. This construction is similar to the one used by Durocher [79] to prove the inapproximability of the kinetic k–center problem. For the case of aerial robots, however, we show how to additionally take into account non-zero $z_{\min}$ and ϕ values.

Let ALG be any algorithm that maintains a quality $Q(\tau) \geq \alpha Q^*(\tau)$. If we set $d_{12} > \frac{\sqrt{2}}{\alpha}$, then ALG cannot use the same robot to cover t_1 and t_2 at time $\tau = 0$. Else, $Q(0) < \alpha = \alpha Q^*(0)$ which violates the approximation guarantee. Hence, ALG uses separate robots to cover t_1 and t_2 at time 0.

Similarly, we can show that for any time $\tau > \frac{d_{12}}{\alpha}$, ALG must use separate robots to cover t_3 and t_4. Else $Q(\tau) < \frac{\sqrt{2}}{\tau} < \alpha Q^*(\tau)$ violating the approximation guarantee.

One of the two separate robots, say r, covering t_1 and t_2 initially, must cover either t_3 and t_4 at time $\tau > \frac{d_{12}}{\alpha}$. In time τ, r must travel at least $d_{23} - \frac{1}{\alpha} - \frac{d_{12}}{\sqrt{2}\alpha}$ distance. Here, $\frac{1}{\alpha}$ and $\frac{d_{12}}{\sqrt{2}\alpha}$ come from the condition that $Q(0) \geq \alpha$ and $Q(\tau) \geq \alpha \frac{\sqrt{2}}{d_{12}}$.

Consider a time $\tau = \frac{2d_{12}}{\alpha}$. At this time, r covers a maximum distance of $\beta\tau = \beta\frac{2d_{12}}{\alpha}$. Set $d_{23} > \beta\frac{2d_{12}}{\alpha} + \frac{1}{\alpha} + \frac{d_{12}}{\sqrt{2}\alpha}$. r cannot simultaneously cover at least one of t_1 or t_2 at time 0, and at least one of t_3 or t_4 at time τ, which is a contradiction. Hence, ALG cannot maintain an α approximation of Q^* for all times.

$\square$

The instance created in the proof above uses minimum flying altitude $z_{\min} = 1$ and camera field-of-view angle $\phi = \pi/4$. We can create corresponding instances for any other values of these parameters. In light of Theorem 6, we drop the requirement that all targets must always be tracked. Instead we focus on the case when the robots are allowed to track a fraction of all targets.

5.4 1/2 Approximation Algorithm

In this section, we present the main algorithm to maximize the number of targets tracked, or maximize the quality of tracking. We divide the time into rounds of fixed duration. We consider the scenario where using measurements from previous rounds, the robots are able to predict the motion of the targets for the current round. For each robot, we create a set of m candidate trajectories that can be followed for the current round. For example, these trajectories can be generated using existing grid-based or sampling-based methods [84]. Our goal is to choose a trajectory for each of the robots for the current round.

Figure 5.3 shows a simple instance with two robots and three candidate trajectories per robot. The camera footprint along two such trajectories as well as the set of targets covered by these trajectories are shown. Note that the trajectories need neither be restricted to any discretized grid, nor have uniform length or uniform speed.

Let $R_j(x)$ denote the set of targets predicted to be covered by x^{th} trajectory followed by j^{th} robot. We create a set system $(X, \mathcal{R})$ where X is the set of all targets and $\mathcal{R}$ is a collection of all $R_j(x)$ sets. We group sets in $\mathcal{R}$ into k collections, one per robot. Each group contains m sets each. That is,

$$\mathcal{R} = \{ \underbrace{R_1(1), \ldots, R_1(m)}_{\text{candidate trajectories for } r_1}, \ldots, \underbrace{R_k(1), \ldots, R_k(m)}_{\text{candidate trajectories for } r_k} \} \tag{5.1}$$

A valid assignment of trajectories can be represented by a map, $\sigma : [1, \ldots, k] \rightarrow [1, \ldots, m]$, indicating trajectory $\sigma(j)$ (i.e., the set $R_j(\sigma(j))$) is chosen for the j^{th} robot. We can remove a target from the set $R_j(x)$ if it does not satisfy a given minimum quality of tracking requirement.

5.4.1 Maximizing Number of Targets

First consider the case of maximizing the number of targets tracked by k robots. This problem is a generalization of the maximum coverage problem[1] stated as: *Choose k subsets to maximize the cardinality of the union of all subsets.* In our case, we cannot arbitrarily pick k subsets since they must belong to distinct groups (i.e., the same robot cannot be assigned to two trajectories).

[1]See Section 2.1 for a review of the maximum coverage problem.

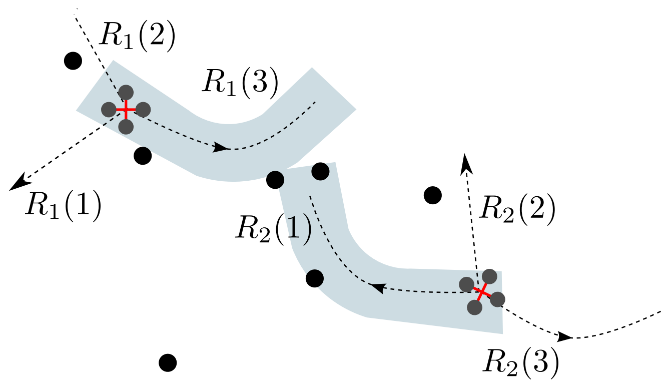


Figure 5.3: At the start of each round, we have a set of m candidate trajectories per robot. The trajectories may be non-uniform and of varying speeds. Using the predicted motion of the targets, we can determine which targets will be covered for a given trajectory and the corresponding quality of tracking.

The maximum coverage problem, under group constraints, can be stated as: *Choose k subsets of $\mathcal{R}$ given by a map, $\sigma : [1, \ldots, k] \rightarrow [1, \ldots, m]$ such that the union of all subsets is maximized.* The constraint that the same robot cannot be assigned to two trajectories is enforced by requiring the output be a map σ. This problem is known as the Maximum Group Coverage (MGC) problem. Chekuri and Kumar [73] proved that the greedy algorithm yields a 1/2 approximation for MGC which is also the best possible approximation. Their algorithm can directly be applied to track half the number of targets as an optimal algorithm. Our contribution is to extend the analysis to the weighted case, which is used for maximizing the quality of tracking.

5.4.2 Maximizing Quality of Tracking

For the case of maximizing the overall quality of tracking, we formulate a weighted version of MGC. Let $q_i(R_j(x))$ be the quality of tracking target t_i with robot r_j following the x^{th} trajectory. $q_i(R_j(x))$ can represent the expected quality of tracking as described in Section 5.2. The weight of any set $R_j(x) \in \mathcal{R}$ is given by the sum of qualities of all targets tracked by $R_j(x)$. The objective is to maximize the sum of quality of tracking for all targets.[2]

The greedy algorithm for the unweighted MGC can be modified for the weighted setting (Algorithm 2). In each iteration, we choose a set $R_j(x)$ greedily that maximizes the total weight. We add $R_j(x)$ to the solution, and discard all other sets belonging to the same group, i.e., all other candidate trajectories for the same robot r_j. This proceeds until we have chosen a trajectory for all robots.

Algorithm 2: Greedy Weighted MGC Algorithm

```
1 $C \leftarrow \emptyset$, $I \leftarrow \emptyset$
2 for $p = 1$ to $k$ do
3     Find $R_i(x)$ such that $Q(R_i(x) \cup C)$ is greatest, and $i \notin I$
4     $\sigma(i) \leftarrow x$
5     $C \leftarrow C \cup R_i(x)$
6     $I \leftarrow I \cup \{i\}$
7 end
8 Return $\sigma$
```

Theorem 7. *Algorithm 2 gives a $(1/2-\epsilon)$ approximation for the weighted MGC problem for any $\epsilon > 0$ in polynomial time.*

The analysis by Chekuri and Kumar [73] for the unweighted case can be modified for this weighted case. We present our full proof in Appendix C.1, for completeness.

We now evaluate the greedy algorithm through simulations and preliminary experiments.

5.5 Simulations

In this section, we describe our implementation of the algorithm, and evaluate its performance through simulations. We carried out the simulations using the SwarmSimX simulation environment [85]. SwarmSimX is a real-time multi-robot simulator designed

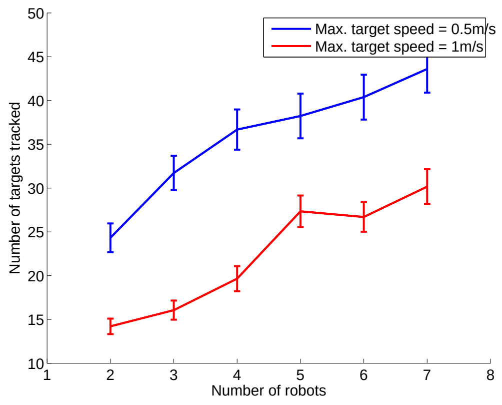
[2]The bottleneck version of maximizing the minimum quality of tracking over all targets cannot be applied since not all targets are tracked.

for modeling rigid-body dynamics in 3D environments. Models of the MikroKopter Quadrotor [86] were used to simulate the motion of the robots.

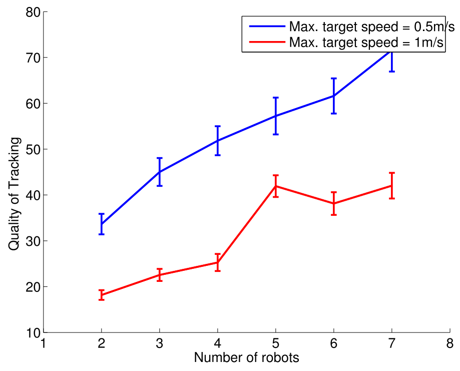
For simulating the targets, we generated random trajectories as follows. Each target randomly chooses a speed and direction and moves along this direction for a random interval of time, drawn from a normal distribution. This class of trajectories is motivated by wildlife monitoring applications where foraging animals have been found to follow such mobility models [87]. The mean and standard deviation of the normal distribution were set to 10 s and 1 s, respectively in the simulations.

The target trajectories were restricted to 20×20 m square on the ground plane. The initial locations of all targets were chosen uniformly at random near the robot locations. A moving average filter of window length 5 running at 10 Hz was used to estimate the position and velocity of the observed targets for the next planning round. A measurement for a target was obtained only if it was contained within the field-of-view of some robot.

For each robot, we created the following set of candidate trajectories: (a) stay in place, and (b) radially symmetric along 8 horizontal directions with a speed of 0.5 m/s. Thus, each robot could choose from a set of 9 trajectories in a round. Each round was set to a duration of 2 s. A trial consisted of 50 rounds.



(a)



(b)

Figure 5.4: (a) Number of targets covered out of 50 targets in the environment. (b) The average quality of tracking. The weight $q_i(R_j(x)$ is computed as the inverse of the minimum distance between the target and the robot along $R_j(x)$.

Figures 5.4(a) and 5.4(b) show the effect of the number of robots and the maximum speed of the targets. As expected, the number of tracks and quality of tracking increases as the number of robots increase. Increase in the maximum speeds of the targets has the effect of spreading them further apart, which further reduces the number of targets that can be tracked. For these trials, the height of the robots was fixed to 3.5 m (i.e., the size of the camera footprint was fixed). Figure 5.5 shows the total number of targets tracked in one representative trial as a function of the time. Once the robots have lost track of a particular target, they do no receive any position information about that target. Thus, they cannot predict the future locations for a lost target, unless it appears again in the field-of-view of some robot.

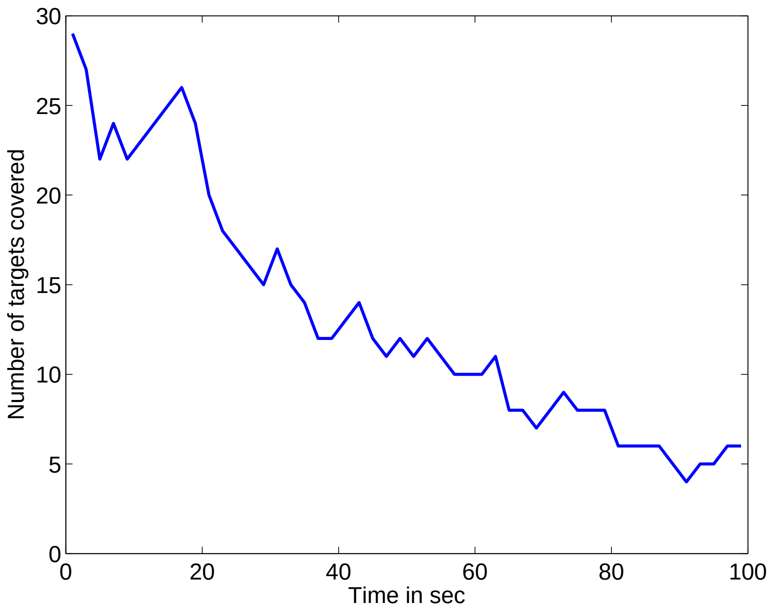


Figure 5.5: The number of targets tracked in one trial. As the targets spread the total number of targets that can be tracked decreases. Once a target moves out of the field-of-view, the robots cannot predict their future locations.

For the simulations, we did not incorporate the uncertainty due to sensing. In the next section, we validate the uncertainty model and present results from a preliminary experiment using 4 aerial robots.

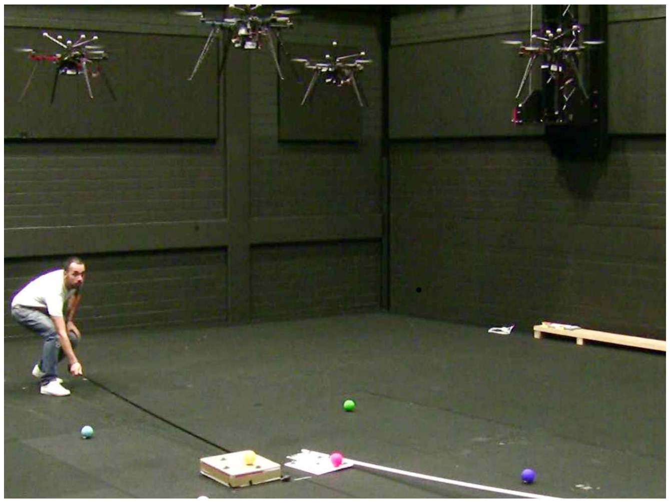


Figure 5.6: Experimental setup. Each robot is fitted with a downward facing wireless camera. All robots directly communicate with a central computer.

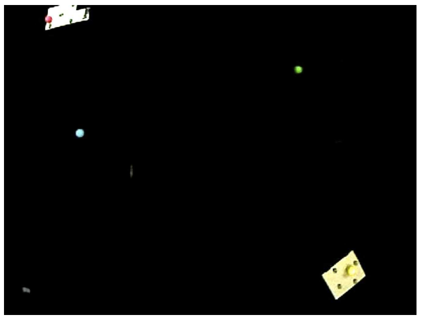
5.6 Experiments

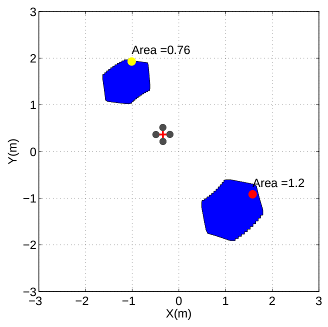
In order to validate our sensing model and the algorithm, we performed trials on an indoor setup (Figure 5.6). The setup consisted of four quadrotors controlled using the TeleKyb framework [88]. All robots communicated directly with a central computer via a wireless XBee link. Each robot was fitted with a downward facing camera. The cameras streamed the live images wirelessly directly to the central computer. An indoor motion capture system was used for position feedback and the orientation was stabilized on-board.

5.6.1 Validating the Sensing Model

We first conducted trials to validate the sensing model presented in Section 5.2. A robot was programmed to fly along a given trajectory at heights of 1 m and 1.5 m. The motion of the robot was smoothed, so as to ensure that the roll and pitch angles remained close to zero. Colored balls were placed on the ground (Figure 5.6). The pink and the yellow colored balls were fixed to motion capture markers to record their ground truth locations. All cameras were calibrated to obtain they camera parameters.

Figure 5.7 shows an image obtained using the on-board camera, along with the


(a)



(b)

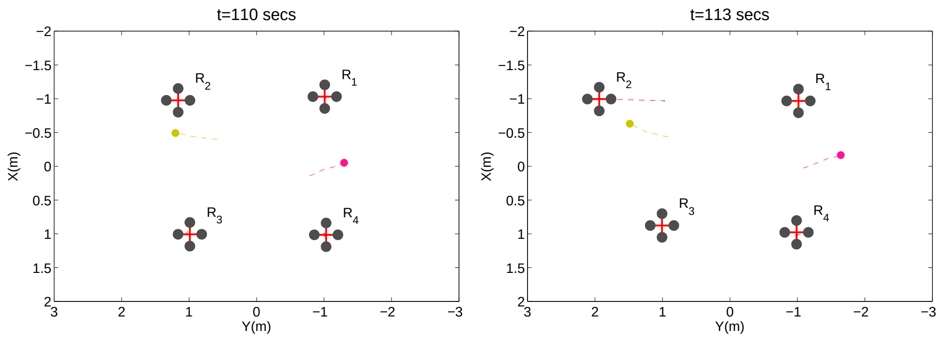
Figure 5.7: Validating the sensing model. (a) On-board camera image. (b) The true target location (colored circles) in the global frame, and the estimated locations using the method described in Section 5.2.

estimated and true locations of the balls. The backprojection area was computed considering ± 5 maximum measurement error in pixels, $\pm 5\,\mathrm{cm}$ maximum error in robot position, $\pm \pi/18$ radians maximum error in the yaw angle, and $\pm \pi/48$ radians maximum error in the roll and pitch angles. The average area of backprojection (for 50 images which contained either the yellow or pink balls) was $0.46\,\mathrm{m}^2$. The average error between the centroid of the projected area and the true location was $0.28\,\mathrm{m}$, with a standard deviation of $0.3\,\mathrm{m}$.

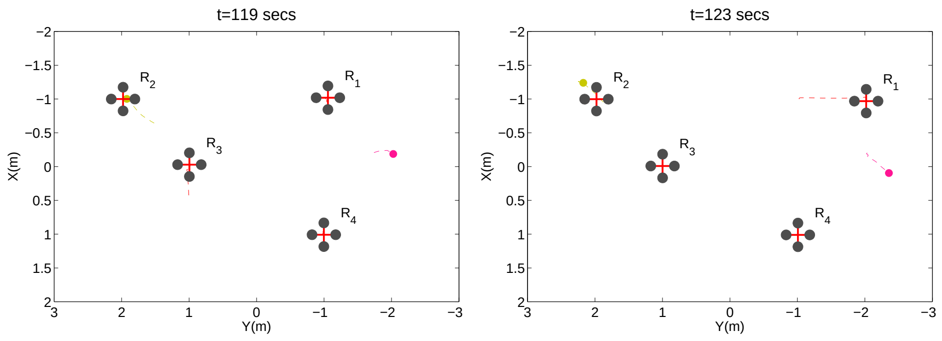
5.6.2 Tracking Experiment

We implemented the greedy algorithm on four robots. The controller on-board the robot was set to operate the robots smoothly in near-hovering mode at an average speed of $0.5\,\mathrm{m/s}$. Each round lasted for 3 seconds. The pink and yellow balls were moved manually (Figure 5.6). For this trial, the locations of the targets were obtained from the motion capture system. The robots used a moving average filter to predict the locations of the targets, based on previous measurements. A radius of $\sqrt{2}\,\mathrm{m}$ was found empirically to correspond to the camera footprint when the robots operated at a height of $2.5\,\mathrm{m}$. The robots had one of the four grid neighbors in the $z = 2.5\,\mathrm{m}$ plane as candidate trajectories.

Figure 5.8 shows the locations of the robots and the targets before and after two key



(a) At $t = 110\,\mathrm{s}$, R_3 chose the trajectory moving to the left to keep tracking the yellow target.



(b) At $t = 119\,\mathrm{s}$, R_1 chose the trajectory moving to the right to keep tracking the pink target.

Figure 5.8: Start (left figures) and end (right figures) of two rounds. Dashed trail shows the locations of the robots and targets in the preceding 5 secs.

rounds: at times 110 s and 119 s. The two rounds show events when the robots predicted that the target would move out of the coverage area in the next round. Hence, as an outcome of the greedy algorithm, the robots chose corresponding trajectories in order to continue to track the targets.

The sensing validation and tracking trials presented here demonstrate a proof-of-concept implementation of the components of our system. Our future efforts are directed towards performing large scale experiments with this system.

5.7 Conclusion

In this chapter, we studied a visual tracking problem in which a team of robots equipped with cameras are charged with tracking the locations of targets moving on the ground. We discussed the sources of uncertainty that affect the quality of estimating the locations of ground targets using overhead images. We showed the infeasibility of tracking all targets while maintaining the optimal quality of tracking, or any factor of the optimal quality, at all times. We then formulated the target tracking problem where the goal is to assign trajectories for each robot in order to maximize the quality of tracking. When we are given a set of candidate robot trajectories, we showed how the problem can be posed as a combinatorial optimization problem. A simple and easy-to-implement greedy algorithm applied to this problem yields a 1/2 approximation. Finally, we presented results from simulations and preliminary experiments validating the sensing model and demonstrating the feasibility of implementing the algorithm.

Chapter 6

Sampling Algorithms with Aerial and Ground Robots (Precision Agriculture)

In Chapters 3 and 4, we saw how to cover an environment using stationary sensors. In this chapter, we will study coverage problems for mobile robots. Unlike stationary sensors, robots have to travel to various locations in order to cover the environment. Hence, not all points in the environment will be sensed simultaneously. Consequently, we must optimize the motion of the robots in order to minimize the coverage time. Furthermore, robots have limited on-board energy. Hence, they may not be able to cover the environment completely. Again, we must optimize their motion so as to cover as many points as feasible. In this chapter, we study how to find coverage tours for ground and aerial robots that address the aforementioned challenges.

We start by introducing a new problem of planning a minimum time coverage tour when we want to optimize not just the travel time for the robots, but also the time for measurements. The input to this problem is given by a set of disks, not all of the same radius, lying in the plane. The coverage tour must obtain a measurement in each disk. The total time is given by the sum of the traveling time and the measurement time (number of measurements times some fixed time per measurement). Our objective is to minimize the total time by choosing a sampling location in each disk, and a tour that visits the chosen sampling locations. We term this as the *Sampling Traveling Salesperson Problem with Neighborhoods* (SamplingTSPN).

SamplingTSPN generalizes the classical Euclidean Traveling Salesperson Problem (TSP) [89]. In Euclidean TSP, the objective is to find a minimum length tour that visits a given set of points lying in the plane. Arkin and Hassin [52] introduced a variant of TSP, termed TSP with Neighborhoods[1] (TSPN), where instead of visiting each point exactly, we are given a set of geometric neighborhoods (e.g. disks), and we want to find a minimum length tour that visits at least one point in each neighborhood. In SamplingTSPN, the total time, which is a combination of both the tour length, and the number of samples, is to be minimized. We can reduce the total time by combining samples of overlapping disks. As we show in Section 6.2, a TSPN tour may perform arbitrarily worse when directly applied to SamplingTSPN.

In the first part of this chapter, we present an $\mathcal{O}\left(\frac{r_{\max}}{r_{\min}}\right)$ approximation algorithm for SamplingTSPN, where $r_{\min}$ and $r_{\max}$ are the smallest and largest radii of the input disks. SamplingTSPN models the problem of obtaining ground measurements in our motivating application of precision agriculture. Precision agriculture is a data-driven technique to estimate and predict the health of crops in a farm, and use this to design targeted fertilizer treatment plans [18]. A key component of precision agriculture is data collection. We propose a prototype robotic system consisting of an Unmanned Ground Vehicle (UGV) and an Unmanned Aerial Vehicle (UAV) for autonomous soil and aerial data collection, respectively. Obtaining a soil measurement with a UGV takes some time. We show how to formulate the problem of obtaining ground measurements as a SamplingTSPN instance, and show how to apply our SamplingTSPN algorithm.

In the second part, we study the corresponding coverage problem for the UAV. Unlike soil measurements, aerial measurements, i.e., multi-spectral aerial images in our application, can be obtained instantaneously. However, small UAVs have a limited battery life. Visiting all input points in a large plot may not be feasible. Hence, we study the problem of maximizing the number of points visited subject to the maximum battery life. The general problem of visiting the most number of points subject to a budget is called orienteering. Instead of using the UAV alone, we consider the scenario where the UAV can land on the UGV, and use the UGV to travel to the next take-off

[1]See Section 2.3 for a review of TSP and TSPN problems.

locations. Consequently, the number of points visited will be higher subject to the small energy cost of take-off and landing. We show how to model this capability in the form of a metric graph, which allows applying constant factor approximation algorithms for this problem.

Finally, we present results from simulations using real data collected from an agriculture plot. We also present results from preliminary field experiments for the UAV conducted in an agriculture plot.

Our contributions can be summarized as follows as follows: We begin by presenting the related work in Section 6.1. We then formulate the SamplingTSPN problem, and present the $\mathcal{O}\left(\frac{r_{\max}}{r_{\min}}\right)$ approximation algorithm. In Section 6.3, we introduce our motivating application of precision agriculture. In Section 6.4, we show how to plan for the symbiotic UAV+UGV paths for obtaining ground and aerial measurements. Simulation results based on field data are presented in Section 6.5, and preliminary field experiments are presented in Section 6.6. We finally conclude with a discussion of future work in Section 6.7.

6.1 Related Work

The problem of designing sensor trajectories and the related problem of selecting sensor locations has recently received much attention. Low et al. [90] presented a control law to minimize the probability of misclassification in a Gaussian Process map. The authors enforce measurements to be taken continuously, and sensors to only move along a 4-connected grid. Zhang and Sukhatme [91] presented an adaptive search algorithm for finding the optimal sensor path to estimate a scalar field. Song et al. [92] presented an algorithm to localize multiple radio sources using a mobile robot. They presented upper bounds on the time required to localize the sources up to a desired probability. In all these works, the sensing model is assumed to be continuous (i.e. no time cost), unlike our work where we penalize discrete measurements explicitly.

Instead of labeling certainty, Krause et al. proposed Mutual Information as a measure of uncertainty [93]. An algorithm to place sensing locations was given which can closely approximate the optimal increase in Mutual Information. The work was extended to mobile sensor routing in [94] and multiple robots in [95]. Since we are designing algorithms for a heterogeneous sensor network, and use different objective functions, these results are not directly applicable.

The SamplingTSPN problem generalizes the TSPN problem. Dumitrescu and Mitchell [2] presented an 11.15-approximation algorithm for TSPN when the neighborhoods are possibly-overlapping unit disks centered at each site. The main difference in SamplingTSPN and TSPN is that our cost is not just the traveling time of the tour, but also the total time taken for obtaining soil measurements. Finding a minimum length/time path does not necessarily ensure that the robot takes fewer soil measurements, and the cost for the UGV tour is not necessarily minimized.

Bhadauria et al. [96] studied the problem of computing a minimum time data collection tour for k robots tasked with wirelessly collecting data from deployed sensors by visiting a point in the sensor's communication range. In their model, robots spends time for both traveling and downloading data from robots. Tekdas et al. [97] extended this model to the case where the communication range consists of two disks centered at the sensor and the inner ring requires less download time than the outer. In these problems, the robot has to separately query each sensor whereas in our model, the robot can combine soil measurements for multiple points by sampling the intersection of their neighborhoods.

In [98], Alt et al. studied the problem of covering a given set of points with k radio antennas with circular ranges, where the algorithm has to choose the center and radius r_i for each circle. They consider a cost function which is a weighted sum of the length of the tour and the sum of r_i^α for each disk (α models the transmission power for the antennas). The main difference between this problem formulation and ours is that we do not require the number of samples i.e., k, to be fixed. Instead our formulation penalizes higher k in the cost function.

The problem of maximizing the number of points visited by the UAV subject to a battery lifetime constraint is modeled as an orienteering problem. Blum et al. [19] presented a 4-approximation to the orienteering problem for complete graphs with metric edges. We show in Section 6.4 how to model the problem of selecting most input points can be solved as an orienteering problem by constructing a complete graph with metric edges.

Recently, there has been a significant interest in developing cooperative aerial and ground/surface/underwater robot systems. Grocholsky et al. [99] described a system with coordinating aerial and ground vehicles for the application of detecting and locating targets. Sujit and Saripalli [100] studied the problem of exploring an area to detect targets using an UAV and inspecting the targets with Autonomous Underwater Vehicles (AUV). The authors compared in simulations three strategies to address the trade-off between quickly exploring the environment for all targets, and minimizing the latency between detection with UAVs and inspection with AUVs. Tanner [101] presented control laws for the UGVs to form a grid of sensors and UAVs to fly in a formation over the grid, such that a target moving on the ground can be detected if it moves from one grid cell to the other.

The main difference between existing literature and our work is that we explicitly consider that the UAV can be carried between takeoff locations by the UGV in the sensor planning phase. The resulting plan found by our algorithm may consist of multiple deployments for the UAV, which increases its coverage with limited battery.

Next, we formulate the SamplingTSPN problem and present our main algorithm.

6.2 Sampling TSPN Problem

The SamplingTSPN problem is defined as follows: *Given a set of disks with centers at points X and maximum and minimum radii $r_{\max}$ and $r_{\min}$ respectively, find a tour τ of N distinct sample locations to minimize the cost $len(\tau) + C_g \cdot N$ such that each disk contains a sample location, where C_g is the cost of obtaining each measurement.*

SamplingTSPN generalizes TSPN with disk neighborhoods. The objective in TSPN is to minimize only the length of the tour. A natural strategy for finding a SamplingTSPN tour would be to first find a TSPN tour, and then choose sampling locations on this tour. However, this approach may lead to bad solutions (see Figure 6.1). Instead, we present an algorithm which finds a tour whose length is at most $\mathcal{O}\left(\frac{r_{\max}}{r_{\min}}\right)$ of the optimal length, and which obtains at most a constant times the optimal number of measurements, yielding a $\mathcal{O}\left(\frac{r_{\max}}{r_{\min}}\right)$ approximation.

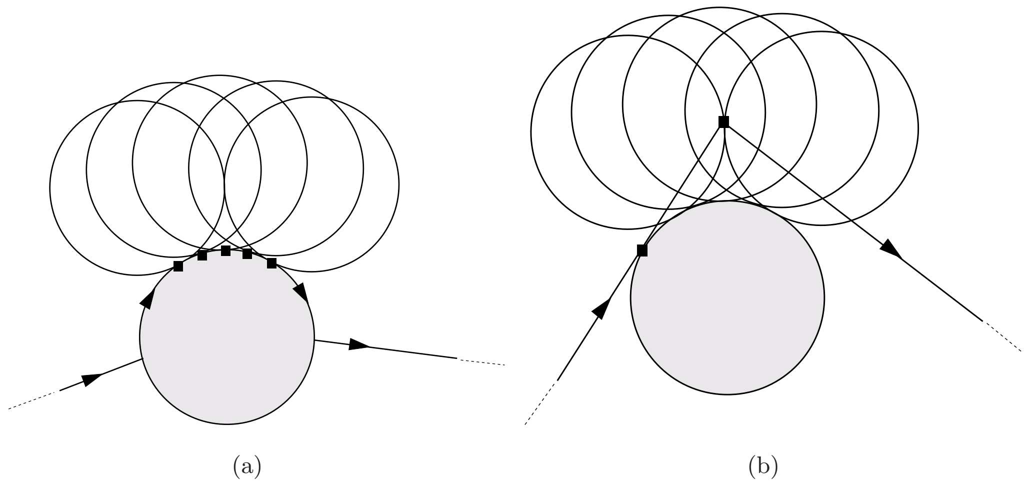


Figure 6.1: Solving SamplingTSPN by first finding a TSPN tour, and then choosing sampling locations on this tour can lead to bad results. (a) The TSPN tour presented in [2] visits all the disks by touring the circumference of each disk in the Maximal Independent Set (one shown shaded). This tour will be forced to take a separate measurement for each outer disk and thus have $O(n)$ measurement locations. (b) In general, we are not forced to move along the circumference and can visit a smaller number of locations where the disks overlap.

6.2.1 Overview of the GridSample Algorithm

Our algorithm works in three main stages. In the first stage, we find a tour that gives us the order in which to visit a carefully chosen subset of the disks. In the second stage, we find a set of candidate sampling locations for each disk. In the third stage, we perform local detours to the tour to visit sampling locations for all disks. The details of each stage are presented next. We will refer to our algorithm as GridSample.

Stage 1. The first stage of GridSample is similar to the standard algorithm for finding a TSPN tour of unit disks [2]. We replace each disk in X with a larger disk of radius $r_{\max}$. Let X' be this new set of disks. We find a Maximal Independent Set (MIS), i.e. a set of non-overlapping disks, in X'. Then, we find a TSP tour of the centers of the disks in the MIS. This tour, denoted by T_C, gives us the order in which to visit the disks in M.

This completes Stage 1. Note that T_C visits only a subset of all disks in X. We will now find a candidate set of sampling locations for all the disks, and add local detours to T_C to visit the set of sampling locations.

Before we present our approach, consider the following straight-forward approach for finding the set of candidate sampling locations. Draw a square grid with side $r_{\min}$ about each disk in the MIS. Extend the grid so that any disk that intersects with a disk in the MIS lies within the grid. Now add a local detour to T_C as follows: every time a new disk in the MIS is visited, visit and obtain a sample at all the grid locations near this disk.

This guarantees that we obtain a sample for each of the input disk in X. However, the number of grid locations will be $\mathcal{O}\left(\frac{r_{\max}}{r_{\min}}\right)$ *per disk in the MIS.* On the other hand, there could possibly be a single location within each MIS disk, where an optimal algorithm can obtain a sample. Thus, this approach would yield an $\mathcal{O}\left(\frac{r_{\max}^2}{r_{\min}^2}\right)$ approximation with respect to the number of measurements. Instead, the following procedure will obtain a constant factor approximation with respect to the number of measurements, and guarantee an overall $\mathcal{O}\left(\frac{r_{\max}}{r_{\min}}\right)$ approximation for SamplingTSPN.

Stage 2. We first find the smallest sized set of points, such that there exists at least one point in the interior of every disk in X. Consider the arrangement of disks in X in the plane. We create a set of points P by placing a point in each face of the arrangement. For each point in $p \in P$, let R be the set of disks containing p. Let $\mathcal{R}$ be the collection of such sets for all points in P. We then solve a *geometric hitting set* problem for the set system $(P, \mathcal{R})$. The hitting set solution finds the minimum number of points in P such that each disk in X has at least one such point in its interior. Finding the minimum number of points is NP-complete in general. However, there exist efficient approximation algorithms, e.g., $(1+\epsilon)$–approximation in [102], that we can use. Let C be the result from the hitting set algorithm.

Stage 3. We will add local detours to the tour T_C computed in Stage 1 to visit the candidate sampling locations C. Instead of visiting a point in C, however, we will impose a grid and visit grid points neighboring C. This allows us to bound the length of the tour, while simultaneously bounding the number of measurements.

Let x be some center along the tour T_C. Let $N(x) \subset C$ be the set of all candidate locations corresponding to disks in X whose centers are at most $2r_{\max}$ from x. These are disks that are completely contained inside a disk of radius $3r_{\max}$ centered at x.

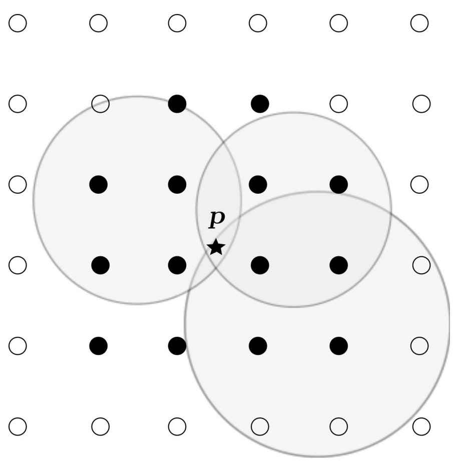


Figure 6.2: The point p marked by a star is the output from solving a hitting set problem. We compute grid locations $G(p)$ (filled circles) at a distance of at most $2r_{\min}$ from p. Lemma 18 guarantees that any disk containing p having radius greater than $r_{\min}$, also contains at least one point from $G(p)$.

Let $p \in N(x)$ be any such candidate location (Figure 6.2). We impose a grid of resolution $r_{\min}$ over the entire plane. Let $G(p)$ be the set of all grid points within distance $2r_{\min}$ from p. There are at most 25 such grid points. In Lemma 18 we will show that any disk having radius greater than $r_{\min}$ that contains p must contain at least one grid point in $G(p)$. Hence, we can restrict our samples to only grid points without missing any disks.

The final tour is obtained by modifying the TSP tour of the centers as follows: After having visited x, instead of continuing on to the next center, the tour visits the set of all grid points within a distance of $3r_{\max}$ of x. Along this tour, we add a sampling location every time a grid point belonging to $G(p)$ for some $p \in N(x)$ is encountered. Let S be the set of all sampling locations.

In general, S contains more sampling locations than necessary. As post-processing, we greedily choose a smaller subset of S such that each disk contains one sampling location. Having found this subset, we can compute a TSP tour of just these locations as the final tour. The construction described above allows us to conveniently bound the performance of the algorithm.

6.2.2 Performance Analysis of the GridSample Algorithm

In Lemma 18 we show the correctness of the algorithm by proving that S has at least one sampling location in each input disk (not just the larger disk). In Lemma 19 we upper bound the number of candidate sampling locations, and in Lemma 20 we bound the total distance traveled. Finally, these results are combined to prove the approximation ratio of our algorithm in Theorem 8.

Lemma 18. *Let S be the set of all candidate sampling locations in* GridSample. *Then, for each disk in X, there exists a point in S lying in its interior.*

Proof. The set S of sampling locations is computed based on the solution C to the hitting set problem. For any disk in X centered at x, there exists a point $p \in C$ lying in its interior. Since we choose sampling locations from the grid, our algorithm may not be able to choose p. However, we show that by including at most 25 points for each point in the hitting set, we can hit all disks.

$G(p)$ is the set of grid points within $2r_{\min}$ of p. Instead of sampling at p, we sample at some grid point in $G(p)$ (Figure 6.2). Let D be any disk in X that contains p. We will show that at least one grid point, say $p' \in G(p)$, is also contained in D. Draw a disk D_2 centered at p, with radius $2r_{\min}$. Any disk of radius $r_{\min}$ contained completely within D_2 must also contain at least one point of $G(p)$. Replace D by a smaller disk, say D_1, such that D_1 has a radius $r_{\min}$, D_1 is contained completely within D, and D_1 contains p. D_1 is completely contained within D_2. Hence, D_1 contains some point of $G(p)$.

Next, we will show that this point p' is also included in S. We have one of two cases, either the larger disk centered at x lies in the MIS or not. If it lies in the MIS, then p' is within $3r_{\max}$ of x and we are done. If not, then the larger disk of radius $r_{\max}$ intersects some other larger disk, centered at say x', lying in the MIS. Hence, the distance between x and x' is at most $2r_{\max}$, which implies that p' is at most $3r_{\max}$ away from x'. Hence, in both cases, p' will be in S. $\square$

Lemma 19. *If N^* is the number of samples by an optimal algorithm for the general* SamplingTSPN *problem and S is the set of grid locations computed in* GridSample, *then $|S| \leq 25(1+\epsilon)N^*$.*

Proof. N^* is the minimum number of points, such that there exists at least point per disk in X. The set C can be found using any constant-factor approximation for this hitting set problem. For example, using the algorithm in [102], we have $|C| \leq (1+\epsilon)N^*$. For each point in C, we add at most 25 points in S. Hence, $|S| \leq 25|C| \leq 25(1+\epsilon)N^*$. □

Lemma 20. *Let T_{ALG} be the tour constructed by* GridSample*, and T^* be the tour for the optimal* SamplingTSPN *algorithm. Then len(T_{ALG}) $\leq \mathcal{O}\left(\frac{r_{\max}}{r_{\min}}\right) T^*$.*

Proof. For ease of notation, in this proof we refer both a tour and its length by T, and T^* refers to an optimal tour.

Denote by T_I and T_C the TSPN tour of the MIS and TSP tour of the center of the MIS respectively. Let n be the total number of disks in the MIS. Now

$$T_C^* \leq T_I^* + 2nr_{\max} \tag{6.1}$$

$$\leq T^* + 2nr_{\max}. \tag{6.2}$$

The first inequality follows from the fact that a tour of the centers can be constructed by taking a detour of at most $2r_{\max}$ for each disk from the tour of the disks. The second inequality comes from the fact that the optimal tour is also a tour of the disks in the MIS.

T_{ALG} consists of a TSP tour of the centers of the disks in MIS and a tour of the grid locations within $3r_{\max}$ of the center. Using the $(1+\epsilon)$-approximation for the TSP tour [51] we get,

$$T_{ALG} \leq (1+\epsilon)T_C^* + 36n\frac{r_{\max}}{r_{\min}}r_{\max} + 6n(1+\sqrt{2})r_{\max} \tag{6.3}$$

Here, $6\frac{r_{\max}}{r_{\min}}$ are the number of horizontal rows in the grid traversed, each horizontal row has a length of $6r_{\max}$. The last term accounts for moving from the center of the disk to the start and end of the grid, and moving vertically along one column.

Using Theorem 1 from [97], we know that the length of any tour that visits n non-overlapping disks of radius $r_{\max}$ is at least $\frac{n}{2}0.4786r_{\max}$. That is, $T^* \geq \frac{n}{2}0.4786r_{\max}$. This gives, $nr_{\max} \leq \frac{2}{0.47}T^*$.

Therefore,

$$\begin{aligned} T_{ALG} &\leq (1+\epsilon)\left(T^* + 2nr_{\max}\right) && (6.4)\\ &+ \left(36\frac{r_{\max}}{r_{\min}} + 6 + 6\sqrt{2}\right) nr_{\max} && (6.5)\\ &\leq (1+\epsilon)T^* + \left(36\frac{r_{\max}}{r_{\min}} + \mathcal{O}(1)\right) nr_{\max} && (6.6)\\ &\leq \mathcal{O}\left(\frac{r_{\max}}{r_{\min}}\right) T^* && (6.7) \end{aligned}$$

□

Theorem 8. GridSample *gives a valid* SamplingTSPN *tour with cost* $\mathcal{O}\left(\frac{r_{\max}}{r_{\min}}\right)$ *times that of the optimal tour, where* $r_{\max}$ *and* $r_{\min}$ *are the radii of the largest and the smallest of the input disks, respectively.*

Proof. Let C^* be the cost of the optimal algorithm for the general SamplingTSPN problem. Therefore, $C^* \geq T^* + N^* \cdot C_g$, where T^* is the optimal TSPN tour visiting all disks, and N^* is the minimum number of sample locations such that each disk has at least one sample location.

Consider the cost of our algorithm,

$$\begin{aligned} C_{ALG} &= T_{ALG} + |S| \cdot C_g, && (6.8)\\ &\leq \mathcal{O}\left(\frac{r_{\max}}{r_{\min}}\right) T^* + \mathcal{O}(1) N^* \cdot C_g, && (6.9)\\ &\leq \mathcal{O}\left(\frac{r_{\max}}{r_{\min}}\right) C^*. && (6.10) \end{aligned}$$

where the first inequality comes from Lemmas 19 and 20. □

In the next sections, we will formulate the informative path planning problem for our motivating application of precision agriculture as a SamplingTSPN instance. We begin by describing the application.

6.3 Motivating Application: Precision Agriculture

Precision agriculture is a data-driven technique to determine the status of crops and the corresponding fertilizer treatment plans for a specific agriculture plot. We use nitrogen deficiency as a proxy for the status of the crops. We want to label each point in the plot accurately based on the level of nitrogen present at that point. If any point has a high probability of being mislabeled, we can obtain ground and aerial measurements near this point and reduce the risk of being mislabeled. In this section, we show how to identify points whose probability of being mislabeled, based on a prior nitrogen map, is above a threshold. We term these as Potentially Mislabeled (PML) points. The set of PML points thus identified will form as the input to the sensor planning algorithms.

Our approach can be summarized as follows:

1. We first identify the set of PML points, $\mathcal{X}_{pml}$, from a given prior nitrogen map (Section 6.3.1). We then show how to compute a disk centered at each such point, such that an expected measurement within this disk is sufficient to reduce the mislabeling probability below a user-defined threshold.

2. Next, we find (an approximation to) the largest subset of the PML points, $\mathcal{X}_s \subseteq \mathcal{X}_{pml}$, that can be visited by the UAV using the symbiotic UAV+UGV system, subject to its maximum battery lifetime constraint (Section 6.4.1). The UAV obtains aerial measurements for each PML point in $\mathcal{X}_s$.

3. Finally, we compute the UGV tour to obtain ground measurements for each PML point in $\mathcal{X}_s$ (Section 6.4.2). This tour is obtained by applying the SamplingTSPN algorithm from Section 6.2 to the set of disks, corresponding to $\mathcal{X}_s$, computed in step (1) above.

We begin by describing how to compute the PML points.

6.3.1 Finding Potentially Mislabeled Points

Our operating environment is a farm plot discretized into a set of points $\mathcal{X} = \{x_1, x_2, \cdots, x_n\}$. We want to estimate the level of Nitrogen (N) at each point in $\mathcal{X}$ by combining ground and aerial measurements. We use Gaussian Process regression to estimate the N levels using the two types of measurements [103].

