Multi-Robot Coordination for Hazardous Environmental Monitoring

Yoonchang Sung

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Electrical Engineering

Pratap Tokekar, Chair
Ryan K. Williams
A. Lynn Abbott
Vassilis Kekatos
Manish Bansal

September 27, 2019
Blacksburg, Virginia

Keywords: Environmental Monitoring, Multi-Robot Coordination, Planning Algorithms, Decision Making

Multi-Robot Coordination for Hazardous Environmental Monitoring

Yoonchang Sung

(ABSTRACT)

In this thesis, we propose algorithms designed for monitoring hazardous agents. Because hazardous environmental monitoring is either tedious or dangerous for human operators, we seek a fully automated robotic system that can help humans. However, there are still many challenges from hardware design to algorithm design that restrict robots to be applied to practical applications. Among these challenges, we are particularly interested in dealing with algorithmic challenges primarily caused by sensing and communication limitations of robots. We develop algorithms with provable guarantees that map and track hazards using a team of robots.

Our contributions are as follows. First, we address a situation where the number of hazardous agents is unknown and varies over time. We propose a search and tracking framework that can extract individual target tracks as well as estimate the number and the spatial density of targets. Second, we consider a team of robots tracking individual targets under limited bandwidth. We develop distributed algorithms that can find solutions in bounded amount of time. Third, we propose an algorithm for aerial robots that explores a translating hazardous plume of unknown size and shape. We present a recursive depth-first search-based algorithm that yields a constant competitive ratio for exploring a translating plume. Last, we take into account a heterogeneous team of robots to map and sample a translating plume. These contributions can be applied to a team of aerial robots and a robotic boat monitoring and sampling a translating hazardous plume over a lake. In this application, the aerial robots coordinate with each other to explore the plume and to inform the robotic boat while the robotic boat collects water samples for offline analysis. We demonstrate the performance of our algorithms through simulations and proof-of-concept field experiments for real-world environmental monitoring.

This thesis is supported by the National Science Foundation under Grant No. 1637915.

Multi-Robot Coordination for Hazardous Environmental Monitoring

Yoonchang Sung

(GENERAL AUDIENCE ABSTRACT)

Quick response to hazards is crucial as the hazards may put humans at risk and thorough removal of hazards may take a substantial amount of time. Our vision is that the introduction of a robotic solution would be beneficial for hazardous environmental monitoring. Not only the fact that humans can be released from dangerous or tedious tasks, but we also can take advantage of the robot's agile maneuverability and its precise sensing. However, the development on both hardware and software is not yet ripe to be able to deploy autonomous robots in real-world scenarios. Moreover, partial and uncertain information of hazards impose further challenges. In this these, we present various research problems addressing these challenges in hazardous environmental monitoring. Particularly, we are interested in overcoming challenges from the perspective of software by designing planning and decision-making algorithms for robots. We validate our proposed algorithms through extensive simulations and real-world experiments.

To my lovely wife, Dasom, my family,
and the memory of my mother.

Acknowledgements

Firstly, I would like to express my gratitude to my advisor, Professor Pratap Tokekar, for his guidance and sincerity of support throughout this process. I am deeply grateful for the opportunity to work with him. In addition, I would like to thank my committee members, Professor Ryan K. Williams, Professor A. Lynn Abbott, Professor Vassilis Kekatos, and Professor Manish Bansal, for their valuable comments and suggestions on this dissertation. I would also like to thank Professor Brian Lattimer for giving me the opportunity to study in the States.

I would like to thank my labmates and other peers for their friendship and support as we maneuvered through our doctoral program together.

My sincere appreciation also goes to my friends both in Blacksburg and Korea, too numerous to enumerate, who helped keep me motivated to keep going.

My deepest appreciation belongs to my family for their love and untiring support. This dissertation is specially dedicated to the memory of my mother who continues to inspire me in everything that I do. She had a great soul, and I was fortunate to be her son.

Finally, and most importantly, huge thanks to my wonderful wife, Dasom Kim, for her full support and encouragement.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Environmental monitoring is becoming an important application area for robotics. A prime reason is that many environmental monitoring tasks are either tedious or dangerous for human operators, naturally seeking for the development of an autonomous robotic system to help humans. This thesis is particularly interested in monitoring agents that are hazardous. For truly autonomous operations, we need algorithms that plan the motion of the robots for efficient environmental monitoring. In this thesis, we propose algorithms designed for hazardous environmental monitoring with provable guarantees.

(a) Fukushima Daiichi nuclear disaster in Japan [4].

(b) BP oil spill in the Gulf of Mexico [5].

(c) Detecting intruders [6].

Figure 1.1: Examples of disaster situations that demonstrate the necessity of developing an autonomous robotic monitoring system.

Figure 1.1 shows the Fukushima Daiichi nuclear disaster and the BP oil spill that occurred in Japan in 2011 (Figure 1.1 (a)) and in the Gulf of Mexico in 2010 (Figure 1.1 (b)), respectively. In case of the Fukushima nuclear accident, four reactor buildings were damaged by the tsunami and even to this day no definite plans for decommissioning the plant were made due to the release of toxic radioactive materials [4]. The BP oil spill is known to be the

largest oil spill in history which had become years of the cleanup operation [5]. However, early response and accurate prediction of oil diffusion could have eased the worst outcome.

Hazardous agents are not only limited to plumes (as in the above examples) which have a relatively large size, but also individual targets, such as intruders in a secured area (Figure 1.1 (c)) and poisonous animals in the wild. Tracking and mapping hazards is critical to secure the safety of human lives but doing so manually is difficult and dangerous. Therefore, robots can be a good alternative to humans. Moreover, robots can collect data more precisely, over larger areas, for longer periods of time, than humans. Access to such data can lead to better understanding of the underlying phenomenon.

The advantage of adopting a robotic platform is its agile maneuverability, enabling to explore an environment of interest to track and map hazards in a short amount of time. It also has a powerful sensing capability that can monitor hazards from a further distance. Moreover, the usage of multiple robots dramatically increases these capabilities by coordinating each other through wireless communication.

Possible objectives of these tasks include accurate modeling, data collection, persistent monitoring, etc. These tasks have been verified and applied to practical applications, such as, wildlife habitat monitoring (*e.g.*, a group of birds [7] and fishes [8]), scientific monitoring (*e.g.*, wind [9], soil [10] and water temperature [11]), and hazardous material monitoring (*e.g.*, bacteria [12], chemical source [13], and fire [14, 15]). There, however, still remains many restrictions coming from the challenges of hardware design, sensor development, human-robot interaction, battery lifetime, communication system, and algorithm design. In particular, we tackle algorithmic challenges in this thesis.

To successfully carry out hazardous environmental monitoring, robots must be able to observe hazardous agents and estimate their states (*e.g.*, position). These robots must be capable of searching for the adversarial targets in the environment as quickly as possible. The robots must plan their actions carefully to track them over time and avoid losing the targets from their Field-Of-Views (FOVs). Communication among robots to share their information and coordination of their trajectories are vital for taking advantage of using multiple robots. It is also important to note that robots can physically collect samples of hazardous agents in special cases such as when analyzing samples offline is needed. This thesis contributes algorithms that deal with these challenges.

To employ a robotic system for environmental monitoring, the following questions must be precisely addressed: What objectives must robots care about? How do robots perceive and model an environment that they work on? How do they characterize hazardous agents that they have to monitor? How do they deal with their limited sensing capability? How do they coordinate with other robots? Answers for these questions enable us to construct algorithms and to select suitable robotic systems. In the following section, we propose research problems of our interest corresponding to these questions.

1.2 Research Problems

We study four problems that address many of the challenges. Figure 1.2 presents a diagram of possible challenges that must be handled algorithmically from the perspective of hazardous agents, sensing, and multi-robot coordination. In the following we discuss each of these challenges and possible consequences if not treated appropriately.

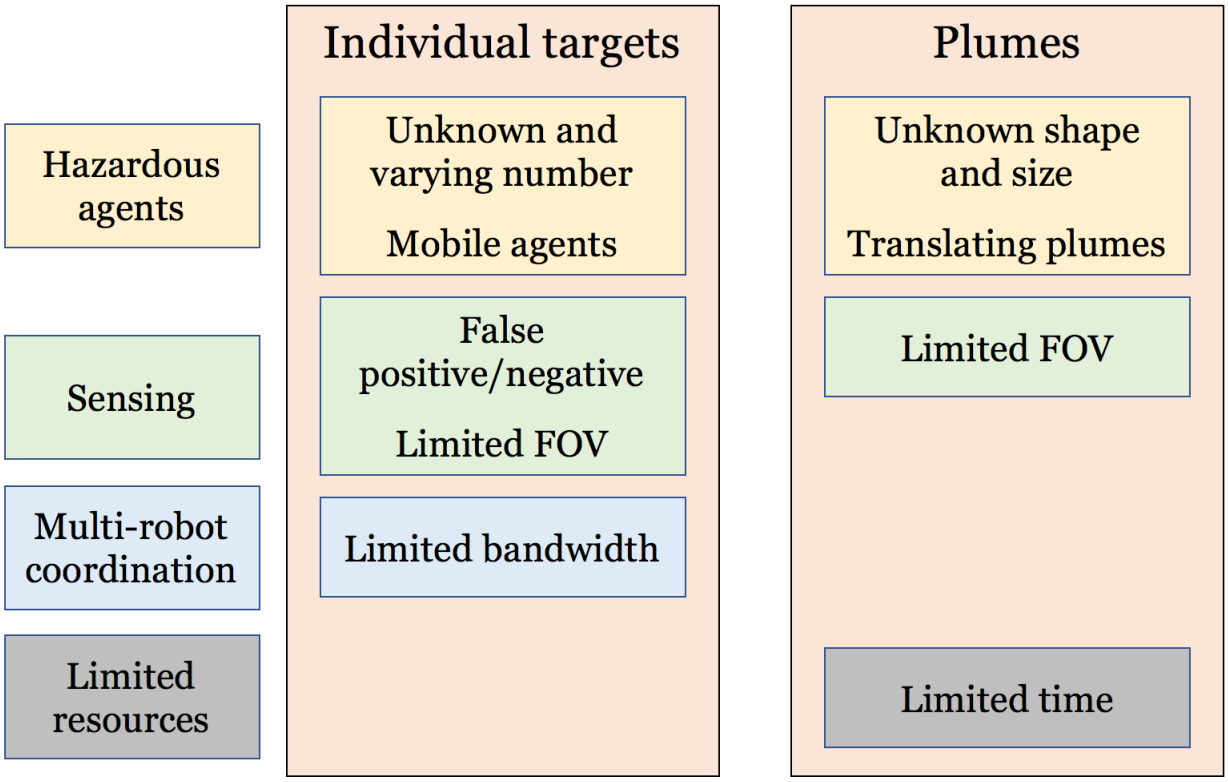


Figure 1.2: Diagram of challenges in hazardous environmental monitoring.

We specifically take into account two types of hazardous agents as they induce different planning strategies for robots: *individual targets* and *plumes*. Individual target entities imply targets that are countable as an integer value and observable as long as they lie within the FOV of a robot. On the other hand, plumes may be of pollutants and radioactive materials. Thus, we need algorithms that plan the motion of the robots to map the plume and track it over time.

We address research challenges caused by possible situations with respect to hazardous agents. In many monitoring individual target entities, the number of targets is usually incorrectly informed or not known in advance due to an imprecise prior knowledge. For example, knowing the exact number of intruders a priori is difficult if the given environment is large. Therefore, it is beneficial to devise a planning algorithm that handles an unknown

number of targets and estimates their states. Additionally, we address a scenario where the number of targets itself also varies over time and targets are mobile.

In the case of plumes, we have an analogous challenge in that the shape and size of the plume may not be known a priori. Generally, we are only informed the existence of hazardous plumes or their locations in some cases. Although robots find the plume, their FOVs might not be able to observe the entire area of plume due to its largeness. For these reasons, figuring out the size and the shape of plumes is the core task for monitoring hazardous plumes. Moreover, sometimes the plume might translate in one direction over time. This, for example, can be seen when the plume translates due to a water flow in lakes and a wind effect.