Previously obtained measurements are used to build a prior N level map. For each point we associate a most likely estimate as $N(x_i)$, with variance of the estimate given by $\sigma^2(x_i)$. Our task is to find regions in the plot with similar N levels. For example, the task can be to classify each point in the plot into three labels: low N, medium N, and high N. In general, we are given a set of labels, and each label L_i is specified by a minimum and maximum N level, L_i^-, L_i^+ respectively.

Since we do not have access to the true N levels and instead have a distribution $N(x_i)$, we associate with each label a probability of being correct. We define $\mathrm{P}_{lj}(x_i)$ as the probability that the label j for point x_i is correct $\mathrm{P}_{lj}(x_i) = \mathrm{P}(L_j^- \leq N(x_i) < L_j^+)$. Labels can then be assigned to points based on which is most likely to be correct, given the estimates of N levels at each point. We use the shorter notation $\mathrm{P}_l(x_i)$ to denote the probability of the most likely label.

We define PML points as all points in $\mathcal{X}$ for which the probability of the most-likely label being incorrect is below a user-desired value $\mathrm{P}_d \in (0, 1)$.

$$\mathcal{X}_{pml} = \{x_i \in \mathcal{X} : p_{\text{mislabeled}}(x_i) \leq \mathrm{P}_d\}. \tag{6.11}$$

Our goal is increase the probability of the label being correct by taking soil and aerial measurements near the PML points.

The previous equation expresses an upper bound on the probability that $N(x_i)$ is below the minimum value of the current label, $L^-(x_i)$, or above the maximum, $L^+(x_i)$. Let $\Phi(a)$ denote the Gaussian cumulative distribution function. Then we have,

$$p_{\text{mislabeled}}(x_i) \leq \mathrm{P}_d \tag{6.12}$$

$$\therefore\ \Phi\left(\frac{L^-(x_i) - N(x_i)}{\sigma(x_i)}\right) + 1 - \Phi\left(\frac{L^+(x_i) - N(x_i)}{\sigma(x_i)}\right) \leq \mathrm{P}_d \tag{6.13}$$

Taking measurements near x_i will reduce $\sigma(x_i)$ due to the spatial correlation of the N values. For any value of $N(x_i)$, there exists a corresponding $\sigma(x_i)$ such that Equation 6.13 as follows.

First, we define the constant $\Delta(x_i)$ for each PML point,

$$\Delta(x_i) = \min\left(|L^+(x_i) - N(x_i)|, |L^-(x_i) - N(x_i)|\right) \tag{6.14}$$

Now Equation 6.13 can be expressed more conveniently as,

$$2 \cdot \Phi\left(\frac{-\Delta(x_i)}{\sigma(x_i)}\right) \leq p_{\text{mislabeled}}(x_i) \leq \mathrm{P}_d \tag{6.15}$$

Rearranging the previous equation yields the desired value for $\sigma(x_i)$ as,

$$\frac{-\Delta x}{\Phi^{-1}\left(\frac{\mathrm{P}_d}{2}\right)} \geq \sigma(x_i) \tag{6.16}$$

We will use the shorthand σ_d for the left hand side of Equation 6.16 since σ_d can be calculated from prior data, and can be treated as a constant. For each point, there will be a different σ_d depending on the exact value of $N(x_i)$, and the current most likely label $L(x_i)$.

Let the measurement location be denoted z, and the sensor noise of the measurement be σ_s. The correlation between the N levels at z and x_i is modeled by the Gaussian Process equations [103]. Thus the *new* variance at x_i, conditioned on the measurement at point z, satisfies,

$$\sigma^2(x_i|z) = \sigma^2(x_i) - K(x_i, z)[K(z, z) + \sigma_s^2]^{-1} K^T(x_i, z) \tag{6.17}$$

The function $K(\cdot, \cdot)$ is the *covariance* or *kernel* function of the Gaussian Process [103]. We fix $K(\cdot, \cdot)$ to be the *squared exponential* function, which is commonly used in precision agriculture [104].

Recall from Equation 6.16, $\sigma^2(x_i|z)$ should be no greater than σ_d^2. Given the measurement location z, Equation 6.17 simplifies as follows,

$$\sigma_d^2 - \sigma^2(x_i) \geq -\sigma_f^4(\sigma_f^2 + \sigma_s^2)^{-1} \exp(-\frac{1}{2l^2}||x_i - z||^2) \tag{6.18}$$

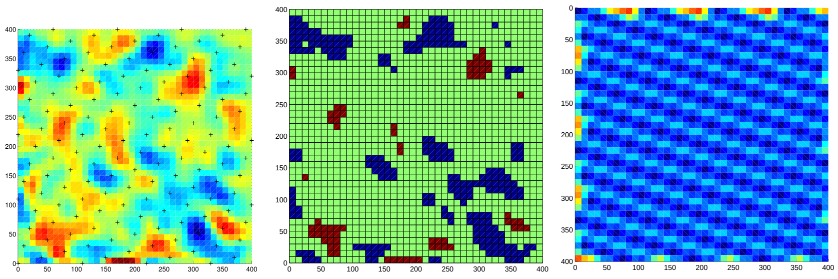
σ_f and l are the *hyperparameters* of the covariance function, which are previously learned from the data.

After further rearrangement and taking the natural log of both sides,

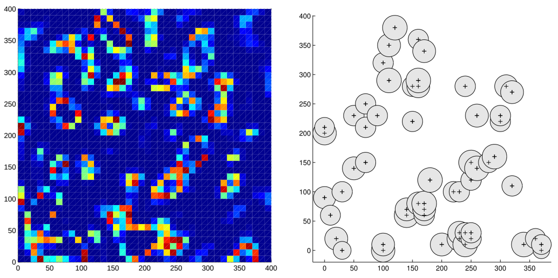
$$||x_i - z||^2 \leq -2l^2 \log[(\sigma^2(x_i) - \sigma_d^2)(\sigma_f^2 + \sigma_s^2)\sigma_f^{-4}]. \tag{6.19}$$

Denote the right hand side of Equation 6.19 by r_i. Thus, for every PML point $x_i \in \mathcal{X}_{pml}$ (i.e., points where N estimates do not satisfy Equation 6.16), we can find a disk of radius r_i centered at x_i. A sample obtained inside this will yield sufficiently small variance on $N(x_i)$ to determine the proper label with probability higher than P_d. An example of a field, the field labels, and the points with high mislabeling probability are shown in Figure 6.3.

In the next section, we show how to plan for the ground and aerial measurements where the input is the set of PML points and their corresponding disks.



(a) (b) (c)



(d) (e)

Figure 6.3: A generated random field using the GP parameters learned from the soil dataset. (a) The ground-truth samples obtained at location marked by a cross, along with the GP regression. (b) The data partitioned into three labels: low, med, high. (c) The variance of the sampling. The variance has a regular pattern, since the samples were obtained along a grid. (d) The mislabel probability. Note that it is high in many places, even though the variance is roughly uniform and low since the mislabel probability also depends on the value $N(x)$. (e) The points at which the labeling certainty is below P_d, and the corresponding ranges described in Equation 6.19.

6.4 Symbiotic UAV+UGV Path Planning

In this section, we describe the algorithms to find the UAV and UGV tours that visit the PML points. We first show how to compute the UAV tour, and then apply the SamplingTSPN algorithm to find the UGV tour.

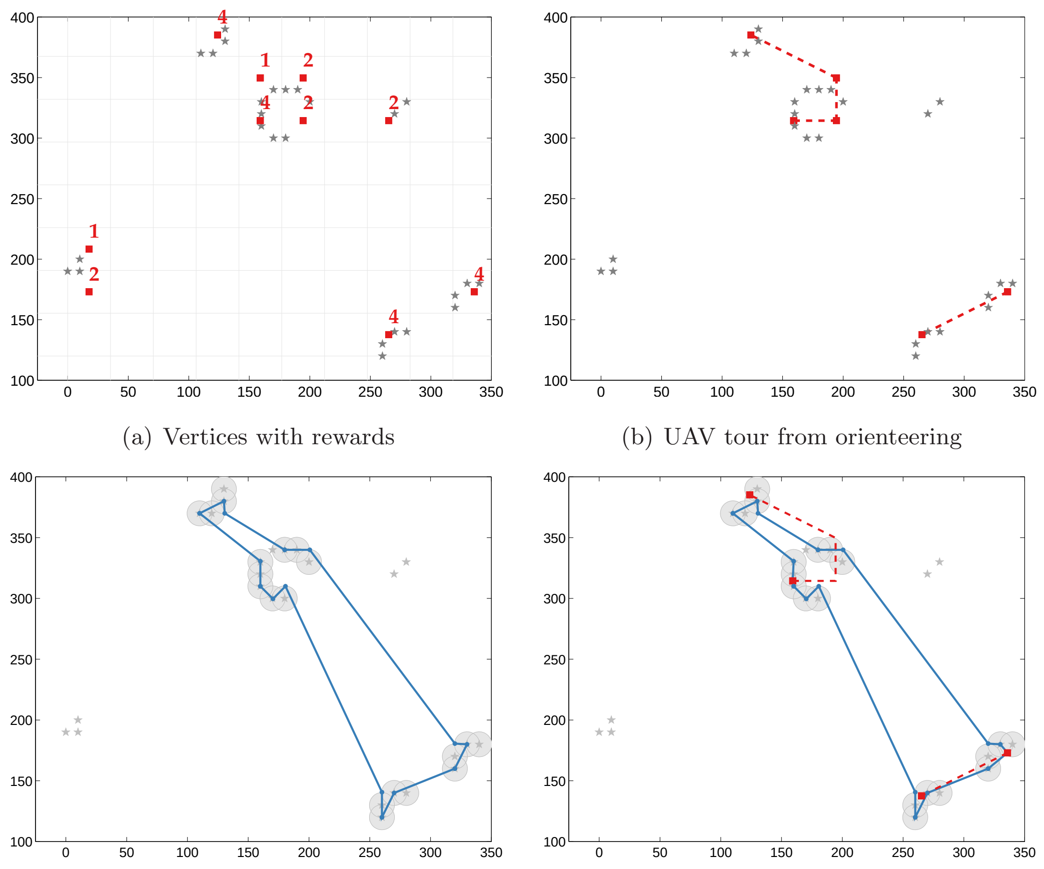
6.4.1 Planning for Aerial Measurements

The main limitation for the UAV is the limited on-board energy. The UAV may not be able to visit all input PML points. Consequently, we consider the problem of maximizing the number of PML points visited subject to the maximum battery lifetime. We reduce this to the orienteering problem. Let $G(V, E, \pi, w)$ be a graph with weights $w(u, v)$ on edges, and rewards $\pi(v)$ on the vertices. The objective in the orienteering problem is to find a tour of a subset of vertices collecting maximum reward, with the constraint that the sum of weights of edges on the tour is less than a given budget.

Instead of using the UAV alone, we consider the scenario where the UAV and UGV operate together, in order to increase the number of points visited. The UAV can land on the UGV, and the UGV can carry the UAV between deployment locations, thus saving energy. However, the UAV still spends some energy taking-off and landing on the UGV. We show how to model this trade-off for the symbiotic UAV+UGV system as an orienteering instance.

First consider the case of finding the maximum subset of points in a UAV-only system. For simplicity, let the camera footprint be a single point for now. Let the vertices of the graph be the set of PML points and let each vertex have unit reward. We add an edge to G between every pair of points with weight equal to the Euclidean distance between the points. The budget for the UAV equals the battery lifetime minus $2C_a$ to account for the single takeoff and landing. The solution for the orienteering problem for this instance will be a path traversing a set of PML points (with a single landing and take-off location).

Since the edge weights are Euclidean distances, this graph is a complete metric graph. Blum et al. [19] presented a 4-approximation for orienteering problems on undirected metric graphs. Applying this algorithm to the graph we constructed above will yield a UAV tour visiting at least $1/4^{\text{th}}$ of the PML points visited by the optimal algorithm.



(a) Vertices with rewards

(b) UAV tour from orienteering

(c) Final Sampling TSPN

(d) Final UGV+UAV Tours

Figure 6.4: Path Planning Algorithm. (a) Square grid of resolution $C/\sqrt{2}$. The reward for visiting each grid point (red square) is the number of PML points (gray star) falling within the grid. (b) UAV tour found using orienteering on the graph of grid points. For this instance, UAV budget was 500 secs out of which 200 secs are spent traveling and 240 secs are spent for the 2 ascents/descents. (c) Sampling TSPN tour (Section 6.2) for the UGV. (d) Final UGV tour including UAV take-off locations (red squares).

Now consider the case of a UAV+UGV system. The UGV can transport the UAV between two PML locations, without affecting the UAV's battery life. Furthermore, since the UAV carries a camera with a footprint of diameter C, it can sample a point without flying directly over it. Hence, we will also modify the set of vertices. The detailed construction of the input graph for the orienteering problem is given in Algorithm 3.

```
Algorithm 3: Creating Input Graph $G$.
1 Create a square grid of resolution $C/\sqrt{2}$ over the plane. Each point in $\mathcal{X}_{pml}$ is
  associated with its nearest grid location (Figure 6.4(a)). Store the number
  (denoted by $\pi(v)$) of PML points associated with a grid location.
2 Let $V$ be the set of grid vertices with at least one PML point associated. For
  each $v \in V$, let $\pi(v)$ be the number of associated PML points (Figure 6.4(a)).
3 Build a complete undirected graph $G = \{V, E, \pi, w\}$. For each edge between
  $(u, v) \in V$, add a weight $w(u, v) = \min\{d(u, v), 2C_a\}$. This implies there are two
  types of edges between grid points: The UAV can either use the UGV to travel
  paying only for the ascent/descent ($2C_A$) or travel directly between points paying
  the distance cost ($d(u, v)$).
```

The following lemma shows that the resulting graph G is a metric graph.

Lemma 21. *The graph G constructed in Algorithm 3 is a metric graph.*

Proof. We verify G is a metric graph. Consider a triple of vertices u, v, w. We know $w(u, v), w(v, w), w(w, u) \leq 2C_a$. It is easy to see the triangle inequality holds when two or three edges have weights equal to $2C_a$. Consider the case when only one edge has weight equal to $2C_a$, say $w(u, v) = 2C_a$. Now, $w(v, w) + w(w, u) = d(v, w) + d(w, u) \geq d(u, v)$. Since $w(u, v) = \min\{2C_a, d(u, v)\} = 2C_a$, we have $d(u, v) \geq 2C_a$. Hence, $w(v, w) + w(w, u) \geq w(u, v)$. And since $w(u, v) = 2C_a$ and $w(v, w), w(w, u) < 2C_a$, $w(u, v) + w(w, u) \geq w(v, w)$ and $w(u, v) + w(v, w) \geq w(w, u)$. For the case when all three edges have weights less than $2C_a$, the weights are equal to Euclidean distances. Hence, weights satisfy triangle inequality in addition to symmetry, identity and non-negativity. Hence, the graph constructed above is a complete metric graph. $\square$

Since G is a metric graph, we apply the algorithm in [19] to obtain a 4-approximation for this problem.

6.4.2 Planning for Ground Measurements

The main limitation for the UGV is the time required to obtain a soil measurement. The UGV tour must minimize the total time required to obtain all measurements. The input consists of a set of PML points, along with their corresponding disks. The SamplingTSPN algorithm presented in Section 6.2 can directly be applied to this problem. The resulting tour yields an $\mathcal{O}\left(\frac{r_{\max}}{r_{\min}}\right)$ approximation to the optimal tour (Figure 6.4(c)). This UGV tour does not include the UAV landing and take-off locations. We can add all the take-off locations to measurement locations determined from SamplingTSPN. A TSP tour of the combined set of points yields the final tour (Figure 6.4(d)).

Next, we study the performance of the two sensing algorithms through simulations based on field data.

6.5 Simulations

In the previous sections, we showed theoretical bounds on the number of PML points selected and the distance traveled by our algorithm with respect to optimal. We expect the UAV+UGV system to sample more PML points as compared to a UAV-only system with the same battery limitations. We investigate this through simulations based on actual system parameters and real data collected from an agricultural plot.

6.5.1 System Description

We present the details of the robotic system we are developing to motivate the choice of our simulation parameters. Our UGV is a Husky A200 by Clearpath Robotics [105]. The UGV has a typical battery life of two hours on a single charge. The operating lifetime can be extended to over six hours easily with additional batteries. The UGV will measure soil organic matter as a proxy for soil N supply to the crop using a Minolta SPAD-502 Chlorophyll meter [106].

Our UAV is a Hexa XL by MikroKopter [86]. This UAV can operate for a maximum of 25 mins (under ideal conditions). Deploying the UAV to approximately 100 meters

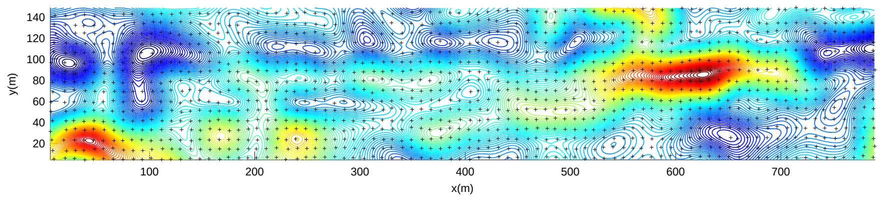


Figure 6.5: Soil organic matter data set from [3]. Dense sampling was collected by hand (black crosses) and used to train a Gaussian Process. The resulting estimate of nitrogen levels is shown as the contour map. From this data set we learn the sensor noise values σ_a and σ_g, as well as model the underlying soil organic matter for larger simulations (Figure best viewed in color).

height gives the camera a 50 meter diameter coverage with a single image. The UAV takes about 2 minutes to ascend/descend this height. The images include multi-spectral information, such as near-infrared reflectance, which is used to estimate the crop N status [3].

6.5.2 Modeling

To generate realistic data, we need a generative model of nitrogen levels and realistic values for the sampling noise for both systems. We will briefly discuss how we obtained these from an existing nitrogen remote sensing and soil sampling dataset [3]. The data from [3] consists of 1375 soil measurements taken manually in a 50m by 250m corn field, along with corresponding 1m spatial resolution remote sensing images in the green (G), red (R) and Near Infrared (NIR) portions of the spectrum. The samples were taken along a dense uniform coverage (see Figure 6.5) and provided the levels of soil Organic Matter (OM). R and NIR are known to be inversely related to crop N status [3].

We used OM as a proxy for the initial quantity of soil N supplied to the crop. We modeled the UGV as taking direct measurements of OM, corrupted by some sensor noise with variance σ_g^2, and the UAV as measuring the Normalized Difference Vegetation Index (NDVI), which is a combination of NIR and R levels [18]. We assume the NDVI levels are corrupted by sensor noise with variance σ_a^2. The noise variances were estimated to be $\sigma_a = 0.31$ and $\sigma_g = .05$ for our dataset, using the following procedure.

To model the spatial patterns of the OM levels, we used GP regression over the set of sample points and OM measurement values. This densely-sampled GP defined the hyperparameters which were used to generate new ground-truth N maps in our simulations. We used the GPML Toolbox [107] for performing the GP regression.

As part of the ground-truth GP regression, we can estimate the sample noise at each point from the data directly (σ_s in Equation 6.17). We used this value directly as σ_g, since we assumed the robot would have the same sensing capability as the human operators. To estimate σ_a, we calculated the sample covariance between NDVI (from the hand-measured R and NIR levels) and OM (measured directly), yielding the 2×2 matrix,

$$\begin{bmatrix} \sigma^2_{\text{OM}} & \sigma_{\text{OM,NDVI}} \\ \sigma_{\text{NDVI,OM}} & \sigma^2_{\text{NDVI}} \end{bmatrix} \tag{6.20}$$

From the above equation, we can find the variance in OM given a measurement of NDVI, and use this as the UAV sensor noise as,

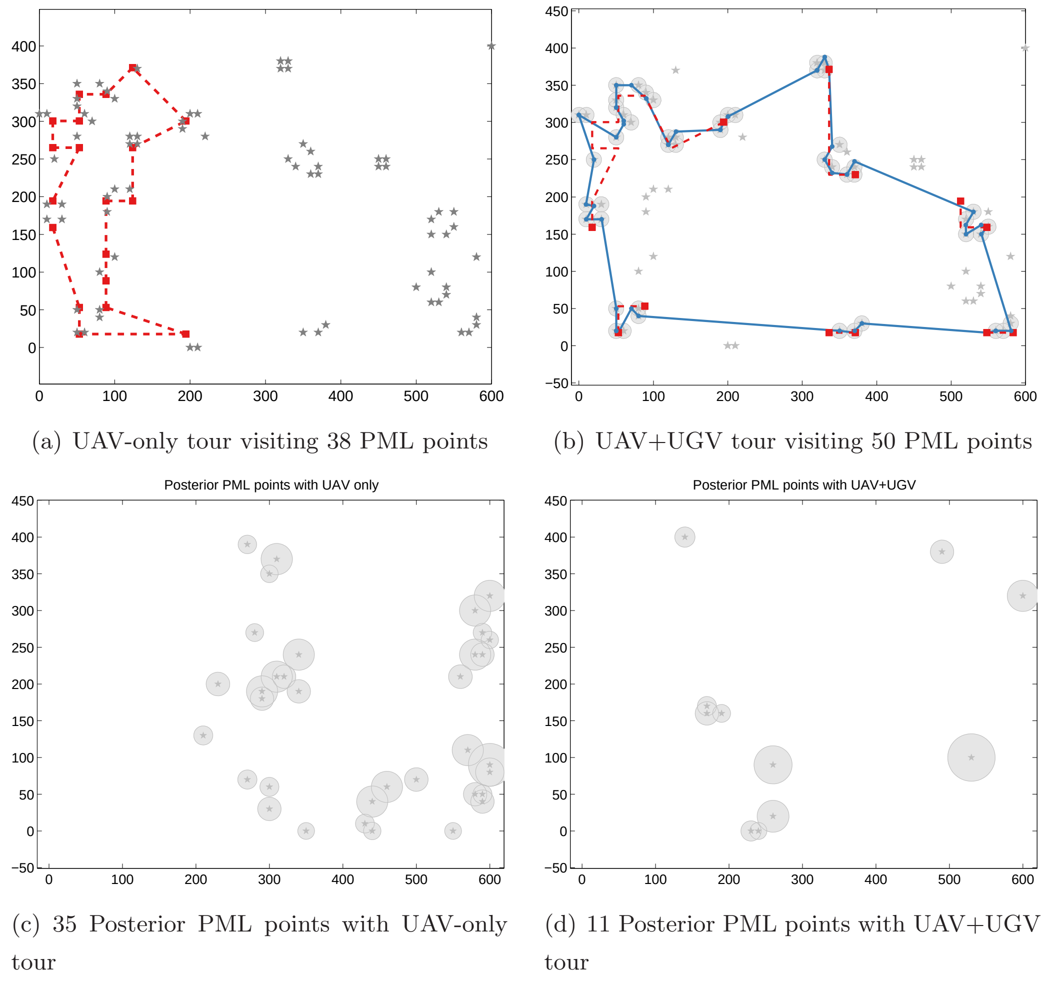
$$\sigma_a^2 = \sigma^2_{\text{OM}|\text{NDVI}} = \sigma^2_{\text{NDVI}} - \frac{\sigma^2_{\text{OM,NDVI}}}{\sigma^2_{\text{OM}}} \tag{6.21}$$

In simulations, we formed a prior estimate of OM levels by down sampling each randomly-generated ground-truth N-map by a factor of 20 and fitting a new GP. We randomly generated 100 N-maps for a 600×400 m field. For each randomly generated prior GP, we found the PML point set as described in Section 6.3 using a desired labeling probability of 0.6. The number of PML points in any instance depends on the randomly generated map.

6.5.3 Results

We first compare the number of PML points covered by the UAV+UGV system versus an UAV-only system. We use the procedure described in Section 6.4 for finding the subset of PML points visited by the UAV-only and the UAV+UGV system, subject to the battery constraint of 25 mins. We used the implementation from the SFO Toolbox [108] for finding an orienteering tour, and the Concorde TSP solver [109] as a subroutine in the Sampling TSPN algorithm implementation.

Figure 6.6 shows a sample run from the simulations. We observe that the UAV-only tour is constrained to only one part of the field, whereas the UAV+UGV system



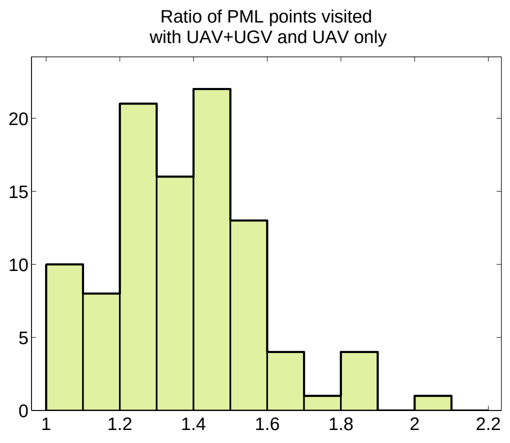
(a) UAV-only tour visiting 38 PML points

(b) UAV+UGV tour visiting 50 PML points

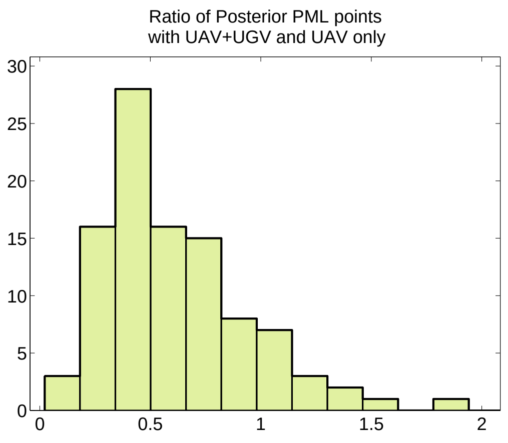
(c) 35 Posterior PML points with UAV-only tour

(d) 11 Posterior PML points with UAV+UGV tour

Figure 6.6: Sample simulation instance. (a) & (b) shows the tours found using a UAV-only and UAV+UGV system. The input consists of 75 PML points. The UAV+UGV tour consists of 6 subtours. (c) & (d) shows the PML points found in the updated N level map after incorporating aerial and ground measurements. The UGV allows the UAV to transport to farther locations in the plot which is reflected in fewer posterior PML points.



(a)



(b)

Figure 6.7: Histograms of the ratio of (a) number of PML points visited, and (b) number of posterior PML points generated after updating the N map with simulated measurements for UAV+UGV system and a UAV-only system, for 100 random instances. Both systems are given an equal budget of 25 minutes.

can obtain measurements from farther away locations. This input instance consisted of 75 PML points, the UAV-only tour covers 38 points whereas the UAV+UGV tour covers 50 points. Figure 6.7 shows a histogram of the ratio of the points covered by the UAV+UGV and the UAV-only tours for 100 random instances. As expected, the ratio is always greater than 1 as the UAV+UGV system is at least as good as a UAV-only system in terms of the number of points visited. Table 6.1 shows the effect of varying the budget on the percentage of input PML points visited.

Table 6.1: Percentage of input PML points visited (avg. of 30 instances).

Budget (sec)	UAV-only	UAV+UGV
500	19	25
1000	36	49
1500	55	72

The UAV+UGV system can cover points that are spread across the field. Intuitively, if the measurements are distributed across the field, we expect the resulting map (after incorporating the measurements) to have fewer mislabeled points than if all measurements are nearby. After calculating the desired UAV/UGV tours, random measurements for the sensors were sampled directly from OM values given the dense (ground truth) GP. We added noise to the measurements using estimated variances $\sigma_a = 0.31$ and $\sigma_g = 0.5$ as described in Section 6.5.2. These values were then used to update the prior GP, which was then used to find the posterior PML points. We observe the posterior PML points in Figures 6.6(c) & 6.6(d). For a fair comparison, we add UGV measurements for each PML point visited by a UAV-only tour, in obtaining the updated N level map.

Figure 6.7(b) shows a histogram of the ratio of the posterior PML points with a UAV+UGV system and a UAV-only system. Since the number of PML points depend on both the variance, and the estimated $N(x)$ values, occasionally there are instances when the number of posterior PML points with UAV-only system are lesser than that of UAV+UGV system. However, as we can observe in Figure 6.7(b) the UAV+UGV system often outperforms the UAV only system in terms of number of posterior PML points.

6.6 Field Experiments

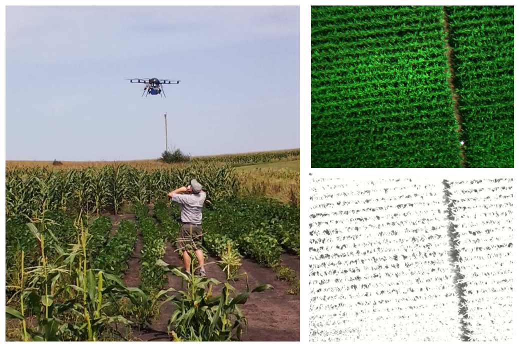
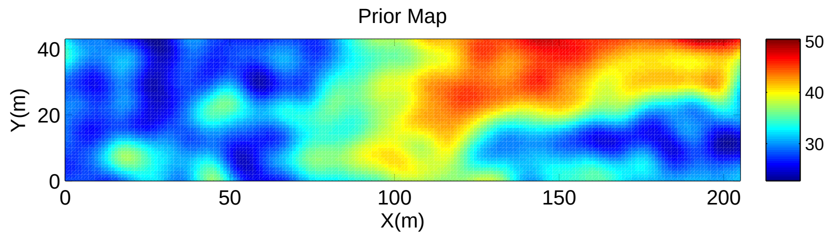


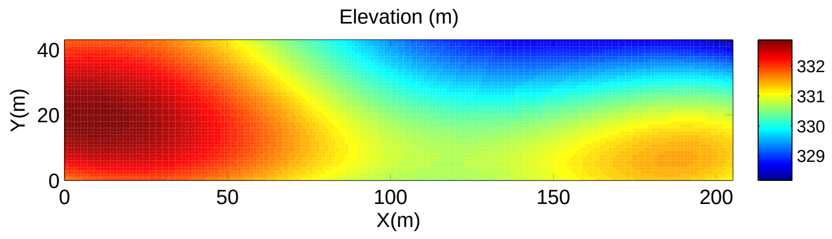
Figure 6.8: Preliminary field experiment with a UAV carrying a multi-spectral camera in a corn plot in Janesville, MN, U.S.A. Visible and near-infrared images shown were obtained from an altitude of 30 m during the experiment.

We conducted proof-of-concept field experiments with the prototype system which is under development. Our current system capabilities include data collection with an autonomous UAV (Figure 6.8). The experiments were conducted in a corn plot at Janesville, MN, U.S.A. The corn plot is a 122 m $\times$ 61 m site for studying the effect of fertilizer treatments on nitrogen stress. The UAV was fitted with a multi-spectral camera from Tetracam [110]. Figure 6.8 shows the visible and NIR images obtained from 30 m altitude. A camera footprint of diameter $C = 30$ m was determined empirically for a flying altitude of 30 m.

A prior estimate was built using dipole data as a proxy for the nitrogen level map. The dipole data consists of ground measurements of electrical conductivity of the soil. The conductivity, in turn, depends on the land elevation, soil moisture, and soil texture. Elevation changes lead to water run-off leading to changes in the nitrogen levels due to leaching. The prior estimate built using the dipole data is shown in Figure 6.9(a), and the elevation map of the plot is shown in Figure 6.9(b).



(a)



(b)

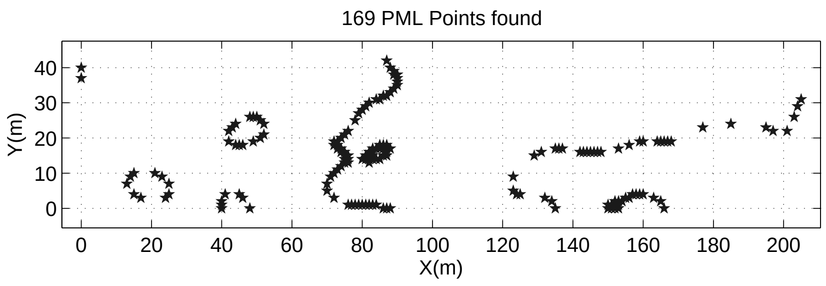
Figure 6.9: (a) Prior map built using dipole measurements as proxy for nitrogen levels. Dipole measurements of the soil conductivity depend on the elevation and soil moisture and texture, which in turn affect nitrogen levels. (b) Elevation map of the test site. (Figures best viewed in color)

Each point in the prior map was labeled as either high or low, using the average prior map value as the threshold. The desired maximum probability of mislabeling was set of $P_d = 0.45$. 169 PML points were identified based on this (Figure 6.10(a)). These points were partitioned into a grid of resolution $C/\sqrt{2}$. The orienteering algorithm was run on the graph constructed using the grid. All the tours were computed considering a nominal UAV speed of 4 m/s. The waypoint following controller on-board the UAV was programmed to maintain this speed. The typical battery lifetime of the UAV at this speed was observed to be approximately 600 s. The UAV can potentially cover all the input PML points in 600 s. Since the goal of this experiment was to demonstrate the proof-of-concept implementation for a larger setup, we restricted the battery lifetime to 200 s.

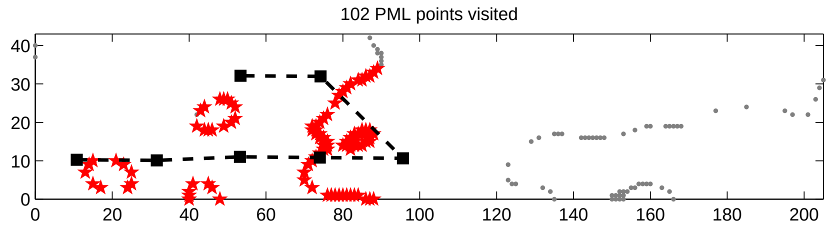
Figure 6.10(c) shows the UAV tour found for a UAV-only system. The number of points visited by the UAV increase from 102 with a UAV-only system to 134, using a UAV+UGV system. Figure 6.10(b) shows two deployments computed for a UAV+UGV system. Figure 6.10(d) shows the GPS coordinates of the UAV during actual execution of the two deployments given in Figure 6.10(b). We programmed the UAV to take-off and land from the same location at one of the corners of the plot where we were stationed (instead of the locations computed by the algorithm) for safety purposes.

6.7 Conclusion

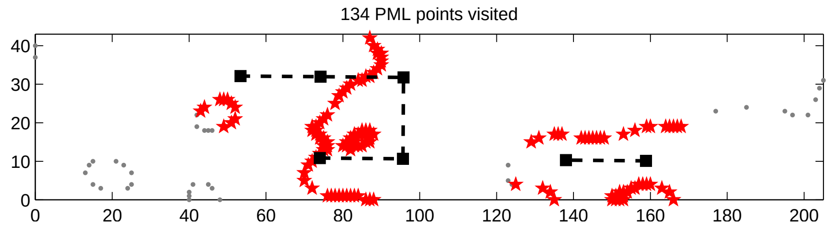
In this chapter, we studied two coverage problems motivated by the use of robots in precision agriculture. Precision agriculture uses data from ground and aerial sensors in order to estimate and predict the status of crops. Obtaining a soil measurement from a ground robots requires spending some time. With this as motivation, we introduced a new coverage problem, termed SamplingTSPN, which penalizes the time spent in traveling and the time spent for obtaining measurements, given a set of input disks within which measurements must be obtained. We presented an $\mathcal{O}\left(\frac{r_{\max}}{r_{\min}}\right)$ approximation algorithm where $r_{\min}$ and $r_{\max}$ are the minimum and maximum radius of input disks. Aerial images, on the other hand, can be obtained instantaneously. However, small UAVs have limited on-board energy. We studied the problem of maximizing the number of points visited by the UAV, subject to its maximum battery lifetime. Unlike traditional approaches, our algorithm takes into consideration the situation where



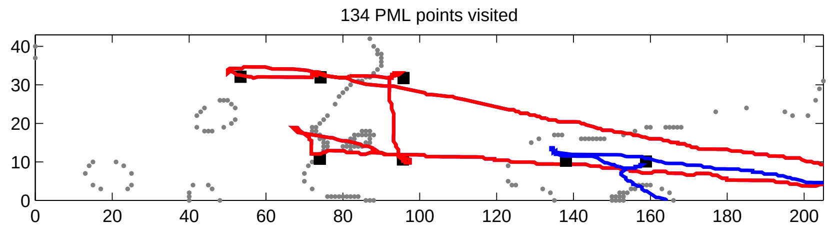
(a)



(b)



(c)



(d)

Figure 6.10: (a) 169 PML points were found after classifying the prior map into two labels. (b) The camera footprint along a UAV-only tour covered only 102 PML points (marked by larger star, in red) with a maximum budget of 200 s. (c) UAV+UGV tour consisted of two paths, and covered 134 of the PML points with the same budget. (d) GPS coordinates of the UAV for executing the tours in (c).

the UAV can land on the UGV and thus be carried between points without expending energy.

We have started building the complete system using a Clearpath Husky A200 as the ground robot. In order to execute the algorithms presented in this chapter, additional capabilities such as soil sampling and autonomous landing are necessary, which are part of our future work.

Chapter 7

Coverage and Tracking with Autonomous Boats (Monitoring Invasive Fish)

In this chapter, we study a practical application for the coverage and tracking tasks seen in the previous chapters. Our motivating application is that of autonomously monitoring radio-tagged invasive fish in Minnesota lakes. We describe a prototype robotic sensing system designed for this application along with coverage and tracking algorithms that are motivated by sensing constraints observed in practice.

Invasive fish, such as the common carp, pose a major threat to the ecological integrity of freshwater ecosystems around the world. Presently, these fish are controlled using non-specific toxins which are expensive, ecologically damaging, and impractical in large rivers and lakes. Recent studies in small lakes have established that common carp aggregate densely at certain times and regions within the lakes [10]. Their population can be controlled by targeting these aggregations using netting. To predict the locations of aggregations within a lake, biologists surgically implant radio tags on some fish, and use radio antennas to track them periodically (Figure 7.1). Manually locating tagged fish in large, turbid bodies of water is difficult and labor-intensive making them a prime candidate for automation using a robotic sensing system.

In this chapter, we describe our system developed for autonomously monitoring the

Figure 7.1: **Left:** Fish biologists manually tracking carp. **Right:** Targeted removal of the carp aggregation in Lake Lucy, MN, USA, using a large under-ice seine. Over 95% of all carp in this lake were captured in 4 hours. Photos courtesy of Peter Sorensen.

radio-tagged carp using robotic boats (Figure 7.2). In winter, the system uses a wheeled robot to operate on the frozen lakes. The process of locating tagged fish consists of a coverage phase followed by an active target localization phase. We formulate the coverage and localization problems by grounding them in practical constraints observed in our system.

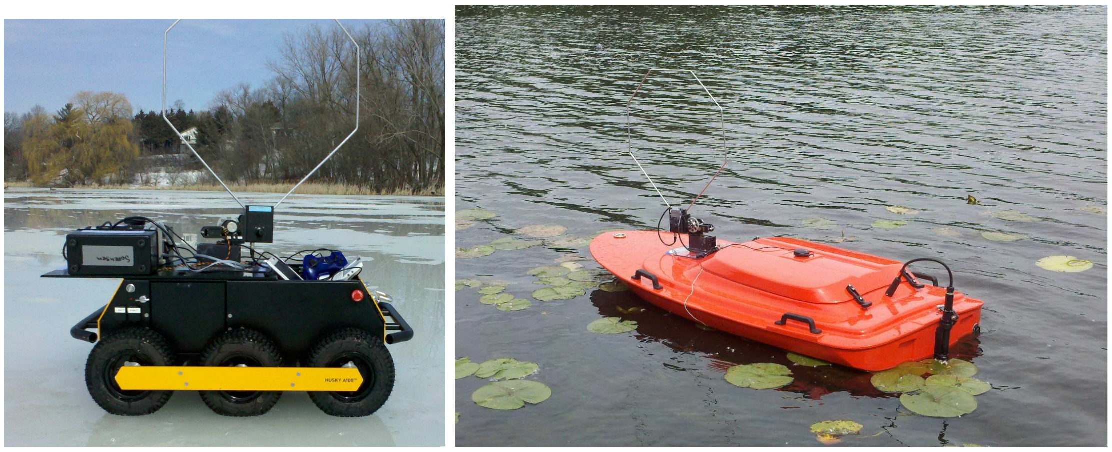


Figure 7.2: Robotic sensing system for monitoring radio-tagged carp in lakes. A directional radio antenna is mounted on a pan-tilt unit. In winter, the system employs a wheeled robot to operate on frozen lakes.

The goal in the coverage phase is to locate the tagged fish within the sensing range of the radio antenna (around 50 m). Common carp are known to loiter in a region (called home ranges) for long periods of time [21]. Instead of covering the entire lake, the coverage algorithm takes as input a set of regions likely to contain the fish. We present approximation algorithms for finding a minimum length tour to cover all regions and report results from field executions.

In the active localization phase, the goal is to accurately estimate the location of the tagged fish, assuming they remain relatively stationary. The radio antenna, in addition to detecting the tags, also has directional properties. The antenna can be rotated to obtain a bearing measurements towards the tag. However, these measurements are noisy. We propose three active localization strategies to compute measurement locations so as to localize stationary tagged fish precisely. We evaluate these through simulations and compare their performance using experiments with a reference radio tag. Finally, we present large scale field experiments where the robots carry out both coverage and active localization tasks in order to locate reference radio tags as well as radio-tagged fish.

The rest of the chapter is organized as follows: We start with the related work in Section 7.1. We describe the system architecture, low-level navigation algorithms and preliminary tests with the radio antennas in Section 7.2. The coverage algorithms along with their analysis and field trials are presented in Section 7.3. In Section 7.4, we describe the three active localization strategies, followed by results from simulations and experimental evaluation. Finally in Section 7.5, we report results from field experiments where the robot executed both coverage and localization phases for locating reference tags and radio-tagged fish.

7.1 Related Work

Marine robotics has seen significant activity in recent years. Numerous groups across the world are involved in designing and developing marine robotic systems for various applications such as tracking dynamic phytoplankton [111], autonomous bathymetry [112], ship hull inspection [113], guarding naval assets [114], and collecting biological data from stationary sensors [115]. The underwater robotic system developed by Clark et al. [116] to track tagged leopard sharks is closest to our application. The recent survey by Dunbabin and Marques [117] provides an excellent overview of environmental monitoring systems, many of which employ marine robots.

The robotic coverage problem has been well studied in the literature (c.f. [118]). Typically, the emphasis is on covering unknown environments and the objective is to guarantee that all points are eventually covered. In our problem, the environment itself is known and the goal is to cover all points in the minimum time. This is closely related to the Traveling Salesperson Problem (see Section 2.3 for a review).

In the localization phase, we perform active, bearing-only target localization. In one of the earlier works on this problem, Hammel et al. [119] used the determinant of the Fisher Information Matrix (FIM) as the objective function to be maximized, and numerically computed an optimal open-loop trajectory for a robot in the case where measurements are obtained continuously. The resulting trajectory follows a spiral shape, but is an open-loop trajectory which does not depend on the actual measurements the robot obtains. Frew [75] presented a feedback strategy for tracking targets with bearing measurements obtained using monocular vision. The strategy is based on a state-exploration tree, and a trajectory is obtained using breadth-first search for the minimum uncertainty. Recently, Zhou and Roumeliotis [120] considered the active localization problem for a team of robots capable of taking range and/or bearing measurements towards a moving target. They consider maximum speed and minimum sensing range constraints, and plan for the next best sensing location using the trace of the posterior covariance matrix as the uncertainty measure.

What differentiates our problem from these works is that each measurement in our system takes a long time, and the uncertainty in measurements is considerably larger. We address these factors in our strategy by using the best worst-case behavior as our objective function, and limit the number of measurements as part of our planning process. Since the completion of the work presented in this chapter, we have developed a new active localization algorithm [121] which provides strong theoretical performance guarantees, as well as performs robustly in practice.

We start with an overview of the system and present our search and active localization strategies in Sections 7.3 and 7.4 respectively.

7.2 System Description

Our system consists of a robotic platform with on-board sensors, radio tags and receiver, and a directional antenna. We discuss each of these in turn.

7.2.1 Robots

The hull of our robotic platform (Figure 7.1) is the *QBoat* designed by Oceanscience [122]. The QBoat has dimensions of 182cm $\times$ 71cm, and can carry a payload of approximately 40kg. The boat is capable of a maximum speed of 1.65m/s. An on-board 12V, 30Ah NiMH battery allows approximately two hours of continuous operation. In winter, once the lakes are frozen, we use a wheeled robot as the main platform.

The electronics on-board the robots (Figure 7.3) consist of a laptop running high-level software, an Atmel micro-controller board for low level interface, radio receiver equipment (described in the following section), a digital compass and a Garmin 18x Global Positioning System (GPS) unit. We also have a remote override radio control system that can directly control the boat, if desired.

We have a modular software architecture based on the Robot Operating System (ROS), comprising of packages for navigation, localization, simulation, and reading sensor data. ROS allows remote monitoring of data from another computer on the shore via an ad-hoc network formed with the on-board laptop. The electronics and software can be easily transferred to between the two platforms. At the core of the navigation package for the boat are the waypoint following routine described next and an EKF-based localization routine similar to the catamaran solution presented in [123].

7.2.2 Navigation Routine

The boat uses on-board GPS and compass sensors as feedback for navigation. A simple control algorithm is used to generate the steering angle θ for the boat. The propeller is always set to move in the forward direction. While going from starting point A to destination point B, the angle by which the boat should steer depends upon the current heading $H_{heading}$, the angle made by the line AB, denoted by H_{start} and the angle made by the line joining the current position of the boat to the destination position, called

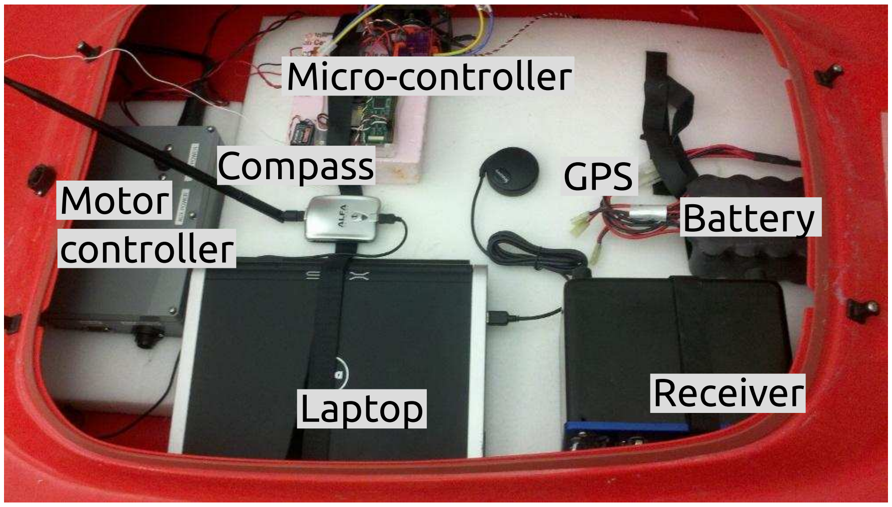


Figure 7.3: Top view of the electronics compartment: On-board electronics comprises of a laptop, GPS, micro-controller board, batteries, digital compass, motor controller, and radio receiver.

H_{dest}. The desired change in the heading, ΔH, of the boat is then given by,

$$\Delta H = \alpha(H_{start} - H_{heading}) + (1 - \alpha)(H_{dest} - H_{start}) \tag{7.1}$$

$$\theta = k_p \Delta H \tag{7.2}$$

where α is a weighting factor.

The first term in Equation 7.1 gives the error between the starting heading and the current heading, where as the second term calculates the error between the current heading and the desired heading. The steering angle θ is set proportional to the error ΔH. The weighting factors α, β and the constant of proportionality, k_p, were determined experimentally.

Waves and wind can potentially throw the boat off the straight line which would require large error correction. Hence, we constantly check to see if the boat is within a particular band drawn about the line AB. If the boat drifts outside of this band on either side, we make a new call to the method with the current position and heading of the boat as the starting point towards the same destination.

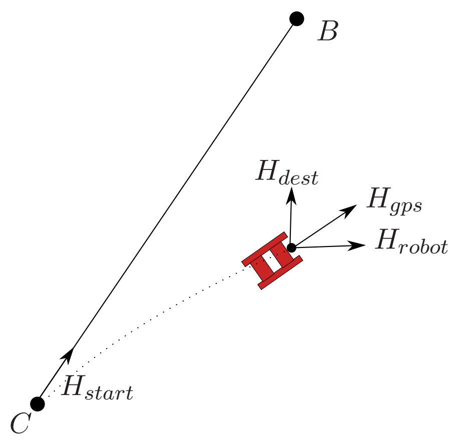


Figure 7.4: Waypoint navigation between points A and B. H_{gps} is the track obtained from the GPS and H_{robot} is the heading obtained of the robot. H_{robot} is shown with slight error with respect to true heading of the boat. H_{dest} is the desired heading.

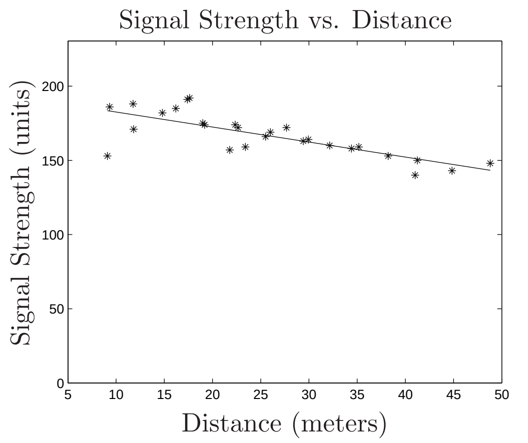
7.2.3 Radio Tag and Receivers

For sensing the fish, we use radio tags manufactured by Advanced Telemetry Systems (ATS) [124]. A complete fish sensing system by ATS consists of radio tags, a loop antenna connected to a radio receiver, and a data logger which provides the computer interface for the receiver. Each radio tag emits a short pulse roughly once per second. The radio antenna (shown on top of the boat in Figure 7.2) is used to detect these pulses.

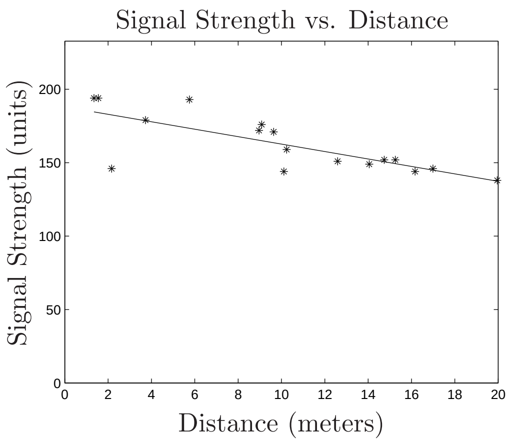
The radio receiver reports the received signal strength of the pulse. We conducted a set of experiments[1] to understand the relationship between signal strength and detection distance. These experiments were conducted at Lake Riley in Eden Prairie, Minnesota. A reference frequency tag was inserted under the water in the middle of the lake at a depth of about 1 meter. The robot was directed in a straight line away from the tag, with the antenna was always pointing towards the tag (i.e. the received signal strength was always at the maximum for the directional antenna).

The plot of the observed readings with respect to distance is shown in Figures 7.5(a) for a depth of 1 meter. The maximum distance from the tag up to which the signal was

[1]The experiments reported in this subsection were conducted using an earlier version of the system described in [125].



(a) Depth: 1m, Maximum Detection Distance: 49m



(b) Depth: 2m, Maximum Detection Distance: 20m

Figure 7.5: Plot of signal strength vs. distance with least squares linear fit. A reference tag was immersed at different depths under water and the corresponding signal strength was measured by moving the boat along a straight line away from the tag.

received for this trial was approximately 49 meters. It can be observed that the signal strength decreases with respect to distance, in general. However, this relation is also a function of the depth of the tag in the water. We repeated the same experiment varying the tag depth to 2 and 4 meters. The plots for the experiment with depth of 2 meters is shown in Figure 7.5(b). In this case, the tag was only detected up to a distance of 20 meters, while for a depth of 4 meters, this distance further reduced to 10 meters. This makes localizing with only signal strength measurements difficult since the depth of the fish is not known. Therefore we rely only on the directional nature of the antenna, and obtain a bearing measurement towards the tag. Our method for estimating the bearing is presented in Section 7.4.

The tag on each fish is assigned a unique frequency. The receiver can be programmed to tune in on one or more frequencies. In the coverage phase, we program the receiver to loop through a list of frequencies of tagged fish present in the lake. To reliably detect a pulse from a tag, the receiver needs to stay tuned on the corresponding frequency for more than one second since the tags emit pulses at about 1Hz. After detecting a radio tagged-fish in the coverage phase, we program the receiver to tune only to the corresponding frequency and switch to the localization phase. In the coverage phase, described in the next section, we do not rotate the antenna or obtain bearing measurements, since the goal of this phase is only to detect the presence of a tag.

7.3 Coverage

The first phase of our monitoring task is to search for all tagged fish in the lake. One of the current models for carp mobility suggests that each fish moves to its preferred region in the lake during day time, and remains in that region for long periods of time [21]. To increase the efficiency of our system, we restrict our attention to only those regions of the lake which are likely to contain the fish (Figure 7.6). We assume that these regions are connected in the sense that there is a path between any two points. We also assume the fish remain stationary for the duration of covering a region.

The radio antenna has limited sensing range. A given robot path is said to cover a point if the point lies within the sensing range of the robot at some instance along the path. The coverage problem can be defined as follows: *Given a set of connected regions $R = \{R_1, R_2, \ldots, R_n\}$, find a minimum length tour which covers every point in each region $R_i \in R$.*

A possible approach for solving this problem is based on the Traveling Salesperson Problem (TSP). We can discretize each region with a resolution dependent on the sensing range (Figure 7.6), and compute a TSP tour of this set of points. However, approximation algorithms for TSP usually require a metric graph, which is typically represented as a complete graph whose vertex set is the point set to be covered. In such a representation the number of edges is quadratic in the number of points. As the lake size or the sampling granularity increases, maintaining and operating on this large graph can become infeasible.

The coverage problem defined above is a generalization of Euclidean TSP and consequently NP-Hard. Next, we present a general approach for solving the coverage problem and show that the length of the path using this approach does not deviate significantly from the length of an optimal tour.

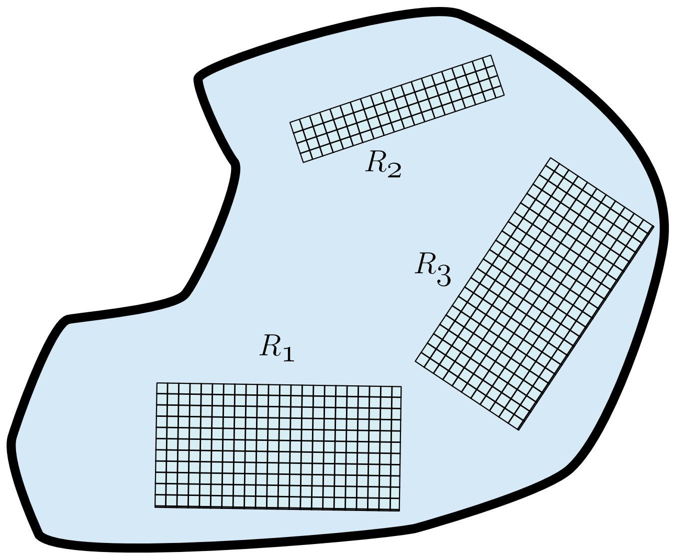


Figure 7.6: We incorporate domain knowledge to restrict the coverage region for the fish to a given set of regions, e.g. $R = \{R_1, R_2, R_3\}$. A simple approach of discretizing these regions and finding a TSP tour becomes infeasible when the regions are large.

7.3.1 Algorithm Description

Our general approach is composed of two steps. First, we compute a tour τ_R that visits all the regions in R exactly once. We say that region R_i is *visited* if *any* point in R_i is visited by the tour. The tour τ_R imposes an ordering on the regions, and defines (possibly same) entry and exit points for each region. The entry and exit points are where τ_R intersects a region for the first and the last time. Such a tour is not necessarily a solution to the original problem, since it is not guaranteed to cover all points in each region. We compute a coverage tour C_{R_i} for each region $R_i \in R$ independently, starting and finishing at the entry and exit points for each R_i. The final tour τ is constructed by augmenting the coverage tours of each region to the visiting tour τ_R.

We now analyze the performance of this algorithm. Let OPT be an optimal tour which visits *and* covers all the regions in R in minimum time. Let τ_R^* be the optimal tour which visits all the regions in R. Since OPT also visits all the regions in R, we have $|OPT| \geq |\tau_R^*|$, where $|\tau|$ denotes the length of tour τ. Let $C_{R_i}^*$ be the optimal coverage tour for a region R_i. OPT covers every region in R therefore, we have $|OPT| \geq$

$\sum_{R_i \in R} |C^*_{R_i}|$.

Suppose we use an α-approximation algorithm for computing τ_R and a β-approximation algorithm for finding the coverage tour of each region. Then the tour τ obtained by visiting the regions according to the order given by τ_R, and covering each region independently when it is visited, has a cost of at most $\alpha|\tau^*_R| + \sum_{R_i \in R} \beta |C^*_{R_i}|$. Equivalently,

$$\begin{aligned} & |\tau| \leq \alpha|\tau^*_R| + \sum_{R_i \in R} \beta|C^*_{R_i}| \leq \alpha|OPT| + \beta|OPT| \\ \therefore \quad & |\tau| \leq (\alpha + \beta)|OPT| \end{aligned}$$

Therefore, this approach costs at most a factor $(\alpha + \beta)$ of an optimal algorithm.

We now present algorithms for the two components of the strategy: computing a tour that visits the regions and covering the regions with specified entry and exit points.

7.3.2 Visiting the Regions: TSPN and the Zookeeper Problems

Computing a tour τ_R that visits all the regions depends on the geometric properties of the regions. This problem is commonly known as TSP with Neighborhoods (TSPN). Most geometric instances of the TSPN problem are NP-Hard. In general, we can use constant-factor approximation algorithms for TSPN (e.g., [54, 55]) to find τ_R and α.

In our application, it is reasonable to model the lake as a simply-connected region, i.e., without any obstacles. Furthermore, regions of interest where the fish may lie are usually close to the shore. If the regions are convex polygons touching the boundary of a simply-connected lake then the tour can be computed using the so-called zookeeper's route [126]. This special case of the TSPN can be solved optimally ($\alpha = 1$) in polynomial time due to the following lemma.

Lemma 22 ([126])**.** *Let $R = \{R_1, R_2, \ldots, R_i, \ldots, R_n\}$ be a set of convex regions located along the perimeter of a simply connected polygon P. There exists an optimal solution for visiting the regions in R which visits them in the order they appear along the boundary of P.*

Once the ordering of the regions is known, the shortest tour visiting all regions can be calculated using dynamic programing. The exact solution is given in [126]. We use a simpler solution by discretizing the boundary of the regions for determining the entry and exit locations for each region. We build a table $C(i, s_i)$ which stores the length of a tour that enters the region R_i at location s_i for the first time. The entries of the table are computed using the following recurrence:

$$C(i, s_i) = \min_{t_{i-1}} \left[\min_{s_{i-1}} (C(i-1, s_{i-1})) + d(t_{i-1}, s_i) \right], \tag{7.3}$$

where s_{i-1} and t_{i-1} lie on the boundary of R_{i-1} and $d(x, y)$ is the Euclidean distance between points x and y. The cost of entering the region R_i at point s_i is equal to the minimum cost of reaching the previous region, R_{i-1}, entering at location s_{i-1}, plus the shortest distance from R_{i-1} to R_i, $d(t_{i-1}, s_i)$. By Lemma 22, the ordering of the regions is optimal. Since we cover all possible values of t and s, the algorithm computes an optimal solution up to the discretization error.

To turn these tours into coverage paths, we need a way to cover a region with specified entry and exit points. We next present such a technique when the regions are arbitrarily oriented rectangles. Rectangles are both easy to specify and general enough for practical purposes.

7.3.3 Covering Regions with Given Entry and Exit Points

The algorithm presented in Section 7.3.2 generates an entry and exit point for each region. These points impose a constraint on our algorithm for finding a path that covers the rectangle. The following lemma shows that we can cover a rectangle satisfying this constraint, and be only a constant factor away from an optimal coverage path without such constraints. We assume that the rectangle has an $x \times y$ grid imposed on it, such that visiting all grid cells covers the rectangle.

Lemma 23. *Let R be a rectangle with a grid imposed on it. Let s and t be two grid points on the boundary specified as entry and exit points. There exists a tour T which starts at s, visits every grid point and exits at t such that the length of T is at most twice that of an optimal tour which visits every grid point but can start and end at any points on the boundary of R, not necessarily s and t.*

Proof. Let π be the optimal path to cover R without any restrictions on the starting and ending points. When R is a rectangle, π is a boustrophedon path which visits every point exactly once.

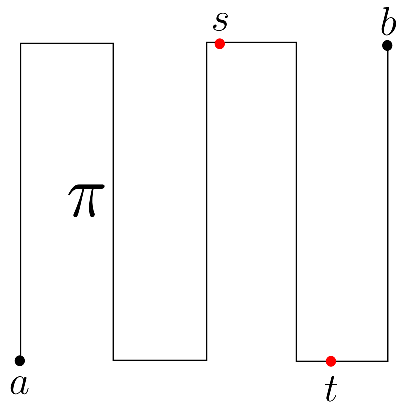


Figure 7.7: Covering a rectangle with given entry and exit points (s and t) with a 2-approximation: Follow π from s to a, complete π, follow π from b to t. In the worst-case the optimal path π is covered twice.

Suppose that π starts at a and ends at b. Note that $s, t \in \pi$. Without loss of generality, we assume that t is between s and b along π (See Figure 7.7). We form a coverage path from s to t using π as follows: From s, go to a along π and come back to s by retracing these steps. Then go to b from s along π (passing through t). Finally, arrive at t from b along π. This path visits every point on π and has length at most twice that of π. $\square$

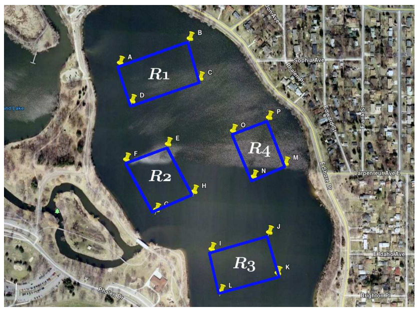
The result is tight; when the input is a rectangle with one side length equal to r, and $s = t$, each point is covered twice.

To summarize, we showed that the following algorithm for covering rectangles along the boundary of a lake is an $(\alpha + \beta)$-approximate algorithm with $\alpha = 1$ and $\beta = 2$.

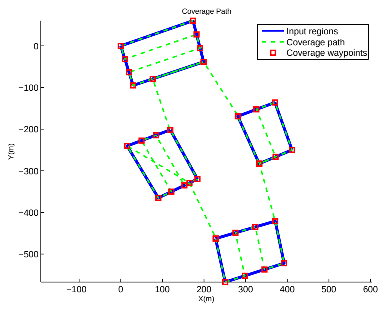
1. Compute the shortest tour τ_R which visits each region in R using the dynamic programming solution presented in Section 7.3.2. This algorithm returns an entry and exit point for each rectangle, along with their ordering.

2. Follow τ_R as follows: Starting from the entry point of an arbitrary region, whenever a region is visited, cover it using the strategy given in Lemma 23 which ends at the exit point. Move to the entry point of the next region given by τ_R and repeat till all regions are visited.

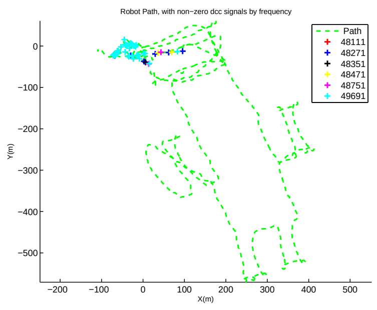
7.3.4 Experiments



(a) Coverage regions



(b) Coverage path found using our algorithm.



(c) Actual path executed by the robot.

Figure 7.8: Coverage experiment conducted at Lake Phalen, MN, USA. **(a)** the four input regions, **(b)** the path found by the algorithm described in Section 7.3, **(c)** the actual path followed by the robot during coverage. The robot traveled a total distance of 5.6km in about 87min. Locations where signals from radio-tags were detected are also marked, along with their frequencies.

We implemented this coverage algorithm on our robotic boat and evaluated it through field trials at Lake Phalen, MN, USA. The input regions for one such trial are shown in Figure 7.8(a) (chosen arbitrarily for testing our algorithm). The series of offline waypoints generated by our algorithm are shown in Figure 7.8(b). We used an empirically determined sensing range of 50m to generate the boustrophedon paths. The actual trajectory executed by the robot is shown in Figure 7.8(c). The robot traveled a total distance of 5.6km in about 87min while executing this trajectory.

While moving along the coverage path at certain locations, the robot detected signals from five radio-tagged fish and a reference tag present in the lake. These robot locations are marked with the corresponding tag frequency in Figure 7.8(c). The actual position of the fish can be anywhere within a distance equal to the sensing range from these locations. To better localize the fish, we switch to the active localization phase whenever we detect a signal on one of the tuned frequencies. We describe our algorithms for the localization phase next.

7.4 Active Localization

The objective of the localization phase is to use bearing measurements from the radio antenna to localize a tagged fish accurately once it is found during the coverage phase. The robot must choose sensing positions which provide the most information about the location of the tag. We use an EKF to estimate the position of the tag and represent the uncertainty in the position of the tag with its covariance. We seek sensing locations which minimize the determinant of the covariance matrix.

We begin by describing how to use the signal strength output of the radio receiver to obtain a bearing towards the tag.

7.4.1 Measurement Model

The received signal strength varies with the relative angle of the plane of the loop antenna with the tag. If the tag is directly aligned with this plane, the signal strength is highest. Since the antenna is mounted on a pan-tilt unit we can rotate it and sample the signal strength as a function of the relative angle from the robot.

Figure 7.9 shows a subset of the samples obtained by rotating the antenna in steps of $15°$ over $[-90°, 90°]$. The true bearing angle is $-30°$. As we can see, the values around the true bearing are all high. This makes finding a single bearing direction from the maximum signal value difficult. Instead, we fit a function to the samples obtained, and find the maximum value of this function. After trying different models and fitting methods, we concluded that least squares fitting of a cubic polynomial works best for our system. We found this reliably estimated the bearing to the tag to within $15°$.

Note that the bearing obtained is an infinite line (as opposed to a directed ray),

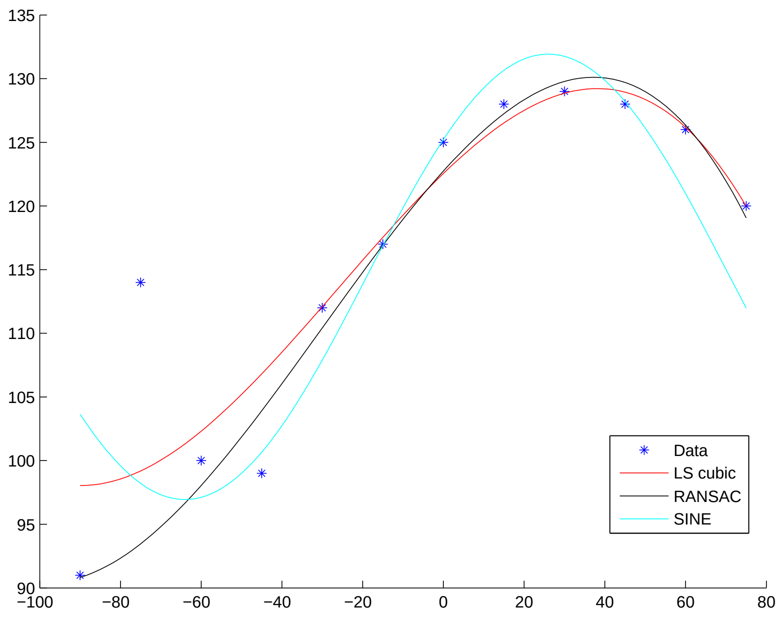


Figure 7.9: A coarse sampling and the corresponding estimates. The horizontal axis is the bearing and the vertical is the measured signal strength. In general, least squares estimation of a cubic polynomial provided the best estimates.

and hence there is ambiguity in the obtained bearing. For example, if α is the direction with maximum signal strength, then α and $\alpha + \pi$ are both valid bearing measurements. We disambiguate by moving along either α or $\alpha + \pi$ and checking if the signal strength increases or decreases. A detailed description of other methods for disambiguating the measurements is given in [127].

7.4.2 Optimization of Robot Motion

Since measuring the bearing takes time (about 1min), the estimation must be performed using a small number, say k, of measurements. Further, these locations must be chosen in an online fashion as the measurements become available. We present three strategies to compute k sensing locations, and compare their performance in simulations and real-world experiments. All three active localization strategies require an initial estimate of the tag. In [128], we presented a scheme to initialize the target based on two bearing measurements taken from different sensing locations. In [129], we presented a slower but more robust initialization scheme which uses only the radio detection signals. Given this initial estimate, we propose the following three strategies to determine the next k sensing locations.

FIM

The Cramer-Rao Lower Bound (CRLB) for an unbiased estimator is a lower bound on the estimation error covariance. This lower bound is equal to the inverse of the FIM (denoted by I) for the k measurements. The determinant of I is inversely proportional to the square of the area of the 1-σ uncertainty ellipse, and is commonly used as the objective function to be maximized. For k bearing measurements with zero-mean Gaussian noise, the determinant of I (denoted by $|I|$) is given as,

$$|I| = \frac{1}{\sigma^4} \sum_{i=1}^{k} \sum_{j=1}^{k} \left[\frac{\sin(\theta_i - \theta_j)}{d_i d_j} \right]^2 \tag{7.4}$$

where θ_i and d_i is the angle and distance from the i^{th} sensing location to the true target location.

We impose a grid of size $n \times n$ centered at the current position of the robot. To compute the k sensing locations, we exhaustively consider each of the $\binom{n^2}{k}$ combinations as a candidate trajectory, and compute $|I|$ for each. An optimal trajectory can then be chosen as one with the minimum value of $|I|$.

Greedy

Instead of computing a fixed path for the k measurements, we can use an online greedy strategy which picks the next sensing location based on the current estimate and uncertainty of the position of the tag. Given the current robot and tag estimates, Greedy considers all neighboring locations of the robot as candidate sensing locations, and computes the posterior covariance by simulating an EKF update at each sensing location with a discretized set of possible bearing measurements. Greedy then picks the candidate location where the determinant of the posterior is minimum.

Enumeration Tree

We extend the objective function of Greedy to look ahead k measurements, in the Enumeration tree strategy. We build a min-max tree that explores the set of all sensing locations and all possible measurements that can be obtained, since the uncertainty depends on both. The tree consists of two types of nodes at alternate levels (Figure 7.10): MAX nodes (u_i) represent neighboring robot locations to the current, and MIN nodes (z_i) represent the discretized set of possible measurements. Each node stores an estimate of the target's state and covariance by simulating EKF updates based on the sensing locations and bearing measurements stored along the path in the tree. Details are presented in [128].

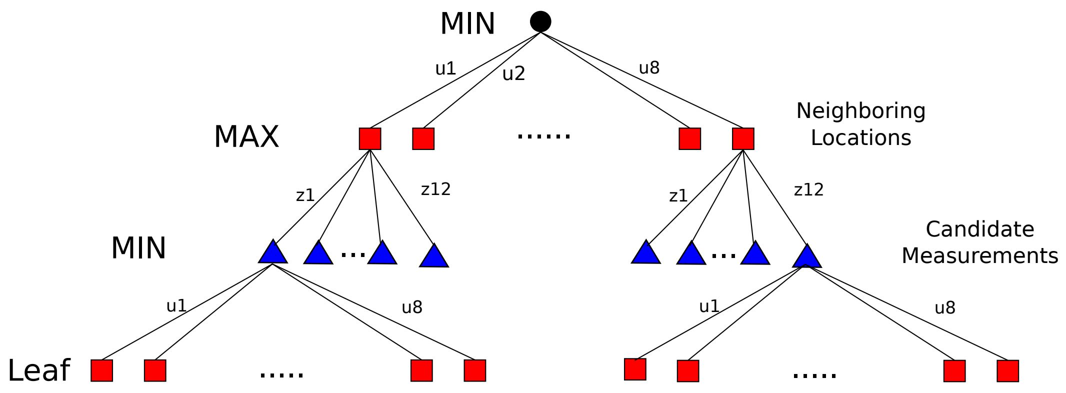


Figure 7.10: Min-max tree: $u_1, \ldots, u_8$ are the neighboring locations for the robot, and $z_1, \ldots, z_{12}$ are candidate bearing measurements.

Once the tree is built, the min-max values for each node are propagated bottom-up starting with the leaf. The min-max value for the leaf nodes is defined as the determinant of the simulated posterior covariance matrix stored at that node. The min-max value for all MAX nodes is the max of min-max values of its children, and that for non-leaf MIN nodes is min of min-max values of its children.

During execution, the robot chooses the MAX node with the minimum min-max value as next sensing location at each iteration. The MIN node is chosen as per the actual bearing obtained. Since we use discrete measurement samples, there might not be a node with bearing exactly equal to the actual measurement. In addition, there is uncertainty associated with the position of the robot itself. Hence, we use the Bhattacharya Distance [130] to find a MIN node with posterior covariance closest to the covariance after the measurement update.

7.4.3 Simulations and Experiments

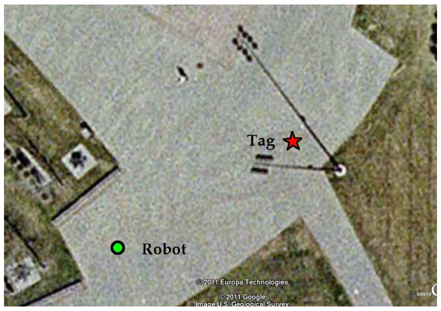
We first compared the performance of the three strategies in simulation. We ran 100 random trials with the true tag 25m away from the starting position of the robot in each trial. A grid with side length 3m was used to generate sensing locations for 3 measurements. We generated noisy bearing measurements by corrupting the true bearing with Gaussian noise ($\sigma = 15°$).

The mean errors for FIM, Greedy, and Enumeration tree were 6.30m, 5.98m, and 5.73m, and the mean determinant of the final covariances were 54.81, 40.59, and 48.36 units respectively. The poor performance for the FIM strategy can be attributed to the fact that it is an open-loop strategy which depends on the initial estimate. Further, it computes locations which minimize the lower bound on final uncertainty of an "efficient estimator" (i.e. estimator whose variance is equal to the CRLB). Since EKF is not an efficient filter, there is no guarantee that it would achieve this lower bound. On the other hand, the Enumeration tree and the Greedy compute the actual covariance of the EKF estimator and pick the location which would minimize its determinant.

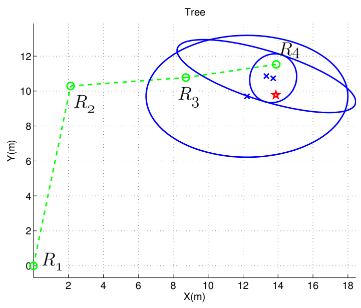
We conducted experiments with the wheeled robot as well as the boats. The experimental setup for the wheeled robot is shown in Figure 7.11(a). A reference tag was kept at a location marked with a star, with its GPS coordinates noted for ground truth. We conducted three trials each using the Enumeration tree and the Greedy strategy to determine two measurement locations.

The results from one such trial with the Enumeration tree and Greedy are shown in Figures 7.11(b) and 7.11(c), respectively. The robot's mean estimated positions are labeled by green circles, while estimates of fish locations are shown as blue crosses. The true location of the tag is marked with a red star. The final estimate of the target from the EKF was $(13.72, 10.73)$, which was within 2 meters of the true position. The final covariance from the EKF is shown in Figure 7.11(b) with the smallest blue ellipse.

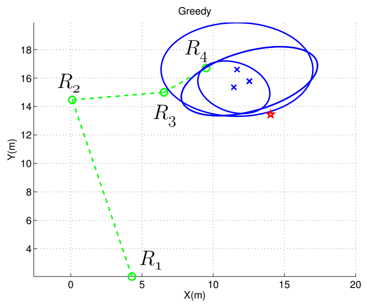
The results from other trials are given in Table 7.1. The Enumeration tree strategy performs slightly better than Greedy, both in terms of final error and final uncertainty.



(a) Experimental setup.



(b) Enumeration Tree Strategy (depth 2)



(c) Greedy Strategy

Figure 7.11: Setup and experimental results for two strategies. R_1 and R_2 are the two locations used for initialization. R_3 and R_4 are the measurement locations obtained using the corresponding strategy. The estimates at R_2, R_3 and R_4 are shown along with the corresponding 1-σ ellipse bounds. The true location of the tag is marked by a star.

However, the performance gains are a trade-off with respect to the significant computation time and space required for building the tree. Hence, we decided to use the Greedy strategy on our system in field experiments with the boat.

For the boat tests, a reference tag submerged in the lake at a known position. The results from one such trial are shown in Figure 7.12. Sensing locations r_4, r_5, r_6 were obtained by running the Greedy strategy. The resulting 1-σ uncertainty ellipses are shown (in blue) along with the tag estimates (as red crosses). The true location of the tag is marked by a black star. The final error of this triangulation was 1.21m, with 1-σ bounds of 3.3m and 2.7m in the x and y directions, respectively.

Table 7.1: Experimental results with the ground robot for depth 2

Method	Final error	Final uncertainty
Enumeration Tree	0.97	3.53
	3.32	8.57
	5.35	6.04
Greedy	3.21	20.52
	3.29	11.93
	8.65	11.34

The field experiments demonstrate that our system is capable of localizing stationary reference tags reliably. Next, we present complete field experiments where the robot executed both the coverage and localization phases.

7.5 Field Experiments

In this section, we report results from two field tests conducted at Lake Gervais in MN, USA. In the first test, a reference tag was deployed at a known location. In the second test, we searched for tagged fish present in the lake with the boat.

Figure 7.13 shows results from the first experiment with a reference tag deployed at the location marked in Figure 7.13(b). The robot executed the coverage pattern while continuously monitoring the radio antenna on the frequency of this reference tag. After detecting signal from this radio tag at r_1 (Figure 7.13(c)), the robot switched from the search phase to the localization phase.

During the localization phase, the robot executed the Greedy strategy with $k = 2$ measurements, in addition to the two initialization measurements taken at r_1 and r_3. The measurement at r_2 is used to distinguish which side of the bearing at r_1 the fish is located by comparing their signal strengths. The robot then moved to r_4 and r_5 and obtained bearing measurements as shown. The 1-σ uncertainty ellipse after each step is also shown. After completing the localization, the robot continued to cover the rest of the regions. The robot covered a total path of approximately 2km in 49min.

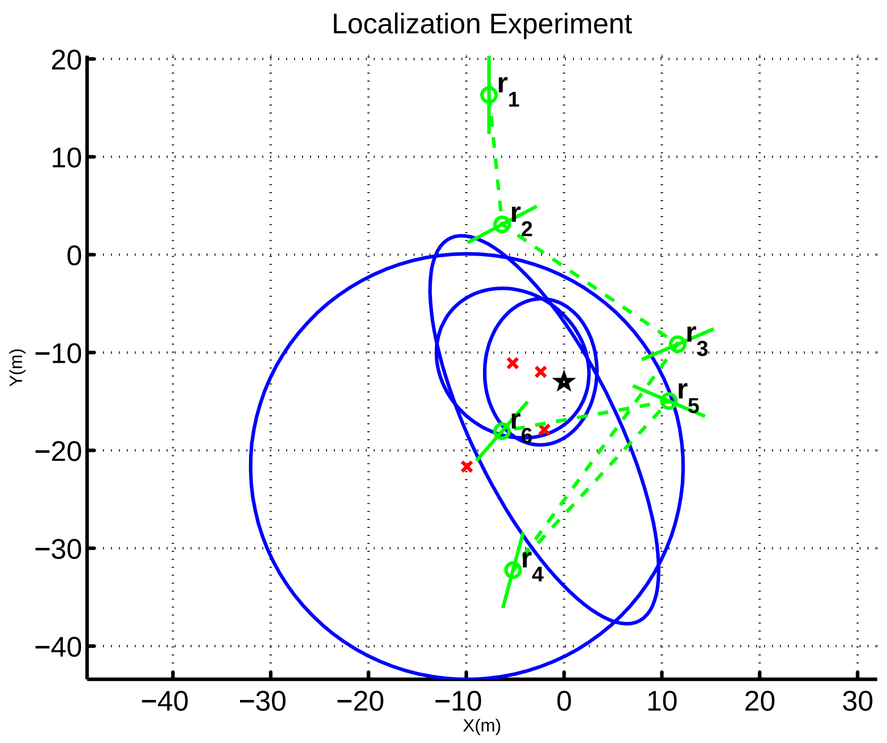
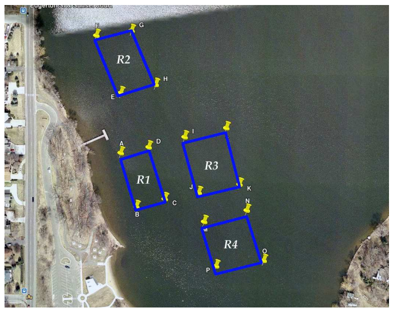
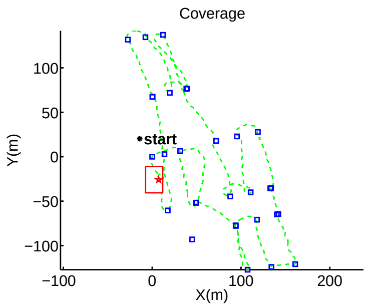


Figure 7.12: Localization experiments with the Greedy strategy. Ellipses shown encompass the 1-σ uncertainty after each measurement. We use the second measurement to disambiguate which side the tag lies from bearing obtained at r_1. Bearing measurements are shown as solid green lines.

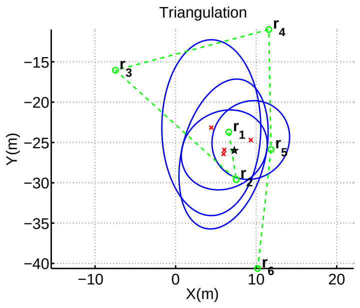
The second field trial (Figure 7.14) was conducted in the same lake without a reference tag. We programmed the robot to search for frequencies corresponding to actual radio-tagged fish present in this lake. While searching the first region, the robot detected one of the frequencies in the list, executed the localization strategy, and returned to the search plan. Figure 7.14(b) shows the coverage path followed by the robot. The red box marks the area where the robot followed the localization strategy to obtain additional bearing measurements and localize the unknown tag. Figure 7.14(c) shows a closeup of the localization phase. Since this was an actual radio-tagged fish, the ground truth is unknown.



(a) Coverage regions



(b) Coverage track of the robot.

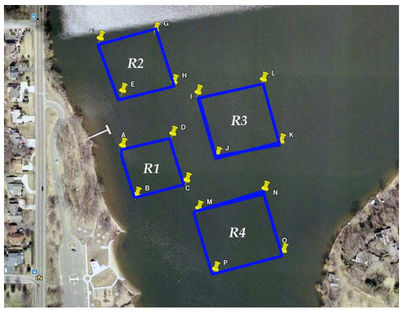


(c) Localization with reference tag.

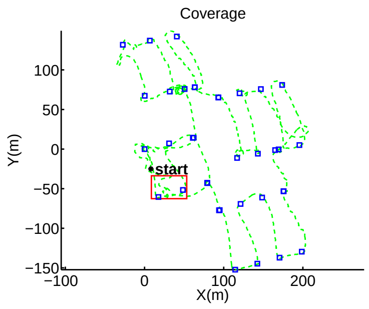
Figure 7.13: A field trial on Lake Gervais, MN. The robot covered the regions via the path shown in the middle figure. On the right, a closeup of the localization of a reference tag is shown. The true location of the reference tag is marked with a black star. The red box in the middle figure corresponds to the triangulation area on the right.

7.6 Conclusion

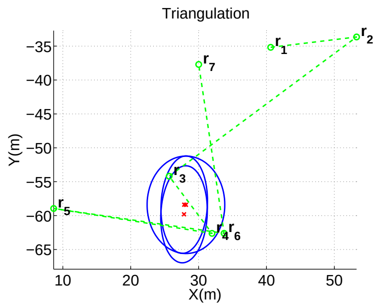
In this chapter, we presented a prototype robotic sensing system for the application of monitoring radio-tagged invasive fish. We divided the monitoring task into two subtasks of finding the tagged fish and localizing them accurately. For the first task, we presented a coverage algorithm for finding a tour whose length is at most a constant factor away from an optimal tour. For actively localizing the tagged fish, we first showed how the bearing of the tag can be estimated by using measurements obtained by rotating a directional antenna. Next we addressed the problem of actively choosing sensing locations to reduce localization uncertainty. We compared three algorithms in simulations and field experiments, and incorporated the most effective one into our



(a) Coverage regions



(b) The track of the robot during coverage.



(c) Close up of the measurements.

Figure 7.14: Field experiments at Lake Gervais, MN. Figure 7.14(a) shows the areas we wish to cover. The computed search path and the trajectory executed by the robot are shown in Figure 7.14(b). Upon detecting a tag, the robot executed a localization strategy as shown in Figure 7.14(c). Because the robot attempted to triangulate a tagged fish, we do not know the true location of the tag.

system. We concluded the chapter with additional field trials.

Chapter 8

Energy-Optimal Trajectory Planning

In previous chapters, we presented path planning algorithms to solve coverage and tracking problems. In doing so, we ignored many low-level aspects of motion planning. The output of the path planners was a series of waypoints to be followed by the robot. However, planning for the path and/or velocity to drive towards the points was left to a low-level controller that is present on most robots. In this chapter, we will focus on a low-level planning aspect. Specifically, we will show how to compute velocity profiles and paths using energy as the optimization criterion.

Energy optimization is a fundamental requirement to achieve long term autonomous deployments. One of the main bottlenecks for robots is the limited lifetime of on-board batteries. To extend the lifetime, it is critical to optimize the energy consumption of the robot, in addition to harvesting additional energy. Motion is a major source of energy consumption for mobile robots. In this chapter, we study the problem of minimizing the energy consumption by optimizing the motion of the robots.

In particular, we focus on car-like robots powered by Direct Current (DC) motors. It is well-known that the energy consumption of a DC motor depends on its angular speed and acceleration [131]. The angular speed and acceleration of the driving DC motor in turn controls the translational velocity and acceleration. We study the problem of computing paths and velocity profiles for a forward-only car-like robot that minimizes the energy consumption in a flat, obstacle-free environment.

First, we focus on the case of finding the energy optimal velocity profile when the path is given. Depending on the application, a high-level planner can specify the exact path to be followed by the robot. However, often the velocity along the path is free to be arbitrarily set. For such situations, we present a closed form solution for the velocity and acceleration profile that minimizes the energy consumption as given by our model.

Second, we consider the problem of computing the minimum energy path itself, given a start and goal position and orientation (pose) for the robot. Computing closed-form paths and velocity profiles simultaneously is difficult. Instead, we will show how to leverage the closed form solution for velocity profiles to obtain energy optimal paths leading to computational savings.

In computing minimum energy paths we face a trade-off between the length of the paths, turning radius, frictional forces, velocity and acceleration of the robot. Unlike minimum length paths, a minimum energy path may contain segments with varying turning radius (Figure 8.1). To accommodate this, we construct a graph (termed *Energy Roadmap*) which incorporates the closed-form solution for optimal velocity profiles. We show how to build this structure efficiently, and present details of an implementation. Finally, we investigate the structure of minimum energy paths found using our algorithm, and highlight instances when these paths deviate from the minimum length paths.

The rest of the chapter is organized as follows: The energy model and the formal problem statement are presented in Section 8.2. We derive the optimal velocity profiles with and without a maximum velocity bound for a path with single segment in Sections 8.3 and 8.4 respectively and for multiple segments in Section 8.5. The application of these results to simultaneously compute the minimum energy path and velocity profiles is presented in Section 8.6. Experiments on our custom-robot are presented in Section 8.7 along with a calibration procedure for estimating the parameters of the energy model in Section 8.7.1. We conclude with a discussion on the utility of our results in Section 8.8. Proofs and derivations for all the results are presented in Appendix D.

We start by discussing the related work on energy-optimal trajectory planning.

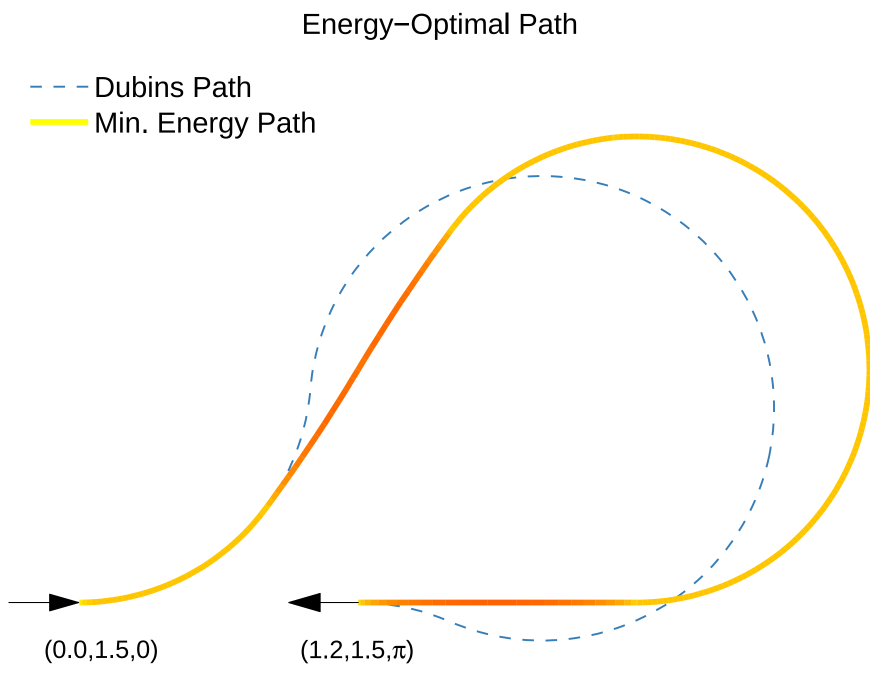


Figure 8.1: The minimum length path consists of 3 circular segments, whereas the minimum energy path consists of a straight line segment and 5 circular segments of varying radii. The optimal velocity profile along the path is given by the color in the heat map along the path. The straight line and circular segments with higher turning radius allow the robot to move at a higher speed and thus for a lesser time leading to lower energy consumption (despite being longer). We explore this trade-off between velocity, turning radius, path length and energy in this chapter.