Imperfect sensing capability of robots produces multiple challenges as follows. Limited FOV of sensors imposes robots to actively search for individual target entities and explore a plume of unknown size and shape. When dealing with target objects, false positive and false negative measurements[1] can be generated by an imperfect sensor. These false positives and false negatives would completely misinform the number of targets, resulting in the failure of the monitoring task. Also, there are cases where obtained samples by robots are often not possible to analyze online. These ex-situ samples can only be studied offline later in the lab once the task is finished. Robots cannot collect all samples of interest due to their limited capacity, and therefore, they must carefully decide whether or not to collect an ex-situ sample along trajectory.

A team of robots can coordinate to speed up the monitoring task. However, this requires them to coordinate with each other, in order to efficiently search for hazardous agents. Furthermore, limitation in communication range and bandwidth can degrade the monitoring performance and even fail the task. Robots must be able to cope with the discrepancy between their available local information and global information (obtained when all robots can communicate with each other) due to the communication limitation.

The environment size may usually be larger than the area covered by the UAV with a single battery life. Due to limited resources (*e.g.*, operation time and fuel), the UAV must plan its trajectory intelligently in order to achieve its objective in limited time. In some scenarios where the goal is to map the boundary of the plume, we consider binary measurements of the plume (*i.e.*, measurement comes from either the plume or the non-plume). In this case, we are interested in finding a minimum-time path of the UAV that can be comparable to the optimal path. In other scenarios where the objective is to identify a maxima of an unknown plume intensity, taking into account continuous measurements that yield a real intensity value is appropriate. To identify a maxima of an unknown intensity function of the plume from stochastic measurements, the UAV has to balance its time spending between exploiting the current knowledge of a learned function and exploring unexplored regions to learn about the rest of the environment.

[1]False positive implies wrongly recognized targets even though they do not exist. It is also known as *false alarms* in literature. False negative corresponds to true targets that were not recognized as targets.

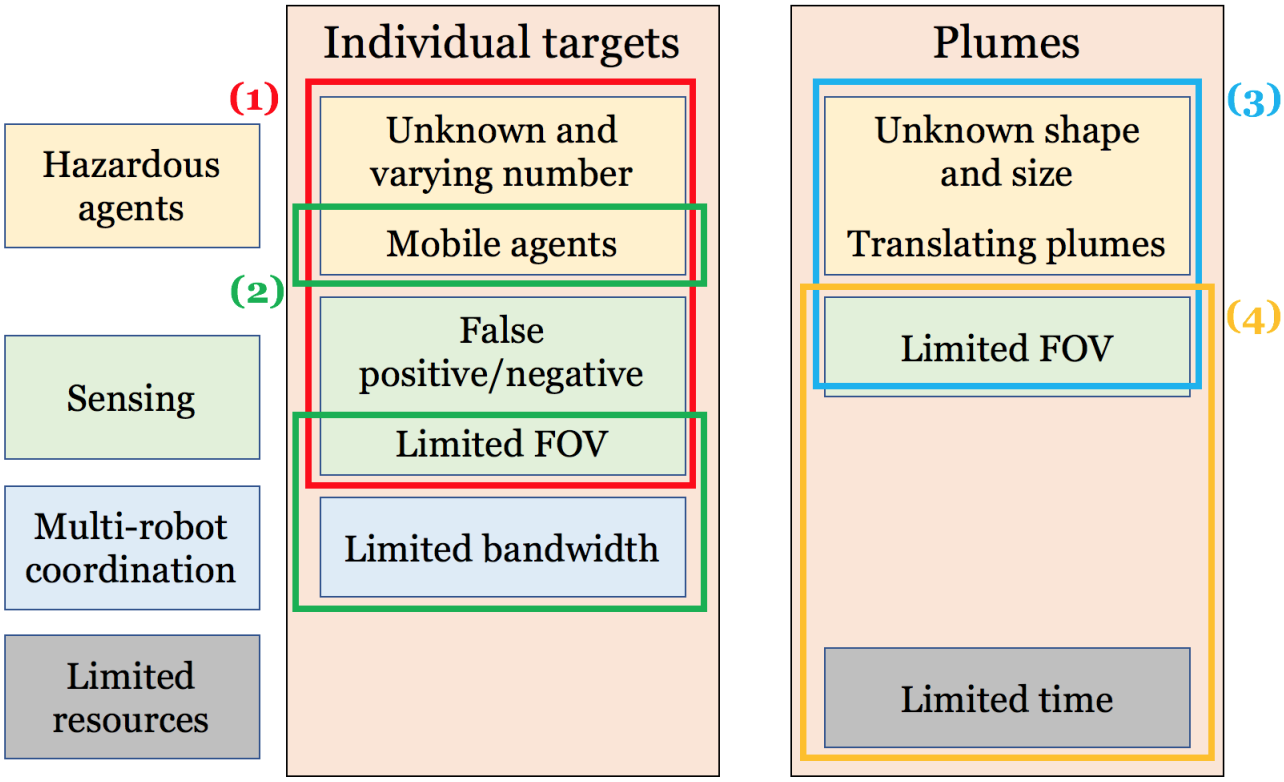


Figure 1.3: Diagram of proposed research problems.

We study four problems that address some of the aforementioned challenges, as shown in Figure 1.3.

Tracking of an unknown and varying number of hazardous targets

Firstly, we study the problem of searching for and tracking a collection of moving individual targets using a robot with a limited FOV sensor. The actual number of targets present in the environment is not known a priori and varying over time. We consider erroneous sensing measurements which yield false positives and false negatives.

Coordinated multi-target tracking under limited bandwidth

Secondly, we study the problem of tracking multiple moving individual targets using a team of robots. Each robot has a set of motion primitives to choose from in order to collectively maximize the number of targets tracked or the total quality of tracking. Our focus is on scenarios where communication is limited and the robots have limited time to share information with their neighbors.

Coordinated online exploration of a translating plume

Thirdly, we study the problem of exploring a translating plume with a team of aerial robots. The shape and the size of the plume are unknown to the robots. The objective is to find a tour for each robot such that they collectively explore the plume. Specifically, the tours must be such that each point in the plume must be visible from the field-of-view of some robot along its tour.

Hotspot identification in limited time

Lastly, we study the problem of exploring a plume using a USV equipped with a single downward-facing camera. To identify the best possible sampling location, the UAV must learn an unknown intensity of the plume in the environment. The goal is to identify a hotspot in the environment in limited time due to a given battery life of the UAV. This problem can be extended to a schenario where we have the UAV explore the plume with its agile maneuverability while the USV physically collects water samples from locations informed by the UAV, which is heterogeneous teaming for environmental monitoring.

1.3 Thesis Contributions

In this section, we briefly summarize the main contributions of the proposed problems before we go into the details in separate chapters.

Tracking of an unknown and varying number of hazardous targets

We propose a search and tracking framework based on the concept of Bayesian Random Finite Sets (RFSs). Specifically, we generalize the Gaussian Mixture Probability Hypothesis Density (GM-PHD) filter which was previously applied for tracking problems to allow for simultaneous search and tracking with a limited FOV sensor. The proposed framework can extract individual target tracks as well as estimate the number and the spatial density of targets. We also show how to use the Gaussian Process (GP) regression to extract and predict non-linear target trajectories in this framework. We demonstrate the efficacy of our techniques through representative simulations and a real data collected from an aerial robot. This work was presented at IEEE International Conference on Robotics and Automation (ICRA 2017) [16] and has been submitted to IEEE Transactions on Automation Science and Engineering (T-ASE) [17].

Coordinated multi-target tracking under limited bandwidth

We present distributed algorithms that can find solutions in bounded amount of time. We present two algorithms: (1) a greedy algorithm that is guaranteed to find a 2–approximation to the optimal (centralized) solution albeit requiring $|R|$ communication rounds in the worst-case; and (2) a *local* algorithm that finds a $\mathcal{O}\left((1+\epsilon)(1+1/h)\right)$–approximation algorithm in $\mathcal{O}(h \log 1/\epsilon)$ communication rounds.[2] Here, h and ϵ are parameters that allow the user to trade-off the solution quality with communication time. In addition to theoretical results, we present empirical evaluation including comparisons with centralized optimal solutions. This work was presented at IEEE International Conference on Robotics and Automation (ICRA 2018) [18] and has been published in Autonomous Robots (AURO) [19].

Coordinated online exploration of a translating plume

We propose a recursive Depth-First Search (DFS)-based algorithm that yields a constant competitive ratio for the exploration problem. The competitive ratio[3] is $\frac{2(S_r+S_p)(R+\lfloor \log R \rfloor)}{(S_r-S_p)(1+\lfloor \log R \rfloor)}$

[2]The approximation ratio of an algorithm is the ratio between the cost obtained by the algorithm and the cost obtained by an optimal algorithm.

[3]The competitive ratio is the worst-case ratio between the cost obtained by an online algorithm and the cost obtained by an offline, optimal algorithm.

where R is the number of robots, and S_r and S_p are the robot speed and the plume speed, respectively. We also consider a more realistic scenario where the plume shape is not restricted to grid cells but an arbitrary shape. We show our algorithm has $\frac{2(S_r+S_p)(18R+\lfloor \log R \rfloor)}{(S_r-S_p)(1+\lfloor \log R \rfloor)}$ competitive ratio under the *fat* condition. We empirically verify our algorithm using simulations. This work was presented at IEEE International Conference on Robotics and Automation (ICRA 2019) [20] and has been submitted to IEEE Transactions on Robotics (T-RO) [3].

Hotspot identification in limited time

We are motivated by environmental monitoring tasks where finding the global maxima (*i.e.*, hotspot) of a spatially varying field is crucial. We investigate the problem of identifying the hotspot for fields that can be sensed using a UAV equipped with a downward-facing camera. The UAV has a limited time budget which it must use for learning the unknown field and identifying the hotspot. Our first contribution is to show how this problem can be formulated as a novel variant of the GP multi-armed bandit problem. The novelty is two-fold: (i) unlike standard multi-armed bandit settings, the arms ; and (ii) unlike standard GP regression, the measurements in our problem are image (*i.e.*, vector measurements) whose quality depends on the altitude at which the UAV flies. We present a strategy for finding the sequence of UAV sensing locations and empirically compare it with a number of baselines. We also present experimental results using images gathered onboard a UAV. This work was submitted to IEEE International Conference on Robotics and Automation (ICRA 2020) [21].

1.4 Thesis Outline

The rest of the thesis is organized as follows. We begin by the problem of tracking of an unknown and varying number of hazardous targets in Chapter 2. In Chapter 3, we present the problem of coordinated multi-target tracking under limited bandwidth. Then, we propose the problem of coordinated online exploration of a translating plume in Chapter 4. Chapter 5 addresses the problem of identifying a hotspot in limited time using a UAV. We conclude remarks in Chapter 6.

Chapter 2

Tracking of an Unknown and Varying Number of Hazardous Targets

2.1 Introduction

We study the problem of searching for and tracking a set of targets using a robot with a limited FOV sensor. This problem is motivated by robotic search-and-rescue [22, 23, 24], surveillance [25], crowd/traffic monitoring [26, 27], and wildlife habitat monitoring [8, 28, 29]. We specifically consider the scenarios where the number of targets being searched is not known a priori. The targets may move during the search process and the motion model of the targets is not known exactly. As the targets are mobile, the robot is also tasked with tracking the target trajectories.

Search and tracking problems can be loosely distinguished depending on whether or not a target is in the FOV: *tracking* when targets are in the FOV, and *search* when targets are out of the FOV. Once all targets are observed by sensor platforms, the search task is accomplished. To successfully conduct the tracking task, the states of targets must be estimated at each time and trajectories of individual targets must be maintained over time. A robust tracking technique must be able to deal with clutter (false positive) measurements which is especially challenging since the true number of targets is not known.

Several techniques have been proposed to unify the search and tracking problems [22, 30]. These include the sequential Monte Carlo filter [31, 32] as well as the Probability Hypothesis Density (PHD) filter [26]. However, the existing works focus on estimating the number of targets and their spatial densities but cannot estimate trajectories of individual targets. On

the other hand, there are existing works on estimating individual target trajectories but assuming an unlimited FOV [33]. Our main contribution is to generalize tracking algorithms for unlimited FOV sensing to the case of limited FOV. We also show how to extend tracking to non-linear motion models by leveraging a GP regression [34] based on the prior work in [26, 35].