8.1 Related Work

The classical problem of optimizing the path and velocity profiles for mobile robots while satisfying velocity and/or acceleration constraints is known as kinodynamic planning [132]. The pioneering work for finding minimum *length* paths for a forward-only car-like robot was done by Dubins [133]. Reed and Shepps [134] extended this work for a car that can go forward and backward. Balkcom and Mason [135] used an optimal control formulation to derive the time optimal trajectories for bounded velocity differential drives. Recently, Chitsaz et al. [136] used similar techniques to give the complete characterization for minimum wheel rotation paths for differential drive robots.

Existing literature on finding minimum energy paths for robots includes the work of Sun and Reif [137] who considered the problem of computing the optimal path for robots traversing a terrain. Under the assumption that the friction coefficients are known across the terrain, they showed how to compute a path that requires minimum energy to overcome frictional forces. This work generates the path but does not yield an optimal velocity and acceleration profile. Furthermore, the paths found are piecewise linear which cannot be directly applied for car-like robots.

With recent advancements in hybrid and electric vehicles technology, power management and optimization has received considerable interest in the automotive sector (see e.g. [138]). Research studies in this area target power optimization based on the users' input and driving profiles. However, there has been little work on finding energy efficient trajectories for vehicles that navigate autonomously. Energy optimal trajectory planning has also been studied for robotic manipulators. Gregory et al. [139] studied the problem of finding energy-optimal control inputs for a manipulator with two revolute joints to follow a prescribed path. Wigstrom et al. [140] studied the problem of scheduling jobs for possibly multiple industrial robots, where each job requires the robot to optimize its control profile with respect to energy and follow a prescribed path. Our work differs from this literature in that we use the kinematic and energy model for a car-like robot. In addition, we focus on simultaneously computing the energy optimal path and velocity profile along this path.

In order to compute velocity profiles, the power consumption needs to be modeled. Mei et al. [141] modeled the power consumption as a sixth-degree polynomial of the robot's speed using experimentally collected data. However, their model does not incorporate acceleration. More importantly, they use this model to compare velocity profiles but do not address the problem of computing an optimal profile.

Kim and Kim [142] computed the optimal velocity profile for a robot moving on a straight line, when the total time to travel is fixed. However, this solution does not incorporate any bound on maximum velocity of the robot. In [143], they proposed a rotational trajectory planner that minimizes the energy consumption. They do not present a systematic method to combine the solutions for translational and rotational trajectories. Thus, it is not clear if this approach yields an optimal solution. Wang et al. [144] studied the problem of finding a minimum energy trapezoidal velocity profile. As we will show shortly, a trapezoidal profile itself is not optimal in terms of total energy consumption. In addition, they do not consider any upper bound on the velocity of the robot. Further, their technique is only applicable for turn-in-place-move-forward type of motion for differential drives, and is not experimentally verified.

Broderick [145] et al. studied the problem of computing energy-efficient velocity profiles for a tracked robot. The path of the robot was computed using a boustrophedon coverage pattern and decomposed into straight-line segments and turns. The goal was to compute the velocity profiles for the left and right tracked wheels along each segment. The cost function for each segment penalized a linear combination of the control inputs, efficiency of the motors, and the fraction of area not covered by the trajectories before the start of the current segment. Based on this cost-function, the paper presented trade-offs between the control inputs and the area covered by the robots. Instead, we focus on optimizing only the control inputs (i.e., velocity and path). We describe the problem formulation next.

8.2 Problem Formulation

First consider the problem of computing the optimal velocity profile when given a path τ on which the robot will move. The instantaneous position of the robot along τ is parameterised by a single variable of time $x(t)$. The linear velocity and acceleration of the robot along this path are represented by $v(t)$ and $a(t)$ respectively. We define the state of the robot by $\mathbf{X}(t) = [x(t), v(t)]^T$. The state transition equation can be written as,

$$\dot{\mathbf{X}}(t) = \begin{bmatrix} \dot{x}(t) \\ \dot{v}(t) \end{bmatrix} = \begin{bmatrix} v(t) \\ a(t) \end{bmatrix} \tag{8.1}$$

where $a(t)$ is the control input.

We first describe the energy consumption model for the robot, before formally stating the problem.

8.2.1 Energy Model

Consider a robot with car-like steering, with forward, translational velocity provided by a DC motor. We use the model described in [131] for energy consumption in a brushed DC motor. This detailed model takes into account the energy dissipated in the resistive winding, the energy required to overcome internal and load friction and the mechanical power delivered to the output shaft. The instantaneous current $i(t)$ in the motors is given by,

$$i(t) = \frac{1}{K_T}\Big[T_F + T_L + D_f\omega(t) + (J_M + J_L)\frac{d\omega(t)}{dt}\Big] \tag{8.2}$$

and the voltage $e(t)$ across the motor is given by,

$$e(t) = i(t)R + K_E\omega(t) \tag{8.3}$$

where $\omega(t)$ is the angular velocity of the motor, K_E and K_T are back-electromotive force and torque constants, T_F and T_L are internal and load frictional torques, D_f is the internal damping, and J_M and J_L are motor and load moments of inertia.

Since linear velocity of the robot and angular velocity of the motor for a car-like robot are proportional to each other, we can rewrite Equations 8.2 and 8.3 to yield the energy consumption for traveling from $t = 0$ to $t = t_f$ as,

$$\begin{aligned} E &= \int_0^{t_f} \Big[e(t)i(t)\Big]dt. \\ &= \int_0^{t_f} \Big[c_1a^2(t) + c_2v^2(t) + c_3v(t) + c_4 + c_5a(t) + c_6v(t)a(t)\Big]dt. \end{aligned} \tag{8.4}$$

where constants $c_1, \ldots, c_6$ are combinations of the motor parameters, and $v(t)$ and $a(t)$ are the linear velocity and acceleration of the robot obtained from $\omega(t)$ and the radius of the wheel. When the initial and final velocity values are the same for τ, the net contribution by the terms corresponding to c_5 and c_6 is zero and can be ignored [131]. Hence, we can rewrite the energy model as,

$$E = \int_0^{t_f} \Big[c_1 a^2(t) + c_2 v^2(t) + c_3 v(t) + c_4\Big] dt. \tag{8.5}$$

The constants $c_1, \ldots, c_4$ depend on the motor parameters which in turn depend on the robot design and the surface on which the robot is moving. These parameters can be obtained using the calibration procedure presented in Section 8.7.1.

The robot's wheels may slip when it is making a sharp turn at a high speed. The maximum speed with which the robot can move along τ is a function of the instantaneous turning radius, the inertia of the robot and the frictional forces with the surface. We assume the maximum centrifugal force without slipping can be specified by a parameter $F_{\max}$. Thus the maximum safe translational speed without slipping will be,

$$v_m(t) = \sqrt{\frac{F_{\max} r(t)}{m}}, \tag{8.6}$$

where $r(t)$ is turning radius and m is the mass of the robot. Any other function of the form $v_m(t) = f(r(t))$ can be easily incorporated in our formulation.

8.2.2 Problem Statement

Let D be the total length of τ. The energy consumption for a velocity profile $v(t)$ traversing τ is given by Equation 8.5. The final time t_f can be fixed or kept free. The robot starts from and returns to rest over τ. This gives us the following boundary conditions,

$$v(0) = 0,\ v(t_f) = 0,\ x(0) = 0,\ x(t_f) = D \tag{8.7}$$

We study four problems of increasing generality. For the first three problems, the objective is to find a velocity profile $v(t)$ to minimize E, subject to the constraints given below:

Problem 4. *τ consists of a single segment. There is no bound on the maximum velocity of the robot, i.e., $v(t) \geq 0$ for $0 \leq t \leq t_f$. Find the optimal velocity profile $v^*(t)$ minimizing Equation 8.5 subject to state transition and boundary constraints given by Equations 8.1 & 8.7.*

Problem 5. *τ consists of a single segment. The maximum velocity of the robot over τ is bounded by constant v_m, i.e., $0 \le v(t) \le v_m$ for $0 \le t \le t_f$. Find the optimal velocity profile $v^*(t)$ minimizing Equation 8.5 subject to state transition and boundary constraints given by Equations 8.1 & 8.7.*

Problem 6. *τ consists of N segments composed of straight lines and curves. There is a separate velocity bound for each segment i given by $v_m(i)$. $v_m(i)$ is constant over the i^{th} segment. Let $D(i)$ be the distance to travel for each $1 \le i \le N$. Find the optimal velocity profile $v^*(t)$ minimizing Equation 8.5 subject to state transition and boundary constraints given by Equations 8.1 & 8.7.*

Finally, we consider the problem of computing the path τ itself. τ is specified by the steering control input $\phi(t)$ and the translational velocity $v(t)$. The robot starts at and returns to rest. We do not consider the cost of steering, and assume for simplicity that the robot can instantaneously switch the steering input. There are existing techniques [146, 147] to compute continuous trajectories for car-like robots where the rate of change of the steering input is bounded. The physical interaction between the surface and the steering wheel has also been extensively studied [148]. Since our algorithm first computes a graph (as presented in Section 8.6), smoothness constraints and the steering cost can be included while searching for the optimal solution in the graph. We assume that there are no obstacles in the environment. Many sampling-based planning algorithms that consider obstacles often require a subroutine that computes the optimal cost and path between two poses in an obstacle-free environment (see e.g. [149, 150]). Hence, we focus on the fundamental case of finding energy-optimal paths without considering obstacles, which can be used as subroutines for the general case.

Problem 7. *Given start and goal poses, compute a path τ and a velocity profile along this path for a car-like robot to minimize Equation 8.5. The velocity at all times must obey the constraint given by Equation 8.6. The robot starts at and returns to rest.*

The solutions for Problems 4, 5, 6 & 7 are presented in Sections 8.3, 8.4, 8.5 & 8.6 respectively. Problems 4 & 5 form special cases of the last two problems and provide insight into the structure of general optimal velocity profiles. We use the generalized solutions of the first two problems, with non-zero boundary conditions, as subroutines for solving Problems 6 & 7. The full proofs for subsequent lemmas and theorem are given in the appendix.

8.3 Optimal Velocity Profile without Bounds

In this section, we present the solution to Problem 4, when the path τ consists of a single section with no bound on the maximum velocity of the robot. We first state the necessary conditions and present the closed form solution for the optimal velocity profile. Then, we discuss and provide insights for the structure of the optimal profile. Finally, we compare the optimal profile with the commonly-used trapezoidal velocity profile.

8.3.1 Solution to Problem 4

When there is no bound on the maximum velocity, the Hamiltonian [151] for this problem can be obtained as,

$$H(\mathbf{X}(t), a(t), \boldsymbol{\lambda}(t), t) = c_1 a^2(t) + c_2 v^2(t) + c_3 v(t) + c_4 + \lambda_1(t) v(t) + \lambda_2(t) a(t) \quad (8.8)$$

where $\lambda_1(t)$ and $\lambda_2(t)$ are the Lagrange multipliers and acceleration $a(t)$ is the control.

The three necessary conditions for $a^*(t)$ to optimize the Hamiltonian for all times $t \in [0, t_f]$ are given as,

$$\dot{\mathbf{X}}^*(t) = \frac{\partial H}{\partial \boldsymbol{\lambda}}, \quad \dot{\boldsymbol{\lambda}}^*(t) = -\frac{\partial H}{\partial \mathbf{X}}, \quad 0 = \frac{\partial H}{\partial a} \quad (8.9)$$

Applying these necessary conditions, we can solve the resulting partial differential equations for the optimal control and states to get,

$$a^*(t) = ks_1 e^{kt} - ks_2 e^{-kt} \quad (8.10)$$

$$v^*(t) = s_1 e^{kt} + s_2 e^{-kt} - \left(\frac{c_3 + s_3}{2c_1}\right) \quad (8.11)$$

$$x^*(t) = \frac{s_1 e^{kt}}{k} - \frac{s_2 e^{-kt}}{k} - \left(\frac{c_3 + s_3}{2c_1}\right) t + s_4. \quad (8.12)$$

where $k = \sqrt{\frac{c_2}{c_1}}$ and $s_1, \ldots, s_4$ are constants.

We can solve for $s_1, \ldots, s_4$ in terms of the final time t_f by substituting the boundary conditions given in Equation 8.7 for $v^*(t)$ and $x^*(t)$. We obtain,

$$\begin{aligned}
s_1 &= -\frac{Dk}{kt_f + e^{kt_f}(kt_f - 2) + 2}, \\
s_2 &= s_1 e^{kt_f}, \quad s_3 = 2c_1(s_1 + s_2) - c_3 \\
s_4 &= -\frac{s_1 - s_2}{k}.
\end{aligned} \tag{8.13}$$

By substituting in Equations 8.10-8.12 we obtain,

$$\begin{aligned}
a^*(t) &= D\left(\frac{c_2}{c_1}\right)\left(\frac{e^{k(t_f-t)} - e^{kt}}{kt_f + e^{kt_f}(kt_f - 2) + 2}\right), \\
v^*(t) &= D\sqrt{\frac{c_2}{c_1}}\left(\frac{(1 + e^{kt_f} - (e^{k(t_f-t)} + e^{kt}))}{kt_f + e^{kt_f}(kt_f - 2) + 2}\right), \\
x^*(t) &= D\left(\frac{(e^{k(t_f-t)} - e^{kt}) - (e^{kt_f} - 1) + kt(e^{kt_f} + 1)}{kt_f + e^{kt_f}(kt_f - 2) + 2}\right).
\end{aligned} \tag{8.14}$$

Since the final time is free, it can be solved for using the additional boundary condition (known as the transversality condition) given by,

$$H(\mathbf{X}^*(t_f), a^*(t_f), \boldsymbol{\lambda}^*(t_f), t_f) = 0. \tag{8.15}$$

Substituting Equations 8.10-8.12 and 8.13 above results in,

$$(D\frac{c_2}{c_1} + 2)(1 - e^{kt_f}) + \sqrt{\frac{c_4}{c_1}}kt_f(1 + e^{kt_f}) = 0, \tag{8.16}$$

which is an equation in a single variable t_f (all other terms are constant) and can be solved using any existing solver for transcendental equations. We used MATLAB's `fzero` function. Alternatively, if the final time is fixed, we can directly substitute this given value in Equation 8.14 to find $v^*(t)$.

Figure 8.2 shows the optimal velocity profile obtained for traveling a distance of $50m$ using Equation 8.14. It can be observed that the profile consists of symmetric acceleration and deceleration curves with an almost-constant velocity region in the middle. From Equations 8.14 and 8.16, we can show that the peak velocity is reached at $t = t_f/2$ and is given by, $v^*\left(\frac{t_f}{2}\right) = \sqrt{\frac{c_4}{c_2}}\frac{\left(e^{\frac{k}{2}t_f} - 1\right)}{\left(e^{\frac{k}{2}t_f} + 1\right)}$. The corresponding optimal control profile

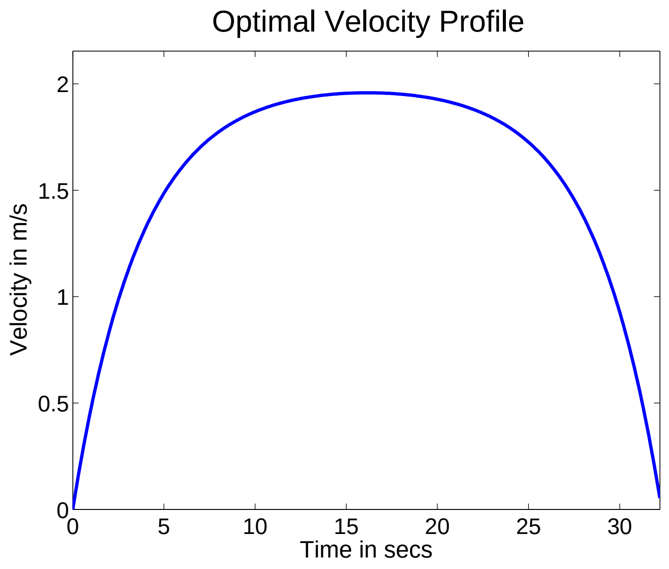


Figure 8.2: The optimal velocity profile $v^*(t)$ for a distance $D = 50m$ using $c_1, \ldots, c_4$ obtained during calibration in Section 8.7.1. The optimal profile consists of symmetric exponential curves, reaching a maximum velocity at $t = t_f/2$.

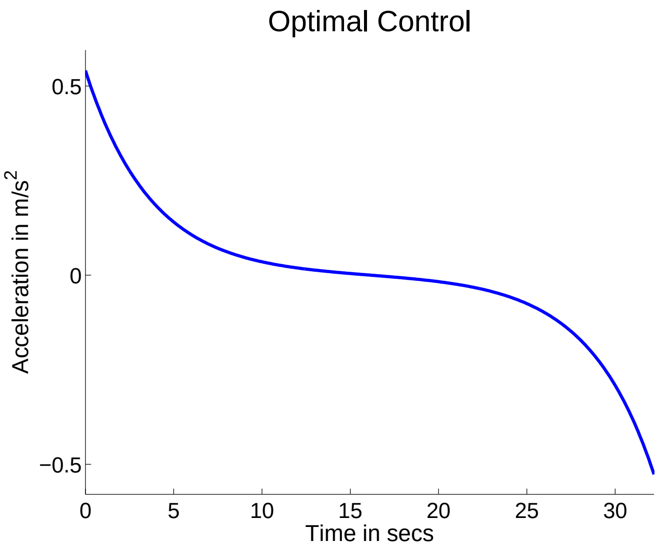


Figure 8.3: Optimal Control $a^*(t)$ obtained for traveling a distance of $D = 50m$ corresponding to the optimal velocity profile shown in Figure 8.2.

$a^*(t)$ is shown in Figure 8.3. The acceleration profile is a smooth exponentially decreasing function. The acceleration is almost zero in the middle region (exactly zero at $t = t_f/2$).

8.3.2 Structure of the Optimal Profile

The optimal velocity profile shows similar structure when the distance to travel D varies. Figure 8.4 shows the optimal velocity profiles for traveling four different distances. The optimal profile reaches the same peak velocity and does not go faster even if the distance to travel increases.

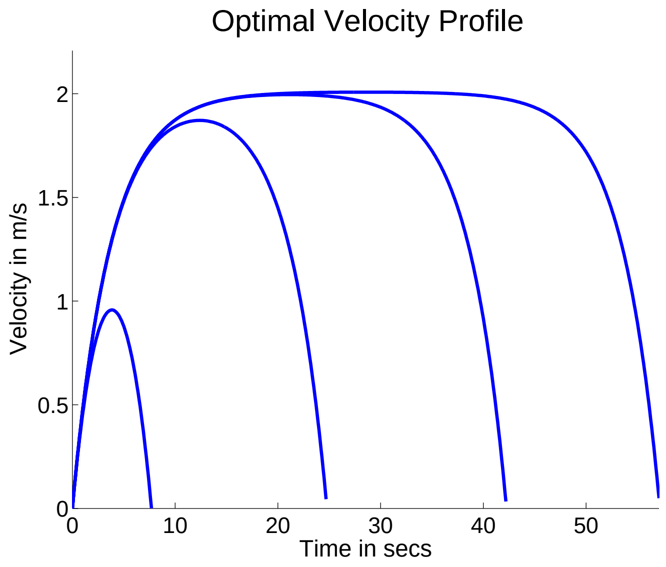


Figure 8.4: Optimal Velocity $v^*(t)$ profiles obtained for traveling distances $D = 5, 35, 70, 100m$ follow a similar structure.

From the cost function (Equation 8.5), we see that both higher velocities (through terms c_2 and c_3) and longer times (through c_4) are penalized by higher energy cost. Consider a time-optimal trajectory where the solution would be to move as fast as possible, subject to maximum acceleration and deceleration. Such a trajectory would pay a much higher instantaneous cost (through terms c_1, c_2, c_3) but integrated over a shorter time. The energy-optimal trajectory, on the other hand, achieves the optimal energy trade-off between moving faster (and consequently for a lesser time) and moving slower (and for longer times). In contrast to a time-optimal trajectory, the solution for the energy-optimal trajectory does not exceed the peak velocity of $\sqrt{\frac{c_4}{c_2}}$. The following lemma sheds light on this underlying structure for the optimal velocity profiles.

Lemma 24. *Consider an arbitrary velocity profile $v(t)$ traveling a distance D. Let the total energy consumption of $v(t)$ be E. If the given profile crosses $\sqrt{\frac{c_4}{c_2}}$ at times t_i and t_{i+1}, we can replace this section of $v(t)$, $t_i \leq t \leq t_{i+1}$ by a constant velocity section of $v_c = \sqrt{\frac{c_4}{c_2}}$, so that the resulting velocity profile covers the same distance and consumes energy less than $v(t)$.*

8.3.3 Comparisons with trapezoidal velocity profile

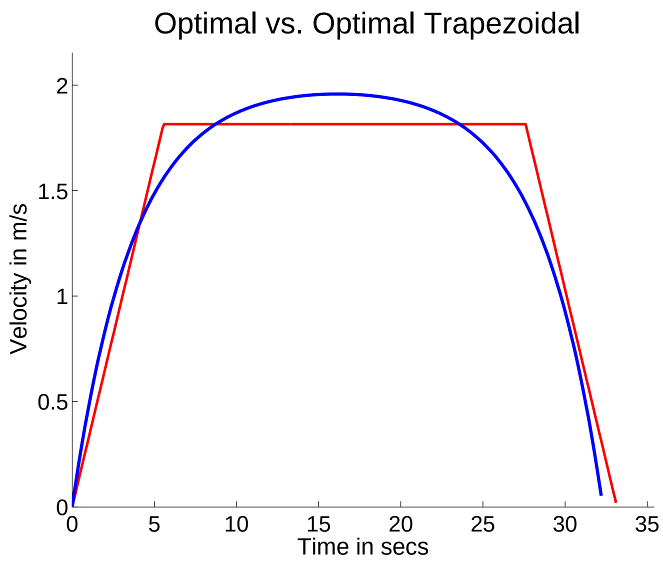


Figure 8.5: Optimal trapezoidal profile computed using the same energy function shown together with the general optimal profile for traveling $D = 50m$. The general optimal profile we compute gains higher savings with respect to the trapezoidal profile while accelerating and decelerating. This yields higher energy savings when the total distance to travel is less, a scenario commonly seen when the robot has to frequently start and stop.

A trapezoidal velocity profile is commonly used for its ease of implementation. A trapezoidal velocity profile (see Figure 8.5) consists of a constant acceleration section, followed by a constant velocity section, followed by a constant deceleration section. In [144], Wang et al. computed the optimal trapezoidal velocity profile for traveling a given distance D. However, their result is only applicable in the case when there is no bound on the maximum velocity of the robot. Figure 8.5 shows the general optimal profile and optimal trapezoidal profile computed for traveling a distance of $D = 50m$, with no maximum velocity constraints.

The general optimal profile we compute gains higher savings with respect to the trapezoidal profile while accelerating and decelerating. For example, the optimal profile yields 1.94% savings when traveling $1m$, while the savings drop to 0.32% when $D = 100m$ for the parameters calculated on our custom robot. In situations where the robot has to frequently stop, following an optimal profile would result in more energy savings and a longer lifetime. In addition, these figures are highly system-specific. The velocity profile computed in this chapter is guaranteed to minimize the energy consumption for the stated assumptions.

8.4 General Solution Incorporating Maximum Velocity Bound

The optimal profile given in Section 8.3 does not satisfy any bound on the maximum velocity imposed by the physical limitations of the robot. In this section, we solve for the optimal velocity profile for Problem 5, with a bound on the maximum velocity $v(t) \leq v_m$. We now derive the analytical solution for Problem 5 by first discussing the possible structures of an optimal profile. Depending on the value of v_m and D, the optimal velocity profile can belong to one of the following two cases.

8.4.1 Unconstrained optimal profile does not violate bound $v(t) \leq v_m$

In the case that the optimal velocity profile computed in Section 8.3 does not exceed the bound v_m, then this profile is a valid solution for the constrained case too. This happens when $v_m \geq \sqrt{\frac{c_4}{c_2}}$. Additionally, in the case when the distance to travel D is small, the optimal velocity profile may not have enough time to reach v_m or $\sqrt{\frac{c_4}{c_2}}$. We can observe this situation in Figure 8.4 when $D = 5m$.

8.4.2 Unconstrained optimal profile violates the bound $v(t) \leq v_m$

If the unconstrained optimal profile violates the bound v_m, the constrained optimal velocity profile will consist of unconstrained $\mathbf{U}$ ($v(t) < v_m$) and constrained arcs $\mathbf{C}$ ($v(t) = v_m$) joined together at *corner points*. We show that there exists an optimal profile with a $\mathbf{U}-\mathbf{C}-\mathbf{U}$ sequence (or one of its degenerate cases $\{\mathbf{U}-\mathbf{C}, \mathbf{C}-\mathbf{U}, \mathbf{C}\}$) having *corner points* at times $t = t_1$ and $t = t_f - t_2$ (degeneracy occurs when either or both of t_1 and t_2 equal to 0).

By definition, there cannot be any $\mathbf{U}-\mathbf{U}$ or $\mathbf{C}-\mathbf{C}$ sequence, as these do not include any *corner points*. Combining this observation with the following lemma, we show that the constrained velocity profile is limited to a $\mathbf{U}-\mathbf{C}-\mathbf{U}$ sequence or one of its degenerate case.

Lemma 25. *The optimal velocity profile cannot consist any sequence of the form* $\mathbf{C}-\mathbf{U}-\mathbf{C}$.

The proof, given in the appendix, follows a process similar to that in Lemma 24. We show that any $\mathbf{C}-\mathbf{U}-\mathbf{C}$ sequence can be replaced by a single $\mathbf{C}$ segment to reduce the energy consumption.

We now show how to obtain the solution for this case in closed form. Specifically, we show how to obtain $v^*(t)$ for the unconstrained and constrained arcs and compute the corner points t_1 and t_2.

We begin by writing the velocity constraint in the form of state inequality $\bar{S} = (v(t) - v_m) \leq 0$. We convert the state inequality $\bar{S}$ into a control equality $\bar{S}^{(1)}$ and interior point constraint G by differentiating $\bar{S}$ once, leading to $\bar{S}^{(1)} = \dot{v}(t) = u$ and $G = \xi(v(t) - v_m)$. The Hamiltonian is augmented with the control equality constraint between $[t_1, t_f - t_2]$ and is given by $\hat{H} = H + \mu(t)a(t)$. Here, $\mu(t)$ is the slack variable associated with the control constraint and H is given by Equation 8.8.

We use the three necessary conditions given in Equation 8.9 to obtain the optimal profile in the time interval $[0, t_1]$ and $[t_f - t_2, t_f]$. On the constraint boundary, i.e., $t \in [t_1, t_f - t_2]$, the following necessary conditions must hold [152],

$$\dot{\mathbf{X}}^*(t) = \frac{\partial \hat{H}}{\partial \boldsymbol{\lambda}} \quad \dot{\boldsymbol{\lambda}}^*(t) = \frac{\partial \hat{H}}{\partial \mathbf{X}} \quad 0 = \frac{\partial \hat{H}}{\partial a} \tag{8.17}$$

Additionally, on the two corners ($t = t_1$, $t = t_f - t_2$), the following conditions must hold for the optimal solution,

$$
\begin{aligned}
H(t_1^+) = H(t_1^-) + \left[\frac{\partial G}{\partial t}\right]_{t_1}, \boldsymbol{\lambda}(t_1^+) &= \boldsymbol{\lambda}(t_1^-) - \left[\frac{\partial G}{\partial \mathbf{X}}\right]_{t_1}^T \\
H((t_f - t_2)^+) &= H((t_f - t_2)^-) \\
\boldsymbol{\lambda}((t_f - t_2)^+) &= \boldsymbol{\lambda}((t_f - t_2)^-)
\end{aligned}
\tag{8.18}
$$

Using the conditions given above, we can solve for the optimal control and velocity profile in terms of the constants for the off-boundary exponential curves, and times t_1, t_2 and t_f. The optimal velocity profile in this case is given by,

$$
v^*(t) = \begin{cases} s_1\left(e^{kt} + e^{k(2t_1 - t)} - (1 + e^{2kt_1})\right), & 0 \le t \le t_1 \\ v_m, & t_1 \le t \le t_f - t_2 \\ s_2\left(e^{-k(t_f - t - 2t_2)} + e^{k(t_f - t)} - (1 + e^{2kt_2})\right) & t_f - t_2 \le t \le t_f. \end{cases} \tag{8.19}
$$

We can obtain the values of these constants and times using the initial and final conditions, the transversality condition given in Equation 8.15, and the interior point constraint $v^*(t) = v_m, \quad t_1 \le t \le t_f - t_2$ as,

$$
\begin{aligned}
s_1 &= -\frac{v_m}{(e^{kt_1} - 1)^2}, \\
s_2 &= -\frac{v_m}{(e^{kt_2} - 1)^2}, \\
t_1 = t_2 &= \frac{1}{k} \ln \left[\frac{\sqrt{\frac{c_4}{c_2}} + v_m}{\sqrt{\frac{c_4}{c_2}} - v_m} \right].
\end{aligned}
\tag{8.20}
$$

The final time can then be calculated by using the total distance to travel and the distances traveled in the two exponential curves.

$$
t_f = t_1 + t_2 + \frac{x^*(t_f - t_2) - x^*(t_1)}{v_m}. \tag{8.21}
$$

Figure 8.6 shows the optimal velocity profile obtained for traveling a distance of $25m$ with the maximum velocity bound set to $v_m = 1m/s$. Observe that the optimal velocity profile follows an exponential curve till it hits the boundary at $t_1 = 4.06s$ and

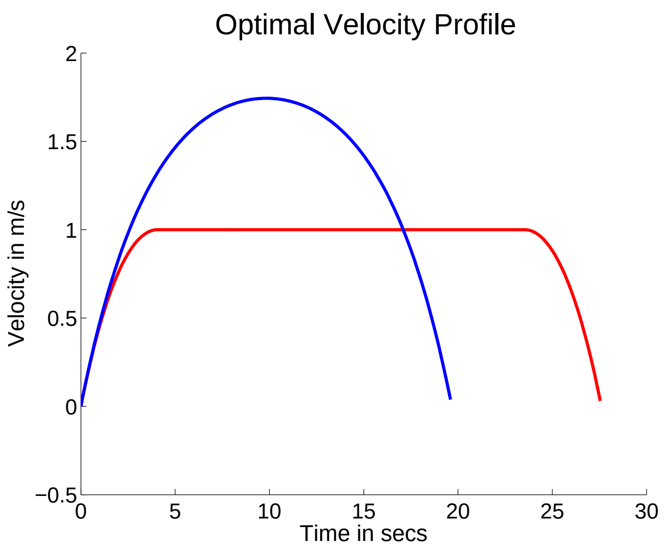


Figure 8.6: Optimal Velocity ($v^*(t)$) profile obtained for maximum velocity $v_m = 1m/s$. The constrained velocity profile consists of exponential acceleration and deceleration curves with the constraint boundary in the middle. This profile is not the same as that obtained from unconstrained solution by setting velocity to v_m wherever it exceeds.

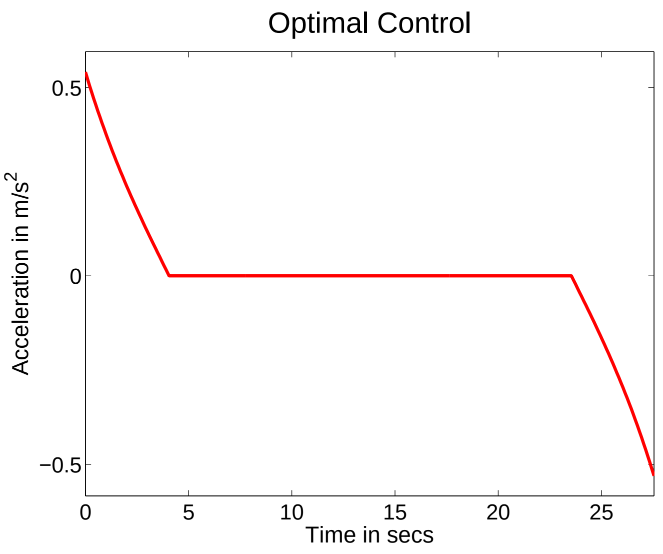


Figure 8.7: Optimal control for the case with bound on maximum velocity. Note that the control is zero whenever the velocity is on the constraint boundary (see Figure 8.6).

then stays on the constraint boundary, before following a symmetric exponential curve to zero. However, this profile is not the same as that obtained from the unconstrained solution by setting velocity equal to v_m wherever it exceeds. This unconstrained optimal velocity profile obtained from Section 8.3 is also shown in Figure 8.6. The corresponding optimal control $a(t)$ is shown in Figure 8.7. The acceleration is zero when $v(t)$ is on the constraint boundary, and follows exponential curves otherwise.

8.5 Optimal Profile over Multiple Segments

In many applications, a high level task planner is used to find the exact path to be followed by the robot. However, the velocity profile of the robot along this path is free to be optimized. We use the solution from the preceding sections to solve for the problem of finding the optimal velocity profile when the given path consists of N segments (see Figure 8.9). We restrict our attention to the case when the paths are composed of straight-line segments and constant curvature turns with possibly different turning radii. For each segment, we are given maximum allowable velocity for the robot $v_m(i)$ (see Equation 8.6) and the distance to travel $D(i)$, $1 \leq i \leq N$.

The robot initially starts at and returns to rest, however the initial and final velocity for the intermediate segments is not constrained to zero. Let $v_0(i)$ and $v_f(i)$ be the initial and final velocities for segment i. Thus, $v_0(1) = 0$ and $v_f(N) = 0$. The velocities $v_0(i)$ and $v_f(i)$ can be non-zero for all other intermediate segments. If we know the $v_0(i)$ and $v_f(i)$ that the optimal uses, we can find the entire velocity profile.

8.5.1 Velocity profile subroutines

While solving for the optimal profiles in Sections 8.3 and 8.4, we considered only zero initial and final velocity boundary conditions. Here, we extend this result for possibly non-zero v_0 and v_f as initial and final velocities, and use this extension as a subroutine for solving Problem 6. Note that in Problem 6, the first and the last segments have zero initial and final velocities respectively, and hence the energy model (which ignores the terms c_5 and c_6 because they cancel-out) remains valid (see proof in appendix).

For segments with no bound on the maximum velocity, by following a process similar to that described in Section 8.3 we get,

$$\begin{aligned}
s_1 &= \frac{(v_0 - v_f)(1 - e^{-kt_f} - kt_f) + Dk(1 - e^{-kt_f})}{e^{kt_f}(2 - kt_f) + e^{-kt_f}(2 + kt_f) - 4}, \\
s_2 &= \frac{(v_0 - v_f)(1 - e^{kt_f} + kt_f) - Dk(1 - e^{kt_f})}{e^{kt_f}(2 - kt_f) + e^{-kt_f}(2 + kt_f) - 4}, \\
s_3 &= 2c_1(s_1 + s_2) - c_3 - v_0, \\
s_4 &= -\frac{s_1 - s_2}{k}.
\end{aligned} \tag{8.22}$$

The resulting profiles can be obtained by substituting the above in Equations 8.10-8.12.

Similarly for segments with a maximum velocity bound v_m, the optimal velocity profile is given by,

$$v^*(t) = \begin{cases} s_1\left(e^{kt} + e^{k(2t_1-t)} - (1 + e^{2kt_1})\right) + v_0, & 0 \le t \le t_1 \\ v_m, & t_1 \le t \le t_f - t_2 \\ s_2\left(e^{-k(t_f-t-2t_2)} + e^{k(t_f-t)} - (1 + e^{2kt_2})\right) + v_f, & t_f - t_2 \le t \le t_f. \end{cases} \tag{8.23}$$

where,

$$\begin{aligned}
s_1 &= -\frac{(v_m - v_0)}{(e^{kt_1} - 1)^2}, \quad s_2 = -\frac{v_m - v_f}{(e^{kt_2} - 1)^2}, \\
t_1 &= \frac{1}{k}\ln\left(\frac{c_4 + c_2v_m^2 - 2c_2v_0v_m}{c_4 - c_2v_m^2} + \frac{2(c_2v_m(c_4 - c_2v_0v_m)(v_m - v_0))^{\frac{1}{2}}}{c_4 - c_2v_m^2}\right), \\
t_2 &= \frac{1}{k}\ln\left(\frac{c_4 + c_2v_m^2 - ca_2v_fv_m}{c_4 - c_2v_m^2} + \frac{2(c_2v_m(c_4 - c_2v_fv_m)(v_m - v_f))^{\frac{1}{2}}}{c_4 - c_2v_m^2}\right).
\end{aligned} \tag{8.24}$$

Figure 8.8 shows the optimal velocity profile obtained for traveling a distance of $30m$, with velocity bound $v_m = 0.4m/s$ and initial and final velocities $v_0 = 0.3m/s$ and $v_f = 0.1m/s$ respectively. Note that the acceleration and deceleration times are different in this case.

The following theorem summarizes the results for all the cases considered.

Theorem 9. *The optimal velocity profile that minimizes the energy consumption given by Equation 8.5 for a segment with distance D is given by,*

- *Equations 8.13 and 8.14 when there is no maximum velocity bound or if $v_m > \sqrt{\frac{c_4}{c_2}}$, and initial and final velocities for the segment are both zero. The final time t_f is obtained from Equation 8.15.*

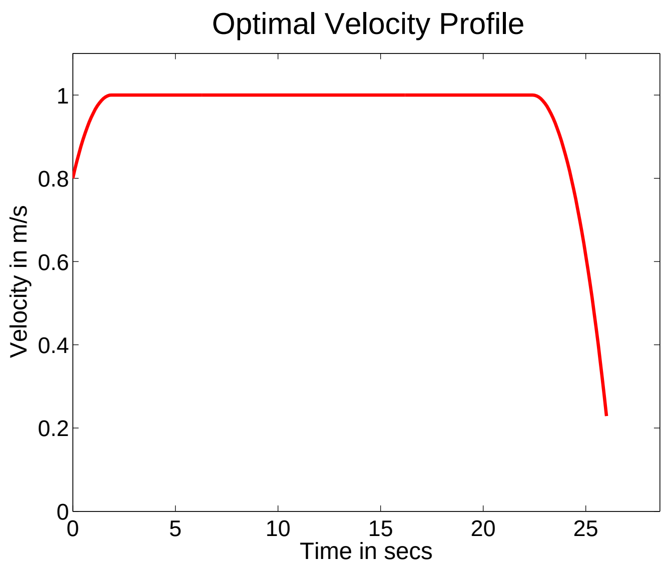


Figure 8.8: Optimal velocity profile with $v_0 = 0.3m/s$, $v_m = 0.4m/s$ and $v_f = 0.1m/s$ for traveling $30m$.

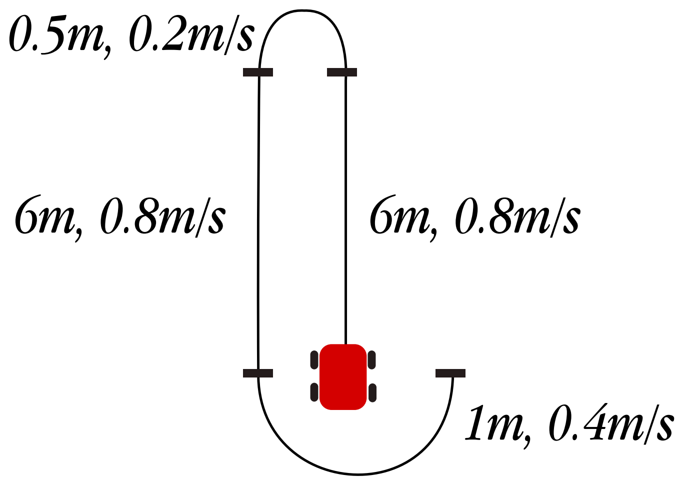


Figure 8.9: Typical path for a robot composed of two straight line segments and two turns of different radii. Segments have different maximum allowable velocities, depending on their radii.

- *Equations 8.22 and 8.14 when there is no maximum velocity bound or if* $v_m > \sqrt{\frac{c_4}{c_2}}$*, and at least one of initial or final velocities for the segment is non-zero. The final time* t_f *is obtained from Equation 8.15.*

- *Equations 8.19 and 8.20 when the maximum velocity bound* $v_m \leq \sqrt{\frac{c_4}{c_2}}$ *and initial and final velocities are both zero for the segment. The final time* t_f *is obtained from Equation 8.21.*

- *Equations 8.23 and 8.24 when the maximum velocity bound* $v_m \leq \sqrt{\frac{c_4}{c_2}}$*, and initial or final velocity is non-zero for the segment. The final time* t_f *is obtained from Equation 8.21. The initial and final velocity for the first and last segment respectively is zero.*

We can use the separate cases of this theorem as subroutines to compute the optimal velocity profile for multiple segments using dynamic programming. Note that the last case is only valid when the initial and final velocity of the first and the last segment is zero (i.e., the net effect of c_5 and c_6 is zero).

8.5.2 Dynamic Programming

Let $V_{max} = \max\{v_m(1), v_m(2), \ldots, v_m(i), \ldots, v_m(N)\}$. We then discretize the velocity space at the segment boundary into $M+1$ equal partitions $v^{(k)} = \frac{k}{M}V_{max}, 0 \leq k \leq M$. Let $C(v^{(k)}, i)$ be the cost to reach velocity $v^{(k)}$ at the i^{th} segment boundary. Let $E(v_0, v_m, v_f)$ be a function which gives the energy consumption for an optimal velocity profile in a segment starting with v_0 and ending with v_f, using the solution in Theorem 9. If either $v_0 > v_m$ or $v_f > v_m$ then the function returns the cost as $E(v_0, v_m, v_f) = \infty$.

We can then use the following recurrence for the i^{th} segment boundary:

$$C(v^{(k)}, i) = \min_{0 \leq j \leq M} \left(C(v^{(j)}, i-1) + E(v^{(j)}, v_m(i), v^{(k)}) \right), \quad 1 \leq k \leq M.$$

Since the robot initially starts from rest, we have the following,

$$C(v^{(k)}, 0) = \begin{cases} 0 & k = 0, \\ \infty & 1 \leq k \leq M. \end{cases}$$

The solution can be obtained by backtracking from $C(v^{(0)}, N)$ and finding optimal segment boundary velocity values. The optimal velocity profile can then be constructed using these optimal boundary velocity values to find individual segment profiles using Theorem 9.

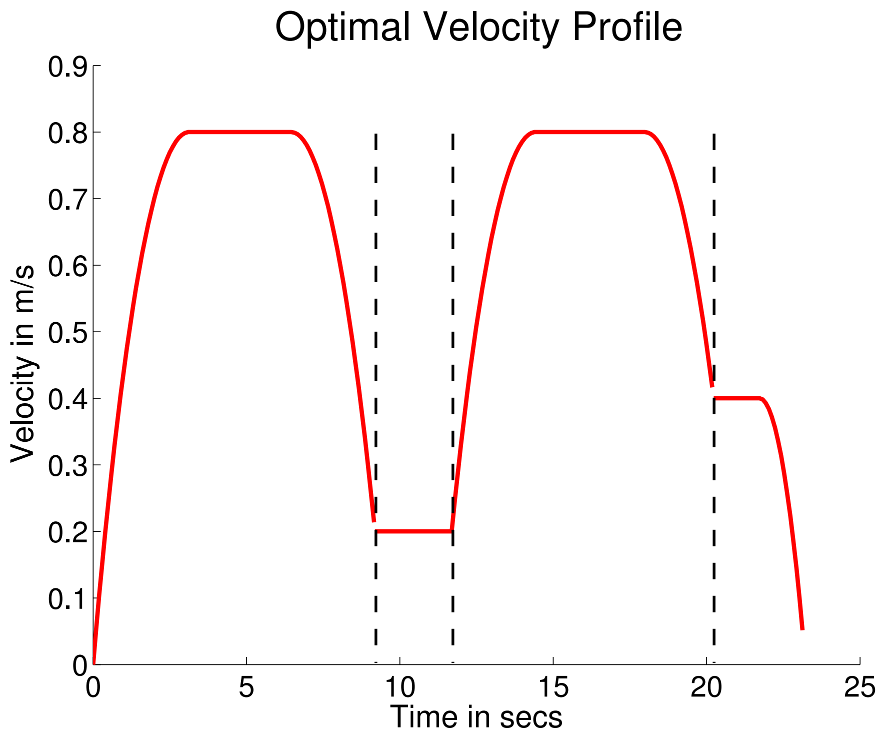


Figure 8.10: Optimal velocity profile with different bounds for different segments. The given path consists of 4 segments with bounds $v_m = \{0.8, 0.2, 0.8, 0.4\} m/s$ and distances $D = \{6, 0.5, 6, 1\} m$

Figure 8.10 shows the optimal velocity profile obtained for a path consisting of 4 segments. The velocity bounds for these segments are $v_m = \{0.8, 0.2, 0.8, 0.4\} m/s$ and the distances $D = \{6, 0.5, 6, 1\} m$ respectively. By discretizing velocity at the junction boundaries, we obtain the set of transition velocities using the recurrence given above as,

$$v_0(1) = v_f(4) = 0 m/s, \qquad v_f(1) = v_0(2) = 0.2 m/s,$$
$$v_f(2) = v_0(3) = 0.2 m/s, \qquad v_f(3) = v_0(4) = 0.4 m/s.$$

The profiles between the boundaries are computed using Theorem 9.

For building the table C, we consider $M + 1$ discretized velocities at transition boundaries of N segments. The table has size $\mathcal{O}(MN)$ and can be constructed in time $\mathcal{O}(M^2N)$. This discretization can be avoided when a segment is sufficiently long so that the robot can accelerate (or decelerate) to the bound for the next segment. In this case a greedy approach which chooses the transition velocity at the i^{th} segment boundary using the following rule suffices:

$$v(i) = \begin{cases} 0, & i = 0, \\ \min\{v_m(i-1), v_m(i)\}, & 0 < i < N, \\ 0, & i = N. \end{cases}$$

We can then use Theorem 9 to compute velocity profiles for each segment i using $v_0(i) = v(i-1)$ and $v_f(i) = v(i)$.

Using a procedure similar to that in Lemma 24 we can show that at any segment boundary, if a velocity profile decelerates further than $\min\{v_m(i), v_m(i+1)\}$, it consumes more energy than another profile that only decelerates up to $\min\{v_m(i), v_m(i+1)\}$. It can also be shown that the complete velocity profile obtained by combining profiles for each segment is optimal, when each distance $D(i)$ is large. However, when the distances are small, this strategy forces the velocity profile to achieve $v_f(i) = v(i)$ leading to higher energy consumption. The optimal solution on the other hand will reach a much lower value for $v_f(i)$. The dynamic programming solution presented here covers this possibility by incorporating all boundary velocity values.

8.6 Energy Optimal Paths

In this section, we study the problem of finding an energy optimal path and a velocity profile along this path, given a start and goal poses for a car-like robot (refer Problem 7). Dubins [133] first showed that the minimum *length* curves between two poses consists of at most three segments. Each segment is either a left or a right turn of minimum turning radius or a straight line path, and no other type. The maximum feasible speed along a curve depends on the turning radius of the robot (Equation 8.6). In the absence of any constraints on the maximum speed, we know from the discussion in Section 8.3 that the energy consumption is a monotonically increasing function of the length of the paths. This suggests that for a car-like robot capable of traveling at more than $\sqrt{\frac{c_4}{c_2}}$ at the minimum turning radius, the minimum length paths are also the minimum energy paths. The optimal profile for such paths will be those given in Section 8.3.

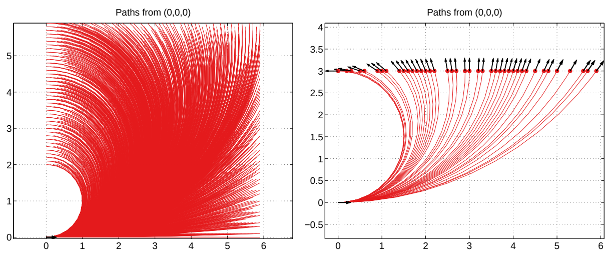
In general, computing the energy optimal paths cannot be decoupled from finding the velocity profiles. The structure of the minimum energy paths will depend on the trade-off between turning radii $r(t)$, maximum feasible speed as a function of turning radii $v_m(t)$, the length of the path and the energy parameters. While finding a general solution where the turning radius varies continuously in time seems difficult, we find an approximate solution by restricting the robot to move along a sequence of constant curvature paths.

To find such a path, we build a weighted graph (which we term as the *Energy Roadmap*) $\mathbf{G}(\mathbf{V}, \mathbf{E})$, where each vertex represents a discretized pose and velocity, i.e. $\mathbf{V} = \{(x, y, \theta, v)\}$.[1] We add an edge between two vertices $\mathbf{v_i} = (x_i, y_i, \theta_i, v_i)$ and $\mathbf{v_j} = (x_j, y_j, \theta_j, v_j)$ if (i) there exists a (directed) circular arc (or straight line) from (x_i, y_i, θ_i) to (x_j, y_j, θ_j), and (ii) v_i and v_j are both less than or equal to the maximum feasible speed along this circular arc. The weight on the edge from $\mathbf{v_i}$ to $\mathbf{v_j}$ is set to the energy for the minimum energy velocity profile along this circular arc with start and end speeds set to v_i and v_j. The energy is computed using the result in Theorem 9.

The minimum energy path from the start and goal vertex can then be computed by any shortest path algorithm on $\mathbf{G}$, e.g. A* search. The shortest (minimum energy) path will be a sequence of poses and discretized velocities; the entire robot path can be obtained by connecting the sequence of poses with circular arcs or straight line segments and the optimal profile along the path can be obtained by applying Theorem 9 to the corresponding sequence of velocities.

In the Energy Roadmap, although the poses are discretized, we allow connecting any two poses with a circular arc (Figure 8.11). Note that we do not impose any grid connectivity or fixed radius turns. Further, although we discretize velocities at a pose, we use the optimal energy profiles leveraging Theorem 9 to interpolate the velocity

[1]In this section, x refers to the X-coordinate of the robot, and not the parametric position of the robot along a path as used in the preceding sections.



(a) (b)

Figure 8.11: In the Energy Roadmap we connect any two discretized poses by a circular path, if it exists. (a) All possible circular paths starting from $(0, 0, 0)$. The minimum turning radius is set to $1m$. A total of 2254 paths exists from $(0, 0, 0)$ using side resolution of $0.1m$ and orientation resolution of $\frac{\pi}{64}$. (b) Paths starting from $(0, 0, 0)$ reaching all discretized vertices with $(x, 3, \theta)$. There exists a unique circular path starting from a given pose reaching a given *position*.

between two vertices (as opposed to enforcing any fixed profile).

The complete algorithm is presented in Algorithm 4. The main subroutines `GetMinEnergy` and `GetMinEnergyProfile` are applications of Theorem 9. The subroutine `GetPath` finds the directed circular arc or straight line path. The rest of the subroutines are obvious from their names.

If $|X|$, $|\Theta|$, $|V|$ are the number of discretized positions, orientations and velocities respectively, then the Energy Roadmap has $|\mathbf{V}| = |X| \cdot |\Theta| \cdot |V|$ vertices. Checking for a feasible path between every pair of vertices would require $\mathcal{O}(|X|^2|\Theta|^2|V|^2)$ checks. Instead we can reduce the number of checks to $\mathcal{O}(|X|^2|\Theta|)$ by observing that there is exactly one circular arc or line from a given pose (x_i, y_i, θ_i) to a position (x_j, y_j) as shown in Figure 8.12. Hence, we only check each pose with every other position for a feasible path (Lines 4–6 in Algorithm 4). Looking up a vertex from a pose or position while adding the edge (`FindVertex` in Lines 12–13) can be done in constant time by maintaining a map of pointers.

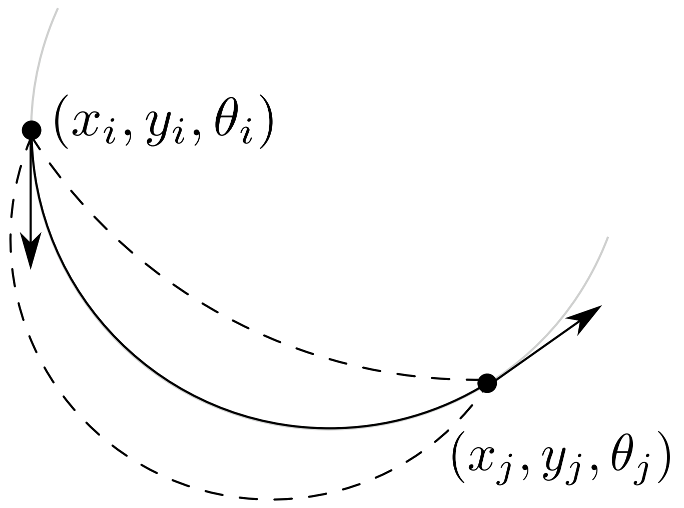


Figure 8.12: There exists only one circle passing through a pose (x_i, y_i, θ_i) and a position (x_j, y_j). All other circular arcs (shown dashed) passing through the same pair of points will not have a tangent aligned along θ_i at (x_i, y_i). Hence, in building the Energy Roadmap, instead of searching over all pairs of poses ($\mathcal{O}(|X|^2 \cdot |\Theta|^2)$), we search over pairs of poses and positions ($\mathcal{O}(|X|^2 \cdot |\Theta|)$).

8.6.1 Implementation

We implemented[2] the algorithm in C++. We used the GNU Scientific Library [153] to perform numerical integration in computing energy and for solving the transversality condition given in Equation 8.15. To find the Dubins' paths, we used the Open Motion Planning Library [154]. Our implementation makes the following optimizations to reduce the runtime and storage requirements:

- In general, the number of edges in **G** can be $\mathcal{O}(|X|^2|\Theta||V|^2)$. For a fine discretization, the storage can become prohibitively high. We reduce the storage requirement to $\mathcal{O}(|X||\Theta||V|^2)$ by observing that the paths between two poses are invariant to rotation and translation in the plane. Hence, instead of computing and storing edges between all possible pairs of vertices, we initially create a lookup table consisting of outgoing edges from $(0, 0, \theta, v_0)$, for all $\theta \in \Theta$ and $v_0 \in \mathbf{V}$ to all other vertices. While finding the shortest path using A* search, each time a new

[2]Code is available to download from `http://rsn.cs.umn.edu/index.php/Downloads`

vertex, say (x, y, θ, v), is discovered, we first transform all other vertices in the relative coordinate frame centered at (x, y). We can then extract its neighbors by looking up the relative coordinates in the table. This approach trades the running time of the search phase with the running time for building the Energy Roadmap (Lines 4–18) and the storage required for the Energy Roadmap.

- To further speed up the A* search, we use a lower bound on the energy as a heuristic function. For any vertex $(x_i, y_i, \theta_i, v_i)$, a lower bound on the energy to reach the goal $(x_t, y_t, \theta_t, v_t)$ can be computed as $c_4 \frac{d}{v_m}$ where d is the Euclidean distance between (x_i, y_i) and (x_t, y_t).

A discretization of $0.1m$, $\frac{\pi}{64}$rad and $0.5m/s$ was used for finding minimum energy trajectories. Energy parameters $c_1, \ldots, c_4$ were set to 1, $F_{\max} = 0.05$, $m = 1$ and minimum turning radius was set to $1m$ for each instance. The graph, thus created consisted of 6.4M vertices. The lookup table to store potential edges (as described above) used 10GB memory. Computing the minimum energy path typically took under 15mins on a 3.0GHz computer. To find the optimal velocity profile along the Dubins' path a resolution of $0.02m/s$ was used for dynamic programming.

8.6.2 Comparison with Dubins' Paths

Figure 8.13 shows the energy-optimal paths and velocity profiles obtained using Algorithm 4 for four start and goal poses. These four instances are representative of the trade-off between the turning radius, maximum velocity and energy. Figure 8.13(a) shows an instance where the Dubins' path consisted of three consecutive circular segments of minimum turning radius. The maximum allowable speeds along turns of minimum radii using Equation 8.6 was $0.22m/s$. Hence, the optimal velocity profile along the Dubins' path (right column) was forced to move at a slower speed, for a longer time consequently paying a higher energy cost. On the other hand, the optimal path consisted of a straight line segment and turns with greater turning radii, allowing the robot to move at a higher velocity. The resulting path, although longer than the Dubins' path, takes a lesser amount of time to travel and pays a lower energy cost.

Figure 8.13(b) shows an instance where the minimum energy path does *not* contain a straight line segment, whereas the Dubins' path does. Both paths begin and end with circular segments of minimum turning radius. The minimum energy path, however, spends lesser time on the minimum turning radius segments and switches to segments with higher turning radius (consequently lower energy) in the middle. We can observe that one of the characteristics of minimum energy paths is to avoid turns with minimum turning radius. Figure 8.13(c) shows an instance where the minimum energy path does not contain any segment of minimum turning radius.

We observed that as the length of the minimum radius turns becomes smaller than length of the straight line segment of the Dubins' path, the energy overhead of traveling at slower speeds decreases. Figure 8.13(d) shows one such instance.

It must be noted that these observations are a function of the system parameters. For example, if the robot is capable of moving at very high speeds at minimum turning radius, then the minimum energy path will coincide with the minimum length paths. Nevertheless, Algorithm 4 will find the minimum energy path, subject to the discretization.

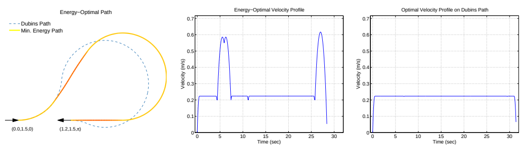
8.7 Calibration and Experiments

To test the validity of our results, we performed experiments using our custom-built robot. We first describe a simple procedure to find the energy model (Equation 8.5) of the robot for a given flat surface.

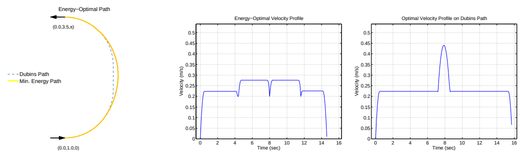
8.7.1 Calibration

We use a custom-built robot (see Figure 8.14) for experiments. Two DC motors with their output shafts coupled together through a gearbox drive the robot. The robot has car-like steering controlled by a servo motor through a fixed steering rod (unlike Ackermann steering). We use separate batteries to drive the DC motors and power the rest of the electronics on the robot.

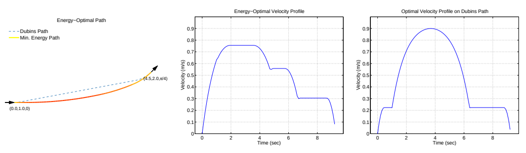
Our method utilizes a simple current and voltage measurement circuit (Figure 8.14) connected between the output of the motor and the motor driver circuit. This circuit measures the current flowing through and the voltage across the motor. An optical encoder installed on one of the robot's wheels measures its linear velocity. In the calibration procedure described next, we fix the steering of the robot so that it drives in a



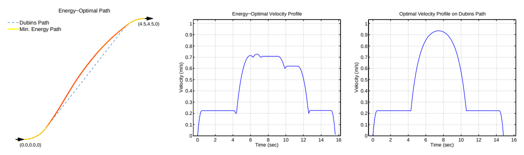
(a) Energy-Optimal trajectory $(39.4J, 7.6m)$. Dubins' path with energy-optimal velocity profile $(40.1J, 6.9m)$. Dubins' path consists of C-C-C segments, whereas the minimum energy path found has 1 straight line and 5 circular segments.



(b) Energy-Optimal trajectory $(19.1J, 3.7m)$. Dubins' path with energy-optimal velocity profile $(20.7J, 3.6m)$. The Dubins' path consists of C-S-C segments, whereas the minimum energy path found consists of 4 circular segments.



(c) Energy-Optimal trajectory $(17.4J, 4.7m)$. Dubins' path with energy-optimal velocity profile $(17.8J, 4.6m)$. Dubins' path consists of C-S-C segments, whereas the minimum energy path found consists of 1 straight line initially and 3 circular segments.



(d) Energy-Optimal trajectory $(26.29J, 6.55m)$. Dubins' path with energy-optimal velocity profile $(26.35J, 6.59m)$. The Dubins' path consists of C-S-C segments, whereas the minimum energy path found consists of 1 straight line and 5 circular segments.

Figure 8.13: The left column shows the energy-optimal paths found using Algorithm 4. The color profile along the path indicates the optimal velocity profile, also shown in the middle column. The dashed path is the minimum length Dubins' paths. The energy-optimal velocity profiles along the Dubins' paths (using the dynamic programming presented in Section 8.5) are shown in the right column. Energy parameters $c_1, \ldots, c_4$ were set to 1, $F_{\max} = 0.05$, $m = 1$ and minimum turning radius was set to $1m$ for each instance. A discretization of $0.1m$, $\frac{\pi}{64}$rad and $0.5m/s$ was used for finding minimum energy trajectories. Resolution of $0.02m/s$ was used for dynamic programming.

Algorithm 4: Minimum Energy Trajectories

```
Input: $s, t$: Start and goal pose
Data: $X, \Theta, V$ discretized positions, orientations, speeds
Output: $\{\phi(t), v(t)\}$: Steering angle and translational velocity profiles.
1  $P \leftarrow \{(l \in X, \theta \in \Theta)\}$                    /* discretized poses */
2  $\mathbf{V} \leftarrow \{(p \in P, v \in V)\}$                     /* vertices */
3  $\mathbf{E} \leftarrow \emptyset$
   /* There exists exactly one circle/line through given pose &
      position                                                  */
4  forall the $p \in P$ do
5      forall the $l \in X$ do
6          $(\theta, \mathrm{len}, \mathrm{rad}) \leftarrow$ GetPath $(p, l)$
7          if $\theta \in \Theta$ then
8              $v_m \leftarrow$ GetMaxVel $(\mathrm{rad})$
9              forall the $v_0 \in V$ $AND$ $v_0 \leq v_m$ do
10                 forall the $v_f \in V$ $AND$ $v_f \leq v_m$ do
11                     $E \leftarrow$ GetMinEnergy $(\mathrm{len}, v_0, v_f, v_m)$
12                     $\mathbf{v_i} \leftarrow$ FindVertex $(p, v_0)$
13                     $\mathbf{v_j} \leftarrow$ FindVertex $(l, \theta, v_f)$
14                     $\mathbf{E} \leftarrow \mathbf{E} \cup$ Edge $(\mathbf{v_i}, \mathbf{v_j}, E)$
15                 end
16             end
17         end
18     end
19 end
20 $s \leftarrow$ FindVertex $(s, 0)$
21 $t \leftarrow$ FindVertex $(t, 0)$
22 Path$\leftarrow$A* Search $(\mathbf{V}, \mathbf{E}, s, t)$
23 $\phi(t) \leftarrow$GetSteering (Path)
24 $v(t) \leftarrow$GetMinEnergyProfile (Path)
25 return $\{\phi(t), v(t)\}$
```

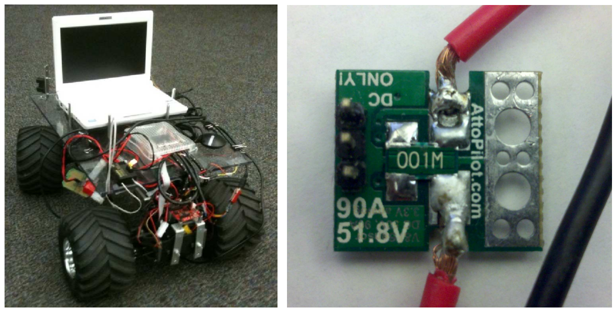


Figure 8.14: **Left:** Custom-built robot used in our experiments. **Right:** Attopilot voltage and current measurement circuit from SparkFun Electronics.

straight line.

We can write Equations 8.2 and 8.3 as,

$$\begin{aligned} i(t) &= b_1 + b_2 v(t) + b_3 a(t), \\ e(t) &= b_4 + b_5 v(t) + b_6 a(t) \end{aligned} \tag{8.25}$$

where $b_1, \dots, b_6$ are linear combinations of the internal parameters of the motors. The calibration procedure to obtain the energy parameters consists of the following steps:

STEP 1: Drive the robot at a constant velocity (v_{set}) for some time interval (we used $10s$ in our calibration experiments). Log the current and voltage across the motor. Repeat for different v_{set} values ranging from the minimum to the maximum achievable velocity for the robot. Figure 8.15(a) shows some of the actual profiles obtained during calibration for v_{set} from $0.5m/s$ to $2.5m/s$.

STEP 2: Compute the average current and voltage for each of the above trials disregarding the initial acceleration phase. Using Equation 8.25, we can find the parameters b_1, b_2, b_4 and b_5 using least-squares linear fitting to the data (see Figure 8.15(b and c)).

STEP 3: To find the remaining two terms b_3 and b_6 in the model, program the robot to drive from rest at various set acceleration values a_{set} to reach some velocity value (we used $1.6m/s$ for our system, see Figure 8.15(d)).

STEP 4: Compute the values of b_3 and b_6 by substituting a_{set} and b_1, b_2, b_4 and b_5 values obtained above in Equation 8.25 and taking the average of all the readings.

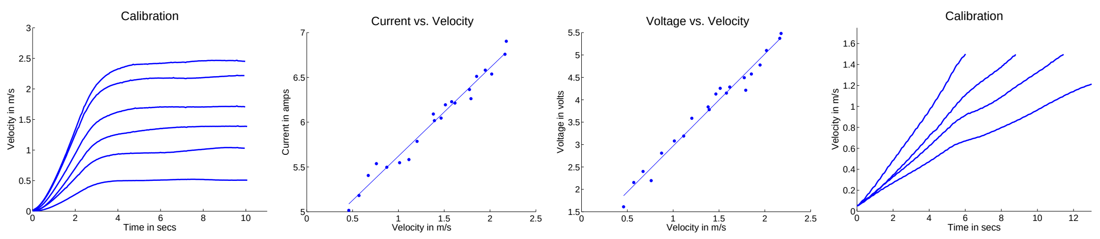


Figure 8.15: Figures obtained during calibration on the corridor surface. Left to right: **(a)** The robot initially accelerates from rest to various set velocity values. We compute the average current and voltage for the region where the robot moves at v_{set} (STEP 1). **(b)** Current consumption as a linear function of the velocity, when the motor is not accelerating (STEP 2). **(c)** Voltage applied to the motor as a linear function of the velocity, when the motor is not accelerating (STEP 2). **(d)** Calibration procedure to determine the parameter c_1 in the energy model. We accelerate the robot with various set acceleration values a_{set} while logging current and voltage values (STEPS 3 and 4).

STEP 5: Finally, calculate the required parameters $c_1, \ldots, c_4$ in Equation 8.5 using $c_1 = b_3 b_6$, $c_2 = b_2 b_5$, $c_3 = b_1 b_5 + b_2 b_4$, and $c_4 = b_1 b_4$.

Table 8.1: Energy model parameters (*SI units*) obtained using the calibration procedure.

Surface	c_1	c_2	c_3	c_4
Corridor	17.75	1.16	10.46	4.70
Concrete	5.47	0.77	10.10	4.24
Grass	8.10	5.28	28.01	25.07

Using the above procedure, we calibrated our robot on three surfaces: indoors on a corridor and outdoors on concrete and grass. The corridor surface was flat whereas the two outdoor surfaces had uneven terrain, the grassy area more so. Figure 8.15 shows plots for the complete calibration procedure with the corridor surface. Figure 8.16 shows the current and voltage plots for the grass surface. The current consumption and voltage required for driving the robot are higher for grass than for the corridor surface. Since the surfaces outdoors are uneven, the plots contain more noise than those for corridor. The model parameters computed for all surfaces are shown in Table 8.1.

8.7.2 Experiments

We conducted experiments on the smooth corridor surface to experimentally validate the optimal velocity profiles found in Section 8.5 and compared with two other profiles. We first computed the analytical solution for the velocity profile to travel the given distance. We then sampled this profile at $10Hz$ and stored the values in a look-up table.

Figure 8.17 shows the optimal profile computed using the dynamic programming solution presented in Section 8.5, for a given path three segments with distances $D = \{10, 3, 10\}m$ and maximum velocity constraints as $v_m = \{1, 0.2, 1\}m/s$. The computed velocity profile is shown as a dashed curve. The total energy consumed over the entire profile was $595J$. The actual profile executed has small deviations arising due to noise and disturbances on the surface. In this work, we pre-compute the optimal trajectory for the robot. A useful extension to this could be to design an optimal velocity feedback controller which minimizes the energy consumption.

We compare the energy consumption of our optimal profile with two commonly-used trapezoidal profiles. We chose the maximum speeds for these profiles as $1m/s$ and $2m/s$, so that the robot covers the same distance taking more and less time than the optimal respectively. We perform these comparisons for $D = 20m$ and $D = 45m$.

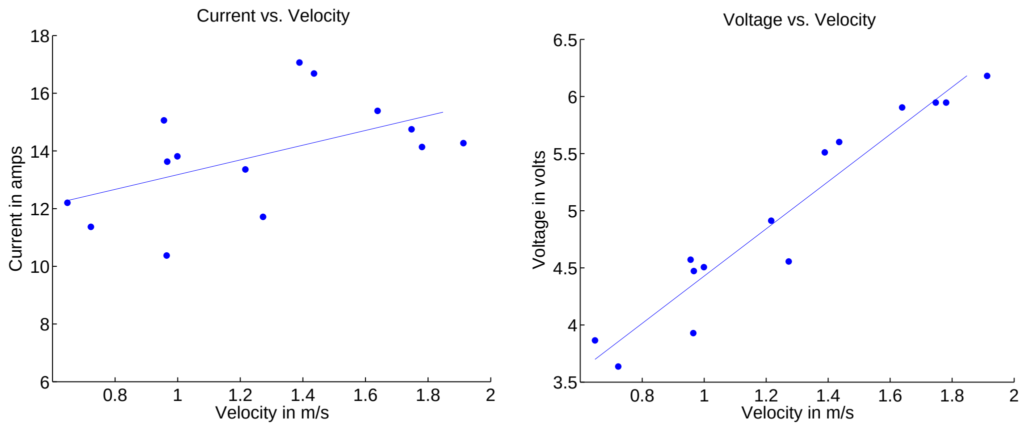


Figure 8.16: Current and voltage as a function of velocity, for the grass surface outdoors. Since the surface outdoors is not flat, the plots contain more noise than the corridor surface (Figure 8.15(b) & (c))

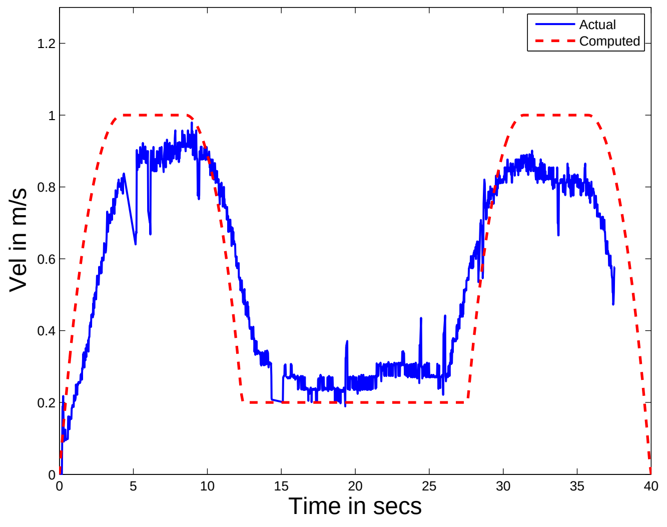


Figure 8.17: Optimal velocity profile executed by the robot for multiple segments. The dashed curve shows the optimal profile computed using the dynamic programming solution for segments with $D = \{10, 3, 10\}m$ and maximum velocity constraints $v_m = 1, 0.2, 1m/s$.

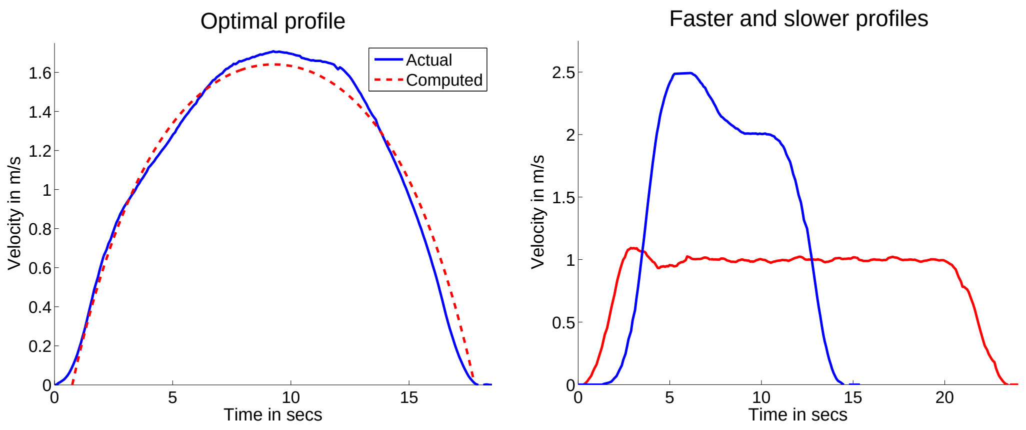


Figure 8.18: **Left:** Optimal velocity profile executed by the robot for traveling $20m$ in $18.4s$ while consuming $296J$ energy. The optimal profile is shown as dashed. **Right:** Sub-optimal velocity profiles executed by the robot for traveling $20m$ at maximum set velocities of $1m/s$ and $2m/s$. The energy consumption for these profiles is $303J$ and $319J$.

Figure 8.18 shows the optimal, slower and faster velocity profiles executed by the robot in the corridor. The optimal profile computed is also shown in Figure 8.18 as dashed. Table 8.2 shows the comparison of the energy consumption for all the trials conducted. As we can observe, the optimal profile consumes lesser energy than the two sub-optimal profiles. Also, the energy savings become more significant as the distance traveled increases.

Table 8.2: Energy consumption during experiments. The numbers in parentheses indicate the percentage of extra energy consumption with respect to E_{opt}.

D (m)	$E_{opt}(J)$	$E_{slow}(J)$	$E_{fast}(J)$
20	296	303 (2.4%)	319 (7.8%)
45	656	694 (5.8%)	696 (6.1%)

8.8 Conclusion

Optimizing the energy consumption is important for long-term deployments of robotic sensor systems. In this chapter, we studied the problem of computing trajectories for a car-like robot so as to minimize the energy consumed while traveling on a flat surface. We separately considered the problem of computing the energy optimal velocity profiles, and that of simultaneously computing the energy optimal paths and velocity profiles. We presented closed form solutions for the velocity profiles for two cases: no constraints on the robot's speed, and a single upper-bound on the speed. For the general problem of computing both paths and trajectories, a discretized graph search algorithm that leverages our closed form solution for optimal velocity profiles was presented. Using an implementation of this algorithm, we investigated the structure exhibited by minimum energy paths and highlighted instances when these paths differ from the minimum length (Dubins') paths. The closed-form velocity profiles and the obstacle-free trajectories can be used as subroutines by sampling-based planners for computing trajectories in the presence of obstacles. In addition, we presented a calibration procedure for obtaining robot's internal energy parameters. We demonstrated the utility of the calibration procedure and the algorithms through experiments performed on a custom-built robot.

Chapter 9

Conclusion and Discussion

In this dissertation, we studied two main types of robotic sensing problems: coverage and target tracking. The goal for coverage problems was to ensure each point in the environment (or a given region-of-interest) is covered by one or more sensors. The notion of covering a point depends on the specific application. We investigated coverage problems motivated by applications such as visual inspection and video-conferencing where seeing an object from all sides is important, as well as applications such as surveillance and location-aware services where locating an object is important. In the target tracking task, our goal was to sense only those regions in the environment which are likely occupied by one or more targets. The goal was to design motion strategies for the robots in order to track these regions as the targets move. We presented algorithms with theoretical performance guarantees (often in the form of constant-factor approximation algorithms) and described the design of prototype robotic systems developed for two practical applications.

In the following, we will summarize our contributions and highlight some follow-up research questions and open problems. We will then conclude with a broader discussion of future research for robotic sensing systems.

9.1 Summary of Contributions and Open Problems

In Chapter 3, we studied the problem of covering all points in a polygonal environment with omnidirectional cameras (called guards). A point was said to be covered if it satisfied the so-called $\triangle$-guarding constraint: the point is seen by one or more guards and it lies within the convex hull of the guards that see the point. This constraint ensures that any convex object within the environment will be inspected from all sides. This is a typical requirement in applications such as surveillance and video-conferencing. We showed that $\Omega(\sqrt{n})$ guards are necessary for $\triangle$-guarding any n–sided polygon. The bound is tight for polygons with holes. We conjecture that for polygons without holes $\Omega(n)$ guards are always necessary.

For the optimization problem of placing guards, we presented a log factor approximation for $\triangle$-guarding the interior with vertex guards. We then formulated the following version for which we presented a constant factor approximation: We are given a set of chords (representing for example paths a person may take) and our goal is to $\triangle$-guard one point per chord. This constant factor approximation is particularly encouraging since few such results exist for visibility-based coverage problems. An immediate research question is whether we can obtain similar results for other types of input regions such as convex subpolygons instead of chords, and for the standard notion of visibility (without $\triangle$-guarding constraint).

In Chapter 4, we studied the problem of coverage with bearing sensors in order to precisely localize a target anywhere in the environment. Each sensor measures a bearing corrupted by an unknown, but bounded amount of noise. Measurements from all sensors are combined in order to find an estimate for the target's location. We analyzed the trade-off between the number of sensors and worst-case uncertainty in a simple, obstacle-free environment. In particular, we showed that at most nine times as many sensors in a triangular grid placement achieve at most six times the uncertainty of an unknown optimal placement.

Future work would be to show whether a triangular placement is optimal. There is some evidence in this direction. Yfantis et al. [65] showed a triangular grid placement is more efficient than a square placement. Future work would also include addressing sensing limitations such as visibility constraints and placement in complex environments, as in the case of Chapter 3.

In Chapter 5, we studied the problem of tracking mobile targets using a team of aerial robots. We consider the scenario where each robot is carrying a camera that is facing downwards. The quality of tracking a ground target is a function of the distance between the robot and the target. A major difference between visibility-based coverage (considered in Chapter 3) and visibility-based tracking is that it may not be feasible to track all targets with a limited number of robots. We show a negative result in this direction. We show that k robots may not be able to track $n > k$ targets while maintaining some bound on the quality of tracking. Our proof relies on the existence of a sufficiently large environment. Future work would be to study tracking in bounded environments.

We then formulated the problem of tracking the maximum number of targets or the maximum quality of tracking as a variant of the maximum k–coverage problem. It has been shown [73] that a simple greedy algorithm achieves a 1/2 approximation for this variant which we adapted for our problem. One avenue of extending this work is to seek better approximation guarantees, perhaps by restricting the motion of the targets (e.g., bound on their maximum speed). Future work would also include investigating the problem under inter-robot communication constraints.

In Chapters 6 and 7, we studied coverage and tracking problems motivated by two practical environment monitoring applications: precision agriculture and autonomously monitoring radio-tagged fish, respectively. We described prototype systems developed for these applications and studied coverage and tracking problems motivated by practical constraints. In Chapter 6, we introduced a new coverage problem termed SamplingTSPN, where we want to obtain samples in a set of disks in the plane and the objective is to minimize the total travel and sampling time. We presented a $\mathcal{O}\left(\frac{r_{\max}}{r_{\min}}\right)$ approximation algorithm for this problem. In Chapter 7, we introduced a new coverage formulation where the goal is to visit and cover a set of regions scattered in the plane. We showed how algorithms for TSPN and coverage can be combined to yield constant factor approximations for our problem.

An immediate extension of our work is to develop planning algorithms with multiple robots to improve the coverage and tracking performance. Additional issues faced when building such a system (e.g. communication and coordination) must be addressed. Future work includes studying the online version of the coverage problems. In the online version, new points are likely to appear while executing the coverage tours. There has been some work on online TSP [155, 156], but online problems for TSP with neighborhoods are currently open.

In Chapter 8, we studied the problem of optimizing the motion of car-like robots in order to minimize their energy consumption. Starting with a known model for energy consumption of DC motors, we showed how to compute energy-optimal velocity profiles in closed-form when the path is given. We also presented a technique to compute energy-optimal paths and velocity profiles which uses the closed-form velocity solution as a subroutine. Extending this work to take into account both higher-level sensing constraints and environmental factors affecting the robot's dynamics, such as the work by Ru and Martinez [157], is an important direction of future work.

9.2 Future Research Directions

These are exciting times for robotics. Robotic technology today has matured to the point where robots capable of operating in the natural environment are available commercially. We regularly see impressive demonstrations of robots performing skillful and challenging tasks. This trend will continue thanks to the renewed interest in robotics by the industry and government.

On the other hand, rapid technological progress can make defining research problems itself challenging. What is considered a challenge today may become ubiquitous by tomorrow. For example, the availability of inexpensive and accurate GPS sensors have largely solved the outdoor robot localization problem.[1] Most modern robots carry a powerful computer by today's standard, that allows us to perform many operations on-board which would have been considered infeasible earlier. For example, the UAV in the current version of the precision agriculture sensing system (Chapter 6) has a sophisticated ODroid U3 computer with a 1.7GHz Quad-Core processor and 2GByte RAM on-board. This is orders of magnitude better than the PDP-10 computer with less than 1MByte memory which was on the Shaky robot, one of the first mobile robots developed in the sixties [158].

In such a situation, theoretical results that rely on some level of abstraction can be helpful in defining and making progress. Such theoretical results are useful is telling us the fundamental performance limits. However, the utility of such results is diminished

[1]There is however, considerable ongoing research on localizing in GPS-denied environments and precise localization beyond GPS-level accuracy.

if they are based on abstractions and models that are far removed from practice. In the following, we list some directions to pursue future research with the ultimate goal of bridging the gap between theoretical models and abstractions and practical constraints for robotic sensing systems.

9.2.1 Realistic Environments

Many of the worst-case bounds for visibility-based coverage problems are based on creating pathologically bad input instances. Realistic environments are typically are more structured. One way of bridging the gap between theory and practice would be to derive the theoretical bounds for realistic environments. However, defining what we mean by "realistic" itself is not easy. We outline a few approaches.

In one of the first attempts, Valtr [159] defined the notion of ϵ–good polygons. A point is called ϵ–good if a guard placed at that point sees at least ϵ fraction of the polygon. The polygon is ϵ–good if all points are ϵ–good. Valtr showed that the number of guards sufficient for covering an ϵ–good polygon depends only on ϵ and the number of holes but is independent of the total number of vertices, which is not the case for general polygons.

We can have polygons that are ϵ–good but have long, thin spikes which go against our intuitive notion of realistic environments. An alternate notion that disallows such instances is that of *fat* polygons. One description of fat polygons are the so-called (α, β)–covered polygons [160]. A polygon P is called (α, β)–covered if for any point p on the boundary of P there exists a triangle $T(p)$ such that, (i) p is a vertex of $T(p)$, (ii) $T(p)$ is completely contained within P, (iii) all three angles of $T(p)$ are at least α, and (iv) the length of each edge of P is at least β times the diameter of P. The intuition is that we can always place a fat triangle $T(p)$ on the boundary of the polygon, thus avoiding long, thin spikes. The consequence of this definition is that the number of guards sufficient to cover the boundary of P depends only on α and β, but not on n [161]. An immediate research problem would be to determine the number of guards sufficient to see the interior of (α, β)–covered polygons.

Both the definitions given above are based on avoiding unrealistic scenarios. Alternatively, we can explicitly define a class of realistic polygons. Orthogonal polygons, in which all edges are aligned along two orthogonal axis, is a class of polygons that is a good approximation for real-life building floorplans [162].

Finally, instead of defining what we mean by realistic environments, we can use examples from the real-world. For example, recently Amigoni et al. [163] presented a data-driven approach to randomly general realistic indoor environments based on what they term as *structural realism*. They used floorplans of 150 actual buildings and assigned semantic labels to all the rooms and corridors in the floorplans. The floorplans themselves were obtained from architectural design textbooks and came with some functional labels already assigned. From this, the authors computed statistics for rooms in each label and relational information between separate labels. These statistics were then used to randomly generate new environments. Using a more data-driven approach may be an interesting direction study more practical versions of coverage problems.

9.2.2 Realistic Robot Motion Models

In addition to developing and using realistic input models, it is important to use realistic motion models for the robot in order to bridge the gap between theory and practice. In Chapters 5–7 we used a simple point model for the robot. We neglected any kinematic constraints (e.g. the steering model) or dynamic constraints (e.g. drifts due to winds/waves) while planning the robot's path. This will cause a gap in the theoretical and experimental performance of the system.

One way of modeling this gap is in a hierarchical fashion with increasing levels of fidelity. Consider the problem of computing TSP tours. The point robot model represents the highest level of abstraction, which gives the strongest theoretical guarantees in the form of a PTAS but possibly at a lower fidelity. At the next level, we can incorporate steering constraints in the robot's motion model improving the fidelity. A constant factor approximation is known [164] when the motion model is a Dubins' car. This algorithm builds on the point motion model of the previous level. At even lower levels of abstraction, we can incorporate further practical constraints, such as robot dynamics and energy, as long as they improve the fidelity. It will be interesting and useful to build and understand this hierarchy for some of the commonly available robotic sensing tasks.

9.2.3 Robustness against Uncertainty in the Model

In this dissertation, we saw how noise in the sensing can limit the quality of coverage and tracking. Unlike passive sensing systems, however, robotic sensors have the capability to reconfigure themselves. By designing adaptive planning algorithms, the robots can overcome the limitations of sensing noise and improve the sensing performance. Sensing noise is one amongst many sources that cause uncertainty for robotic sensing systems. Designing efficient planning algorithms to address other sources of uncertainty with both provable performance guarantees, as well as through rigorous experimentation is a crucial challenge. We discuss one particular source of uncertainty.

In formulating many of the problems in this dissertation, we implicitly assumed the input can be perfectly specified and is a faithful representation of the actual scenario. For example, the coverage algorithm to find radio-tagged carp in Section 7.3 takes as input a set of polygons that represent regions within the lake that are likely to contain carp. The algorithm and its analysis rely only the geometry of the input regions. In practice, however, it may not be possible to specify such regions with certainty. Similarly, for the coverage problems input polygons may not be an accurate representation of the actual environment. Consequently, this will cause the empirical performance of an algorithm to deviate from the theoretical analysis. One possible way of bridging this gap would be to allow the input itself to be uncertain instead of assuming it is perfectly specified. There are two related questions that must be addressed: how to represent or model the uncertainty in the input, and how does this uncertainty affect our algorithms and their theoretical analysis?

When the input itself is estimated from sensor measurements or in a data-driven fashion, the uncertainty can be derived from the relevant estimator. For example, Vasudevan et al. [165] show how Gaussian Process (GP) regression can be used to model an environment using terrain data and use the GP to determine the uncertainty and incompleteness of the model. Similarly, for the target tracking problem, Joseph et al. [166] used GPs to estimate motion models (and the uncertainty in the estimates) from a dataset of time-stamped GPS coordinates of taxis in Boston. On the other hand, if a human is specifying the input, we may have to take into account additional uncertainty factors [167]. For example, users' backgrounds can influence how they view the functionality of a robot [168] which may potentially affect how they specify inputs or tasks. Understanding such factors and developing algorithms that take them into account is an important research direction.

For the second question, there have been recent efforts in understanding the role played by uncertainty in geometric problems that have been studied in mostly deterministic settings. Kamousi and Suri [169] studied the classical TSPN problem for disks, where the radii of disks is a random variable drawn from a known probability distribution. Given some assumptions on the probability distribution, they presented an algorithm which for a given random instance of n disks computes a tour whose length is at most $\mathcal{O}(\log\log n)$ times the *expected* length of the optimal tour. In this case, the radii are all revealed before the tour is computed. If the radii are revealed in an online fashion, then they presented an expected $\mathcal{O}(\log n)$ approximation algorithm.

The effect of uncertainty in the environment representation on visibility has not been studied largely. Cai and Keil [170] showed that the visibility graphs for polygons whose vertices are known only up to an accuracy of ϵ distance can be computed efficiently. Establishing "robust" bounds for visibility-placed coverage algorithms remains an open research problem and an avenue of future research.

9.2.4 Long-Term Planning

All the coverage and tracking problems we studied in this dissertation focused on a single, well-defined task. We formulated these problems as optimization problems with one or sometimes two criteria from amongst sensing quality, time, distance or energy. The situation will be more complex when we consider practical deployments of robotic sensor systems for a longer time period. Instead of relying on either stationary sensors or one or more robots, a combination of various types of sensors and robots may be used. Addressing and planning for the communication between various components of the system will become important. Managing overall system energy will be critical to ensure long-term operation. Moreover, the system could be expected to carry out a number of tasks simultaneously whose objectives may be in conflict with each other.

In recent years, robotics researchers have made progress towards solving separate components, both through theoretical results and through field studies. Going forward, a principled study of the interplay between various components of this system will become important. Our lab's research efforts [171] are aligned in this direction.

9.3 Concluding Remarks

We started this dissertation by noting that industrial robots that carry out complex assembly and manufacturing tasks have been one of the most widespread and successful accomplishments for the field of robotics. We end on an optimistic note. Robots that are no longer constrained to controlled, industrial settings will become ubiquitous in the future and perform tasks deemed too complex, unsafe, or infeasible for humans today. A number of challenges will need to be solved before we get there. Achieving a greater synergy between theory and practice is a crucial challenge. In this dissertation, we investigated some of the algorithmic and system challenges pertaining to robotic sensing systems. Our hope is that engineers and researchers from diverse areas will continue to work together and ultimately overcome the challenges that lie before a robotics revolution.

References

[1] Joseph O'Rourke. *Art gallery theorems and algorithms*. Oxford University Press Oxford, 1987.

[2] Adrian Dumitrescu and Joseph SB Mitchell. Approximation algorithms for tsp with neighborhoods in the plane. *Journal of Algorithms*, 48(1):135–159, 2003.

[3] D. J Mulla, M Beatty, and A. C. Sekely. Evaluation of remote sensing and targeted soil sampling for variable rate application of nitrogen. In *Proceedings of 5th International Conference on Precision Agriculture*. American Society of Agronomy, Crop Science Society of America, and Soil Science Society of America, 2001.

[4] Malaysia missing MH370 plane: "ping area" ruled out, 2014 (accessed August 20, 2014). `http://www.bbc.com/news/world-asia-26514556`.

[5] Chris Urmson. The latest chapter for the self-driving car: mastering city street driving, 2014 (accessed August 20, 2014). `http://googleblog.blogspot.com/2014/04/the-latest-chapter-for-self-driving-car.html`.

[6] The Economist. Telepresence robots: Your alter ego on wheels, 2013 (accessed August 26, 2014). `http://www.economist.com/news/technology-quarterly/21572916-robotics-remotely-controlled-telepresence-robots-let-people-be-two-places`.

[7] Jason M O'Kane. *A Theory for Comparing Robot Systems*. PhD thesis, University of Illinois at Urbana-Champaign, 2007.

[8] Neil H Carter, Bhim Gurung, Andrés Viña, Henry Campa III, Jhamak B Karki, and Jianguo Liu. Assessing spatiotemporal changes in tiger habitat across different land management regimes. *Ecosphere*, 4(10):art124, 2013.