The two main contributions of this chapter are (i) a new mechanism for handling a limited FOV sensor in the GM-PHD framework for simultaneous search and tracking problems; and (ii) integration of GP regression in the GM-PHD framework to deal with unknown motion models of targets.

In addition, owing to GM-PHD, our method can also estimate and track an unknown and varying number of targets.

The rest of this chapter is organized as follows. We begin by describing a problem setup in Section 2.3. We present a brief introduction to the GM-PHD filter in Section 2.4. Our proposed algorithm is presented in Section 2.5. We present results from representative simulations and experiments in Section 2.6 before concluding with a discussion of future work in Section 2.7.

2.2 Related Work

Search and tracking with robot teams can be useful in many high impact applications such as disaster recovery [36], habitat monitoring, surveillance, and patrolling. Murphy *et al.* [37] gives an overview of robotic technology applied to search and rescue during disaster recovery. The References [38, 39, 40] have proposed various strategies for search and rescue response with robots. Robots can be used to collect data that will be useful for biologist and policy-makers from wildlife habitats by searching and tracking for biological phenomena of interest [29, 8, 41, 42]. Patrolling requires a single or a team of robots to move around in a known environment to search and track intruders and possibly capture them [43]. In the rest of this section, we survey existing search and multi-target tracking algorithms and show how they are related to the proposed work.

Search techniques have been applied to a broad range of problems (*e.g.*, [44, 45, 46, 23, 47, 48, 49]). Miller *et al.* [47] investigated planning strategies to drive a robot to a desired position for search theory. Chung and Burdick [48] proposed a decision-making approach to find the optimal control for searching. Ryan and Hedrick [49] presented an information-theoretic approach to minimize entropy during search. Hollinger *et al.* [50] proposed an approximation algorithm that finds multi-robot search path planning in a known environment. The recent survey by Chung *et al.* [51] gives a good overview of the search problem.

For the multitarget tracking problem, Joint Probability Data Association (JPDA) [52] and Multiple Hypothesis Tracking (MHT) [53] have become canonical algorithms. These tech-

niques have been applied to many problems including human following [54], object tracking [55] and human-robot interaction [56]. However, JPDA requires solving the data association problem which is especially costly when the actual number of targets is not known exactly [33]. Conventional Bayes trackers use a vector representation in which the order of the targets and its size is known and fixed. This makes tracking with an unknown number of targets intractable. In [57], the Extended Kalman Filter (EKF) simultaneous localization and mapping algorithm handles a varying number of targets but does not represent an explicit distribution over the number of targets. However, the PHD filter [58] that we use in this chapter avoids these problems with the help of random set representations [59].

Dames *et al.* [60] adopted the PHD filter with a finite FOV to estimate the position of hidden objects. Their approach, however, is based on a discrete grid map whereas we use a continuous representation. They use a binary sensing model where the output of the sensor is 1 only if the sensor detects one or more targets (including false positive ones) in a grid cell. If multiple targets are present in a cell, the sensor will still report 1. In contrast, we consider a sensor that reports the position of all targets separately.

The PHD filter has various approximate variants because solving the PHD recursion exactly is difficult. Approximations include the Sequential Monte Carlo PHD (SMC-PHD) filter, the GM-PHD filter, and the Cardinalized PHD (CPHD) filter. The SMC-PHD filter is based on the particle filtering approach, and thus, it requires clustering of particles in order to interpret target states. Dames *et al.* [26] exploited SMC-PHD for localizing and tracking an unknown number of targets but their framework does not extract individual tracks of targets and this has been extended to a multi-robot version in Dames [61]. We use GM-PHD that makes it more convenient to extract individual targets. CPHD relaxes the restriction of a first-order cardinality distribution [62]. Mahler [63, 64] pointed out that CPHD has $O(m^3n)$ complexity while PHD has $O(mn)$ complexity, where n is the current number of targets and m is the current number of measurements, although CPHD yields a smaller variance in the cardinality distribution [65]. GM-PHD is more intuitive for multi-target tracking since each component can refer to one target or a cluster of targets. This makes the planning process easier (*e.g.*, we use the components to design two simple control laws to guide the robot).

Wasik *et al.* [66] employed GM-PHD for multi-robot formation control considering a limited FOV sensor. However, they used the standard PHD filter without any modification with limited FOV sensing. Instead, we present a new approach (described in Section 2.5.2) and show through simulations that this results in a better performance (Section 2.6.1). Furthermore, their work assumes a known state transition model for robots whereas we learn the model using GP regression.

The outputs of the GM-PHD filter are stacked at each time step as a set of tracks (we discuss with more details in Section 2.5). According to Mahler [63], PHD is more likely JPDA than MHT in spirit as the association between PHD components and tracks takes place in the current time step, whereas MHT considers the possible whole history of track. For track maintenance, a few temporal association schemes have been proposed: Lin *et al.* [67]

proposed the peak-to-track association as a two dimensional assignment problem; Panta *et al.* [68] presented the track-to-estimate association based on SMC-PHD; and the GM-PHD-based track-to-estimate association was proposed by the same authors in the Reference [69]. In this work, we adopt the temporal association proposed by the Reference [69].

2.3 Problem Description

We study the problem of finding and tracking the unknown and varying number of targets of interest moving in an environment using a robotic sensor with a limited sensing range. We consider a scenario where the number of targets present in the environment is not known a priori. Initially, only an estimate of the number of targets and a probability distribution over their initial spatial locations is given. The actual number of targets may be different.

We assume that all targets move independently of each other, and that their motion models are not necessarily known to the robot. We allow for targets to move on a non-linear trajectory, however, we assume that the trajectories be smooth (in the sense, that will become clearer in Section 2.5.2). The robot has an onboard sensor capable of detecting the location of targets that are in the sensor's FOV. If the target is not present in the FOV, then it does not generate any measurement. However, if a target is present in the FOV then it is detected by the robot with probability p_D. If the target is detected, then the sensor returns a measurement of the position of the target. We assume that the measurement noise is additive and Gaussian with known covariance. In addition, at any time step, the sensor may also generate false-positive measurements uniformly at random in the FOV.

We present an estimation framework based on the concept of RFSs to deal with the search and tracking problem. The proposed method can estimate the states of targets and the number of targets at the same time, and initiate and terminate tracks. Throughout the chapter, we present illustrations and simulations assuming that the environment is 2D and obstacle-free, and the robot has a circular FOV. However, the proposed techniques easily extend to more complex scenarios.

2.4 Preliminaries

In multitarget-multisensor tracking, Recursive Bayesian Estimation (RBE) has been a canonical tool to estimate target states from observations obtained by imperfect sensors. A standard assumption is that the number of targets is known exactly. Hence, we can treat the positions of all the targets at any time as a random vector and use RBE for estimation. We

consider scenarios where the number of targets itself is not known. Hence, standard RBE techniques that use a vector representation cannot directly be used since there is uncertainty on the length of the random vector itself making the Bayesian updates intractable.

Mahler [58] developed the PHD filter to tractably solve exactly this class of problems. PHD, also known as the intensity function, when integrated over any subset of the environment yields the expected number of targets present in that subset.[1] The advantage of PHD is that it allows estimation of both target states and the number of targets simultaneously without the necessity of data association. We briefly discuss the PHD filter next but refer the reader to Reference [63] for an in-depth discussion.

PHD is the first-order statistical moment of RFS and denoted by v. We denote the multitarget posterior density by $p_{k|k}(X|Z_k)$, where X is a multitarget state set ($\mathbf{x}_i \in X$ is a state of the i-th target) and Z_k is an observation set ($\mathbf{z}_{j,k} \in Z_k$ is the j-th measurement at time step k). The robot state is denoted by $\mathbf{y}$. $p_{k|k}(\cdot)$ takes all previous measurements into account. The expected number of targets in any region S is:

$$\int |X \cap S| p_{k|k}(X|Z_k)\delta X = \int_S v_{k|k}(\mathbf{x})d\mathbf{x}, \tag{2.1}$$

which is the integral of PHD over S.

Similar to a Kalman Filter, RBE with PHD consists of a prediction step followed by an update step. The prediction and update equations of a PHD are given by $v_{k|k-1}(\mathbf{x}) := v_{k|k-1}(\mathbf{x}|Z_{k-1})$ and $v_{k|k}(\mathbf{x}) := v_{k|k}(\mathbf{x}|Z_k)$, respectively, for notational convenience. The prediction equation [64] is:

$$\begin{aligned} v_{k|k-1}(\mathbf{x}) = &\int p_S(\mathbf{w}) f_{k|k-1}(\mathbf{x}|\mathbf{w}) v_{k-1|k-1}(\mathbf{w})d\mathbf{w} + \\ &\int \omega_{k|k-1}(\mathbf{x}|\mathbf{w}) v_{k-1|k-1}(\mathbf{w})d\mathbf{w} + \beta_k(\mathbf{x}), \end{aligned} \tag{2.2}$$

where $p_S(\cdot)$, $f_{k|k-1}(\cdot|\cdot)$, $\omega_{k|k-1}(\cdot|\cdot)$ and $\beta_k(\cdot)$ denote the probability of survival of existing targets, the Markov transition density, the intensity of spawning new targets from existing targets and the intensity of birthing targets. The update step [64] is:

$$\begin{aligned} v_{k|k}(\mathbf{x}) = &[1 - p_D(\mathbf{x}, \mathbf{y})] v_{k|k-1}(\mathbf{x}) + \\ &\sum_{\mathbf{z} \in Z_k} \frac{p_D(\mathbf{x}, \mathbf{y}) g_k(\mathbf{z}|\mathbf{x}) v_{k|k-1}(\mathbf{x})}{k(\mathbf{z}) + \int p_D(\mathbf{w}, \mathbf{y}) g_k(\mathbf{z}|\mathbf{w}) v_{k|k-1}(\mathbf{w})d\mathbf{w}}, \end{aligned} \tag{2.3}$$

[1] PHD is not a probability density function, meaning that the integral over the entire region of PHD does not necessarily sum to 1.

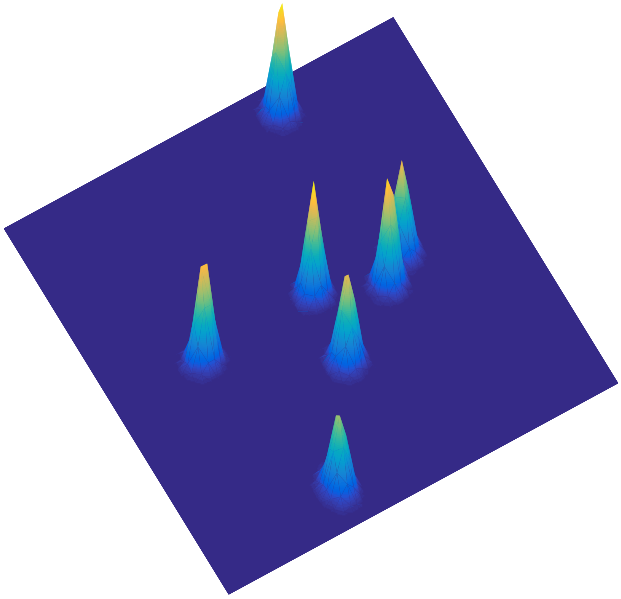


Figure 2.1: The GM-PHD filter with 7 Gaussian components.

where $p_D(\cdot)$, $g_k(\cdot|\cdot)$ and $k(\cdot)$ denote the probability of the detection, the sensor likelihood and the intensity of clutter (*i.e.*, false-positive measurements). The probability of detection depends on the FOV of the sensor as well as the state of targets. The sensor likelihood is given by the likelihood of obtaining a position measurement with additive zero-mean Gaussian noise with known covariance. Clutter and the predicted multitarget RFS follow the Poisson model [70].

The PHD filter propagates the intensity recursively over time through Equations (2.2) and (2.3). The details of the derivation of the PHD recursion are given in Reference [58].