[9] Onur Tekdas, Volkan Isler, Jong Hyun Lim, and Andreas Terzis. Using mobile robots to harvest data from sensor fields. *IEEE Wireless Communications*, 16(1):22, 2009.

[10] P. G. Bajer, C. J. Chizinski, and P. W. Sorensen. Using the Judas technique to locate and remove wintertime aggregations of invasive common carp. *Fisheries Management and Ecology*, 2011.

[11] Ioannis Rekleitis, Eric Martin, Guy Rouleau, Régent L'Archevêque, Kourosh Parsa, and Eric Dupuis. Autonomous capture of a tumbling satellite. *Journal of Field Robotics*, 24(4):275–296, 2007.

[12] Jue Wang, Fadel Adib, Ross Knepper, Dina Katabi, and Daniela Rus. Rf-compass: robot object manipulation using RFIDs. In *Proceedings of the 19th annual international conference on Mobile computing & networking*, pages 3–14. ACM, 2013.

[13] Jocelyn Smith and William S Evans. Triangle guarding. In *Canadian Conference on Computational Geometry*, pages 76–80, 2003.

[14] Alon Efrat, Sariel Har-Peled, and Joseph SB Mitchell. Approximation algorithms for two optimal location problems in sensor networks. In *2nd International Conference on Broadband Networks*, pages 714–723, 2005.

[15] P. Tokekar and V. Isler. Polygon guarding with orientation. In *Proceedings of IEEE International Conference on Robotics and Automation*. IEEE, 2014.

[16] P. Tokekar and V. Isler. Sensor placement and selection for bearing sensors with bounded uncertainty. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 2515–2520. IEEE, 2013.

[17] P. Tokekar, V. Isler, and A. Franchi. Multi-target visual tracking with aerial robots. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014.

[18] David J. Mulla. Twenty five years of remote sensing in precision agriculture: Key advances and remaining knowledge gaps. *Biosystems Engineering*, 114(4):358 – 371, 2013. Special Issue: Sensing Technologies for Sustainable Agriculture.

[19] Avrim Blum, Shuchi Chawla, David R Karger, Terran Lane, Adam Meyerson, and Maria Minkoff. Approximation algorithms for orienteering and discounted-reward tsp. *SIAM Journal on Computing*, 37(2):653–670, 2007.

[20] P. Tokekar, J. Vander Hook, D. Mulla, and V. Isler. Sensor planning for a symbiotic UAV and UGV system for precision agriculture. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5321–5326. IEEE, 2013.

[21] P.G. Bajer, H. Lim, M.J. Travaline, B.D. Miller, and P.W. Sorensen. Cognitive aspects of food searching behavior in free-ranging wild Common Carp. *Environmental Biology of Fishes*, 88(3):295–300, 2010.

[22] D. Bhadauria, V. Isler, A. Studenski, and P. Tokekar. A robotic sensor network for monitoring carp in Minnesota lakes. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 3837–3842. IEEE, 2010.

[23] P. Tokekar, J. Vander Hook, and V. Isler. Active target localization for bearing based robotic telemetry. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 488–493. IEEE, 2011.

[24] P. Tokekar, D. Bhadauria, A. Studenski, and V. Isler. A robotic system for monitoring carp in Minnesota lakes. *Journal of Field Robotics*, 27(6):779–789, 2010.

[25] P. Tokekar, E. Branson, J. Vander Hook, and V. Isler. Tracking aquatic invaders: Autonomous robots for invasive fish. *IEEE Robotics and Automation Magazine*, 20(3):33–41, 2013.

[26] P. Tokekar, N. Karnad, and V. Isler. Energy-optimal velocity profiles for car-like robots. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 1457–1462. IEEE, 2011.

[27] P. Tokekar, N. Karnad, and V. Isler. Energy-optimal trajectory planning for car-like robots. *Autonomous Robots*, 2014.

[28] Uriel Feige. A threshold of ln n for approximating set cover. *Journal of the ACM (JACM)*, 45(4):634–652, 1998.

[29] David S Johnson. Approximation algorithms for combinatorial problems. In *Proceedings of the fifth annual ACM symposium on Theory of computing*, pages 38–49. ACM, 1973.

[30] Vašek Chvátal. A greedy heuristic for the set-covering problem. *Mathematics of operations research*, 4(3):233–235, 1979.

[31] Dorit S Hochbaum and Anu Pathria. Analysis of the greedy approach in problems of maximum k-coverage. *Naval Research Logistics (NRL)*, 45(6):615–627, 1998.

[32] Jorge Urrutia. Art gallery and illumination problems. *Handbook of Computational Geometry*, pages 973–1027, 2000.

[33] Subir Kumar Ghosh. Approximation algorithms for art gallery problems in polygons and terrains. In Md.Saidur Rahman and Satoshi Fujita, editors, *WALCOM: Algorithms and Computation*, volume 5942 of *Lecture Notes in Computer Science*, pages 21–34. Springer Berlin Heidelberg, 2010.

[34] Vasek Chvatal. A combinatorial theorem in plane geometry. *Journal of Combinatorial Theory, Series B*, 18(1):39–41, 1975.

[35] Iliana Bjorling-Sachs and Diane L. Souvaine. An efficient algorithm for guard placement in polygons with holes. *Discrete & Computational Geometry*, 13(1):77–109, 1995.

[36] Frank Hoffmann, Michael Kaufmann, and Klaus Kriegel. The art gallery theorem for polygons with holes. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pages 39–48. IEEE Comput. Soc. Press, 1991.

[37] Steve Fisk. A short proof of chvátal's watchman theorem. *Journal of Combinatorial Theory, Series B*, 24(3):374, 1978.

[38] Joseph O'Rourke and Kenneth Supowit. Some NP-hard polygon decomposition problems. *IEEE Transactions on Information Theory*, 29(2):181–190, 1983.

[39] Der-Tsai Lee and ARTHURK Lin. Computational complexity of art gallery problems. *Information Theory, IEEE Transactions on*, 32(2):276–282, 1986.

[40] S. Eidenbenz, C. Stamm, and P. Widmayer. Inapproximability results for guarding polygons and terrains. *Algorithmica*, 31(1):79–113, 2001.

[41] Hervé Brönnimann and Michael T Goodrich. Almost optimal set covers in finite VC-dimension. *Discrete & Computational Geometry*, 14(1):463–479, 1995.

[42] Alexander Gilbers and Rolf Klein. A new upper bound for the vc-dimension of visibility regions. *Computational Geometry*, 47(1):61–74, 2014.

[43] James King and David Kirkpatrick. Improved approximation for guarding simple galleries from the perimeter. *Discrete & Computational Geometry*, 46(2):252–269, 2011.

[44] Alon Efrat and Sariel Har-Peled. Guarding galleries and terrains. *Information Processing Letters*, 100(6):238–245, 2006.

[45] Ajay Deshpande, Taejung Kim, Erik Demaine, and Sanjay Sarma. A pseudopolynomial time o (log n)-approximation algorithm for art gallery problems. *Algorithms and Data Structures*, pages 163–174, 2007.

[46] Erik A Krohn and Bengt J Nilsson. Approximate guarding of monotone and rectilinear polygons. *Algorithmica*, 66(3):564–594, 2013.

[47] Christos H Papadimitriou. The euclidean travelling salesman problem is np-complete. *Theoretical Computer Science*, 4(3):237–244, 1977.

[48] Nicos Christofides. Worst-case analysis of a new heuristic for the travelling salesman problem. Technical report, DTIC Document, 1976.

[49] Michael Lampis. Guest column: the elusive inapproximability of the tsp. *ACM SIGACT News*, 45(1):48–65, 2014.

[50] Sanjeev Arora. Polynomial time approximation schemes for euclidean traveling salesman and other geometric problems. *Journal of the ACM (JACM)*, 45(5):753–782, 1998.

[51] Joseph SB Mitchell. Guillotine subdivisions approximate polygonal subdivisions: A simple polynomial-time approximation scheme for geometric tsp, k-mst, and related problems. *SIAM Journal on Computing*, 28(4):1298–1309, 1999.

[52] Esther M Arkin and Refael Hassin. Approximation algorithms for the geometric covering salesman problem. *Discrete Applied Mathematics*, 55(3):197–218, 1994.

[53] Adrian Dumitrescu and Csaba D Tóth. The traveling salesman problem for lines, balls and planes. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 828–843. SIAM, 2013.

[54] Joseph SB Mitchell. A constant-factor approximation algorithm for tsp with pairwise-disjoint connected neighborhoods in the plane. In *Proceedings of the twenty-sixth annual symposium on Computational geometry*, pages 183–191. ACM, 2010.

[55] Joseph SB Mitchell. A ptas for tsp with neighborhoods among fat regions in the plane. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 11–18. Society for Industrial and Applied Mathematics, 2007.

[56] Anurag Ganguli, Jorge Cortés, and Francesco Bullo. Distributed coverage of nonconvex environments. In *Networked sensing information and control*, pages 289–305. Springer, 2008.

[57] Tim Danner and EE Kavraki. Randomized planning for short inspection paths. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 2, pages 971–976. IEEE, 2000.

[58] Andrew J Davison, Ian D Reid, Nicholas D Molton, and Olivier Stasse. Monoslam: Real-time single camera slam. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1052–1067, 2007.

[59] Héctor González-Baños and Jean-Claude Latombe. A randomized art-gallery algorithm for sensor placement. In *Proceedings of the seventeenth annual symposium on Computational geometry*, pages 232–240. ACM, 2001.

[60] Paul S Blaer and Peter K Allen. View planning and automated data acquisition for three-dimensional modeling of complex sites. *Journal of Field Robotics*, 26(11-12):865–891, 2009.

[61] Sumio Masuda and Kazuo Nakajima. An optimal algorithm for finding a maximum independent set of a circular-arc graph. *SIAM Journal on Computing*, 17(1):41–52, 1988.

[62] R. D'Andrea. Guest editorial: A revolution in the warehouse: a retrospective on Kiva systems and the grand challenges ahead. *IEEE Transactions on Automation Science and Engineering*, 9(4):638–639, 2012.

[63] D. Bertsekas and I. Rhodes. Recursive state estimation for a set-membership description of uncertainty. *IEEE Transactions on Automatic Control*, 16(2):117–128, 1971.

[64] A. Garulli and A. Vicino. Set membership localization of mobile robots via angle measurements. *IEEE Transactions on Robotics and Automation*, 17(4):450–463, 2001.

[65] E.A. Yfantis, G.T. Flatman, and J.V. Behar. Efficiency of kriging estimation for square, triangular, and hexagonal grids. *Mathematical Geology*, 19(3):183–205, 1987.

[66] M. Younis and K. Akkaya. Strategies and techniques for node placement in wireless sensor networks: A survey. *Ad Hoc Networks*, 6(4):621–655, 2008.

[67] L. Hu and D. Evans. Localization for mobile sensor networks. In *Proceedings of the 10th annual international conference on Mobile computing and networking*, pages 45–57. ACM, 2004.

[68] F. Benbadis, K. Obraczka, J. Cortés, and A. Brandwajn. Exploring landmark placement strategies for topology-based localization in wireless sensor networks. *EURASIP Journal on Advances in Signal Processing*, 2008:138, 2008.

[69] O. Tekdas and V. Isler. Sensor placement for triangulation-based localization. *IEEE Transactions on Automation Science and Engineering*, 7(3):681–685, 2010.

[70] A. Ercan, D. Yang, A. El Gamal, and L. Guibas. Optimal placement and selection of camera network nodes for target localization. In *International Conference on Distributed Computing in Sensor Systems*, pages 389–404. Springer, 2006.

[71] V. Isler and M. Magdon-Ismail. Sensor selection in arbitrary dimensions. *IEEE Transactions on Automation Science and Engineering*, 5(4):651–660, 2008.

[72] Cédric Vermeulen, Philippe Lejeune, Jonathan Lisein, Prosper Sawadogo, and Philippe Bouché. Unmanned aerial survey of elephants. *PloS one*, 8(2):e54700, 2013.

[73] Chandra Chekuri and Amit Kumar. Maximum coverage problem with group budget constraints and applications. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 72–83. Springer, 2004.

[74] John R Spletzer and Camillo J Taylor. Dynamic sensor planning and control for optimally tracking targets. *The International Journal of Robotics Research*, 22(1):7–20, 2003.

[75] Eric W Frew. *Observer trajectory generation for target-motion estimation using monocular vision*. PhD thesis, Stanford University, 2003.

[76] Steven M LaValle, Hector H González-Banos, Craig Becker, and J-C Latombe. Motion strategies for maintaining visibility of a moving target. In *IEEE International Conference on Robotics and Automation*, volume 1, pages 731–736. IEEE, 1997.

[77] Nicholas R Gans, Guoqiang Hu, Kaushik Nagarajan, and Warren E Dixon. Keeping multiple moving targets in the field of view of a mobile camera. *IEEE Transactions on Robotics*, 27(4):822–828, 2011.

[78] Sergei Bespamyatnikh, Binay Bhattacharya, David Kirkpatrick, and Michael Segal. Mobile facility location. In *Proceedings of the 4th International Workshop on Discrete Algorithms and Methods for Mobile computing and Communications*, pages 46–53. ACM, 2000.

[79] Stephane Durocher. *Geometric Facility Location under Continuous Motion*. PhD thesis, The University of British Columbia, 2006.

[80] Mark de Berg, Marcel Roeloffzen, and Bettina Speckmann. Kinetic 2-centers in the black-box model. In *Proceedings of the 29th Annual on Symposium on Computational Geometry*, pages 145–154. ACM, 2013.

[81] Sonia Martinez, Jorge Cortes, and Francesco Bullo. Motion coordination with distributed information. *IEEE Control Systems*, 27(4):75–88, 2007.

[82] Mac Schwager, Brian J Julian, Michael Angermann, and Daniela Rus. Eyes in the sky: Decentralized control for the deployment of robotic camera networks. *Proceedings of the IEEE*, 99(9):1541–1561, 2011.

[83] Sebastian Thrun, Wolfram Burgard, Dieter Fox, et al. *Probabilistic robotics*, volume 1. MIT press Cambridge, 2005.

[84] Colin J Green and Alonzo Kelly. Toward optimal sampling in the space of paths. *Springer Tracts in Advanced Robotics*, 66:281–292, 2010.

[85] J. Lächele, A. Franchi, H. H. Bülthoff, and P. Robuffo Giordano. SwarmSimX: Real-time simulation environment for multi-robot systems. In *3rd International Conference on Simulation, Modeling, and Programming for Autonomous Robots*, Tsukuba, Japan, Nov. 2012.

[86] Mikrokopter, 2014 (accessed August 31, 2014). `http://www.mikrokopter.de/`.

[87] Simon Benhamou. How many animals really do the levy walk? *Ecology*, 88(8):1962–1969, 2007.

[88] V. Grabe, M. Riedel, H. H. Bülthoff, P. Robuffo Giordano, and A. Franchi. The TeleKyb framework for a modular and extendible ROS-based quadrotor control.

In *6th European Conference on Mobile Robots*, pages 19–25, Barcelona, Spain, Sep. 2013.

[89] Sanjeev Arora. Polynomial time approximation schemes for euclidean tsp and other geometric problems. In *Foundations of Computer Science, 1996. Proceedings., 37th Annual Symposium on*, pages 2–11. IEEE, 1996.

[90] Kian Hsiang Low, Jie Chen, John M Dolan, Steve Chien, and David R Thompson. Decentralized active robotic exploration and mapping for probabilistic field classification in environmental sensing. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 105–112, 2012.

[91] Bin Zhang and Gaurav S. Sukhatme. Adaptive Sampling for Estimating a Scalar Field using a Robotic Boat and a Sensor Network. *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 3673–3680, April 2007.

[92] Dezhen Song, Chang-Young Kim, and Jingang Yi. Simultaneous localization of multiple unknown and transient radio sources using a mobile robot. *Robotics, IEEE Transactions on*, 28(3):668–680, 2012.

[93] Andreas Krause, A. Singh, and C. Guestrin. Near-optimal sensor placements in Gaussian processes: Theory, efficient algorithms and empirical studies. *The Journal of Machine Learning Research*, 9:235–284, 2008.

[94] Alexandra Meliou, Andreas Krause, Carlos Guestrin, and Joseph M Hellerstein. Nonmyopic informative path planning in spatio-temporal models. In *Proceedings of the 22nd national conference on Artificial intelligence-Volume 1*, pages 602–607. AAAI Press, 2007.

[95] Amarjeet Singh, Andreas Krause, and William J Kaiser. Nonmyopic Adaptive Informative Path Planning for Multiple Robots. *International Joint Conference on Artificial Intelligence*, pages 1843–1850, 2008.

[96] Deepak Bhadauria, Onur Tekdas, and Volkan Isler. Robotic data mules for collecting data over sparse sensor fields. *Journal of Field Robotics*, 28(3):388–404, 2011.

[97] Onur Tekdas, Deepak Bhadauria, and Volkan Isler. Efficient data collection from wireless nodes under the two-ring communication model. *The International Journal of Robotics Research*, 31(6):774–784, 2012.

[98] Helmut Alt, Esther M Arkin, Hervé Brönnimann, Jeff Erickson, Sándor P Fekete, Christian Knauer, Jonathan Lenchner, Joseph SB Mitchell, and Kim Whittlesey. Minimum-cost coverage of point sets by disks. In *Proceedings of the twenty-second annual symposium on Computational geometry*, pages 449–458. ACM, 2006.

[99] Ben Grocholsky, James Keller, Vijay Kumar, and George Pappas. Cooperative air and ground surveillance. *Robotics & Automation Magazine, IEEE*, 13(3):16–25, 2006.

[100] P.B. Sujit and S. Saripalli. An empirical evaluation of co-ordination strategies for an auv and uav. *Journal of Intelligent & Robotic Systems*, 70:373–384, 2013.

[101] Herbert G Tanner. Switched uav-ugv cooperation scheme for target detection. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 3457–3462. IEEE, 2007.

[102] Nabil H Mustafa and Saurabh Ray. Improved results on geometric hitting set problems. *Discrete & Computational Geometry*, 44(4):883–895, 2010.

[103] Carl Edward Rasmussen and Christopher KI Williams. *Gaussian processes for machine learning*, volume 1. MIT press Cambridge, MA, 2006.

[104] Donald E Myers. Estimation of linear combinations and co-kriging. *Mathematical Geology*, 15(5):633–637, 1983.

[105] Clearpath robotics. 2013 (accessed Jan 2013).

[106] Spectrum technologies inc. 2013 (accessed Mar 2013).

[107] Carl Edward Rasmussen and Hannes Nickisch. Gaussian processes for machine learning (gpml) toolbox. *The Journal of Machine Learning Research*, 11:3011–3015, 2010.

[108] Andreas Krause. Sfo: A toolbox for submodular function optimization. *The Journal of Machine Learning Research*, 11:1141–1144, 2010.

[109] David Applegate, ROBERT Bixby, Vasek Chvatal, and William Cook. Concorde tsp solver. *URL http://www. tsp. gatech. edu/concorde*, 2006.

[110] Tetracam. 2014 (accessed Feb 2014).

[111] J. Das, F. Py, T. Maughan, T. OReilly, M. Messié, J. Ryan, K. Rajan, and G. Sukhatme. Simultaneous Tracking and Sampling of Dynamic Oceanographic Features with Autonomous Underwater Vehicles and Lagrangian Drifters. In *International Symposium on Experimental Robotics*, 2010.

[112] H. Ferreira, C. Almeida, A. Martins, J. Almeida, N. Dias, A. Dias, and E. Silva. Autonomous Bathymetry for Risk Assessment with ROAZ Robotic Surface Vehicle. In *OCEANS 2009 - Europe*, pages 1–6, May 2009.

[113] Geoffrey A Hollinger, Brendan Englot, Franz S Hover, Urbashi Mitra, and Gaurav S Sukhatme. Active planning for underwater inspection and the benefit of adaptivity. *The International Journal of Robotics Research*, page 0278364912467485, 2012.

[114] Eric Raboin, Petr Švec, Dana S Nau, and Satyandra K Gupta. Model-predictive asset guarding by team of autonomous surface vehicles in environment with civilian boats. *Autonomous Robots*, pages 1–22, 2014.

[115] M. Dunbabin, P. Corke, I. Vasilescu, and D. Rus. Experiments with Cooperative Control of Underwater Robots. *The International Journal of Robotics Research*, 28(6):815, 2009.

[116] Christopher M Clark, Christina Forney, Esfandiar Manii, Dylan Shinzaki, Chris Gage, Michael Farris, Christopher G Lowe, and Mark Moline. Tracking and following a tagged leopard shark with an autonomous underwater vehicle. *Journal of Field Robotics*, 30(3):309–322, 2013.

[117] M. Dunbabin and L. Marques. Robots for environmental monitoring: Significant advancements and applications. *IEEE Robotics and Automation Magazine*, 19(1):24 –39, Mar 2012.

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

[119] S. E. Hammel, P. T. Liu, E. J. Hilliard, and K. F. Gong. Optimal observer motion for localization with bearing measurements. *Computers and Mathematics with Applications*, 18(1-3):171 – 180, 1989.

[120] Ke Zhou and S.I. Roumeliotis. Multi-robot active target tracking with combinations of relative observations. *IEEE Transactions on Robotics*, 27(4):678 –695, Aug. 2011.

[121] J. Vander Hook, P. Tokekar, and V. Isler. Cautious greedy strategy for bearing-only active localization: Analysis and field experiments. *Journal of Field Robotics*, 31(2):296–318, 2014.

[122] Oceanscience, 2011 (accessed November 11, 2011). `http://www.oceanscience.com/`.

[123] Massimo Caccia, Marco Bibuli, Riccardo Bono, and Gabriele Bruzzone. Basic navigation, guidance and control of an Unmanned Surface Vehicle. *Autonomous Robots*, 25(4):349–365, Aug 2008.

[124] Advanced Telemetry Systems, 2011 (accessed November 11, 2011). `http://www.atstrack.com/`.

[125] P. Tokekar, D. Bhadauria, A. Studenski, and V. Isler. A Robotic System for Monitoring Carp in Minnesota Lakes. *Journal of Field Robotics*, 27(6):779–789, 2010.

[126] W.P. Chin and S. Ntafos. The zookeeper route problem. *Information Sciences: an International Journal*, 63(3):245–259, 1992.

[127] J. Vander Hook, P. Tokekar, E. Branson, P. Bajer, P. Sorensen, and V. Isler. Local-Search Strategy for Active Localization of Multiple Invasive Fish. In *International Symposium on Experimental Robotics*, 2012.

[128] P. Tokekar, J. Vander Hook, and V. Isler. Active Target Localization for Bearing-Based Robotic Telemetry. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 488–493, 2011.

[129] J. Vander Hook, P. Tokekar, E. Branson, P. Bajer, P. Sorensen, and V. Isler. Local-search strategy for active localization of multiple invasive fish. In Jaydev P. Desai, Gregory Dudek, Oussama Khatib, and Vijay Kumar, editors, *Experimental Robotics*, volume 88 of *Springer Tracts in Advanced Robotics*, pages 859–873. Springer International Publishing, 2013.

[130] A. Bhattacharyya. On a measure of divergence between two multinomial populations. *Sankhyā: The Indian Journal of Statistics (1933-1960)*, 7(4):401–406, 1946.

[131] *DC Motors, Speed Controls, Servo Systems: An Engineering Handbook*. Electro-Craft Corporation, 1977.

[132] Bruce Donald, Patrick Xavier, John Canny, and John Reif. Kinodynamic motion planning. *Journal of the ACM*, 40(5):1048–1066, 1993.

[133] LE Dubins. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of Mathematics*, 79(3):497–516, 1957.

[134] JA Reeds and LA Shepp. Optimal Paths for A Car That Goes Both Forwards and Backwards. *Pacific Journal of Mathematics*, 145(2):367–393, 1990.

[135] D.J. Balkcom and M.T. Mason. Time optimal trajectories for bounded velocity differential drive vehicles. *The International Journal of Robotics Research*, 21(3):199, 2002.

[136] H. Chitsaz, S.M. LaValle, D.J. Balkcom, and M.T. Mason. Minimum wheel-rotation paths for differential-drive mobile robots. *The International Journal of Robotics Research*, 28(1):66, 2009.

[137] Z. Sun and J.H. Reif. On finding energy-minimizing paths on terrains. *IEEE Transactions on Robotics*, 21(1):102–114, 2005.

[138] L. Guzzella and A. Sciarretta. *Vehicle propulsion systems: introduction to modeling and optimization*, volume 10. Springer Verlag, 2007.

[139] John Gregory, Alberto Olivares, and Ernesto Staffetti. Energy-optimal trajectory planning for robot manipulators with holonomic constraints. *Systems & Control Letters*, 61(2):279–291, 2012.

[140] O. Wigstrom, B. Lennartson, A. Vergnano, and C. Breitholtz. High-level scheduling of energy optimal trajectories. *Automation Science and Engineering, IEEE Transactions on*, 10(1):57–64, 2013.

[141] Y. Mei, Y.H. Lu, YC Hu, and CSG Lee. Energy-efficient motion planning for mobile robots. In *Proceedings of IEEE International Conference on Robotics and Automation*, 2004.

[142] C.H. Kim and B.K. Kim. Minimum-energy translational trajectory generation for differential-driven wheeled mobile robots. *Journal of Intelligent and Robotic Systems*, 49(4):367–383, 2007.

[143] C.H. Kim and B.K. Kim. Minimum-Energy Rotational Trajectory Planning for Differential-Driven Wheeled Mobile Robots. In *Proceedings of 13th International Conference on Advanced Robotics*, pages 265–270, 2007.

[144] G. Wang, M.J. Irwin, P. Berman, H. Fu, and T. La Porta. Optimizing sensor movement planning for energy efficiency. In *Proceedings of the ACM International Symposium on Low power electronics and design*, 2005.

[145] John A Broderick, Dawn M Tilbury, and Ella M Atkins. Optimal coverage trajectories for a ugv with tradeoffs for energy and time. *Autonomous Robots*, 36(3):257–271, 2014.

[146] Florent Lamiraux and J-P Lammond. Smooth motion planning for car-like vehicles. *Robotics and Automation, IEEE Transactions on*, 17(4):498–501, 2001.

[147] Thierry Fraichard and Alexis Scheuer. From reeds and shepp's to continuous-curvature paths. *Robotics, IEEE Transactions on*, 20(6):1025–1035, 2004.

[148] Liang Ding, Zongquan Deng, Haibo Gao, Keiji Nagatani, and Kazuya Yoshida. Planetary rovers' wheel—soil interaction mechanics: new challenges and applications for wheeled mobile robots. *Intelligent Service Robotics*, 4(1):17–38, 2011.

[149] Sertac Karaman and Emilio Frazzoli. Optimal kinodynamic motion planning using incremental sampling-based methods. In *Decision and Control (CDC), 2010 49th IEEE Conference on*, pages 7681–7687. IEEE, 2010.

[150] Steven M LaValle and James J Kuffner. Randomized kinodynamic planning. *The International Journal of Robotics Research*, 20(5):378–400, 2001.

[151] D.E. Kirk. *Optimal Control Theory: An Introduction*. Prentice Hall, 1970.

[152] D.G. Hull. *Optimal control theory for applications*. Springer Verlag, 2003.

[153] Mark Galassi, Jim Davies, James Theiler, Brian Gough, Gerard Jungman, Patrick Alken, Michael Booth, and Fabrice Rossi. *GNU Scientific Library Reference Manual*. 3rd edition, 2007.

[154] Ioan A. Şucan, Mark Moll, and Lydia E. Kavraki. The Open Motion Planning Library. *IEEE Robotics & Automation Magazine*, 19(4):72–82, December 2012. `http://ompl.kavrakilab.org`.

[155] Giorgio Ausiello, Vincenzo Bonifaci, and Luigi Laura. The online prize-collecting traveling salesman problem. *Information Processing Letters*, 107(6):199–204, 2008.

[156] Michiel Blom, Sven O Krumke, Willem E de Paepe, and Leen Stougie. The online tsp against fair adversaries. *INFORMS Journal on Computing*, 13(2):138–148, 2001.

[157] Yu Ru and Sonia Martinez. Coverage control in constant flow environments based on a mixed energy–time metric. *Automatica*, 49(9):2632–2640, 2013.

[158] Nils J Nilsson. Shakey the robot. Technical report, DTIC Document, 1984.

[159] Pavel Valtr. Guarding galleries where no point sees a small area. *Israel Journal of Mathematics*, 104(1):1–16, 1998.

[160] Alon Efrat. The complexity of the union of (α,β)-covered objects. *SIAM Journal on Computing*, 34(4):775–787, 2005.

[161] Greg Aloupis, Prosenjit Bose, Vida Dujmović, Chris Gray, Stefan Langerman, and Bettina Speckmann. Triangulating and guarding realistic polygons. *Computational Geometry*, 47(2):296–306, 2014.

[162] Sándor P Fekete, Sophia Rex, and Christiane Schmidt. Online exploration and triangulation in orthogonal polygonal regions. In *WALCOM: Algorithms and Computation*, pages 29–40. Springer, 2013.

[163] Francesco Amigoni, Matteo Luperto, and Alberto Quattrini Li. Towards more realistic indoor environments for the virtual robot competition. In *RoboCup2014 Team Description Papers*, 2014.

[164] Ketan Savla, Emilio Frazzoli, and Francesco Bullo. Traveling salesperson problems for the dubins vehicle. *Automatic Control, IEEE Transactions on*, 53(6):1378–1391, 2008.

[165] Shrihari Vasudevan, Fabio Ramos, Eric Nettleton, and Hugh Durrant-Whyte. Gaussian process modeling of large-scale terrain. *Journal of Field Robotics*, 26(10):812–840, 2009.

[166] Joshua Joseph, Finale Doshi-Velez, Albert S Huang, and Nicholas Roy. A bayesian nonparametric approach to modeling motion patterns. *Autonomous Robots*, 31(4):383–400, 2011.

[167] Eric Horvitz. Principles of mixed-initiative user interfaces. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pages 159–166. ACM, 1999.

[168] Hee Rin Lee, Selma Šabanovic, and Erik Stolterman. Stay on the boundary: artifact analysis exploring researcher and user framing of robot design. In *Proceedings of the 32nd annual ACM conference on Human factors in computing systems*, pages 1471–1474. ACM, 2014.

[169] Pegah Kamousi and Subhash Suri. Euclidean traveling salesman tours through stochastic neighborhoods. In *Algorithms and Computation*, pages 644–654. Springer, 2013.

[170] Leizhen Cai and J Mark Keil. Computing visibility information in an inaccurate simple polygon. *International Journal of Computational Geometry & Applications*, 7(06):515–537, 1997.

[171] Narges Noori, Patrick A Plonski, Alessandro Renzaglia, Pratap Tokekar, Joshua Vander Hook, and Volkan Isler. Long-term search through energy efficiency and harvesting. Technical Report 13-001, Department of Computer Science & Engineering, University of Minnesota, January 2013. `http://www.cs.umn.edu/research/technical_reports.php?page=report&report_id=13-001`.

Appendix A

Proofs from Chapter 3

A.1 Proof of Lemma 1

Proof. **Convex vertices.** Suppose not. There exists a convex vertex v_i with no guard placed on it. Without loss of generality, say v_i lies at the origin of a coordinate system, with the perpendicular bisector of the interior angle as the Y-axis.

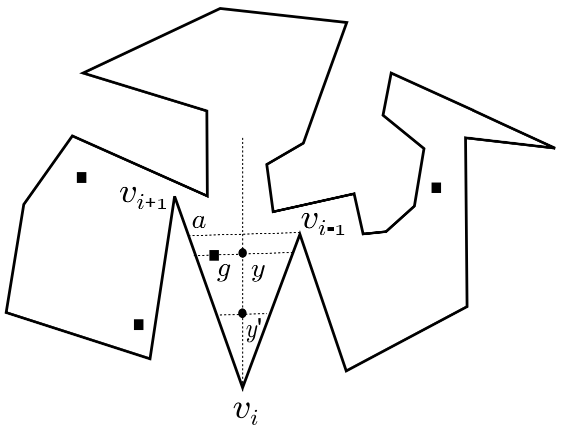


Figure A.1: There exists a guard on every convex vertex of the polygon.

Consider the triangle spanned by v_{i-1}, v_i, and v_{i+1} (see Figure A.1). Without loss of generality, say v_{i-1} has a lower Y-coordinate than v_{i+1}. Draw a line through v_{i-1} parallel to the X-axis. Let a be the point of intersection with the edge v_iv_{i+1}. We have two cases: (a) There exists a guard in the interior of triangle $v_{i-1}v_ia$, or (b) There does not exist a guard in the interior of the triangle $v_{i-1}v_ia$.

For (a), let g be some guard with the smallest Y-coordinate (say y) lying in the triangle. We have $y > 0$, since v lies at the origin. Consider a point, say y' on the Y-axis midway between y and v. Draw a line through y' parallel to the X-axis, and consider the lower half-plane. If there exists a guard visible from y' lying in the lower half-plane, then that contradicts the assumption that g is the guard with the lowest Y-coordinate in the triangle. Hence, there does not exist any guard in the lower half-plane through y'. Thus, y' is not $\triangle$-guarded from Proposition 1, which sets up our contradiction.

For (b), we repeat the same argument as the case (a) above using any arbitrary point y' with Y-coordinate less than that of v_{i-1}.

Edge extensions. We will prove by contradiction. Consider the case when the edge has two reflex vertices on its endpoints, say v_i and v_{i-1}. Let the edge be aligned with the X-axis such that its midpoint is the origin. From all guards, draw a line passing through all vertices of the polygon creating a visibility arrangement (Figure A.2).

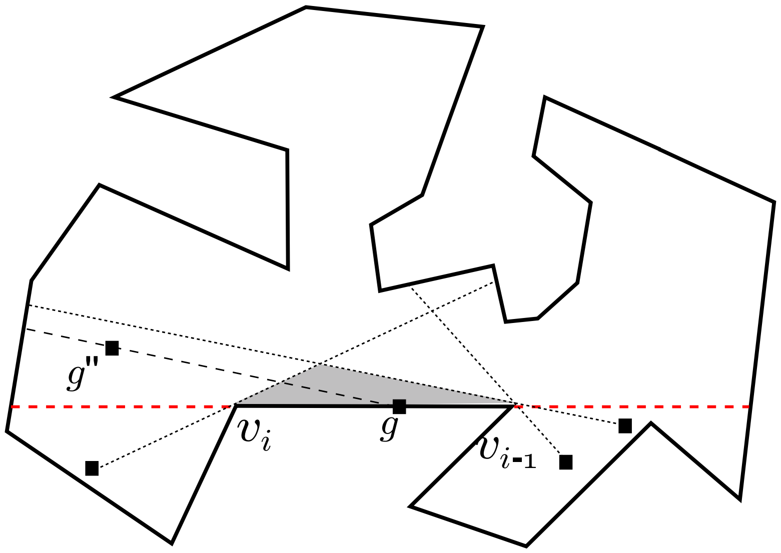


Figure A.2: To $\triangle$-guard all points lying in the cell (shown shaded) near the edge, there must exist a guard on each edge extension.

Consider any cell, A, in the visibility arrangement sharing an edge with v_iv_{i-1}. Let p be any point in the interior of this cell. p is not visible from any guard with negative

Y-coordinate (the visibility of any such guard is blocked by either v_i or v_{i-1}). Let y and y' be the smallest Y-coordinates of guards visible from p and with X coordinate smaller and greater than p, respectively. We denote the corresponding guards by g and g' respectively.

If both y and y' are greater than 0, then draw a line parallel to the X-axis with Y-coordinate equal to $0.5 \min\{y, y'\}$. Let p' be a point on this line contained in cell A. Then the halfplane containing p' extending towards the negative Y-axis does not contain any guard visible from p'. Hence, p' is not $\triangle$-guarded, which is a contradiction.

Suppose only one of y and y' is greater than 0, say y'. Then g must lie on the X-axis. We have either g lies on an edge extension, or g lies in the (open) polygon edge. Suppose g is the left-most point on the X-axis lying on the polygon edge, but not on the edge extension. Let A be the cell sharing with v_i as one of its vertices. Rotate the X-axis about g clockwise till the first guard g'' lying to the right of g is encountered.

Let H be the open halfplane using the line through g and g'' containing v_i. If there exists a point p' lying in $H \cap A$ then draw a line through p' parallel to gg'' and consider the closed lower halfplane. This halfplane does not contain any guard in its interior, and hence p' is not $\triangle$-guarded, which is a contradiction. Hence p' must not exist, which implies g'' lies on the X-axis to the left of g. Since g is the left-most guard on the edge, g'' must lie on the edge extension. The argument for the other edge extension is symmetrical. □

A.2 Proof of Lemma 3

Proof. Without loss of generality let C_i start first along clockwise ordering on the boundary, i.e., $s_i \prec s_j$. If C_i and C_j intersect, then we have $s_i \prec s_j \prec t_i \prec t_j$ (Figure A.3). Hence, A_i cuts A_j.

Consider the other direction. We prove the contrapositive. That is, if C_i and C_j do not intersect then A_i and A_j do not cut each other. If C_i and C_j do not intersect, then we have either $s_i \prec t_i \prec s_j \prec t_j$ or $s_i \prec s_j \prec t_j \prec t_i$ (Figure A.3). These imply either A_i and A_j are disjoint or $A_j \subset A_i$. In both cases, A_i and A_j do not cut each other. □

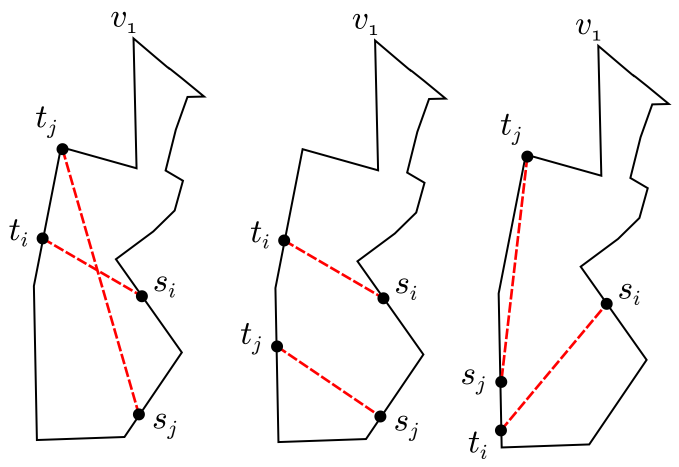


Figure A.3: If C_i and C_j intersect, then the correspondings arcs cut each other. If C_i and C_j do not intersect, either A_j is completely contained in A_i, or A_i and A_j are disjoint (given $s_i \prec s_j$).

A.3 Proof of Lemma 8

Proof. When both G^i and G^j contain Type IV chords, all arcs in G^i and G^j are contained in disjoint arcs in MIS. Hence, A_m and A_n do not cut each other.

If only one group contains Type IV chords, say G^i, then all arcs in G^i lie between two consecutive gaps. On the other hand, arcs in G^j start and terminate in a gap. Hence, all arcs in G^j are either disjoint from arcs in G^i or completely contain arcs in G^i.

The third possibility is both G^i and G^j contain Type III chords.

We have three cases:

1. Both starting and terminal gaps for G^i and G^j are distinct. Without loss of generality, let $s_m \prec s_n$. Hence we have,

   (a) $s_m \prec t_m \prec s_n \prec t_n$: All arcs in G^i and G^j are disjoint.

   (b) $s_m \prec s_n \prec t_n \prec t_m$: All arcs in G^j are completely contained in any arc in G^i.

   (c) $s_m \prec s_n \prec t_m \prec t_n$: A_m and A_n cut each other. That is, C_m and C_n are Type III chords with distinct start or terminal gaps cutting each other. From

Lemma 7 we have that S_2 covers both C_m and C_n. Hence $C_m, C_n \notin C'$ which is a contradiction.

2. Only starting gaps for G^i and G^j are distinct. Without loss of generality, let $s_m \prec s_n$. Hence we have,

   (a) $s_m \prec t_m \prec s_n \prec t_n$: We know t_m and t_n lie in the same gap. Therefore, s_n and t_n lie in the same gap which is a contradiction since Type III arcs span at least one gap.

   (b) $s_m \prec s_n \prec t_n \preceq t_m$: A_n is completely contained in A_m.

   (c) $s_m \prec s_n \prec t_m \prec t_n$: Similar to (1c) above.

3. Only terminal gaps for G^i and G^j are distinct. Without loss of generality, let $t_m \prec t_n$. Hence we have,

   (a) $s_m \prec t_m \prec s_n \prec t_n$: We know s_m and s_n lie in the same gap. Therefore, s_m and t_m lie in the same gap which is a contradiction since Type III arcs span at least one gap.

   (b) $s_n \preceq s_m \prec t_m \prec t_n$: A_m is completely contained in A_n.

   (c) $s_m \prec s_n \prec t_m \prec t_n$: Similar to (1c) above.

□

Appendix B

Proofs from Chapter 4

B.1 Proofs for the Lower Bounds

B.1.1 Proof for Lemma 13

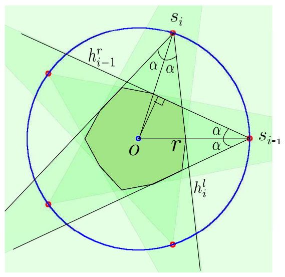


Figure B.1: Five sensors placed uniformly on a circle with radius r. The true target is located at the center of the circle. Each sensor receives a measurement with no noise.

Proof. Before we prove the lower bound for *any* placement, we first consider the case of $n \geq 3$ sensors placed uniformly on the boundary of C (Figure B.1). The true target location x is at the center o and all sensors receive measurements θ_i^m without any noise. Hence, for any point q on any sensing wedge, we have $oq \geq r \sin \alpha$, since $r \sin \alpha$ is the perpendicular distance of o to any of the bounding half-planes.

Recall that $\hat{P}$ denotes the possibly unbounded convex polygonal region of intersection of sensing wedges. We claim that the area of $\hat{P}$ in this case is lower bounded by $\pi r^2 \sin^2 \alpha$. Since o is in the intersection, if we show that there is a circle C centered at

o with radius $r \sin\alpha$ such that it lies in the interior or shares a boundary point with $\hat{P}$ then the claim holds. Suppose there is a point p in the interior of C such that $p \notin \hat{P}$. Let $p' \neq p$ be the point of intersection of the segment op with the boundary of $\hat{P}$. We have $op' < op < r\sin\alpha$. Since p' lies on the boundary of $\hat{P}$, it must also lie on one of the bounding half-planes of some sensing wedge. Hence, $op' \geq r\sin\alpha$ which is a contradiction. Since n here was arbitrary, the result holds for the case when the sensors are everywhere on the boundary of C.

Now consider any placement of sensors S. From Equation 4.1, we know that $U(S)$ is defined as the maximum over all possible true target locations. Hence, $U(S) \geq U(S|x = o)$ which is the uncertainty when the target is located at o. Assume that all sensors receive a measurement with zero noise, i.e., $\theta_i^m = \theta_i^t$ for all $i = 1, \ldots, n$. Hence, $U_D(S)$ and $U_A(S)$ are lower bounded by the diameter and area of intersection of all such sensing wedges.

We further lower bound this by the following construction: Replace each sensor s_i with the point of intersection of the boundary of C with the segment joining o and s_i. Denote this point of intersection by s_i'. Note $W(s_i, \theta_i^t) \supseteq W(s_i', \theta_i^t)$ and hence the intersection area and diameter formed by $W(s_i', \theta_i^t)$ is a lower bound on $U(S)$. Hence,

$$U_A(S) \geq \text{Area}\left(\bigcap_{i=1}^{n} W(s_i, \theta_i^t)\right) \geq \text{Area}\left(\bigcap_{i=1}^{n} W(s_i', \theta_i^t)\right) \geq \pi r^2 \sin^2(\alpha)$$

since the last step covers the case that the sensors are everywhere on the circle including all s_i' locations. Similarly, $U_D(S) \geq 2r\sin\alpha$. $\square$

B.1.2 Proof for Corollary 2

Proof. Suppose the corollary does not hold not. Then there must exist a point, say p, with distance $r^* > \dfrac{U_D^*}{2\sin\alpha}$ (or $r^* > \sqrt{\dfrac{U_A^*}{\pi}\dfrac{1}{\sin\alpha}}$) to the boundary and any sensor in S^*. We can draw a circle lying completely inside $\mathcal{A}$ with radius r^* centered at p, not containing any sensor in its interior. By Lemma 13, $U_D^* \geq 2r^*\sin\alpha$ or $r^* \leq \dfrac{U_D^*}{2\sin\alpha}$ (equivalently $U_A^* \geq \pi r^{*2}\sin^2\alpha$ or $r^* \leq \sqrt{\dfrac{U_A^*}{\pi}\dfrac{1}{\sin\alpha}}$), which is a contradiction. $\square$

B.1.3 Proof for Corollary 3

Proof. We first show that $U_D^* < \sin\alpha \cdot d$ implies $d > 2r^*$. Suppose not, i.e., $d \le 2r^*$. Hence we have, $U_D^* < 2r^* \sin\alpha$. However, from Corollary 2 we have $U_D^* \ge 2r^* \sin\alpha$, which is a contradiction. Similarly we can show $U_A^* < \dfrac{\pi \sin^2\alpha}{4} \cdot d^2$ implies $d > 2r^*$.