Performing exact prediction and update by the general PHD recursion is computationally intractable. Instead, particle filter-based approaches [71] and Gaussian mixture-based approaches [33] have gained attention for the realization of PHD. The particle PHD is suitable for dealing with nonlinear motion of targets. GM-PHD, however, assumes that a target has a linear motion model. Nevertheless, we can use the Extended Kalman Filter (EKF) and Unscented Kalman Filter (UKF) versions of GM-PHD [33]. GM-PHD gives a closed-form solution without requiring a large sample size and clustering techniques to extract multitarget state estimates, which both are necessary for the particle PHD.

In a GM-PHD, the intensity function (*i.e.*, PHD) is approximated as a Gaussian mixture model of one or more Gaussian components (Figure 2.1) and can be expressed as:

$$v(\mathbf{x}) = \sum_{i=1}^{n} w^{(i)} \mathcal{N}(\mathbf{x}; m^{(i)}, P^{(i)}), \tag{2.4}$$

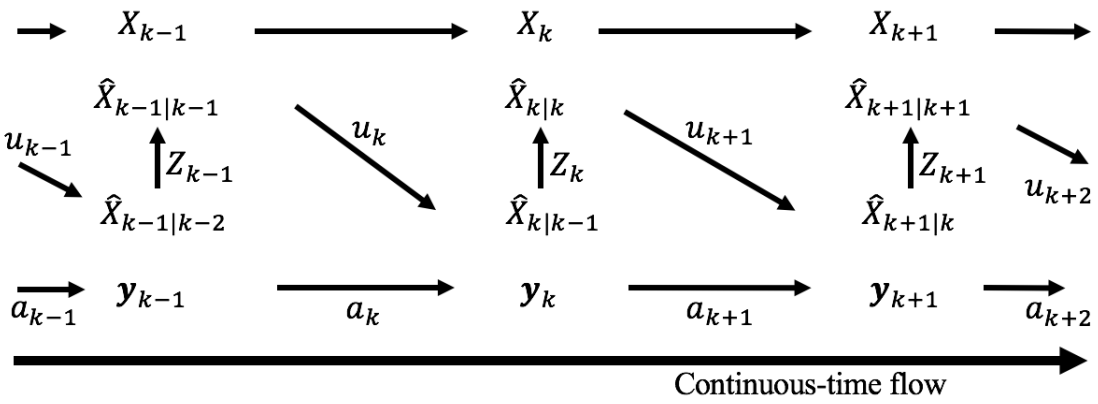


Figure 2.2: Time framework of RBE for targets and a robot. We use $\mathbf{y}$ to denote the state of robot. The true state is denoted by $(\cdot)$ while the estimated state is denoted by $\hat{(\cdot)}$. u and a correspond to the control input of a component and sensor, respectively.

where n is the number of Gaussian components and each Gaussian component is represented by its mean (m), covariance (P), and weight (w). We drop "Gaussian" from "Gaussian component" for simplicity from now on. The weight of a component gives the expected number of targets generated as a result of that component.[2] According to Vo and Ma [33], the GM-PHD prediction equation is:

$$v_{k|k-1}(\mathbf{x}) = v_{S,k|k-1}(\mathbf{x}) + v_{\beta,k|k-1}(\mathbf{x}) + \gamma_k(\mathbf{x}), \tag{2.5}$$

where $v_{S,k|k-1}(\cdot)$, $v_{\beta,k|k-1}(\cdot)$ and $\gamma_k(\cdot)$ correspond to the GM-PHD of survival, spawn and birth RFSs, respectively. The GM-PHD update is:

$$v_{k|k}(\mathbf{x}) = (1 - p_D)v_{k|k-1}(\mathbf{x}) + \sum_{\mathbf{z}\in Z_k} v_{D,k}(\mathbf{z}), \tag{2.6}$$

where $v_{D,k}(\cdot)$ is the GM-PHD induced from the sensor likelihood. We refer the reader to Reference [33] for a detailed discussion of GM-PHD. Figure 2.2 presents the state propagation for targets and a robot over time.

2.5 GM-PHD SAT Algorithm

In this section, we define our main algorithm for GM-PHD based search and tracking. Throughout the chapter, we use the terms *component*, *target* and *track* frequently. These

[2]Note that the weight of a component does not correspond to the expected number of components but the expected number of targets. This implies a component might contain more than a target in it. If $w \ll 1$, then the expected number of targets in that component can be considered as 0.

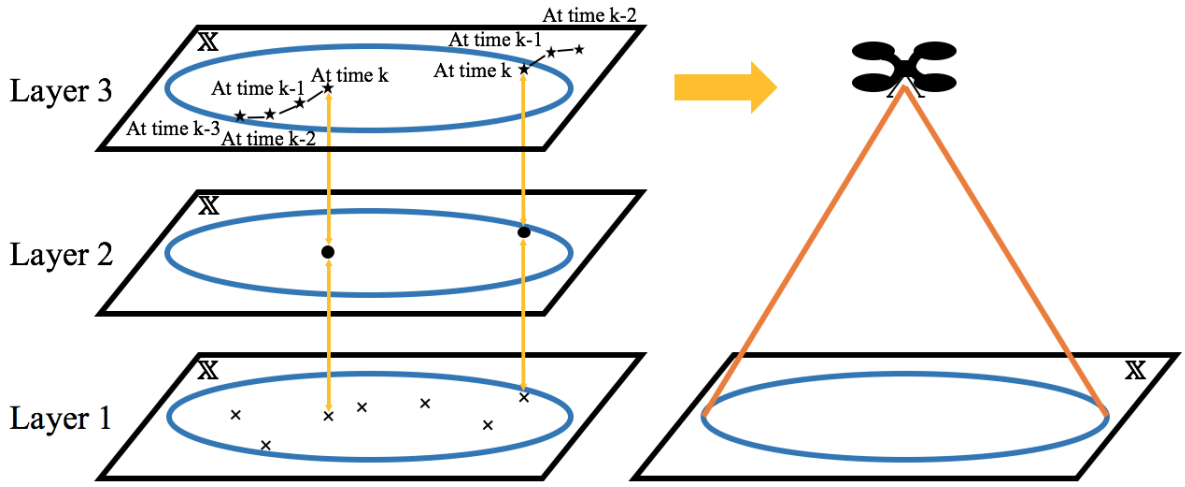


Figure 2.3: Hierarchical layers of the proposed scheme. The x marks of *Layer 1*, the circle marks of *Layer 2* and the star marks of *Layer 3* denote components, targets and tracks, respectively. A robot in the right figure utilizes information of Layer 3 to carry out the search and tracking task.

are defined based on *three layers* in a hierarchical order (Figure 2.3). The core algorithm of GM-PHD SAT works in the lowest layer, *i.e.*, *Layer 1*, consisting of the components of the posterior GM-PHD. Each component is specified by its weight, mean and covariance. Among these components, those with a large weight can then be extracted and considered as targets of interest in *Layer 2*. Components that are not extracted as targets can be viewed as *tentative targets*. In *Layer 3*, trajectories of targets are extracted as tracks that include the history of targets over time. Each track is assigned an ID which is maintained over time. Components and targets, however, do not have IDs. Figure 2.3 gives a more in-depth picture of the hierarchical layers.

Figure 2.4 describes the overall flowchart of the proposed algorithm. We start with an initial estimate of the PHD. Multitarget Bayes filter, *i.e.*, the prediction and update steps, is applied recursively to estimate the state of targets based on a limited FOV. Since PHD is employed, additional data association between targets and measurements is not required. The pruning and merging scheme reduces components with low and similar weights, respectively. Then, multitarget state estimates are extracted from GM-PHD and used for maintaining trajectory states of targets. Finally, an active control strategy is used to the robot to search for and track the targets.

We describe our new contributions and refer the reader to Reference [69] for a discussion of the other blocks. Specifically, we show how to extend GM-PHD to allow for a limited FOV mobile sensor (Section 2.5.2), how to use GP regression to predict non-linear motion models of the targets (Section 2.5.2), extracting and managing tracks of individual targets (Section 2.5.6), and two heuristic strategies for actively controlling the robot's state (Sec-

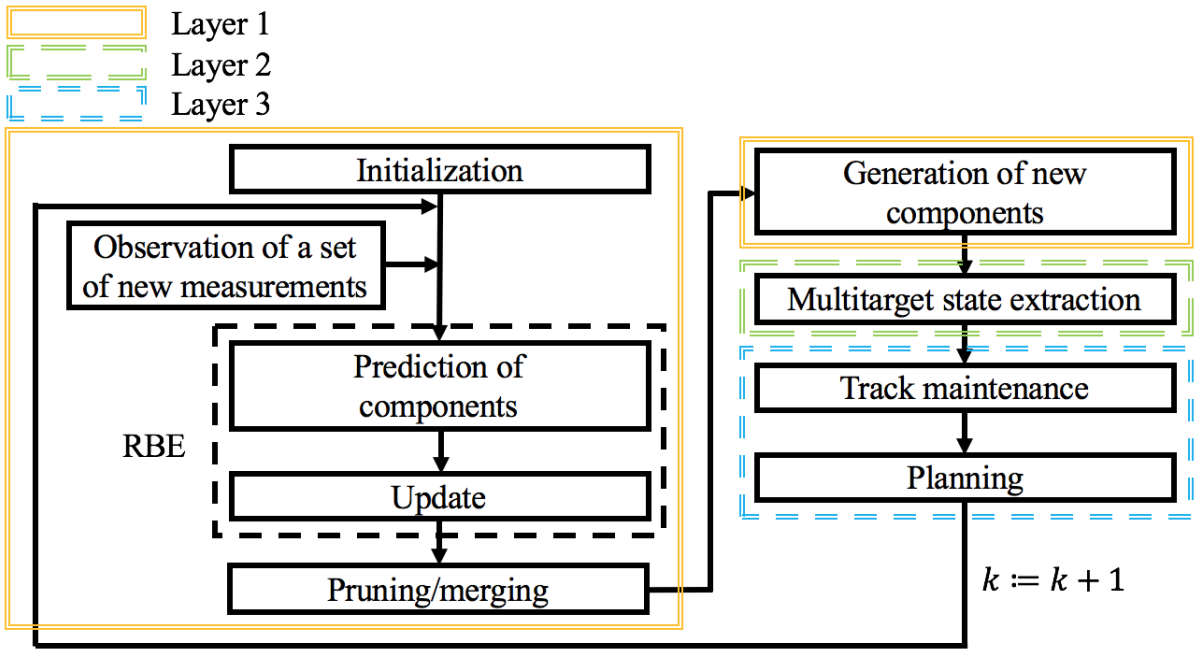


Figure 2.4: Flowchart for the GM-PHD search and tracking algorithm.

tion 2.5.7). In the following, we describe each block in details.

2.5.1 Initialization

The initialization block produces a set of components that constitutes the initial GM-PHD representing the initial belief of targets. To conduct a search mission, initial belief for possible locations of lost targets can be defined a priori from external sources. Examples of external sources include: mayday signals from missing crews in disaster scenarios [22], abandoned dangerous elements in security missions [72], unknown transient radio sources from the sensor network deployed by enemies [73] and high-frequency radio signals from tagged animals for monitoring wildlife habitat [42]. If we know the region where a target may be present, we construct a component covering the region. Nearby components can be clustered into a single component having the weight that corresponds to the summed weight of combined components.

The possible number of targets in any components reported by external sources can be expressed by the weight. The initial GM-PHD may be an underestimate or an overestimate of the true number of targets. We evaluate the consequence of three different cases (including the exact estimate) for the initial belief in simulations.

2.5.2 Recursive Bayesian Estimation

The RBE block takes prior GM-PHD and produces the posterior GM-PHD as output. At the first time step, the prior GM-PHD comes from the initialization block. In subsequent time steps, the prior GM-PHD comes recursively from the posterior components of previous time steps. The RBE block performs the prediction and update steps. We follow a similar procedure as that proposed by Vo and Ma [33] (see Table 1 in Reference [33]) with suitable modifications to account for a limited FOV sensor. Algorithm 1 shows the pseudo-code for RBE.