Consider a square $\mathcal{A}'$ centered at the center of $\mathcal{A}$ but with side length $d - 2r^*$. From the definition of r^*, we observe that any point within $\mathcal{A}'$ will be at most r^* away from a sensor in S^*. This implies that the set of circles of radii r^* centered at each sensor in S^* cover $\mathcal{A}'$. Hence,

$$n^* \pi r^{*2} \ge (d - 2r^*)^2,$$
$$\therefore \quad n^* \ge \frac{(d-2r^*)^2}{\pi r^{*2}}.$$

This completes the proof. $\square$

B.2 Proofs for the Upper Bounds

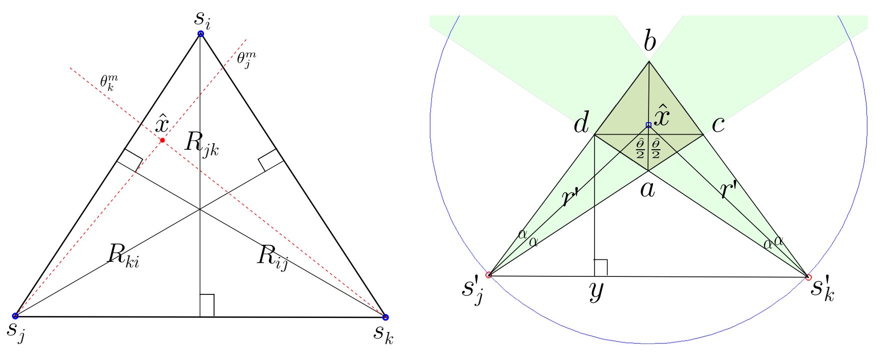
B.2.1 Proof for Lemma 14

We break down the proof for Lemma 14 into the following three lemmas for each of the following cases.

First consider the case of the area of intersection of wedges from two sensors. Let $S_{ALG} = \{s_i, s_j, s_k\}$ be three sensors forming an equilateral triangle of side r (Figure B.2(a)). We divide $\triangle s_i s_j s_k$ into three equal regions R_{ij}, R_{jk} and R_{ki} as shown in Figure B.2(a) using three perpendicular bisectors. Suppose the true target $x \in R_{jk}$. We begin by bounding Area$(W(s_j, \theta_j^m) \cap W(s_k, \theta_k^m))$, where θ_j^m, θ_k^m are any two valid measurements from s_j and s_k respectively. Let $\hat{x}$ be the intersection of rays along θ_j^m and θ_k^m. Note that $\hat{x}$ is not necessarily coincident with x but $\hat{x}, x \in W(s_j, \theta_j^m) \cap W(s_k, \theta_k^m)$. Then we have the following:

Lemma 26 (Intersection of two wedges)**.** *When* $0 < \alpha < \dfrac{\pi}{18}$,

$$\max_{x \in R_{jk}} \max_{\theta_i^m, \theta_j^m} \mathit{Diameter}(W(s_j, \theta_j^m) \cap W(s_k, \theta_k^m)) \le 11.35 r \sin\alpha$$



(a) (b)

Figure B.2: (a) We divide the interior of the $\triangle s_i s_j s_k$ with perpendicular bisectors into R_{ij}, R_{jk}, R_{ki}. For $x \in R_{jk}$, we show that the area of intersection of any valid sensing wedges from s_j and s_k is bounded. (b) Intersection of sensing wedges from s'_j and s'_k is a kite. s'_j and s'_k are obtained by extending s_j and s_k along lines θ_j^m and θ_k^m with $\hat{x}s'_j = \hat{x}s'_k = r'$. This kite bounds the intersection of the original sensing wedges.

and,

$$\max_{x \in R_{jk}} \max_{\theta_i^m, \theta_j^m} \mathit{Area}(W(s_j, \theta_j^m) \cap W(s_k, \theta_k^m)) \leq 23.46 r^2 \sin^2(\alpha)$$

Proof. Consider Figure B.2(a). We have $\frac{\pi}{3} \leq \angle s_j x s_k \leq \frac{2\pi}{3}$ and $d(s_j, x), d(s_k, x) \leq r$, where $d(\cdot,\cdot)$ gives the distance between two points. We now bound $d(s_j, \hat{x})$, $d(s_k, \hat{x})$ and $\angle s_j \hat{x} s_k$. We have: $\angle s_j \hat{x} s_k = \pi - \angle \hat{x} s_j s_k - \angle \hat{x} s_k s_j$.

$\therefore\ \min \angle s_j \hat{x} s_k = \pi - \max \angle \hat{x} s_j s_k - \max \angle \hat{x} s_k s_j \ \therefore\ \min \angle s_j \hat{x} s_k \quad = \pi - 2(\dfrac{\pi}{3} + \alpha) = \left(\dfrac{\pi}{3} - 2\alpha\right).$

Similarly $\max \angle s_j \hat{x} s_k = \pi - \min \angle \hat{x} s_j s_k - \min \angle \hat{x} s_k s_j = \left(\frac{2\pi}{3} + 2\alpha\right)$. Hence, we have $\left(\frac{\pi}{3} - 2\alpha\right) \leq \angle s_j \hat{x} s_k \leq \left(\frac{2\pi}{3} + 2\alpha\right)$. Since both $\angle \hat{x} s_j s_k$ and $\angle \hat{x} s_k s_j$ are acute, $s_j s_k = s_j y + s_k y$. Consider $\triangle \hat{x} y s_k$. By law of sines we have,

$$\frac{\sin(\angle s_j \hat{x} s_k)}{s_j s_k} = \frac{\sin(\angle \hat{x} s_j s_k)}{\hat{x} s_k}$$

$$\hat{x} s_k = \frac{\sin\left(\angle \hat{x} s_j s_k\right)}{\sin\left(\angle s_j \hat{x} s_k\right)} s_j s_k = \frac{\sin\left(\angle \hat{x} s_j s_k\right)}{\sin\left(\angle s_j \hat{x} s_k\right)} r \leq \frac{\sin\left(\frac{\pi}{3} + \alpha\right)}{\sin\left(\angle s_j \hat{x} s_k\right)} r$$

Let $\hat{\theta} \triangleq \angle s_j \hat{x} s_k$ and $r' \triangleq \dfrac{\sin\left(\frac{\pi}{3} + \alpha\right)}{\sin\left(\hat{\theta}\right)} r$. By symmetry we have $d(s_j, \hat{x}), d(s_k, \hat{x}) \leq r'$.

The actual distance varies depending on θ_j^m, θ_k^m. In order to bound Area$(W(s_j, \theta_j^m) \cap W(s_k, \theta_k^m))$ for all possible θ_j^m, θ_k^m, we construct the following instance:

- For any θ_j^m, θ_k^m draw a circle C, centered at $\hat{x}$ with radius r'.
- Extend segment from $\hat{x}$ towards s_j and s_k. Let s_j' and s_k' be its points of intersection with C. Note that $s_j, s_k \in C$, $d(s_j', x) = d(s_k', x) = r'$, and $W(s_j, \theta_j^m) \subseteq W(s_j', \theta_j'^m)$ and $W(s_k, \theta_k^m) \subseteq W(s_k', \theta_k'^m)$.

We have, Area$\left(W(s_j, \theta_j^m) \cap W(s_k, \theta_k^m)\right) \leq$ Area$\left(W(s_j', \theta_j^m) \cap W(s_k', \theta_k^m)\right)$. We first show that the intersection of $h_{i'}^l, h_{j'}^r, h_{k'}^l, h_{l'}^r$ is a bounded quadrilateral $acbd$ (Figure B.2(b)). First consider the intersection of $h_{j'}^r$ with $h_{k'}^l$ denoted by a. We have,

$$\begin{aligned} \angle s_j' a s_k' &= \pi - 2\left(\angle a s_j' s_k'\right) = \pi - 2\left(\frac{\pi}{2} - \frac{\hat{\theta}}{2} - \alpha\right) \\ &= \hat{\theta} + 2\alpha \leq \frac{2\pi}{3} + 4\alpha < \pi. \end{aligned}$$

Hence a lies on the same side of $s_j's_k'$ as $\hat{x}$. Further this implies that the intersection of $h_{j'}^l$ with $h_{j'}^r$ (i.e., s_j') and $h_{k'}^l$ with $h_{k'}^r$ (i.e., s_k') does not lie within or on the intersection region of the sensing wedges. Next consider the intersection of $h_{j'}^l$ with $h_{k'}^r$ denoted by b. Suppose that b does not exist. Let $b_j \neq s_j'$ and $b_k \neq s_k'$ be any points on $h_{j'}^l$ and $h_{k'}^r$ respectively. Since $h_{j'}^l$ and $h_{k'}^r$ do not intersect we have,

$$\begin{aligned} \angle b_j s_j' s_k' = \angle b_j s_j' \hat{x} + \angle \hat{x} s_j' s_k' \geq \frac{\pi}{2} \\ \therefore\ \alpha + \frac{\pi}{2} - \frac{\hat{\theta}}{2} \geq \frac{\pi}{2} \implies \alpha \geq \frac{\pi}{12} \end{aligned}$$

which is a contradiction since $0 < \alpha < \frac{\pi}{18}$. Since a and b exist the intersection of $h_{j'}^r$ with $h_{k'}^r$ (denoted by c) and $h_{j'}^l$ with $h_{k'}^l$ (denoted by d) also exist. By symmetry, it is easy to show that $\triangle bcs_j' \cong \triangle bds_k'$ and $\triangle das_j' \cong \triangle cas_k'$ so that $bc = bd$ and $ac = ad$. Thus, $acbd$ is a kite. Furthermore, $\hat{x}$ is collinear with a and b since $\triangle s_j'\hat{x}s_k'$ and $\triangle s_j'as_k'$ are both isosceles.

By property of a kite, $ab \perp cd$ and Area$(acbd) = 0.5ab \times cd$. Next, we compute the length of the two diagonals ab and cd. First we find $ab = a\hat{x} + \hat{x}b$.

We have $\triangle s'_j \hat{x} a \cong \triangle s'_k \hat{x} a$. Hence, $\angle s'_j \hat{x} a = \dfrac{\angle s'_j \hat{x} s'_k}{2}$. Thus, $\angle s'_j \hat{x} a = \dfrac{\hat{\theta}}{2}$. Further $\angle s'_j a \hat{x} = \pi - \left(\dfrac{\hat{\theta}}{2} + \alpha\right)$ yielding $\angle s'_j b \hat{x} = \angle s'_j ba = \left(\dfrac{\hat{\theta}}{2} - \alpha\right)$. By law of sines in $\triangle s'_j b \hat{x}$,

$$\hat{x}b = \frac{\sin\left(\angle b s'_j \hat{x}\right)}{\sin\left(\angle s'_j b \hat{x}\right)} s'_j \hat{x} = \frac{\sin(\alpha)}{\sin\left(\dfrac{\hat{\theta}}{2} - \alpha\right)} r'.$$

Similarly applying law of sines in $\triangle s'_j \hat{x} a$ we get $a\hat{x} = \dfrac{\sin(\alpha)}{\sin\left(\dfrac{\hat{\theta}}{2} + \alpha\right)} r'$. Hence,

$$ab = a\hat{x} + \hat{x}b = r' \sin\alpha \left[\sin^{-1}\left(\frac{\hat{\theta}}{2} + \alpha\right) + \sin^{-1}\left(\frac{\hat{\theta}}{2} - \alpha\right)\right] \tag{B.1}$$

If we extend segment from b to a onto $s'_j s'_k$, by symmetry we can show that the segment is a perpendicular bisector of $s'_j s'_k$. Since $cd \perp ab$, quadrilateral $abcd$ forms a trapezoid. We have,

$$\angle c s'_k s'_j = \angle c s'_k \hat{x} + \angle \hat{x} s'_k s'_j = \alpha + \left(\frac{\pi}{2} - \frac{\hat{\theta}}{2}\right) = \frac{\pi}{2} - \left(\frac{\hat{\theta}}{2} - \alpha\right).$$

Hence $\angle dcs'_k = \pi - \angle c s'_k s'_j = \dfrac{\pi}{2} + \left(\dfrac{\hat{\theta}}{2} - \alpha\right)$ and $\angle d s'_k s'_j = \dfrac{\pi}{2} - \left(\dfrac{\hat{\theta}}{2} + \alpha\right)$.

Drop a perpendicular from d onto $s'_j s'_k$ and let y be its point of intersection. Since $\angle b s'_k s'_j, b s'_j s'_k < \dfrac{\pi}{2}$, y lies between s'_j and s'_k. Now we get,

$$y s'_k = d s'_k \cos\left(\frac{\pi}{2} - \left(\frac{\hat{\theta}}{2} + \alpha\right)\right) = d s'_k \sin\left(\frac{\hat{\theta}}{2} + \alpha\right)$$

and

$$dy = d s'_k \sin\left(\frac{\pi}{2} - \left(\frac{\hat{\theta}}{2} + \alpha\right)\right) = d s'_k \cos\left(\frac{\hat{\theta}}{2} + \alpha\right)$$

Further,

$$\angle d s'_j s'_k = \angle b s'_j s'_k = \angle b s'_j \hat{x} + \angle \hat{x} s'_j s'_k = \alpha + \frac{\pi}{2} - \frac{\hat{\theta}}{2} = \frac{\pi}{2} - \left(\frac{\hat{\theta}}{2} - \alpha\right).$$

Therefore,

$$s'_j y = dy \cot\left(\angle ds'_j y\right) = dy \tan\left(\frac{\hat{\theta}}{2} - \alpha\right) = ds'_k \cos\left(\frac{\hat{\theta}}{2} + \alpha\right) \tan\left(\frac{\hat{\theta}}{2} - \alpha\right)$$

We can now solve for ds'_k,

$$\begin{aligned} s'_j s'_k &= s'_j y + y s'_k \\ \therefore\ 2r' \sin\left(\frac{\hat{\theta}}{2}\right) &= ds'_k \cos\left(\frac{\hat{\theta}}{2} + \alpha\right) \tan\left(\frac{\hat{\theta}}{2} - \alpha\right) + ds'_k \sin\left(\frac{\hat{\theta}}{2} + \alpha\right) \\ \therefore\ ds'_k &= \frac{2r' \sin\left(\frac{\hat{\theta}}{2}\right)}{\sin\left(\frac{\hat{\theta}}{2} + \alpha\right) + \cos\left(\frac{\hat{\theta}}{2} + \alpha\right) \tan\left(\frac{\hat{\theta}}{2} - \alpha\right)} \end{aligned}$$

By law of sines in $\triangle dcs'_k$,

$$\begin{aligned} \frac{cd}{\sin\left(\angle cs'_k d\right)} &= \frac{ds'_k}{\sin\left(\angle dcs'_k\right)} \\ \therefore\ cd &= \frac{2r' \sin(2\alpha) \sin\left(\frac{\hat{\theta}}{2}\right)}{\sin\left(\frac{\pi}{2} + \left(\frac{\hat{\theta}}{2} - \alpha\right)\right) \left[\sin\left(\frac{\hat{\theta}}{2} + \alpha\right) + \cos\left(\frac{\hat{\theta}}{2} + \alpha\right) \tan\left(\frac{\hat{\theta}}{2} - \alpha\right)\right]} \\ &= \frac{4r' \sin(\alpha) \cos(\alpha) \sin\left(\frac{\hat{\theta}}{2}\right)}{\cos\left(\frac{\hat{\theta}}{2} - \alpha\right) \left[\sin\left(\frac{\hat{\theta}}{2} + \alpha\right) + \cos\left(\frac{\hat{\theta}}{2} + \alpha\right) \tan\left(\frac{\hat{\theta}}{2} - \alpha\right)\right]} \\ &= \frac{4r' \sin(\alpha) \cos(\alpha) \sin\left(\frac{\hat{\theta}}{2}\right)}{\sin\left(\hat{\theta}\right)} = \frac{2r' \sin(\alpha) \cos(\alpha)}{\cos\left(\frac{\hat{\theta}}{2}\right)} \end{aligned} \tag{B.2}$$

Using Equations B.1 and B.2 we get,

$$\begin{aligned} \text{Area}(acbd) &= 0.5ab \times cd \\ &= \frac{0.5 \times 2r'^2 \sin^2(\alpha) \cos(\alpha) \left(\sin^{-1}\left(\frac{\hat{\theta}}{2} + \alpha\right) + \sin^{-1}\left(\frac{\hat{\theta}}{2} - \alpha\right)\right)}{\cos\left(\frac{\hat{\theta}}{2}\right)} \end{aligned}$$

Now, $r' = r\frac{\sin\left(\frac{\pi}{3}+\alpha\right)}{\sin\left(\hat{\theta}\right)} \leq r\frac{\sin\left(\frac{\pi}{3}+\frac{\pi}{18}\right)}{\sin\left(\hat{\theta}\right)} = \frac{0.9397r}{\sin\left(\hat{\theta}\right)}$.

Substituting,

$$\text{Area}(acbd) \leq \frac{0.9397^2 r^2 \sin^2(\alpha)\cos(\alpha)\left(\sin^{-1}\left(\frac{\hat{\theta}}{2}+\alpha\right)+\sin^{-1}\left(\frac{\hat{\theta}}{2}-\alpha\right)\right)}{\sin^2\left(\hat{\theta}\right)\cos\left(\dfrac{\hat{\theta}}{2}\right)}$$

Recall that we have $\left(\frac{\pi}{3}-2\alpha\right) \leq \hat{\theta} \leq \left(\frac{2\pi}{3}+2\alpha\right)$ and $0 < \alpha < \frac{\pi}{18}$. We split this into the following: (a) $\left(\frac{\pi}{3}-2\alpha \leq \hat{\theta} \leq \frac{\pi}{2}\right)$ and (b) $\left(\frac{\pi}{2} < \hat{\theta} \leq \frac{2\pi}{3}+2\alpha\right)$.

For (a) we get,

$$\text{Area}(acbd) \leq \frac{0.9397^2 r^2 \sin^2(\alpha)\left(\sin^{-1}\left(\frac{\pi}{6}\right)+\sin^{-1}\left(\frac{\pi}{18}\right)\right)}{\sin^2\left(\frac{2\pi}{9}\right)\cos\left(\frac{\pi}{4}\right)} \qquad \leq 23.46r^2\sin^2(\alpha).$$

Similarly for (b) we get,

$$\text{Area}(acbd) \leq \frac{0.9397^2 r^2 \sin^2(\alpha)\left(\sin^{-1}\left(\frac{\pi}{4}\right)+\sin^{-1}\left(\frac{7\pi}{36}\right)\right)}{\sin^2\left(\frac{2\pi}{9}\right)\cos\left(\frac{7\pi}{18}\right)} \qquad \leq 19.74r^2\sin^2(\alpha).$$

The diameter for the intersection of wedges is bounded by the diameter for $acbd$. We have,

$$\begin{aligned}\max_{x\in R_{jk}} \max_{\theta_i^m,\theta_j^m} \text{Diameter}(W(s_j,\theta_j^m)\cap W(s_k,\theta_k^m)) &\leq \text{Diameter}(acbd)\\ &= \max\{ab, cd\}\\ &\leq 11.35r\sin\alpha,\end{aligned}$$

where the last step is obtained by substituting the expressions for ab and cd using Equation B.1 and B.2 and finding the bound similar to the area case. □

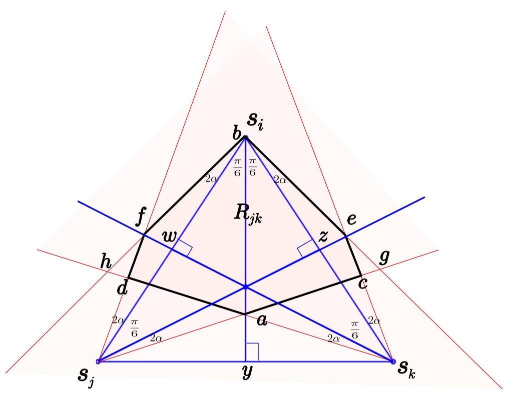
For other cases since $\alpha \geq \frac{\pi}{18}$, we know from Lemma 13 that $U(S^*)$ is lower bounded by $\pi r^{*2}\sin^2\left(\frac{\pi}{18}\right) \approx 0.095r^{*2}$ unlike in Lemma 26 where the lower bound could be arbitrarily small. However since $\alpha \geq \frac{\pi}{18}$ we cannot use just two sensors to get a good approximation especially when the angle between the sensors and the target is close to $\frac{\pi}{3}$. Instead we approximate the sensing wedges from three neighboring sensors in the next lemma.

Lemma 27 (Intersection of three sensing wedges)**.** *When* $\frac{\pi}{18} \leq \alpha < \frac{\pi}{12}$,

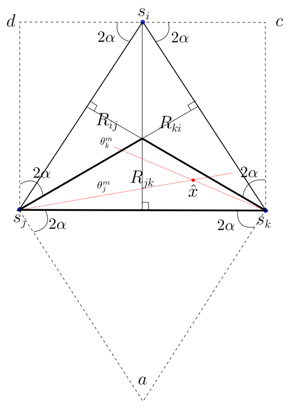
$$\max_{x \in R_{jk}} \max_{\theta_i^m, \theta_j^m, \theta_k^m} \mathit{Diameter}\left(W(s_i, \theta_i^m) \cap W(s_j, \theta_j^m) \cap W(s_k, \theta_k^m)\right) \leq 2.04r$$

and

$$\max_{x \in R_{jk}} \max_{\theta_i^m, \theta_j^m, \theta_k^m} \mathit{Area}\left(W(s_i, \theta_i^m) \cap W(s_j, \theta_j^m) \cap W(s_k, \theta_k^m)\right) \leq \frac{\sqrt{3}r^2}{4} + 10.1r^2 \sin^2 \alpha.$$



(a)



(b)

Figure B.3: Instead of projecting the sensors back, as in Figure B.2, we approximate each sensor by taking the union of all feasible sensing wedges when $x \in R_{jk}$. For (a) $\frac{\pi}{18} \leq \alpha < \frac{\pi}{12}$, and (b) $\frac{\pi}{12} \leq \alpha < \frac{\pi}{6}$. Note for R_{jk} is different for (a) and (b).

Proof. Consider Figure B.3(a) where s_i, s_j, s_k are three neighboring sensors. $x \in R_{jk}$ where R_{jk} is defined as in Lemma 26. We have $\frac{\pi}{3} \leq \angle x s_j s_k, \angle x s_k s_j \leq \frac{2\pi}{3}$ and $0 \leq \angle x s_i s_j \leq \frac{\pi}{3}$. For each sensor we are going to build a larger sensing wedge with the union of sensing wedges $\forall x \in R_{jk}$. Denote such a sensing wedge by $\hat{W}(s_t, R_{jk})$ where $t = i, j, k$. $\hat{W}$ is an approximation to the actual sensing wedges: $W(s_t, \theta_t^m) \subset \hat{W}(s_t, R_{jk})$, $\forall x \in R_{jk}$. We have,

$$\max_{x \in R_{jk}} \max_{\theta_i^m, \theta_j^m, \theta_k^m} \text{Area}\left(\bigcap_{t=i,j,k} W(s_t, \theta_t^m)\right) \leq \text{Area}\left(\bigcap_{t=i,j,k} \hat{W}(s_t, R_{jk})\right)$$

$\hat{W}(s_j, R_{jk})$ and $\hat{W}(s_k, R_{jk})$ have an angular width of $\frac{\pi}{6} + 4\alpha$ whereas $\hat{W}(s_i, R_{jk})$ has $\frac{\pi}{3} + 4\alpha$. Let $acebfd$ be the six-sided convex polygon formed by the intersection of three $\hat{W}$'s as shown in Figure B.3(a). Since there are six bounding half-lines, we have fifteen cases to consider.

We denote the bounding half-planes of $\hat{W}(s_t, R_{jk})$ by $\hat{h}_t^l$ and $\hat{h}_t^r$. Consider point of intersection of $\hat{h}_j^r$ and $\hat{h}_k^l$, denoted by a. In $\triangle s_j a s_k$ we have,

$$\angle s_j a s_k = \pi - 2\left(\frac{\pi}{6} - 2\alpha\right) = \frac{2\pi}{3} + 4\alpha \leq \pi.$$

Hence a lies to the same side of $s_j s_k$ as s_i and thus s_j (intersection of $\hat{h}_j^l$ and $\hat{h}_j^r$) and s_k (intersection of $\hat{h}_k^l$ and $\hat{h}_k^r$) do not lie on or within the intersecting polygon. Now consider the intersection of $\hat{h}_j^l$ and $\hat{h}_i^r$, denoted by f. In $\triangle s_i f s_j$ we have,

$$\angle s_i f s_j = \pi - 4\alpha.$$

Since $\dfrac{\pi}{18} \leq \alpha < \dfrac{\pi}{12}$,

$$\frac{2\pi}{3} < \angle s_i f s_j \leq \frac{7\pi}{9}.$$

Denote by d the point of intersection of $\hat{h}_j^l$ and $\hat{h}_k^l$. We have

$$\angle s_j d s_k = \pi - \left(\frac{\pi}{3} + 2\alpha + \frac{\pi}{6} - 2\alpha\right) = \frac{\pi}{2}.$$

The considerations for e and c (as shown in Figure B.3(a)) are symmetric to f and d respectively. Vertex b is coincident with s_i, hence the points of intersection of $\hat{h}_j^r$ with $\hat{h}_i^r$, $\hat{h}_j^l$ with $\hat{h}_i^l$, $\hat{h}_k^r$ with $\hat{h}_i^r$, and $\hat{h}_k^l$ with $\hat{h}_i^l$ do not lie within or on the intersecting polygon. Similarly, the point of intersection of $\hat{h}_j^l$ with $\hat{h}_k^r$ is at a height of $\left(\dfrac{r}{2}\tan\left(\dfrac{\pi}{3} + 2\alpha\right) > \dfrac{\sqrt{3}}{2}r\right)$ perpendicular to $s_j s_k$ and hence outside $\hat{W}(s_i, R_{jk})$. Finally consider g and h, the points of intersection of $\hat{h}_j^r$ with $\hat{h}_i^l$ and $\hat{h}_k^l$ with $\hat{h}_i^r$ respectively. Since $\angle s_j d s_k = \dfrac{\pi}{2}$ i.e., $\hat{h}_j^l \perp \hat{h}_k^l$ and $\hat{h}_i^r$ intersects $\hat{h}_j^l$, then s_i and h cannot lie on the same side of $\hat{h}_j^l$. Since $s_i = b$ lies on the intersecting polygon, hence h does not. The case for g is symmetric.

We can verify that the vertices appear cyclically on the intersection polygon in the order $acebfd$. Hence, the intersecting polygon of the three $\hat{W}$ wedges is the six-sided convex polygon $acebfd$.

Now we compute and bound the area of the polygon:

$$\text{Area}(acebfd) = \text{Area}(agbh) - 2\text{Area}(\triangle dfh) = 0.5ab \times gh - df \times hd. \tag{B.3}$$

The above equation follows from the properties that $\triangle dfh \cong \triangle ceg$ and quadrilateral $agbh$ is a kite due to symmetry. Next we compute each of the lengths in turn. Drop a perpendicular from a onto s_js_k and denote its point of intersection with y. Since $\triangle s_jas_k$ is isosceles, y, a and b are collinear. We have,

$$\begin{aligned} ab &= by - ay \\ &= \frac{\sqrt{3}r}{2} - \frac{r}{2}\tan\left(\frac{\pi}{6} - 2\alpha\right) \\ &= \frac{r}{2}\left(\sqrt{3} - \tan\left(\frac{\pi}{6} - 2\alpha\right)\right) \end{aligned} \tag{B.4}$$

Similarly f, w and s_k are collinear. Thus,

$$\begin{aligned} s_kf &= s_kw + wf \\ &= \frac{\sqrt{3}r}{2} + \frac{r\tan(2\alpha)}{2} = \frac{r}{2}\left(\sqrt{3} + \tan(2\alpha)\right). \\ df &= s_kf\sin(2\alpha) = \frac{r\sin(2\alpha)}{2}\left(\sqrt{3} + \tan(2\alpha)\right). \\ ds_k &= s_kf\cos(2\alpha) = \frac{r\cos(2\alpha)}{2}\left(\sqrt{3} + \tan(2\alpha)\right). \end{aligned}$$

Consider $\triangle s_jds_k$,

$$ds_k = s_js_k\sin\left(\frac{\pi}{3} + 2\alpha\right) = r\sin\left(\frac{\pi}{3} + 2\alpha\right)$$

Now in $\triangle dfs_k$ we get,

$$df = ds_k\tan(2\alpha) = r\sin\left(\frac{\pi}{3} + 2\alpha\right)\tan(2\alpha) \tag{B.5}$$

In $\triangle dfh$ we have $\angle hdf = \dfrac{\pi}{2}$ and

$$\angle hfd = \pi - \angle s_jfs_i = \pi - (\pi - 4\alpha) = 4\alpha.$$

Therefore,

$$hd = df\tan(\angle hfd) = r\sin\left(\frac{\pi}{3} + 2\alpha\right)\tan(2\alpha)\tan(4\alpha) \tag{B.6}$$

Finally since $agbh$ is a kite, ab is a perpendicular bisector of gh giving

$$gh = 2ah\sin(\angle hab) = 2ah\sin\left(\frac{\angle s_jas_k}{2}\right) = 2ah\sin\left(\frac{\pi}{3} + 2\alpha\right)$$

where,

$$\begin{aligned} ah &= ds_k + hd - s_k a \\ &= r\sin\left(\frac{\pi}{3}+2\alpha\right) + r\sin\left(\frac{\pi}{3}+2\alpha\right)\tan(2\alpha)\tan(4\alpha) - \frac{r}{2\sin\left(\frac{\pi}{3}+2\alpha\right)}. \end{aligned}$$

Substituting we get,

$$gh = 2r\sin^2\left(\frac{\pi}{3}+2\alpha\right)\left[1+\tan(2\alpha)\tan(4\alpha)\right] - r. \tag{B.7}$$

Now since $\frac{\pi}{18} \le \alpha < \frac{\pi}{12}$ from Equations B.4 and B.7 we get,

$$\begin{aligned} ab \times gh &= \frac{r}{2}\left[\sqrt{3} - \tan\left(\frac{\pi}{6}-2\alpha\right)\right] \times \left[2r\sin^2\left(\frac{\pi}{3}+2\alpha\right)\left[1+\tan(2\alpha)\tan(4\alpha)\right] - r\right] \\ &\le \frac{\sqrt{3}r}{2} \times \left[2r\sin^2\left(\frac{\pi}{3}+2\alpha\right)\left[1+\tan(2\alpha)\tan(4\alpha)\right] - r\right] \\ &\le \sqrt{3}r^2\left(1+\tan 2\alpha\tan 4\alpha\right) - \frac{\sqrt{3}r^2}{2} \\ &= \frac{\sqrt{3}r^2}{2} + \frac{8\sqrt{3}r^2\sin^2\alpha\cos^2\alpha}{\cos 4\alpha} \\ &\le \frac{\sqrt{3}r^2}{2} + \frac{8\sqrt{3}r^2\sin^2\alpha\cos^2\left(\frac{\pi}{18}\right)}{\cos\left(\frac{\pi}{3}\right)} \\ &\le \frac{\sqrt{3}r^2}{2} + 26.88r^2\sin^2\alpha. \end{aligned}$$

Also from Equations B.5 and B.6,

$$\begin{aligned} df \times hd &= r^2\sin^2\left(\frac{\pi}{3}+2\alpha\right)\tan^2 2\alpha\tan 4\alpha \\ &\ge r^2\sin^2\left(\frac{\pi}{3}+\frac{\pi}{9}\right)\tan 2\alpha\left(\tan 2\alpha\tan 4\alpha\right) \\ &\ge r^2\sin^2\left(\frac{4\pi}{9}\right)\tan 2\alpha\left(\frac{8\sin^2\alpha\cos^2\alpha}{\cos 4\alpha}\right) \\ &\ge 8r^2\sin^2\alpha\sin^2\left(\frac{4\pi}{9}\right)\tan\left(\frac{\pi}{9}\right)\left(\frac{\cos^2\left(\frac{\pi}{12}\right)}{\cos\left(\frac{2\pi}{9}\right)}\right) \\ &\ge 3.439r^2\sin^2\alpha. \end{aligned}$$

Finally substituting in Equation B.3,

$$\begin{aligned} \text{Area}(acebfd) &\le \frac{\sqrt{3}r^2}{4} + 13.44r^2\sin^2\alpha - 3.439r^2\sin^2\alpha \\ &= \frac{\sqrt{3}r^2}{4} + 10.1r^2\sin^2\alpha. \end{aligned}$$

The bound for diameter follows immediately,

$$\begin{aligned}\max_{x\in R_{jk}} \max_{\theta_i^m,\theta_j^m,\theta_k^m} \text{Diameter}(W(s_i,\theta_i^m)\cap W(s_j,\theta_j^m)\cap W(s_k,\theta_k^m)) &\le \text{Diameter}(acebfd)\\ &\le \text{Diameter}(agbh)\\ &= \max\{ab, gh\}\\ &\le 2.04r\sin\alpha\end{aligned}$$

where the last step is obtained by substituting the values of ab and gh. □

Next we consider $\frac{\pi}{12} \le \alpha < \frac{\pi}{6}$ in the following lemma. Instead of using R_{jk} as in Figure B.3(a) we now use the partition as shown in Figure B.3(b). Similar to the previous lemma, we consider for each sensor the union of all possible sensing wedges, and show that the intersection of such wedges for the three sensors is bounded.

Lemma 28 (Intersection of three sensing wedges)**.** *When* $\dfrac{\pi}{12} \le \alpha < \dfrac{\pi}{6}$,

$$\max_{x\in R_{jk}} \max_{\theta_i^m,\theta_j^m,\theta_k^m} \textit{Diameter}\left(W(s_i,\theta_i^m)\cap W(s_j,\theta_j^m)\cap W(s_k,\theta_k^m)\right) \le \left(1+\frac{1}{\sqrt{3}}\right)r.$$

and

$$\max_{x\in R_{jk}} \max_{\theta_i^m,\theta_j^m,\theta_k^m} \textit{Area}\left(W(s_i,\theta_i^m)\cap W(s_j,\theta_j^m)\cap W(s_k,\theta_k^m)\right) \le \frac{3\sqrt{3}}{4}r^2.$$

Proof. We use a procedure similar to the previous lemma, except that the assigned regions $x \in R_{jk}$ is changed (Figure B.3(b)). We have $\dfrac{2\pi}{3} \le \angle s_j x s_k \le \pi$ and $0 \le \angle s_j x s_i \le \dfrac{\pi}{3}$. For sensors s_j and s_k, we draw sensing wedges $\hat{W}(s_j,R_{jk})$ and $\hat{W}(s_k,R_{jk})$ that approximate any valid sensing wedges. As before, we denote the bounding half-lines of $\hat{W}(s_j,R_{jk})$ ($\hat{W}(s_j,R_{jk})$) by $\hat{h}_j^l$ and $\hat{h}_j^r$ ($\hat{h}_j^l$ and $\hat{h}_j^r$ respectively). We consider the intersection of all the bounding half-lines and bound the area using these points.

Since $\alpha < \dfrac{\pi}{6}$, we draw $\hat{h}_j^r$ by rotating line through $s_j s_k$ clockwise by $\dfrac{\pi}{3}$ about s_j. Similarly $\hat{h}_j^l$ is drawn by rotating the line through s_j and perpendicular to $s_i s_k$ counter-clockwise by $\dfrac{\pi}{3}$ about s_j. $\hat{h}_k^l$ and $\hat{h}_k^r$ are drawn symmetrically. $\hat{h}_i^l$ ($\hat{h}_i^r$) are drawn by rotating line $s_i s_k$ ($s_i s_j$) counter-clockwise (clockwise) by $\dfrac{\pi}{3}$ about s_i. Note that for all $x \in R_{jk}$, for any valid θ_j^m and θ_k^m we have $\hat{W}(s_j,R_{jk}) \supseteq W(s_j,\theta_j^m)$ and $\hat{W}(s_j,R_{jk}) \supseteq W(s_j,\theta_j^m)$.

Denote by d the intersection of $\hat{h}_i^r$ with $\hat{h}_j^l$, by c the intersection of $\hat{h}_i^l$ with $\hat{h}_k^r$, and by a the intersection of $\hat{h}_j^r$ with $\hat{h}_k^l$. We can see that dcs_ks_j is a rectangle with sides $s_js_k = dc = r$ and $s_jd = s_kc = \dfrac{\sqrt{3}r}{3}$. Further $\triangle s_js_ka$ is an equilateral triangle with side r. Hence,

$$\begin{aligned}&\max_{x\in R_{jk}} \max_{\theta_i^m,\theta_j^m,\theta_k^m} \text{Area}\left(W(s_i,\theta_i^m)\cap W(s_j,\theta_j^m)\cap W(s_k,\theta_k^m)\right)\\ &\leq \text{Area}\,(s_js_kcd) + \text{Area}\,(s_js_ka) = \frac{\sqrt{3}}{2}r^2 + \frac{\sqrt{3}}{4}r^2 = \frac{3\sqrt{3}}{4}r^2.\end{aligned}$$

The bound on the diameter follows:

$$\begin{aligned}\max_{x\in R_{jk}} \max_{\theta_i^m,\theta_j^m,\theta_k^m} \text{Diameter}\left(W(s_i,\theta_i^m)\cap W(s_j,\theta_j^m)\cap W(s_k,\theta_k^m)\right) &\leq \text{Diameter}(as_jdcs_k)\\ &= ad = \left(1+\frac{1}{\sqrt{3}}\right)r\end{aligned}$$

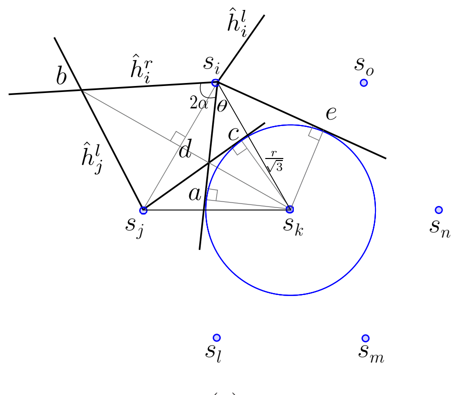
□

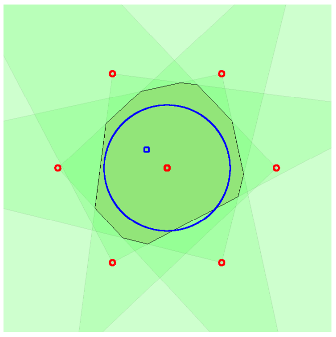
Lemma 14. Using Lemmas 26, 27, and 28. □

B.2.2 Proof for Lemma 15

Proof. Consider Figure B.4(a). Sensors $s_i,\ldots,s_o$ are seven neighboring sensors in S_{ALG}: s_k is the center of a regular hexagon of side length r formed by $s_i, s_j, s_l, s_m, s_n, s_o$. Instead of considering that the target lies in one of the triangular faces, we instead draw a circle of radius $r' = \dfrac{r}{\sqrt{3}}$ centered at each sensor and bound the uncertainty when the true target lies in one of such circles. The union of all such circles covers $\mathcal{A}$ and hence, the uncertainty bound holds for any true target location. Denote by C_k the circle corresponding to sensor s_k as shown in Figure B.4(a), and let $x \in C_k$.

As in the previous lemmas, we approximate the sensing wedges from each sensor and then bound their intersection area. Consider sensor s_i in Figure B.4(a). When $x \in C_k$, θ_i^t is bounded between the two tangents from s_i to C_k i.e., between within $\angle as_ie$. From $\triangle s_ias_k$ we have, $\angle as_is_k = \sin^{-1}\left(\frac{as_k}{s_is_k}\right) = \sin^{-1}\left(\frac{r}{\sqrt{3}r}\right) = \sin^{-1}\left(\frac{1}{\sqrt{3}}\right)$. Define $\theta \triangleq \sin^{-1}\left(\frac{1}{\sqrt{3}}\right)$. By symmetry, $\angle ds_is_k = \angle cs_js_k = \theta$. Recall that the sensing wedge is defined as the intersection of two bounding half-planes.





(a) (b)

Figure B.4: Bounding the intersecting area with 6 sensors $s_i, s_j, s_l, s_m, s_n, s_o$. The target can be anywhere within a circle C of radius $\frac{r}{\sqrt{3}}$ centered at sensor s_k. (a) We ignore s_k and approximate each other sensor by taking the union of all valid sensing wedges when $x \in C$. The resulting shape is a star with six petals. (b) An instance of execution with randomly generated target (blue square) and measurements with $\pm\alpha = \frac{\pi}{4}$.

We now approximate *all* sensing wedges from s_i (when $x \in C_k$) by a larger wedge $\hat{W}(s_i, C_k)$ using the following operations: (a) Draw $\hat{h}_i^r$ by rotating line $s_i a$ pivoted at s_i, clockwise by an angle of 2α. (b) Draw $\hat{h}_i^l$ by rotating line $s_i b$ pivoted at s_i, counter-clockwise by an angle of 2α. (c) $\hat{W}(s_i, C_k)$ is defined as the intersection of the two half-planes given by $\hat{h}_i^l$ and $\hat{h}_i^r$. We can see that $\hat{W}(s_i, C^k)$ as defined above ensures that $W(s_i, \theta_i^m) \subset \hat{W}(s_i, C_k)$ for all θ_i^m. We can similarly define $\hat{W}$ for all other sensors. The intersection of any combination of $W(s_t, \theta_t^m)$ sensing wedges lie completely inside the intersection of $\hat{W}$ sensing wedges. Hence, we compute the area of intersection of all $\hat{W}$ sensing wedges next.

Consider two adjacent sensors s_i and s_j as shown in Figure B.4(a). We first show that $\hat{h}_i^r$ and $\hat{h}_j^l$ intersect at a point b such that $s_k s_j b s_i$ forms a kite. Suppose not. Let b_i and b_j be any points on $\hat{h}_i^r$ and $\hat{h}_j^l$ lying on opposite sides of the line $s_i s_j$ as s_k. Hence we must have $\angle b_i s_i s_j, b_j s_j s_i \geq \frac{\pi}{2}$. Since $\angle b_i s_i s_k = \theta + 2\alpha$, we get $2\alpha + \theta - \frac{\pi}{3} \geq \frac{\pi}{2}$. Therefore $2\alpha \geq \frac{5\pi}{6} - \sin^{-1}\left(\frac{1}{\sqrt{3}}\right) \implies \alpha \geq \frac{\pi}{3.313}$. But $\alpha \leq \frac{\pi}{4}$, hence its a contradiction and $\hat{h}_i^r$ and $\hat{h}_j^l$ must intersect at a point b. Furthermore by symmetry, $s_k s_j b s_i$ is a kite and $b s_k$ is a perpendicular bisector of $s_i s_j$. We compute the area of the kite next, and use that to bound the total area.

We have,

$$\begin{aligned} bs_k &= bd + ds_k \\ &= \frac{r}{2}\tan\left(\angle bs_js_i\right) + \frac{\sqrt{3}}{2}r \\ &= \frac{r}{2}\left[\tan\left(\theta + 2\alpha - \frac{\pi}{3}\right) + \sqrt{3}\right] \end{aligned}$$

By the property of kite,

$$\begin{aligned} \text{Area}(s_ks_jbs_i) &= 0.5bs_k \times s_is_j \\ &= \frac{r^2}{4}\left[\tan\left(2\alpha - \left(\frac{\pi}{3} - \theta\right)\right) + \sqrt{3}\right] \\ &= \frac{r^2}{4}\left[\frac{\sin 2\alpha\cos\left(\frac{\pi}{3} - \theta\right)}{\cos\left(2\alpha - \left(\frac{\pi}{3} - \theta\right)\right)} - \frac{\cos 2\alpha\sin\left(\frac{\pi}{3} - \theta\right)}{\cos\left(2\alpha - \left(\frac{\pi}{3} - \theta\right)\right)} + \sqrt{3}\right] \\ &= \frac{r^2}{4}\left[\frac{2\sin\alpha\cos\alpha\cos\left(\frac{\pi}{3} - \theta\right) - \cos^2\alpha\sin\left(\frac{\pi}{3} - \theta\right) + \sin^2\alpha\sin\left(\frac{\pi}{3} - \theta\right)}{\cos\left(2\alpha - \left(\frac{\pi}{3} - \theta\right)\right)} + \sqrt{3}\right] \\ &\leq \frac{r^2}{4}\left[\frac{2\sin\alpha\cos\frac{\pi}{6}\cos\left(\frac{\pi}{3} - \theta\right) - \cos^2\frac{\pi}{4}\sin\left(\frac{\pi}{3} - \theta\right) + \sin^2\alpha\sin\left(\frac{\pi}{3} - \theta\right)}{\cos\left(\frac{\pi}{2} - \left(\frac{\pi}{3} - \theta\right)\right)} + \sqrt{3}\right] \\ &= \frac{r^2}{4}\left(sin^2\alpha + 3.76\sin\alpha + 1.232\right). \end{aligned}$$

Now the complete intersection polygon is made up of the union of six such kites (one per adjacent sensor pair). Hence, the uncertainty when $x \in C_k$ is bounded by $\frac{6r^2}{4}\left(sin^2\alpha + 3.76\sin\alpha + 1.232\right)$.

The diameter can be bounded as,

$$U_D(\{s_1, \ldots, s_6\}) \leq 2bs_k \leq r\left(\sin^2\alpha + 3.76\sin\alpha + 1.232\right)$$

$\square$

B.2.3 Proof for Lemma 16

Proof. The construction is similar to that in Lemma 15. Let o be the center of the circle C of radius $\dfrac{r}{\sqrt{3}}$ that contains the target. Let s_i be a sensor on the triangular grid at a distance $r' = kr$ from o (let o lie on a grid location). Draw two tangents from s_i onto

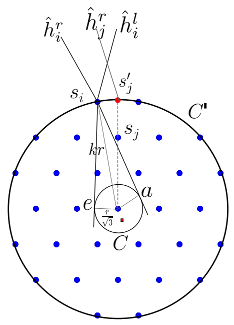


Figure B.5: The target can lie anywhere within C. All sensors inside a circle of radius kr are projected on to its boundary. The resulting set of sensors has bounded intersection for any $\alpha = \frac{\pi}{2} - \epsilon$ if $\sin^{-1}\left(\frac{1}{\sqrt{3}k}\right) + \frac{\pi}{6(k-1)} < 2\epsilon$.