Algorithm 1: RBE

Step 1: A prediction for birth components. Apply a simple linear motion model as proposed in Step 1 of Table 1 from Reference [33].
Step 2: A prediction of existing components. Apply the GP regression over confirmed tracks.
Step 3: A construction of PHD update components (Step 3 of Table 1 from Reference [33]).
Step 4: An update.
Require: the number of predicted components.

for $i \in \{1, ..., n_{k|k-1}\}$ **do**
  Compute $p(\mathcal{F})$ using Equation (2.10).
  Compute $p(\mathcal{D})$ using Equation (2.9) for all components.
  The *no detection* event: $w_{k|k}^{(i)} = (1 - p_D) w_{k|k-1}^{(i)}$. $P_{k|k}^{(i)} = P_{k|k-1}^{(i)}$.
  if $threshold_{lower} \leq p(\mathcal{D}) \leq threshold_{upper}$ **then**
    Apply Equation (2.11) to the mean of the i-th component.
  else
    $m_{k|k}^{(i)} = m_{k|k-1}^{(i)}$.
  end if
end for
for $\mathbf{z} \in Z_k$ **do**
  The *detection* event: refer to the update part with respect to measurements in Step 4 of Table 1 from Reference [33].
end for

The prediction equations (Steps 1 and 2 of Algorithm 1) need a motion model for the components. Rather than assuming a known motion model (*e.g.*, linear), we use GP regression to estimate a motion model in a data-driven fashion. The PHD prediction requires knowing the motion model, $f_{k|k-1}$, for each of the targets. In previous works, a simple linear motion model was applied [33]. Instead, we aim at dealing with an unknown motion model by using GP regression [34] which is a non-parametric, Bayesian, and non-linear regression technique which requires specifying a kernel function. In our previous works, we have shown how GP regression can be employed to learn the spatial velocity vectors of targets for a real-world

taxi dataset [26]. Here, we employ GP regression to extrapolate each target's trajectory and predict its future positions.

Noisy measurements of the state of the targets are fed as input to GP regression, which produces a prediction of its future positions. In particular, we use GP regression to estimate d functions, $f_i(t)$ where $i = 1, \ldots, d$, that predicts the evolution of the state of the target along each of its d dimensions, independently. Figure 2.5 shows an example of the 2D case and the result of GP regression applied to a trajectory sample. From a distribution obtained from GP regression, future trajectory mean position with covariance can be extrapolated.

We use the squared exponential function [34] as our kernel. The squared exponential kernel is a function of only the distance between two inputs. Therefore, the squared exponential kernel for two input times, k and k', is:

$$\mathcal{K}(k, k') = \sigma_f^2 \exp\Big[-\frac{1}{2}\Big(\frac{k - k'}{\lambda}\Big)^2\Big], \tag{2.7}$$

where $\sigma_f^2 > 0$ is the signal variance and $\lambda > 0$ is the lengthscale. The hyperparameters (σ_f and λ) for the kernel are learned offline using a training set consisting of noisy observations of the target's motion.

In order to apply a GP regression to predict the motion of each Gaussian in GM-PHD, we must have a *confirmed track* of individual targets (it will be explained in Section 2.5.6). If a Gaussian is not assigned to a confirmed track, then we can use a simple linear motion model for the prediction. Once a track is confirmed (*i.e.*, we have sufficient history of an individual target trajectory) we employ a GP regression to predict its motion.

For any confirmed track, let M^d be the history of mean values of components along the d-th dimension. Let K be the set that records the time steps corresponding to the mean values. The following equation gives the predicted mean at time step k for each corresponding confirmed track along the d-th dimension:

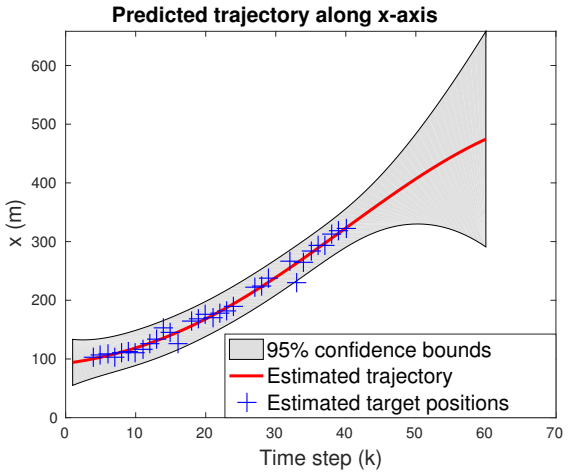
$$m_k^d = \mathcal{K}(k, K)[\mathcal{K}(K, K) + \sigma_n^2 I]^{-1} M^d. \tag{2.8}$$

Here, σ_n^2 is the variance of the measurement noise which is also a hyperparameter learned from the data.

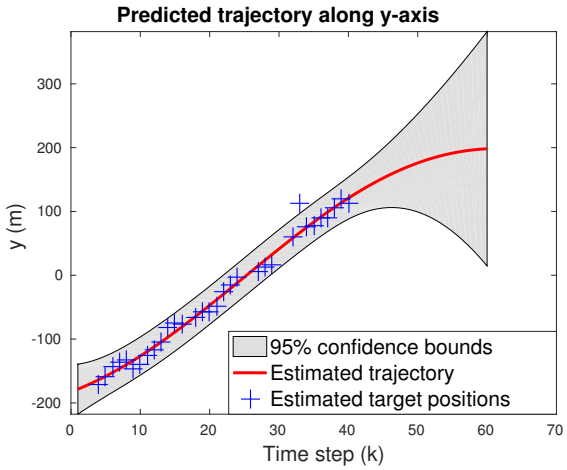
It might be time-consuming to perform GP regression online because the GP prediction has cubic complexity in the length of history of the confirmed track [34]. Thus, we do not use the entire history but only a recent subset.[3]

Figure 2.5 shows an example of the 2D case and the result of GP regression applied to a trajectory sample. From a distribution obtained from GP regression, future trajectory mean position with covariance can be extrapolated.

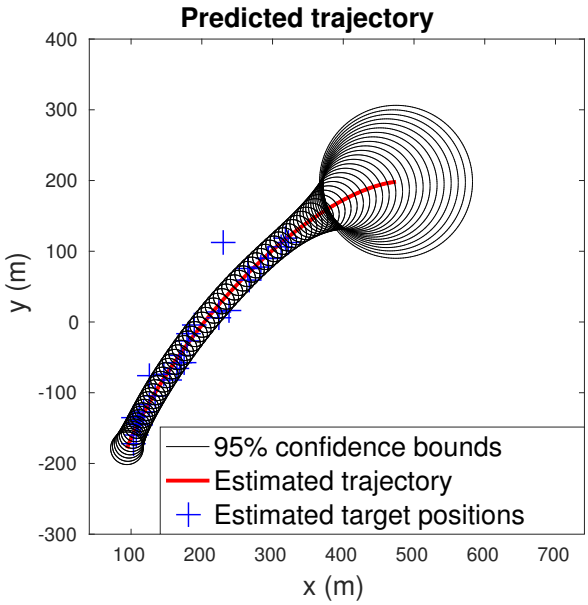
[3]For example, when using the GPML toolbox [34] implemented in MATLAB, it took 0.29, 0.35, and 3.53 seconds to compute Equation (2.8) when the length of the history was 100, 1000, and 5000, respectively.



(a) Predicted trajectory with the interval along x– axis.



(b) Predicted trajectory with the interval along y– axis.



(c) Predicted trajectory in 2D generated by covariance ellipses of the GP regression.

Figure 2.5: Result of the GP regression applied to a 2D trajectory sample.

The update of predicted components has two parts: components with the *no detection* event (lines 1-10 of Algorithm 1); and components compared with all measurements observed in the corresponding time step (lines 11-13 of Algorithm 1). The *no detection* event reflects the possibility of target lost by not assigning any measurements to each component. Thus, the computation complexity for the update is $\Theta(|X||Z+1|)$. We incorporate a limited FOV sensor in the update equations. Specifically, we show how to compute the probability of detection (*i.e.*, p_D) that explicitly considers the limited FOV of the robot.

Without loss of generality, we assume that the robot has a circular FOV with a radius of r and centered at $\mathbf{y} = (y_x, y_y)$. We define two events: $\mathcal{F}$ denotes an event of a target inside the FOV; and $\mathcal{D}$ is an event for a target being detected by the robot. The probability of detection is:

$$p_D := p(\mathcal{D}|\mathcal{F})p(\mathcal{F}), \tag{2.9}$$

where $p(\mathcal{D}|\mathcal{F})$ is a probability of a target being detected given that it is inside the FOV of the robot, which characterizes the performance of sensor [33]. For example, in case of the radar sensor, $p(\mathcal{D}|\mathcal{F})$ corresponds to the probability of having a radar intensity that is above a certain detection threshold when a target exists. Another example is the probability of having a feature in the image when employing a feature detector. $p(\mathcal{F})$ is a probability of having a target inside the FOV and is given by:

$$\begin{aligned} p(\mathcal{F}) = \int_{y_{min}}^{y_{max}} \int_{x_{min}}^{x_{max}} \frac{1}{2\pi\sigma_x\sigma_y\sqrt{1-\rho^2}} \exp\Big(\frac{-1}{2(1-\rho^2)} \\ \times \Big[\frac{(x-\mu_x)^2}{\sigma_x^2} + \frac{(y-\mu_y)^2}{\sigma_y^2} - \frac{2\rho(x-\mu_x)(y-\mu_y)}{\sigma_x\sigma_y}\Big]\Big) dxdy, \end{aligned} \tag{2.10}$$

where μ and σ are the mean and standard deviation of a component. The integral is over the circular region bounded by $x_{min} = y_x - r \leq x \leq y_x + r = x_{max}$ and $y_{min} = y_y - \sqrt{r^2-(x-y_x)^2} \leq y \leq y_y + \sqrt{r^2-(x-y_x)^2} = y_{max}$, and ρ is the Pearson correlation coefficient. Depending on how far components are located away from the robot, Equation (2.9) naturally encodes the amount of influence that the robot affects each component; a component that is far away from the robot barely gets updated.

In the case of a no-detection event, the PHD values inside the FOV decrease more than those outside the FOV. We can see from Line 4 of Algorithm 1, the weights of the components centered in the FOV will decrease significantly whereas those that are centered much farther from the FOV will barely decrease. In our testing, we found that this adversely affected components that were centered near the boundary of the FOV. Line 4 of Algorithm 1 reduced the weights of these components more than desired which caused components that corresponded to true targets to be pruned away in future steps. Therefore, we propose a specific treatment for such components (Lines 5–7 of Algorithm 1).

Existing components that are located near the boundary of the FOV (*i.e.*, p_D lies between thresholds) get pushed outside the FOV in case of a no-detection event. That way, we may be able to prevent components that correspond to true targets from being pruned. As the states of these components can be corrected later if measurements for them are made, this treatment is the trade-off between estimation accuracy and robustness for false-negative errors. The following equation only updates those components which have a p_D between thresholds:

$$\begin{bmatrix} \mu_x \\ \mu_y \end{bmatrix} = \begin{bmatrix} p_D(\mu_x - y_x) + \mu_x \\ p_D(\mu_y - y_y) + \mu_y \end{bmatrix}. \tag{2.11}$$

This attribute can also be considered as a counter-effect of the update step that is being used in general Bayesian filters; targets are attracted by the robot in the detection event whereas targets are repulsed in the no-detection event. As our simulation results show, this additional step improves the performance of the estimator.

2.5.3 Pruning/Merging

The pruning/merging block takes a set of posterior GM-PHD components as input from the RBE block and produces a set of the reduced number of GM-PHD components. In the update step of the GM-PHD SAT algorithm, the number of components increases rapidly as the combination of all measurements and existing components is considered at every time step. Vo and Ma [33] proposed pruning and merging algorithms to eliminate less important components. We prune away all components that have weights smaller than a threshold. We recursively find a component having the largest weight and compute the Mahalanobis distance [74] with respect to all other components. Then, we merge those which have a distance less than a threshold and remove them from candidates for the next recursion. We continue the recursion until either no component is left as candidate or the number of components meets a threshold that bounds the total number of components. In the latter case, we prune away components that are not yet merged. A survived component computes its weight by summing the weight of merged components. The mean and covariance are averaged among merged component.

2.5.4 Generation of New Components

This block takes a set of reduced number of components in GM-PHD and measurements as input from the pruning/merging block and produces a set of GM-PHD components that come from the pruning/merging block as well as a set of new components generated from measurements as output.

The proposed algorithm, so far, cannot handle a target that gets inside the FOV but was not a member of components in the previous time step. Consider a situation where the initial

PHD underestimates the number of true targets. At some point, some target that has no components associated with it may enter the FOV of the robot. When a measurement is obtained, it is not known if it corresponds to such a target that has no associated component or if it is a false-positive measurement. We employ a measurement-driven model proposed by Ristic *et al.* [75] for the generation of new components as follows. We create a new component for every measurement (in addition to updating existing components with this measurement as described in the RBE block). A new component has its mean as the position of the corresponding measurement, its variance as a measurement noise, and its weight as one because each measurement corresponds to a single target.

For those newly generated components we can expect the following three possible consequences. First, if a component comes from a true target that was not recognized beforehand, it will survive. The weight of the component would increase over time as it would have more subsequent measurements. Secondly, if a component turns out to be a result of false-positive measurement, it will eventually be pruned away as it would not have further measurements. Lastly, if a component is generated from a target that is already assigned to any existing components, it will merge to the corresponding existing component.

2.5.5 Multitarget State Estimation

The multitarget state estimation block takes a set of GM-PHD components and produces a set of targets extracted from a set of components that have weights above threshold as output.

Up to the previous sections, the lowest layer, Layer 1, was only considered to propagate components by employing the GM-PHD filter. It is crucial to extract targets of interest from the raw components. This is done in Layer 2. We set a weight threshold for components that survive after pruning and merging which turns some components into targets (if their weights are above the threshold). We set a weight threshold for pruned components and merged components that turn some components into targets if weights are above the threshold. Many false positives would survive if the weight threshold is set to a low value. On the other hand, if it is set to a high value, many false negatives would occur. A reasonable threshold value can be preset by calibrating the value between trade-offs. The weight threshold indicates the tolerance of accepting components as targets based on their measurement likelihoods; the higher the maximum likelihood of true measurement, the higher the weight threshold can be set, and vice versa. This block helps to avoid false-positive targets by not considering trivial components. Table 3 of Reference [33] shows an algorithm for the multitarget state estimation.

2.5.6 Track Maintenance

The track maintenance block takes a set of targets as input and produces a set of tracks for the targets that have survived over time as output.

It is important to keep the track continuity of the PHD filter so that the trajectories of individual targets can be observed and maintained. We achieve the track maintenance in Layer 3 and take a track-to-estimate approach. N_k tracks at time k are denoted by $T_k = \{\mathbf{t}_k^i \mid \forall i \in \{1, ..., N_k\}\}$. The i-th track at time k, $\mathbf{t}_k^i$, is represented as: $\mathbf{t}_k^i = (x_{1,1}, x_{1,2}, ..., x_{1,d}, \ldots, x_{l,1}, x_{l,2}, ..., x_{l,d}, i) \subseteq \mathbb{R}^{d \times l} \times \mathcal{Z}_{\geq 0}$, where d is the dimension of the target state, l is the life length of track, and i is a non-negative integer representing the track ID. Each existing track is associated with targets that lie within the Mahalanobis distance threshold used in the merging step (*i.e.*, the gating condition). If there exist more than one target or no tracks satisfy the threshold for targets, we generate new tracks with corresponding IDs. The details of the track continuity of the particle PHD filter and the GM-PHD filter are explained in References [76] and [69], respectively.

Two types of tracks are defined in Layer 3: *tentative track* if $l < l_{Threshold}$ and confirmed track if $l \geq l_{Threshold}$. The mechanism of tentative track and confirmed track filters out false-positive tracks. One beneficial property (refer to the Remark 30 in Reference [63]) of using the PHD-based tracker is a self-gating property; prior tracks are updated by closer measurements rather than farther ones. Also, each track consists of a tree structure as multiple targets can be survived from a single component, which resembles MHT [53]. This inherently yields a deferred decision-making to infer a correct history of tree afterward.

2.5.7 Planning

The planning block takes a set of tracks generated from a set of targets that have survived over time as input from the track maintenance block and produces a control input to the robot. All the building blocks of the search and tracking algorithm (Figure 2.4) described so far focus on estimating the state of the targets. In this section, we focus on the complementary problem of actively controlling the state of the sensor so as to improve the search and tracking process. A number of approaches have been proposed for active target tracking [77], target search [8], as well as joint search and tracking [30]. In this chapter, we evaluate two simple strategies that are particularly suited to the underlying GM-PHD framework. Investigating better strategies with stronger performance guarantees is part of our ongoing work.

In GM-PHD, the mean of the Gaussian is a local maxima of the PHD (*i.e.*, most likely location to find targets in the local neighborhood), whereas the variance encodes the spatial uncertainty of the location of the targets. We evaluate two control strategies. (i) `nearest-Gaussian`: drive to the nearest mean of all Gaussians in the mixture; and (ii) `largest-Gaussian`: drive to the mean of the Gaussian with the largest covariance in the

mixture.

Intuitively, the `nearest-Gaussian` strategy will track one or more targets for as long as possible, giving good tracking performance but poor search performance. On the other hand, the `largest-Gaussian` strategy will equitably cover the search region giving good search performance but possibly poor tracking performance. We evaluate these two strategies through simulations. There can be a third strategy that switches between these two behaviors while trading off search and tracking objectives, which we leave as a future work.

2.6 Simulations and Experiments

In this section, we present the simulation results that show the performance of the proposed algorithm. We compare different probabilities of false-positive detections, with and without considering repulsion effect in the no-detection event, different initial estimate cases, and the proposed heuristic planning strategies. We then show the experimental results that have been performed in an outdoor environment using a single Unmanned Aerial Vehicle (UAV).

2.6.1 Simulation Results

We carried out simulations of the proposed algorithm using MATLAB. Figure 2.6 shows the simulation scenario, where there is a single robot with limited FOV and ten stationary targets in a given environment. The details of the simulation are given in the caption of Figure 2.6.

We firstly evaluate the case of different probabilities of false-positive detections (*i.e.*, 0%, 10% and 20%) to verify the robustness of the proposed algorithm. We generate one false positive detection, located uniformly at random within the FOV, with the given probability. The robot follows the lawn-mower path (Figure 2.7). Initial estimate has ten components with the average mean offset of 15 from the true position and the average variance given by a diagonal matrix with diagonal elements 50. Tables 2.1 and 2.2 present the results of the simulation. As the clutter rate increases, the estimated number of both components and confirmed tracks increases (Table 2.1). The summation of weights gives larger estimated number of confirmed tracks than the number of tracks, with a larger STD. We conjecture that the summation of weights tends to overestimate the number of targets, and that the number of the tracks might be a better estimate of the number of targets in the GM-PHD framework. The overestimation of weights is due to our mechanism of generating new components (*i.e.*, Section 2.5.4), which assigns a weight of one to newly generated components. This can be observed when the probability of false-positive detections is 0% (no false positives). Average

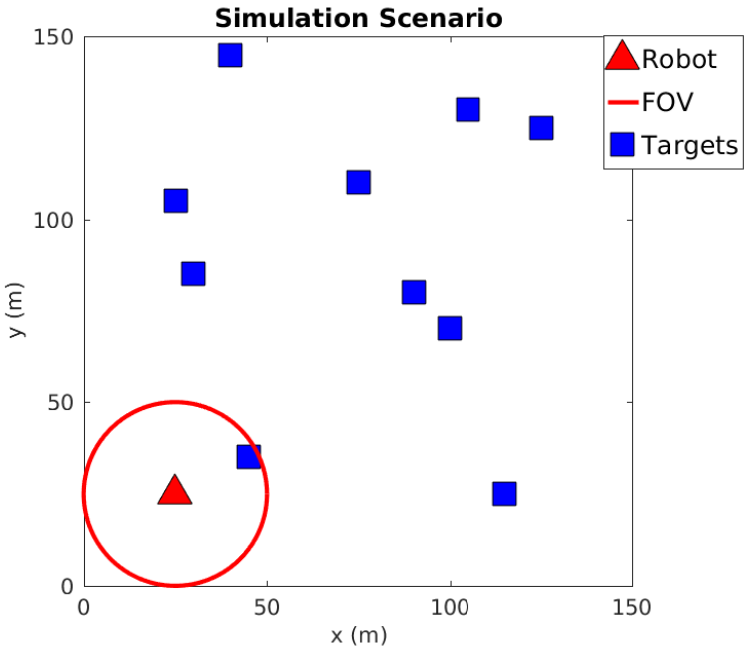
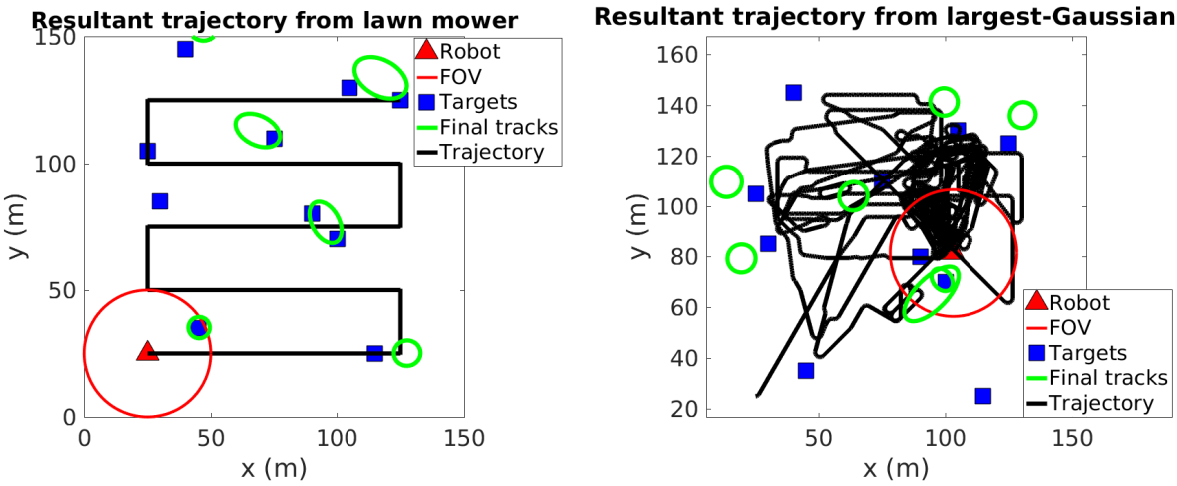


Figure 2.6: Simulation scenario. The FOV of the robot is given by a circular disk with a radius of 25 m. The robot moves at a speed of 0.5 m/s. In the case of lawn mowing, the robot swipes the whole environment (*i.e.*, 150 $m \times 150$ m) and returns to the original position. The robot repeats this three times in $7,278$ time steps. The values we set for parameters are as follows: $p_S = 1$; $p(\mathcal{F}) = 0.98$; lower and upper thresholds for $p(\mathcal{D})$ are 0.4 and 0.6; the weight threshold to be extracted as targets is 0.5; $l_{threshold} = 3$; the pruning weight threshold is 0.001; and the merging threshold (*i.e.*, Mahalanobis distance) is 10.



(a) (b)

Figure 2.7: Resultant trajectories at time step 7,287 for (a) lawn mower and (b) `largest-Gaussian` strategies when the clutter rate of 10% and the exact estimate are applied.

Mahalanobis distance of all true targets to the closest confirmed track does not show any dependence on the clutter rate (Table 2.2).

Next, we compare three estimates (*i.e.*, under-, exact and overestimate) that are given to the robot initially. The clutter rate is set to 10%. In the underestimate case, the initial belief of the robot is five targets where there are ten true targets. On the other hand, the initial belief for overestimate has fifteen targets. The exact estimate implies that the initial belief matches with the true number of targets, which is ten targets in this case. The results in all three cases are similar, as shown in Tables 2.3 and 2.4, except for the underestimate case being less consistent than other estimates. We conclude that as far as the lawn mower is concerned, which covers the entire environment, an overestimate of initial belief does not degrade the performance.