C. Let a and e be their point of intersection with C (see Figure B.5). Draw $\hat{h}_i^l$ and $\hat{h}_i^r$ by rotating s_ia and s_ib about s_i away from s_is_k by an angle of $2\alpha < \frac{\pi}{2}$. $\hat{h}_i^l$ and $\hat{h}_i^r$ enclose any valid sensing wedge for s_i when the target lies within C.

Let b_l and b_r be a point on the half-line $\hat{h}_i^l$ and $\hat{h}_i^r$ (starting at s_i) respectively. We have,

$$\angle b_l s_i o = \angle b_l s_i a + \angle a s_i o = 2\alpha + \sin^{-1}\left(\frac{1}{\sqrt{3}k}\right) < \pi$$
$$\angle b_r s_i o = \angle b_r s_i e + \angle e s_i o = 2\alpha + \sin^{-1}\left(\frac{1}{\sqrt{3}k}\right) < \pi.$$

Draw a circle C' of radius kr centered at o. We project all sensors lying in C' on to its boundary, along the line passing through the origin and the sensor. $\hat{h}_j^l$ and $\hat{h}_j^r$ drawn about the projected sensor location enclose the original sensing wedges for every s_j. Observe that for every integer k, there exist six sensors in the triangular grid that enclose a regular hexagon or side kr. Further there are $k-1$ sensors that lie along each side of this hexagon, not counting the sensors on the vertices. Hence, when the sensors are projected on to a circle of radius kr, it must enclose a hexagon of side at least $k-1$. Further, if s_i' and s_j' are adjacent projected sensors along the boundary, then $\angle s_i' o s_j' \leq \frac{\pi}{3(k-1)}$.

The area of C' is $\pi k^2 r^2$, where r is the grid side. Each triangle in the grid covers an area $\frac{\sqrt{3}}{4}r^2$. Hence, C' contains $\mathcal{O}(k^2)$ sensors.

Now for bounded intersection, we require $\hat{h}_i^l$ to intersect with $\hat{h}_j^r$ (or vice-versa) for every adjacent pair of projected sensors. Suppose the intersection is unbounded for some pair s_i' and s_j'. Then, $\hat{h}_i^l$ does not intersect with $\hat{h}_j^r$. Hence, $\angle b_i s_i' s_j' \geq \frac{\pi}{2}$ where b_i is any point on h_i^l. But $\angle b_i s_i' s_j' = \angle b_i s_i' o - \angle o s_i' s_j'$. Therefore $2\alpha + \sin^{-1}\left(\frac{1}{\sqrt{3k}}\right) - \left(\frac{\pi}{2} - \frac{\pi}{6k}\right) \geq \frac{\pi}{2}$. That is, $\sin^{-1}\left(\frac{1}{\sqrt{3k}}\right) + \frac{\pi}{6(k-1)} \geq 2\epsilon$. This is a contradiction. Hence, the intersection of all valid sensing wedges is bounded. $\square$

B.2.4 Proof for Lemma 17

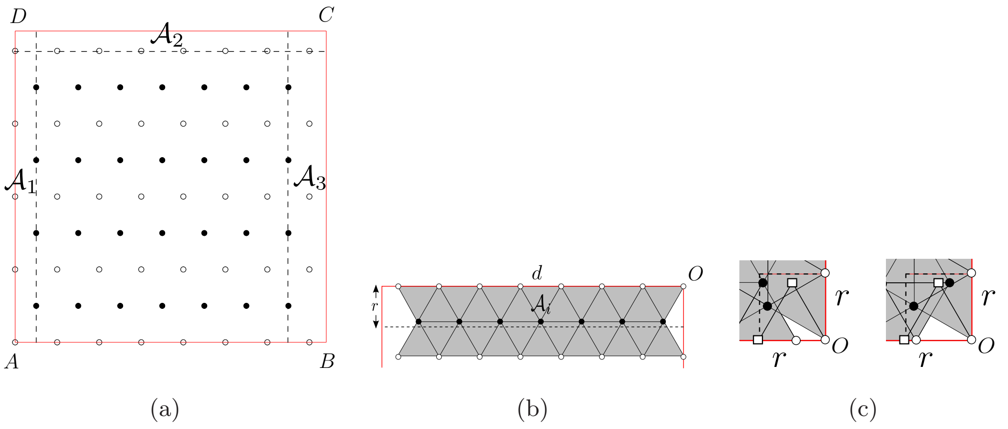


Figure B.6: (a) Regions $\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3$ are not covered by equilateral triangles by sensors placed on a triangular grid. (b) We place additional sensors in these three regions to cover all portions except near the ends. (c) We place two additional sensors per corner (shown in square) to cover the rest of $\mathcal{A}$.

Proof. Consider Figure B.6. Regions $\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3$ are not covered by equilateral triangles. The width of each of these regions is d and the height is at most r. We place additional sensors in each of these regions as shown in Figure B.6(b). Each region will require at most $2w_r + b_r$ of such sensors. Observe that, these sensors cover the three regions ($\mathcal{A}_{1,2,3}$) everywhere except the two ends. We will add additional sensors to ensure that $\mathcal{A}$ is covered by equilateral triangles.

Note that amongst the newly added sensors, one sensor is placed at one of the four corners (marked O in Figure B.6(b)). We align this corner with vertex B, C and D for regions $\mathcal{A}_3, \mathcal{A}_2$ and $\mathcal{A}_1$ respectively. The uncovered portions of $\mathcal{A}$ are now four regions near each of the corners of sides at most r. For each of the corners we have a situation as shown in Figure B.6(c), where already placed sensors are marked as white and black circles. We require two additional sensors per corner (marked as squares) to cover the remaining portion of $\mathcal{A}$. Hence, the total number of sensors required are,

$$n_{ALG} \leq (w_r w_c + b_r b_c) + 3(2w_r + b_r) + 4 \cdot 2.$$

This completes the proof. □

B.3 Proof for Theorem 4

Proof. First consider the number of sensors. Let n_{ALG} be the number of sensors placed by our algorithm for covering $\mathcal{A}$ with equilateral triangles. From Lemma 17,

$$\begin{aligned} n_{ALG} &\leq w_r w_c + b_r b_c + 6w_r + 3b_r + 8, \\ &\leq 2w_r w_c + 9w_r + 8, \\ &\leq 2\left(\frac{d}{r}+1\right)\left(\frac{d}{\sqrt{3}r}+1\right) + 9\left(\frac{d}{r}+1\right) + 8, \\ &= \frac{2}{\sqrt{3}}\frac{d^2}{r^2} + \left(11 + \frac{2}{\sqrt{3}}\right)\frac{d}{r} + 19. \end{aligned} \tag{B.8}$$

From Corollary 3, we now the number of sensors used by an optimal algorithm is bounded by,

$$\begin{aligned} n^* &\geq \frac{(d-2r^*)^2}{\pi r^{*2}}, \\ &= \frac{1}{\pi}\frac{d^2}{r^{*2}} - \frac{4}{\pi}\frac{d}{r^*} + \frac{4}{\pi}. \end{aligned}$$

Recall from Corollary 2, $r^* \leq \dfrac{U_D^*}{2\sin\alpha}$ and $r^* \leq \sqrt{\dfrac{U_A^*}{\pi}\dfrac{1}{\sin\alpha}}$. For the triangular grid placement, we set the grid length $r = \dfrac{U_D^*}{2\sin\alpha}$ and $r = \sqrt{\dfrac{U_A^*}{\pi}\dfrac{1}{\sin\alpha}}$, for the diameter and area uncertainty problems respectively. Hence $r^* \leq r$ which gives,

$$n^* \leq \frac{1}{\pi}\frac{d^2}{r^2} - \frac{4}{\pi}\frac{d}{r} + \frac{4}{\pi}. \tag{B.9}$$

When $d \leq 14r$, we have $d \leq \dfrac{14U_D^*}{2\sin\alpha}$ which implies $U_D^* \geq \dfrac{d}{7\sin\alpha}$ (equivalently $d \leq \dfrac{14}{\sin\alpha}\sqrt{\dfrac{U_A^*}{\pi}}$ which implies $U_A^* \geq \dfrac{\pi d^2 \sin^2\alpha}{196}$). This refers to the case when the desired uncertainty is greater than a constant times the diameter or area of $\mathcal{A}$. In this case, since we know $\frac{d}{r} \leq 14$, we only require a constant number of sensors (n_{ALG}) as given by Equation B.8.

Now consider $d > 14r$. Then we have,

$$\frac{2}{\sqrt{3}}\frac{d^2}{r^2} + \left(11 + \frac{2}{\sqrt{3}}\right)\frac{d}{r} + 19 \leq 9\left(\frac{1}{\pi}\frac{d^2}{r^2} - \frac{4}{\pi}\frac{d}{r} + \frac{4}{\pi}\right)$$
$$\therefore \quad n_{ALG} \leq 9n^*.$$

Next, we bound the uncertainty achieved by our placement with that by an optimal. We consider each of the four cases defined in Lemma 14 and Lemma 15 in turn. For the first three cases, we require the target to lie within an equilateral triangle. From Lemma 17 we know that our placement strategy ensures a cover of $\mathcal{A}$ with equilateral triangles. Recall that we set $r = \frac{U_D^*}{2\sin\alpha}$ and $r = \sqrt{\frac{U_A^*}{\pi}\frac{1}{\sin\alpha}}$, for the diameter and area uncertainty problems respectively.

CASE (a) $0 < \alpha < \frac{\pi}{18}$**:**

For diameter uncertainty,

$$\begin{aligned} U_D(S_{ALG}) &\leq 11.35 r \sin\alpha, \\ &= 11.35\frac{U_D^* \sin\alpha}{2\sin\alpha} \\ \therefore \quad U_D(S_{ALG}) &\leq 5.675 U_D^*. \end{aligned}$$

For area uncertainty,

$$\begin{aligned} U_A(S_{ALG}) &\leq 23.46 r^2 \sin^2\alpha \\ &= 23.46\frac{U_A^*}{\pi}\frac{\sin^2\alpha}{\sin^2\alpha} \\ \therefore \quad U_A(S_{ALG}) &\leq 7.47 U_A^*. \end{aligned}$$

CASE (b) $\frac{\pi}{18} \leq \alpha < \frac{\pi}{12}$**:**

For diameter uncertainty,

$$\begin{aligned} U_D(S_{ALG}) &\leq 2.04r \\ &= 2.04\frac{U_D^*}{2\sin\alpha} \\ &\leq 2.04\frac{U_D^*}{2\sin\frac{\pi}{18}} \\ \therefore \quad U_D(S_{ALG}) &\leq 5.88U_D^*. \end{aligned}$$

For area uncertainty,

$$\begin{aligned} U_A(S_{ALG}) &\leq \frac{\sqrt{3}r^2}{4} + 10.1r^2\sin^2\alpha \\ &= \frac{\sqrt{3}U_A^*}{4\pi\sin^2\alpha} + 10.1\frac{U_A^*\sin^2\alpha}{\pi\sin^2\alpha} \\ &\leq \frac{\sqrt{3}U_A^*}{4\pi\sin^2\frac{\pi}{18}} + 10.1\frac{U_A^*}{\pi} \\ \therefore \quad U_A(S_{ALG}) &\leq 7.76U_A^*. \end{aligned}$$

CASE (c) $\frac{\pi}{12} \leq \alpha < \frac{\pi}{6}$**:**
For diameter uncertainty,

$$\begin{aligned} U_D(S_{ALG}) &\leq \left(1 + \frac{1}{\sqrt{3}}\right)r \\ &= \left(1 + \frac{1}{\sqrt{3}}\right)\frac{U_D^*}{2\sin\alpha} \\ &\leq \left(1 + \frac{1}{\sqrt{3}}\right)\frac{U_D^*}{2\sin\frac{\pi}{12}} \\ \therefore \quad U_D(S_{ALG}) &\leq 3.05U_D^*. \end{aligned}$$

For area uncertainty,

$$\begin{aligned} U_A(S_{ALG}) &\leq \frac{3\sqrt{3}}{4}r^2 \\ &= \frac{3\sqrt{3}U_A^*}{4\pi\sin^2\alpha} \\ &\leq \frac{3\sqrt{3}U_A^*}{4\pi\sin^2\frac{\pi}{12}} \\ \therefore \quad U_A(S_{ALG}) &\leq 6.17U_A^*. \end{aligned}$$

CASE (d) $\frac{\pi}{6} \leq \alpha \leq \frac{\pi}{4}$:

We apply Lemma 15 to bound this uncertainty. In Lemma 15, we constructed an over approximation to the union of all possible sensing wedges from the sensors placed on the hexagon. Without loss of generality let $x \in C_k$. The union of all C_k circles covers $\mathcal{A}$. When x is close to the boundary of $\mathcal{A}$, we may find only a subset of the sensors $i, \ldots, o$, as shown in Figure B.4(a), which are required by Lemma 15. However, the intersection of the available sensing wedges from $t = i, \ldots, o$ with the boundary of $\mathcal{A}$ is a subset of the over approximation used in Lemma 15.

Hence for diameter uncertainty,

$$\begin{aligned} U_D(S_{ALG}) &\leq r\left(\sin^2\alpha + 3.76\sin\alpha + 1.232\right) \\ &= \frac{U_D^*}{2\sin\alpha}\left(\sin^2\alpha + 3.76\sin\alpha + 1.232\right) \\ &\leq \frac{U_D^*}{2}\left(\sin\alpha + 3.76 + \frac{1.232}{\sin\alpha}\right) \\ \therefore \quad U_D(S_{ALG}) &\leq 3.47U_D^*. \end{aligned}$$

For area uncertainty,

$$\begin{aligned} U_A(S_{ALG}) &\leq 1.5r^2\left(\sin^2\alpha + 3.76\sin\alpha + 1.232\right) \\ &= 1.5\frac{U_A^*}{\pi\sin^2\alpha}\left(\sin^2\alpha + 3.76\sin\alpha + 1.232\right) \\ &= 1.5\frac{U_A^*}{\pi}\left(1 + \frac{3.76}{\sin\alpha} + \frac{1.232}{\sin^2\alpha}\right) \\ &\leq 1.5\frac{U_A^*}{\pi}\left(1 + \frac{3.76}{\sin\frac{\pi}{6}} + \frac{1.232}{\sin^2\frac{\pi}{6}}\right) \\ \therefore \quad U_A(S_{ALG}) &\leq 6.42U_A^*. \end{aligned}$$

This completes the proof. □

Appendix C

Proofs from Chapter 5

C.1 Proof of Theorem 7

For the analysis we will convert the weighted version of the problem to an unweighted one. Choose a small $\delta > 0$ such that for all i, j, x, we have $q_i(R_j(x)) - \delta\lfloor \frac{q_i(R_j(x))}{\delta} \rfloor \leq \frac{2\epsilon}{n}$. We now create a new set system $(X', \mathcal{R}')$. For each target we create $\lfloor \frac{q_i}{\delta} \rfloor$ elements in X' labeled $t_i^1, \ldots, t_i^{\lfloor \frac{q_i}{\delta} \rfloor}$. $\mathcal{R}'$ has one range set corresponding to each $R_j(x)$, say $R'_j(x)$. If $q_i(R_j(x)) > 0$, then $R'_j(x)$ contains $t_i^1, \ldots, t_i^{\lfloor \frac{q_i(R_j(x))}{\delta} \rfloor}$.

Let σ and σ^* be the map given by Algorithm 2 and the optimal algorithm, respectively on the original weighted problem. σ and σ^* also define corresponding maps for the modified set system. We use the shorthand $R_i \triangleq R'_i(\sigma(i))$ and $R_i^* \triangleq R'_i(\sigma^*(i))$ to indicate the actions chosen by the greedy and optimal algorithms for each robot. Without loss of generality, relabel the robots such that Algorithm 2 picks a set corresponding to robot 1 in the first iteration, corresponding to robot 2 in the second iteration, and so on.

Let $A(i)$ and $A^*(i)$ denote the set of all elements in X' covered by the greedy and optimal algorithm using robots labeled 1 through i in the modified set system $(X', \mathcal{R}')$. That is,

$$A(i) \triangleq \bigcup_{p=1}^{i} R_p, \quad A^*(i) \triangleq \bigcup_{p=1}^{i} R_p^*.$$

In the case of the greedy algorithm, $A(i)$ also refers to the set of elements covered in the first i iterations.

After the last round, the greedy algorithm covers $|A(k)|$ elements, whereas the optimal algorithm covers $|A^*(k)|$ elements. We have $|A(k)| \leq |A^*(k)|$. Hence, there are at least $|A^*(k)| - |A(k)|$ elements covered by the optimal algorithm, which are not covered by any set selected by the greedy algorithm. Hence, there exists a set of at least $|A^*(k)| - |A(k)|$ elements present in $A^*(k)$ but not in $A(k)$.

Lemma 29. *If U is any set of exactly $|A^*(k)| - |A(k)|$ elements present in $A^*(k)$ but not in $A(k)$, then for any $i = 1, \ldots, k$ $|A(i)| \geq |A^*(i) \cap U|$.*

Proof. We prove this by induction over i. The base case is for $i = 1$. Since $R_1 = R_1'(\sigma(1))$ is the first set picked by the greedy algorithm the base case is trivially true.

Now suppose that the statement holds for some $1 < j < k$. That is,

$$|A(j)| \geq |A^*(j) \cap U|. \tag{C.1}$$

We will show that the statement then also holds for $j+1$.

In the $(j+1)^{th}$ iteration, the greedy algorithm chooses some $R_{j+1} = R'_{j+1}(\sigma(j+1))$. Recall $A(j)$ refers to the set of elements covered by the greedy algorithm in the first j iterations. Hence due to the greedy choice in the original set system we have,

$$\begin{aligned} |A(j) \cup R'_{j+1}(\sigma(j+1))| &\geq |A(j) \cup R'_{j+1}(l)|, \ \forall l = 1, \ldots, m \\ \therefore \ |A(j) \cup R_{j+1}| &\geq |A(j) \cup R'_{j+1}(\sigma^*(j+1))| \\ &= |A(j) \cup R^*_{j+1}|. \end{aligned}$$

We can write R^*_{j+1} as,

$$R^*_{j+1} = \left(R^*_{j+1} \cap U\right) \cup \left(R^*_{j+1} \cap \bar{U}\right)$$

where $\bar{U}$ is the complement of U in $\mathcal{U}$. Therefore,

$$\begin{aligned} |A(j) \cup R_{j+1}| &\geq |A(j) \cup \left(\left(R^*_{j+1} \cap U\right) \cup \left(R^*_{j+1} \cap \bar{U}\right)\right)| \\ &\geq |A(j) \cup \left(R^*_{j+1} \cap U\right)| \\ &= |A(j)| + |R^*_{j+1} \cap U| - |A(j) \cap R^*_{j+1} \cap U| \\ &= |A(j)| + |R^*_{j+1} \cap U| \end{aligned}$$

since $A(j) \cap U = \emptyset$ by definition of U.

Using the induction hypothesis in Equation C.1,

$$\begin{aligned}
|A(j) \cup R_{j+1}| &\geq |A^*(j) \cap U| + |R^*_{j+1} \cap U| \\
&\geq |\,(A^*(j) \cap U) \cup \big(R^*_{j+1} \cap U\big)\,| \\
\therefore |A(j+1)| &\geq |A^*(j+1) \cap U|.
\end{aligned}$$

Hence proved by induction.

□

This gives us,

$$\begin{aligned}
|A(k)| &\geq |A^*(k) \cap U| \\
&\geq |U| \\
&= |A^*(k)| - |A(k)|
\end{aligned}$$

where the second inequality comes from $U \subseteq A^*(k)$. Therefore, $|A(k)| \geq \frac{|A^*(k)|}{2}$.

Finally, we will relate $|A(k)|$ and $|A^*(k)|$ to the correspond weights in the original problem. We have,

$$\begin{aligned}
w^* &= \sum_{\forall t_i} \max_{\forall j} q_i(R_j(\sigma^*(j))) \\
&\leq \sum_{\forall t_i} \max_{\forall j} \frac{2\epsilon}{n} + \delta \lfloor \frac{q_i(R_j(\sigma^*(j)))}{\delta} \rfloor \\
&\leq 2\epsilon + \delta |A^*(k)|.
\end{aligned}$$

On the other hand,

$$\begin{aligned}
w &= \sum_{\forall t_i} \max_{\forall j} q_i(R_j(\sigma(j))) \\
&\geq \sum_{\forall t_i} \max_{\forall j} \delta \lfloor \frac{q_i(R_j(\sigma(j)))}{\delta} \rfloor \\
&= \delta |A(k)|.
\end{aligned}$$

Thus, $w \geq \delta |A(k)| \geq \frac{\delta |A^*(k)|}{2} \geq \frac{w^*}{2} - \epsilon$.

Appendix D

Proofs from Chapter 8

D.1 Proof: Unconstrained Solution

The state transition equation can be written as,

$$\dot{\mathbf{X}}(t) = \begin{bmatrix} \dot{x}(t) \\ \dot{v}(t) \end{bmatrix} = \begin{bmatrix} v(t) \\ a(t) \end{bmatrix} \tag{D.1}$$

The objective is to find a velocity profile $v^*(t)$ which minimizes the total energy required for motion given by the following cost functional,

$$J = \int_0^{t_f} \Big[c_1 a^2(t) + c_2 v^2(t) + c_3 v(t) + c_4 \Big] dt, \tag{D.2}$$

where the final time t_f is kept a free variable. The initial boundary conditions are given as,

$$x(0) = 0, \quad v(0) = 0, \tag{D.3}$$

and the final boundary conditions are given as,

$$x(t_f) = D, \quad v(t_f) = 0. \tag{D.4}$$

Hamiltonian

The hamiltonian $H(\mathbf{X}, \boldsymbol{\lambda}, u, t)$ is defined as,

$$H(\mathbf{X}, \boldsymbol{\lambda}, u, t) = J + \lambda_1(t)\dot{x}(t) + \lambda_2(t)\dot{v(t)}, \tag{D.5}$$

where $\lambda_1(t)$ and $\lambda_2(t)$ are the Lagrange multipliers, also called the co-state variables which include the state transition equations as a constraint to the objective.

When there is no bound on the maximum velocity, the Hamiltonian for this problem can be obtained using Equations D.1 and D.2 as,

$$H(\mathbf{X}(t), a(t), \boldsymbol{\lambda}(t), t) = c_1 a^2(t) + c_2 v^2(t) + c_3 v(t) + c_4 + \lambda_1(t) v(t) + \lambda_2(t) a(t) \quad \text{(D.6)}$$

where the acceleration $a(t)$ is the control.

The three necessary conditions for $a^*(t)$ to optimize the Hamiltonian [151] for all time $t \in [0, t_f]$ are given as,

$$\dot{\mathbf{X}}^*(t) = \frac{\partial H}{\partial \boldsymbol{\lambda}}, \quad \dot{\boldsymbol{\lambda}}^*(t) = -\frac{\partial H}{\partial \mathbf{X}}, \quad 0 = \frac{\partial H}{\partial a} \quad \text{(D.7)}$$

By substituting we get,

$$\begin{aligned}
\dot{x}(t) &= v(t), \\
\dot{v}(t) &= a(t), \\
\dot{\lambda_1}(t) &= 0, \\
\dot{\lambda_2}(t) &= 2c_2 v(t) + c_3 + \lambda_1, \\
\lambda_2(t) &= -2c_1 a(t), \\
\therefore \lambda_2(t) &= -2c_1 \dot{v}(t).
\end{aligned}$$

Using the last two equations, we can write,

$$\begin{aligned}
2c_2 v(t) + c_3 + \lambda_1 &= -2c_1 \ddot{v}(t), \\
2c_1 \ddot{v}(t) + 2c_2 v(t) + c_3 + \lambda_1 &= 0.
\end{aligned}$$

We can solve for this second order differential equation to yield,

$$v^*(t) = s_1 e^{kt} + s_2 e^{-kt} - \left(\frac{c_3 + s_3}{2c_1}\right) \quad \text{(D.8)}$$

where $k = \sqrt{\frac{c_2}{c_1}}$ and $s_1 - s_4$ are constants and $\lambda_1 = s_3$.

Applying the state transition equations, we can get the optimal control and states given as,

$$a^*(t) = k s_1 e^{kt} - k s_2 e^{-kt} \quad \text{(D.9)}$$

$$x^*(t) = \frac{s_1 e^{kt}}{k} - \frac{s_2 e^{-kt}}{k} - \left(\frac{c_3 + s_3}{2c_1}\right) t + s_4. \quad \text{(D.10)}$$

We can solve for $s_1 - s_4$ in terms of the final time t_f by substituting the boundary conditions given in Equations D.3 and D.4 for $v^*(t)$ and $x^*(t)$. We obtain,

$$\begin{aligned}
s_1 &= -\frac{Dk}{kt_f + e^{kt_f}(kt_f - 2) + 2}, \\
s_2 &= s_1 e^{kt_f}, \\
s_3 &= 2c_1(s_1 + s_2) - c_3, \\
s_4 &= -\frac{s_1 - s_2}{k}.
\end{aligned} \tag{D.11}$$

By substituting in Equations D.9-D.10 we obtain,

$$\begin{aligned}
a^*(t) &= D\left(\frac{c_2}{c_1}\right)\left(\frac{e^{(kt_f - t)} - e^{(kt)}}{kt_f + e^{kt_f}(kt_f - 2) + 2}\right), \\
v^*(t) &= D\sqrt{\frac{c_2}{c_1}}\left(\frac{(1 + e^{kt_f} - (e^{k(t_f - t)} + e^{kt}))}{kt_f + e^{kt_f}(kt_f - 2) + 2}\right), \\
x^*(t) &= D\left(\frac{(e^{k(t_f - t)} - e^{kt}) - (e^{kt_f} - 1) + kt(e^{kt_f} + 1)}{kt_f + e^{kt_f}(kt_f - 2) + 2}\right).
\end{aligned} \tag{D.12}$$

Since the final time is free, it can be solved for using the additional boundary condition (known as the transversality condition) given by,

$$H(\mathbf{X}^*(t_f), a^*(t_f), \boldsymbol{\lambda}^*(t_f), t_f) = 0. \tag{D.13}$$

Substituting Equations D.9-D.10 and D.11 above results in,

$$(D\frac{c_2}{c_1} + 2)(1 - e^{kt_f}) + \sqrt{\frac{c_4}{c_1}} kt_f(1 + e^{kt_f}) = 0, \tag{D.14}$$

which is an equation in single variable t_f and can be solved using existing solvers. (We used MATLAB's `solve` function). Alternatively, if the final time is fixed, we can directly substitute this given value in Equation D.12 to find $v^*(t)$.

D.2 Proof for Lemma 24

Proof. Consider any velocity profile $v(t)$ shown in Figure D.1. Let D and E be the total distance covered and energy consumed by $v(t)$. This profile crosses $\sqrt{\frac{c_4}{c_2}}$ between times

$[t_1, t_2]$ and $[t_3, t_4]$. Let d_{12} and d_{34} be the distances covered by $v(t)$ in these sections. The total energy consumption of $v(t)$ is given by,

$$E = E_{01} + E_{12} + E_{23} + E_{34} + E_{45}, \tag{D.15}$$

where E_{ij} refers to the energy consumption to cover the distance d_{ij}.

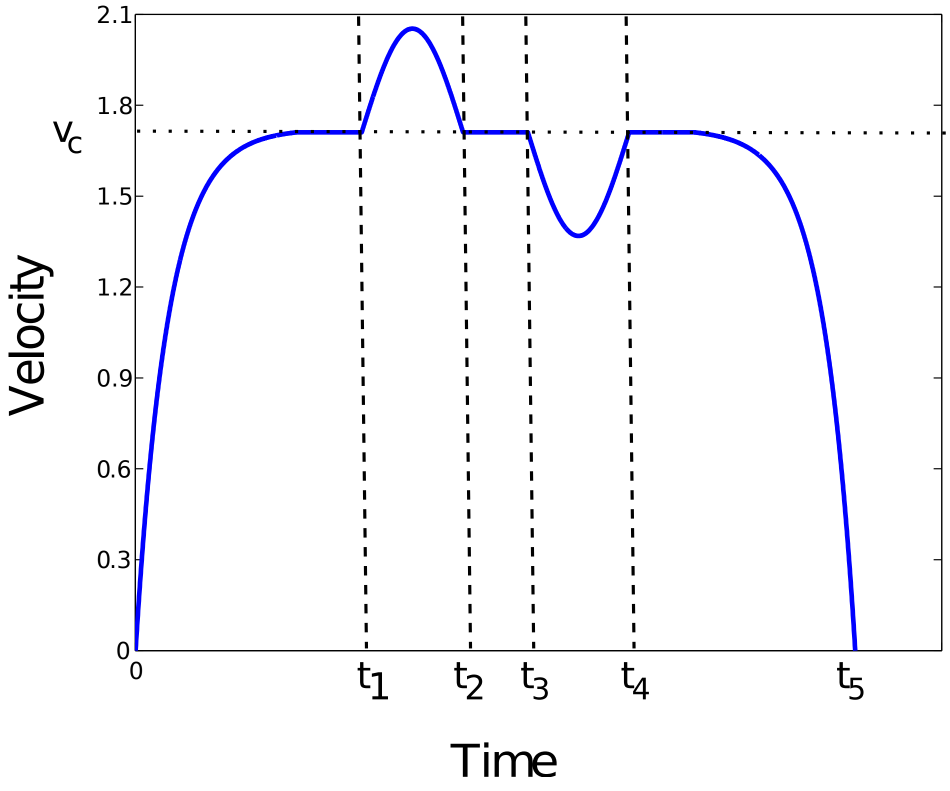


Figure D.1: Sections of this velocity profile crossing ($v_c = \sqrt{\frac{c_4}{c_2}}$) between $[t_1, t_2]$ and $[t_3, t_4]$ can be replaced by constant velocity (v_c) sections resulting in a velocity profile that consumes lesser energy to travel the same distance.

We construct another velocity profile $v'(t)$ by replacing the sections $[t_1, t_2]$ and $[t_3, t_4]$ by constant velocity $v_c = \sqrt{\frac{c_4}{c_2}}$ sections for time $\frac{d_{12}}{v_c}$ and $\frac{d_{34}}{v_c}$ respectively. The total distance traveled by $v'(t)$ is D, same as $v(t)$. The total energy consumption of $v'(t)$ is given by,

$$E' = E_{01} + E'_{12} + E_{23} + E'_{34} + E_{45}, \tag{D.16}$$

since $v'(t)$ is the same as $v(t)$ everywhere except $t \in [t_1, t_2]$ and $t \in [t_3, t_4]$.

We now show that $E' \leq E$ by proving both $E'_{12} \leq E_{12}$ and $E'_{34} \leq E_{34}$. This result can then be generalized to velocity profiles with any number of crossing sections in either directions.

First, consider the energy consumption E_{12} for $v(t)$,

$$E_{12} = \int_{t_1}^{t_2} \Big[c_1 a^2(t) + c_2 v^2(t) + c_4\Big] dt + c_3 d_{12}. \tag{D.17}$$

Now, let us consider E'_{12}. The time taken in this case would be $t_c = \dfrac{d_{12}}{v_c}$. The energy consumption is,

$$\begin{aligned} E'_{12} &= \int_{t_1}^{t_c} \Big[c_1 a^2(t) + c_2 v^2(t) + c_3 v(t) + c_4\Big] dt, \\ &= c_2 v_c d_{12} + c_3 d_{12} + c_4 \frac{d_{12}}{v_c}. \end{aligned} \tag{D.18}$$

The distance d_{12} can also be written as,

$$d_{12} = \int_{t_1}^{t_2} \left[v(t) dt\right]. \tag{D.19}$$

Substituting Equation D.19 in D.18, we obtain,

$$E'_{12} = c_2 \int_{t_1}^{t_2} v_c v(t) dt + c_3 d_{12} + c_4 \int_{t_1}^{t_2} \frac{v(t)}{v_c} dt. \tag{D.20}$$

Using Equations D.17 and D.20, we can write,

$$\begin{aligned} E_{12} - E'_{12} &= c_1 \int_{t_1}^{t_2} a^2(t) dt + \frac{c_2}{v_c} \int_{t_1}^{t_2} \left[v_c(t) - v(t)\right] \left[\frac{c_4}{c_2} - v(t) v_c\right] dt \\ \therefore E_{12} - E'_{12} &\geq 0, \end{aligned}$$

since $v(t) \leq v_c \leq \sqrt{\dfrac{c_4}{c_2}}$. For the section between t_3 and t_4, we can show that $E_{34} - E'_{34} \geq 0$.

In general we can replace any number of such sections crossing $\sqrt{\dfrac{c_4}{c_2}}$ to yield another velocity profile with lower energy covering the same distance moving at $\sqrt{\dfrac{c_4}{c_2}}$. Hence, once the velocity profile hits $\sqrt{\dfrac{c_4}{c_2}}$, there is no reason to deviate from this value except at the boundary (initial and final conditions). $\square$

D.3 Proof: Constrained Solution

We begin by writing the velocity constraint in the form of state inequality $\bar{S} = (v(t) - v_m) \leq 0$. The state inequality $\bar{S}$ is converted into a control equality and interior point constraint by differentiating $\bar{S}$ once leading to,

$$\begin{aligned} \bar{S}^{(1)} &= \dot{v}(t) = u. \\ v(t_1) &= v_m \end{aligned} \tag{D.21}$$

Along the unconstrained arc, the state transition is governed by Equation D.1. On the constrained arc, the state transition is given by,

$$\dot{\mathbf{X}}(t) = \begin{bmatrix} \dot{x}(t) \\ \dot{v}(t) \end{bmatrix} = \begin{bmatrix} v_m \\ 0 \end{bmatrix} \tag{D.22}$$

The Hamiltonian is augmented with the control equality constraint in $[t_1, t_f - t_2]$ and is given by,

$$\hat{H} = c_1 a^2(t) + c_2 v^2(t) + c_3 v(t) + c_4 + \lambda_1(t)v(t) + \lambda_2(t)a(t) + \mu(t)a(t) \tag{D.23}$$

where μ is the slack variable associated with the control constraint. In the interval $[0, t_1]$ and $[t_f - t_2, t_f]$, the Hamiltonian is given by,

$$H = c_1 a^2(t) + c_2 v^2(t) + c_3 v(t) + c_4 + \lambda_1(t)v(t) + \lambda_2(t)a(t) \tag{D.24}$$

The interior point constraint is given by,

$$G = \xi(t)(v(t) - v_m). \tag{D.25}$$

D.3.1 $0 \leq t \leq t_1$

Using the necessary condition $\dot{\lambda} = -\dfrac{\partial H}{\partial x}$ we get,

$$\begin{aligned} \dot{\lambda_1} &= 0, \\ \therefore \quad \lambda_1 &= s_3. \end{aligned}$$

and,

$$\dot{\lambda_2} = -\frac{\partial H}{\partial x}, \tag{D.26}$$

$$\therefore \quad \dot{\lambda_2} = -\left[2c_2 v(t) + c_3 + s_3\right]. \tag{D.27}$$

Applying the third necessary condition, $0 = \dfrac{\partial H}{\partial a}$ we get,

$$0 = 2c_1 a(t) + \lambda_2(t),$$
$$\therefore \quad \lambda_2(t) = -2c_1 a(t).$$

Differentiating the above equation we get,

$$\dot{\lambda_2}(t) = -2c_1 \ddot{v}(t).$$

From Equation D.27 we can write,

$$2c_1 \ddot{v(t)} = 2c_2 v(t) + c_3 + s_3,$$
$$\therefore \quad \ddot{v}(t) - \frac{c_2}{c_1} v(t) - \frac{c_3 + s_3}{2c_1} = 0.$$

The solution for the above differential equation is given as,

$$\begin{aligned}
v^*(t) &= s_1 e^{kt} + s_2 e^{-kt} - \frac{c_3 + s_3}{2c_1}, \\
a^*(t) &= s_1 k e^{kt} - s_2 k e^{-kt}, \\
x^*(t) &= \frac{s_1}{k} e^{kt} - \frac{s_2}{k} e^{-kt} - \frac{c_3 + s_3}{2c_1} t + s_4, \\
\lambda_1^*(t) &= s_3, \\
\lambda_2^*(t) &= -2c_1 a^*(t).
\end{aligned}$$

Using initial conditions $x(0) = 0$ and $v(0) = v_0$, we get,

$$s_4 = -\frac{s_1 - s_2}{k},$$
$$\frac{c_3 + s_3}{2c_1} = s_1 + s_2 - v_0.$$

Putting these together we get,

$$\begin{aligned}
v^*(t) &= s_1 e^{kt} + s_2 e^{-kt} - (s_1 + s_2 - v_0), \\
a^*(t) &= s_1 k e^{kt} - s_2 k e^{-kt}, \\
x^*(t) &= \frac{s_1}{k} e^{kt} - \frac{s_2}{k} e^{-kt} - (s_1 + s_2 - v_0)t - \frac{s_1 - s_2}{k}, \\
\lambda_1^*(t) &= 2c_1 s_1 + s_2 - v_0 - c_3, \\
\lambda_2^*(t) &= -2c_1 a^*(t),
\end{aligned}$$

where s_1 and s_2 are two constants left to be evaluated.

D.3.2 $t_f - t_2 \leq t \leq t_f$

In this section, the system is governed by the same state equation as in the interval $0 \leq t \leq t_1$. Hence, we get a similar form for the optimal state and control given by,

$$\begin{aligned}
v^*(t) &= s_1' e^{-kt_f} e^{kt} + s_2' e^{kt_f} e^{-kt} - \frac{c_3 + s_3'}{2c_1}, \\
a^*(t) &= s_1' k e^{-kt_f} e^{kt} - s_2' k e^{kt_f} e^{-kt}, \\
x^*(t) &= \frac{s_1'}{k} e^{-kt_f} e^{kt} - \frac{s_2'}{k} e^{kt_f} e^{-kt} - \frac{c_3 + s_3'}{2c_1} t + s_4', \\
\lambda_1^*(t) &= s_3', \\
\lambda_2^*(t) &= -2c_1 a^*(t).
\end{aligned}$$

where $s_1' \ldots s_4'$ are the new constants to be solved for. Using the final condition, $x(t_f) = D$, we get

$$\begin{aligned}
D &= \frac{s_1'}{k} - \frac{s_2'}{k} - \frac{c_3 + s_3'}{2c_1} t_f + s_4', \\
\therefore \quad s_4' &= D - \left[\frac{s_1'}{k} - \frac{s_2'}{k} - \frac{c_3 + s_3'}{2c_1} t_f \right].
\end{aligned}$$

Using the second final condition, $v(t_f) = v_f$ we get,

$$\begin{aligned}
v_f &= s_1' + s_2' - \frac{c_3 + s_3'}{2c_1}, \\
\therefore \quad s_3' &= 2c_1(s_1' + s_2' - v_f) - c_3.
\end{aligned}$$

The equations can then be written as,

$$\begin{aligned}
v^*(t) &= s_1' e^{-k(t_f - t)} + s_2' e^{k(t_f - t)} - (s_1' + s_2' - v_f), \\
a^*(t) &= s_1' k e^{-k(t_f - t)} - s_2' k e^{k(t_f - t)}, \\
\lambda_1^*(t) &= 2c_1(s_1' + s_2' - v_f) - c_3, \\
\lambda_2^*(t) &= -2c_1 a^*(t).
\end{aligned}$$

D.3.3 Corner conditions

We can now use the corner conditions to determine the unknown constants s_1, s_2, s_1', s_2'. The corner conditions state that $\lambda((t_f - t_2)^+) = \lambda((t_f - t_2)^-)$ and $v((t_f - t_2)^+) = v((t_f - t_2)^-)$ and $\mu((t_f - t_2)^+) = 0$,

$$\begin{aligned} H(t_2^+) &= H(t_2^-), \\ \therefore \quad a(t_f - t_2) &= 0, \\ \therefore \quad s_2' &= s_1' e^{-2kt_2}. \end{aligned}$$

Using the other corner condition $v((t_f - t_2)^+) = v((t_f - t_2)^-)$ we have,

$$\begin{aligned} v(t_f - t_2) &= v_m, \\ \therefore \quad v_m &= s_1' e^{-kt_2} + s_1' e^{-kt_2} - (s_1' + s_1' e^{-2kt_2} - v_f), \\ \therefore \quad s_1' &= -\frac{v_m - v_f}{(e^{-kt_2} - 1)^2}. \end{aligned}$$

Using similar arguments at the other corner $t = t_1$ we get the final form for the optimal velocity profile as,

$$v^*(t) = \begin{cases} s_1 \left(e^{kt} + e^{k(2t_1 - t)} - (1 + e^{2kt_1}) \right) + v_0, & 0 \le t \le t_1 \\ v_m, & t_1 \le t \le t_f - t_2 \\ s_2 \left(e^{-k(t_f - t - 2t_2)} + e^{k(t_f - t)} - (1 + e^{2kt_2}) \right) + v_f, & t_f - t_2 \le t \le t_f. \end{cases}$$

where,

$$\begin{aligned} s_1 &= -\frac{(v_m - v_0)}{(e^{kt_1} - 1)^2}, \\ s_2 &= -\frac{v_m - v_f}{(e^{kt_2} - 1)^2}, \end{aligned}$$

Using the transversality condition $H(t_f) = 0$, we can determine the times t_1 and t_2 as,

$$\begin{aligned} t_1 &= \frac{1}{k} \ln \left(\frac{c_4 + c_2 v_m^2 - 2c_2 v_0 v_m}{c_4 - c_2 v_m^2} + \frac{2(c_2 v_m (c_4 - c_2 v_0 v_m)(v_m - v_0))^{\frac{1}{2}}}{c_4 - c_2 v_m^2} \right), \\ t_2 &= \frac{1}{k} \ln \left(\frac{c_4 + c_2 v_m^2 - 2c_2 v_f v_m}{c_4 - c_2 v_m^2} + \frac{2(c_2 v_m (c_4 - c_2 v_f v_m)(v_m - v_f))^{\frac{1}{2}}}{c_4 - c_2 v_m^2} \right). \end{aligned} \tag{D.28}$$

and the final time can then be calculated by using the total distance to travel and the distances traveled in the two exponential curves. It is easy to see that if v_0 or v_f is equal to v_m, then $t_1 = 0$ or $t_2 = 0$ respectively.

D.4 Proof for Lemma 25

Proof. Consider any velocity profile consisting of a $\mathbf{C} - \mathbf{U} - \mathbf{C}$ sequence covering distance D. We can replace this $\mathbf{C} - \mathbf{U} - \mathbf{C}$ sequence with a single $\mathbf{C}$ section, so that the resulting velocity profile covers the same distance and consumes energy less than the original profile.

Let $v(t)$ be any velocity profile that contains a $\mathbf{C} - \mathbf{U} - \mathbf{C}$ sequence. That is, $v(t) = v_m(t)$ for $t_0 \leq t \leq t_1$ and $t_2 \leq t \leq t_3$ and $v(t) < v_m$ between $t_1 \leq t \leq t_2$. The energy consumption of this profile for traveling a distance $D = d_{01} + d_{12} + d_{23}$ is $E = E_{01} + E_{12} + E_{23}$, where E_{ij} is the energy spent in traveling d_{ij} for $v(t)$. We construct another velocity profile that is identical to $v(t)$ in $[t_0, t_1]$ and $[t_2, t_3]$ but covers the section d_{12} at $v(t) = v_m$. The energy consumption for this new profile differs only in the d_{12} section.

By following a process similar to that in Lemma 24, we can show that $E'_{12} \leq E_{12}$ leading to $E' \leq E$. Hence, any $\mathbf{C} - \mathbf{U} - \mathbf{C}$ sequence can be replaced by a single $\mathbf{C}$ segment to reduce the energy consumption. Hence, the optimal velocity profile will never consist of a $\mathbf{C} - \mathbf{U} - \mathbf{C}$ sequence. □

D.5 Proof of Energy Model for Non-zero Initial and Final Velocities

Lemma 30. *Let τ be a path with N segments starting and returning to rest, i.e., $v_0(1) = 0$ and $v_f(N) = 0$. Let $E_{14}(i)$ and $E_{16}(i)$ be the minimum energy obtained for the i^{th} segment using Equations 8.5 and 8.4 respectively. Then $\sum_i E_{14}(i) = \sum_i E_{16}(i)$.*

Proof. First, consider the energy obtained using Equation 8.4. For each segment, we have

$$\begin{aligned}
E_{16}(i) &= \int_0^{t_f} \Big[c_1 a^2(t) + c_2 v^2(t) + c_3 v(t) + c_4 + c_5 a(t) + c_6 v(t) a(t)\Big] dt \\
&= \int_0^{t_f} \Big[c_1 a^2(t) + c_2 v^2(t) + c_3 v(t) + c_4\Big] dt + \int_0^{t_f} \Big[c_5 a(t) + c_6 v(t) a(t)\Big] dt \\
&= E_{16}^{14}(i) + E_{16}^{56}(i).
\end{aligned}$$

Now we have,

$$\begin{aligned}
\sum_i E_{16}(i) &= \sum_i E_{16}^{14}(i) + \sum_i E_{16}^{56}(i) \\
&= \sum_i E_{16}^{14}(i) \\
&= \sum_i E_{14}(i).
\end{aligned}$$

The second statement follows since $\sum_i E_{16}^{56}(i) = 0$ when $v_0(0) = v_f(N) = 0$ as given by [131]. That is, the net effect of c_5 and c_6 is zero when the robot starts and returns to rest. □