In addition, we verify the effect of Equation (2.11) to update components (lines 5-7 of Algorithm 1). For this simulation, we chose the exact estimate case, lawn-mower trajectory, and 10% clutter rate. Figures 2.8 (a)-(d) show that ignoring the repulsion effect in the update step generates false-negative targets as well as larger errors in the Mahalanobis distance. We can also see the difference between components and confirmed tracks; components tend to overestimate the number of targets due to their unavailability of preserving a history. Furthermore, we also computed the Optimal Subpattern Assignment (OSPA) metric [78], shown in Figure 2.8 (e). The OSPA metric is commonly used in the literature as it captures both the state estimation error and the cardinality error in a consistent manner. It has two parameters, *i.e.*, p for outlier sensitivity and c for cardinality penalty. In the simulation, we set p and c to 2 and 100, respectively, that are the same values used in Reference [78]. Note

Clutter rate		0%	10%	20%
Number of components/ tracks	Mean from components	9.2995	17.8844	22.5303
	STD from components	1.4761	3.3565	4.9181
	Mean from tracks	6.6800	8.1645	8.8030
	STD from tracks	1.1048	1.1685	2.7396
Sum of weights	Mean from components	16.4074	18.6284	19.4675
	STD from components	4.9371	3.9135	5.2100
	Mean from tracks	14.9232	15.6775	15.4238
	STD from tracks	4.9926	3.5912	4.6435

Table 2.1: Estimated number of components/tracks from the number of components/tracks and by the summation of weights for different clutter rates. All values are computed by averaging the values of elements.

Clutter rate	0%	10%	20%
Mean to closest	3.0311	2.3603	2.5529
STD to closest	2.6494	1.9604	2.6242
Mean to second closest	8.2450	7.0569	7.0451

Table 2.2: Average Mahalanobis distance of confirmed tracks compared to true targets for different clutter rates. All values are computed by averaging the values of ten true targets. STD stands for the standard deviation.

that the lower the OSPA, the better performance. Figure 2.8 (e) presents the importance of Equation (2.11).

Figure 2.9 presents the result of the lawn mower when the targets are dynamic within the given environment. Targets are designed to change their moving directions randomly at every 400 time steps. The targets move at a speed of 0.05 m/s, which is one tenth of the robot speed. The GP regression presented in Section 2.5.2 is applied to predict the state of each target. It is, however, intuitive that the robot is able to accurately handle targets that are within the FOV or near the FOV rather than the ones that are located far away from the robot. As compared with Figures 2.8 (a) and (c), it can be seen that there is no pronounced difference in the estimated number of targets and a slightly increased Mahalanobis distance error for moving targets. Therefore, the performance for the moving-target case is shown qualitatively competitive with the stationary-target case using the proposed metrics unless targets are allowed to move outside the boundary of environment.

Lastly, we compare the lawn mower with the `largest-Gaussian` strategy. We have proposed two heuristic planning approaches in Section 2.5.7; the `nearest-Gaussian` strategy, however, is not compared in simulation due to its desire to stick to the closest target. Instead, we show how the planning strategies can affect all targets of interest. Even though planning

Estimate		Under-	Exact	Over-
Number of components/ tracks	Mean from components	18.4820	17.8844	17.8883
	STD from components	5.2331	3.3565	3.2361
	Mean from tracks	9.5336	8.1645	8.5219
	STD from tracks	2.3428	1.1685	1.7715
Sum of weights	Mean from components	19.4229	18.6284	18.3695
	STD from components	5.7117	3.9135	3.5348
	Mean from tracks	16.3315	15.6775	15.4196
	STD from tracks	5.1293	3.5912	3.8297

Table 2.3: Estimated number of components/tracks from the number of components/tracks and by the summation of weights for lawn mower (clutter rate is 10%).

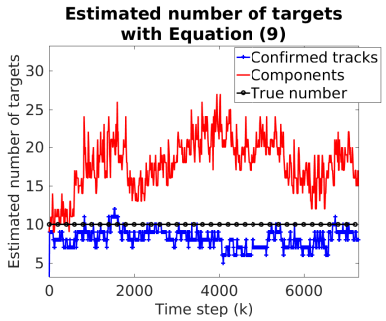
Estimate	Under-	Exact	Over-
Mean to closest	3.0930	2.3603	3.3152
STD to closest	2.5537	1.9604	2.2353
Mean to second closest	7.3006	7.0569	7.5527

Table 2.4: Average Mahalanobis distance of confirmed tracks compared to true targets for lawn mower (clutter rate is 10%).

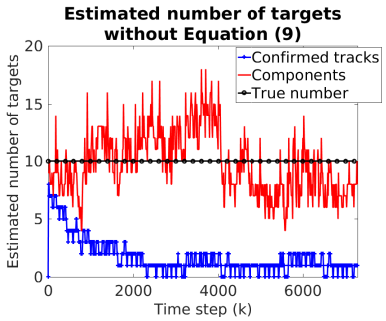
strategies are heuristics, it may be worth analyzing the consequences of these strategies because the lawn mower does not allow the robot to adaptively change its trajectory. We set the clutter rate to 10%. Figure 2.7 shows the resultant trajectories after applying two strategies. Figure 2.10 implies that the advantage of using `largest-Gaussian` over lawn mower is that a smaller worst covariance among all confirmed tracks is achievable. Even though `largest-Gaussian` has a better exploration ability than `nearest-Gaussian`, since the lawn mower explores the whole environment, the lawn mower estimates higher number of targets than `largest-Gaussian`, as shown in Table 2.5. We conjecture that depending on the trade-off between the search and tracking objectives we may be able to adaptively select one of these planning strategies.

2.6.2 Experiments with Real Data

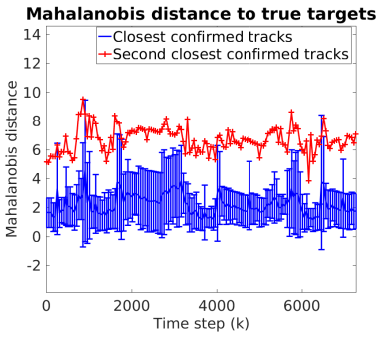
We carried out outdoor experiments using UAV equipped with a single downward-facing camera that detects targets of interest that are located on the ground for the proof of concept. Figure 2.11 shows hardware details of UAV and the snapshot of the field environment. The UAV has Intel NUC (NUC7i7BNH) which runs Ubuntu 16.04 with ROS Kinetic [79]. The onboard software controls the UAV, reads sensor information, and detects targets. Five stationary AprilTag markers [80] were used as stationary targets in a $30m \times 24m$ environment.



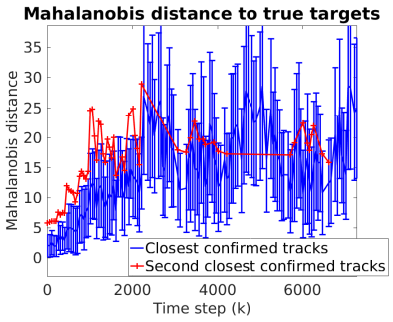
(a) Estimated number of targets with the update of Equation (2.11).



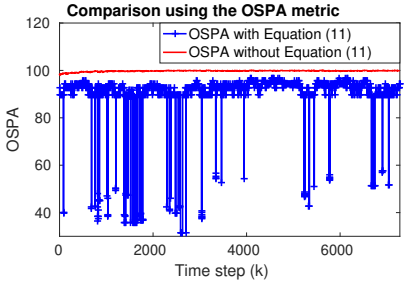
(b) Estimated number of targets without the update of Equation (2.11).



(c) Average Mahalanobis distance of all true targets to the closest confirmed track with the update of Equation (2.11).



(d) Average Mahalanobis distance of all true targets to the closest confirmed track without the update of Equation (2.11).

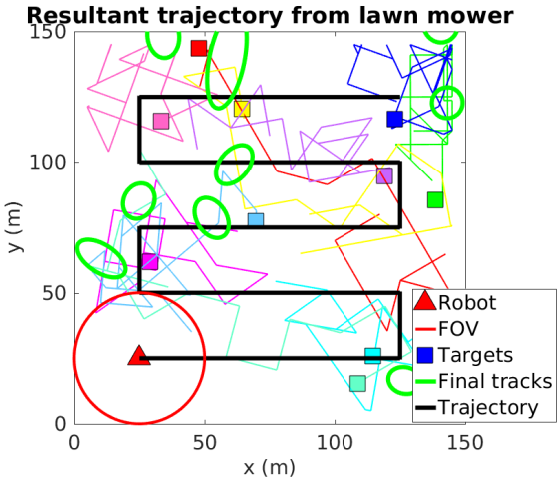


(e) OSPA (when $c = 100$ and $p = 2$) with and without Equation (2.11).

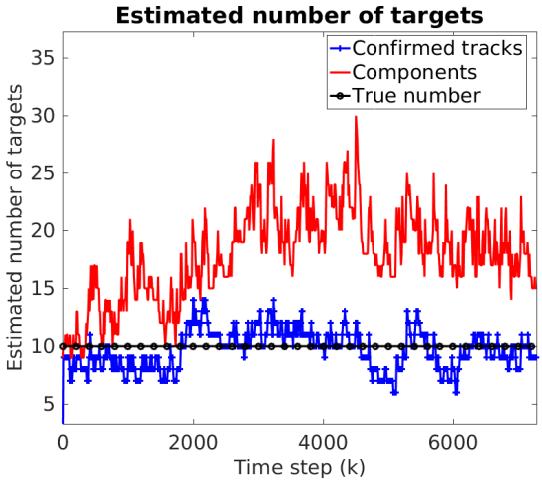
Figure 2.8: Comparison between with and without the update of Equation (2.11). The true number of targets is ten.

The UAV flying at an altitude of $8m$ that yields a circular FOV of a radius of approximately $7.5m$. We chose the lawn-mower strategy with a width of $8m$ to search for and track targets.

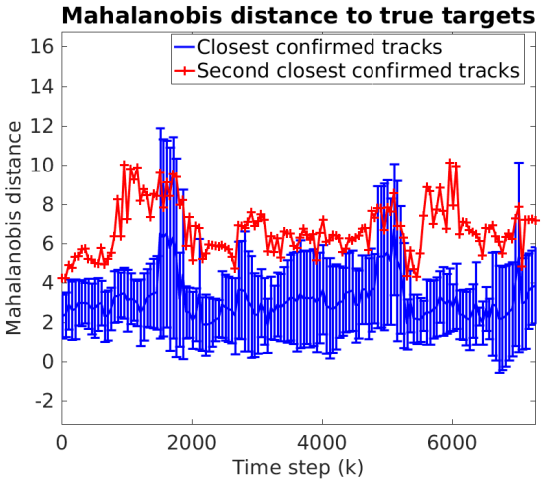
Figure 2.12 plots the measurements observed by the camera and the trajectory of the UAV after going to the end of the environment and coming back to the origin. The measurements were noisy because we did not deliberately calibrate the camera. We also discarded IDs obtained from AprilTag measurements to make them identical. We did this to evaluate the robustness of the proposed algorithm. The initial estimate has zero components. Figure 2.12 shows the final confirmed tracks. In Figure 2.13 the UAV started detecting a component at time step 114. We observe that the UAV detected 5-6 targets most of the time with reasonable Mahalanobis distance. This demonstrates the robust performance of the algorithm with noisy real-world data.



(a) Resultant trajectory at time step 7,287 for lawn mower. The targets are denoted as the square markers as well as the associated trajectories.

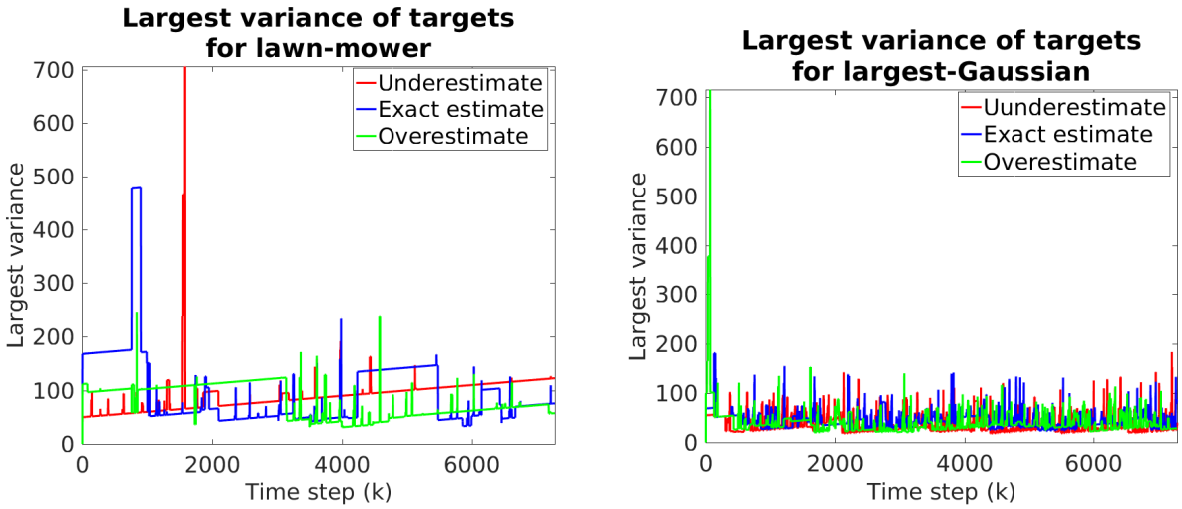


(b) Estimated number of targets.



(c) Average Mahalanobis distance of all true targets to the closest confirmed track.

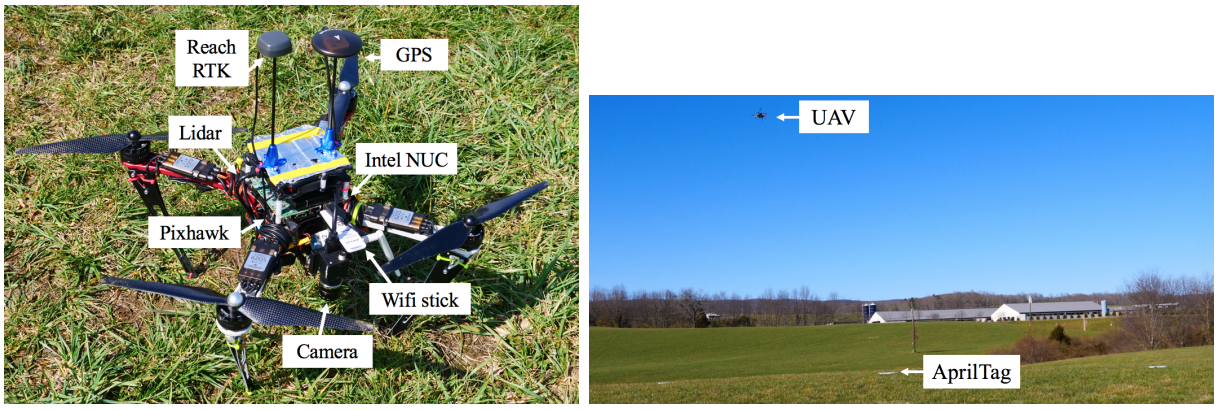
Figure 2.9: Result of the lawn mower when targets are dynamic and change their directions randomly at every 400 time steps. The clutter rate is 10% and it is the exact estimate.



(a) Largest variance of targets of lawn mower for under-, exact-, and overestimate cases.

(b) Largest variance of targets of `largest-Gaussian` for under-, exact-, and overestimate cases.

Figure 2.10: Largest variance of targets for lawn mower and `largest-Gaussian` strategies.



(a) UAV (DJI F450) platform.

(b) Test environment. Five AprilTag markers are placed on the ground.

Figure 2.11: Field experiment carried out in Kentland Farm, Virginia, USA (please refer to the attached video for both the simulation and experiment).

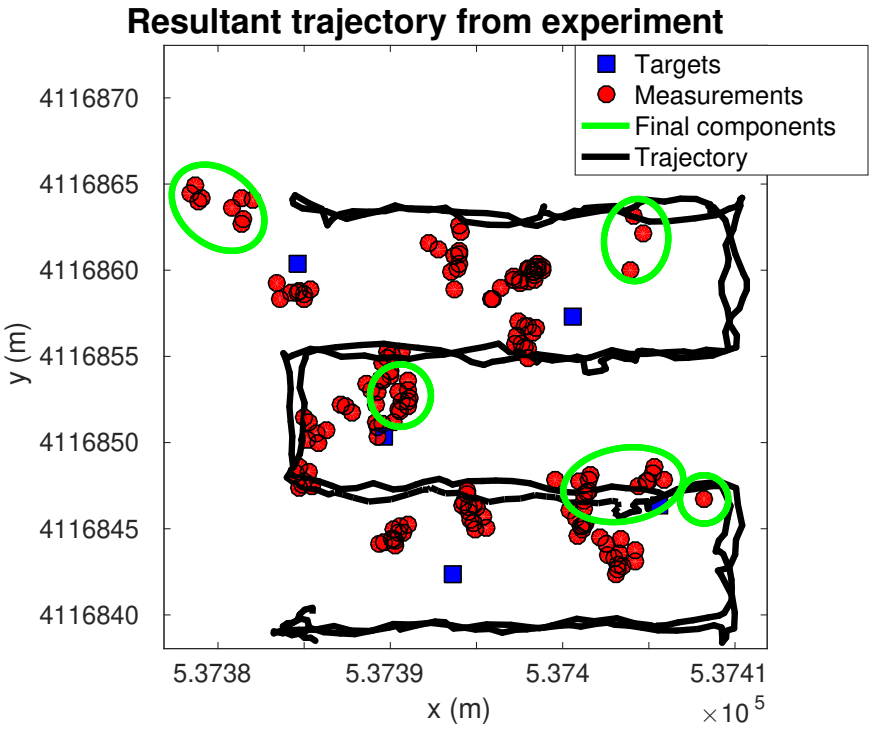
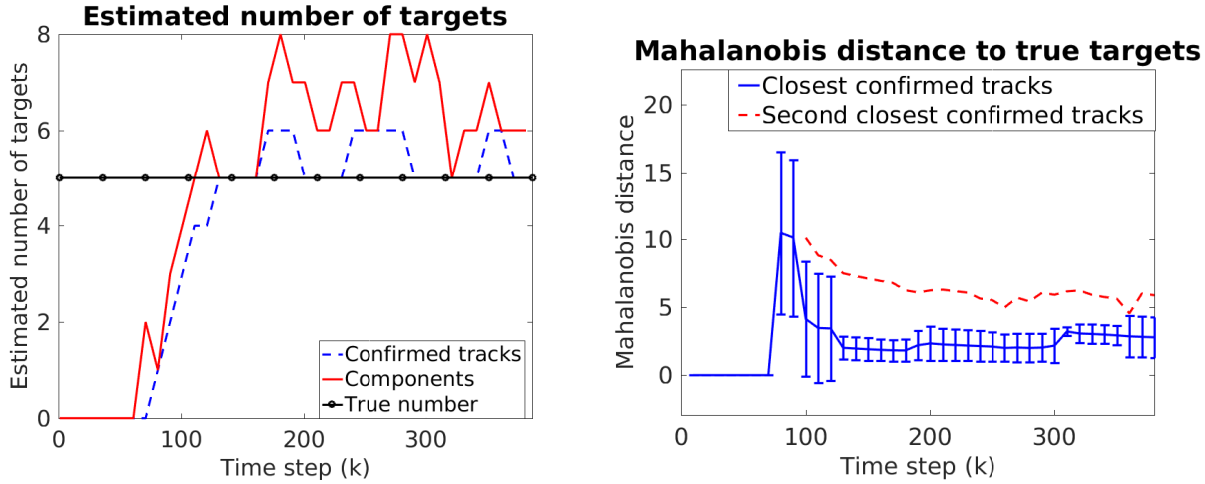


Figure 2.12: Trajectory of the UAV and positions of observed measurements. The total flight time was 6 minutes and 43 seconds.



(a) (b)

Figure 2.13: Results of the outdoor experiment: the plot (a) presents the estimated number of targets by counting the elements in a set of confirmed tracks and that of components; and the plot (b) shows the average Mahalanobis distance among all true targets compared to the closest and the second closest tracks.

Estimate		Under-	Exact	Over-
Number of tracks	Lawn mower	10	7	8
	`Largest-Gaussian`	3	7	5
Sum of weights	Lawn mower	13.3769	18.8052	19.6354
	`Largest-Gaussian`	11.3125	14.1745	13.0396

Table 2.5: Estimated number of tracks from the number of tracks and by the summation of weights for lawn-mower and `largest-Gaussian` strategies (clutter rate is 10%).

2.7 Discussion

Our main contribution in this chapter is to extend the GM-PHD filter, initially proposed for the tracking problem [33], to allow for search and tracking with a limited FOV robot. Our second contribution was to incorporate non-linear target prediction using GP regression. The current form is restricted to a 2D environment and a circular FOV but this can be extended to higher dimensional environments and any shape of sensing models by appropriately modifying Equation (2.10).

We employed the PHD filter and extended the framework to take into account the finite FOV of a mobile sensor. The GM-PHD filter uses a simpler representation (Gaussian mixtures) than the original PHD. Recently, a number of filters have been proposed that estimate the number of targets as well as the state/track of individual targets, unlike GM-PHD, such as the Multi-Target Multi-Bernoulli (MeMBer) filter [81] and the δ-Generalized Labeled Multi-Bernoulli (δ-GLMB) filter [82, 83]. The MeMBer filter is more advantageous for the SMC implementation than PHD because it allows a more reliable and efficient way of extracting target states [81]. However, even the cardinality-balanced MeMBer filter [81] has a similar performance in terms of mean and variance estimate to GM-PHD under the high signal-to-noise ratio condition. As opposed to PHD and MeMBer in which track maintenance is not inherent, δ-GLMB directly estimates the state of tracks by using the labeled RFS. δ-GLMB is also robust to missed detections that significantly reduce the weight of corresponding targets in the PHD framework. Due to high complexity of δ-GLMB, many approximation algorithms have recently been developed [84, 85]. These filters can be a more promising estimator/tracker to the proposed problem. Extending the current approach for limited FOV sensor to these methods is an important avenue of future work.

The immediate future work is to incorporate better planning algorithms. In our previous work on particle PHD filters [26], we defined information-theoretic measures to control the position of the robots. Such approaches can directly be applied to the GM-PHD case. Another possible direction is to incorporate the *ridge-walking* algorithm [73] which plans a tour of level sets in the spatial distribution of the targets. However, this algorithm assumes

that the targets are stationary and would thus need to be generalized to handle mobile target distributions. A decentralized version of Monte Carlo search tree proposed by Best *et al.* [86] can also extend the proposed work to consider multi-robot online planning if reasonable metrics for the objective functions can be found.

Chapter 3

Coordinated Multi-Target Tracking under Limited Bandwidth

3.1 Introduction

We study the problem of assigning robots with limited FOV sensors to track multiple moving targets. Multi-robot multi-target tracking is a well-studied topic in robotics [87, 2, 88, 89, 90]. We focus on scenarios where the number of robots is large and solving the problem locally rather than centrally is desirable. The robots may have limited communication range and limited bandwidth. As such, we seek assignment algorithms that rely on local information and only require limited amount of communication with neighboring robots.

Constraints on communication impose challenges for robot coordination as global information may not always be available to all the robots within the network. As a result, it may not be always possible to design algorithms that operate on local information while still ensuring global optimality. Recently, Gharesifard and Smith [91] studied how limited information due to the communication graph topology affects the global performance. Their analysis applies for the case when the robots are allowed only round of communication with their neighbors. If the robots are allowed multiple rounds of communication, they can propagate the information across the network. Given sufficient rounds of communication, all robots will have access to global information, and therefore can essentially solve the centralized version of the problem. In this chapter, we investigate the relationship between the number of communication rounds allowed for the robots and the performance guarantees. We focus on the problem of distributed multi-robot, multi-target assignment for our investigation (Figure 3.1).

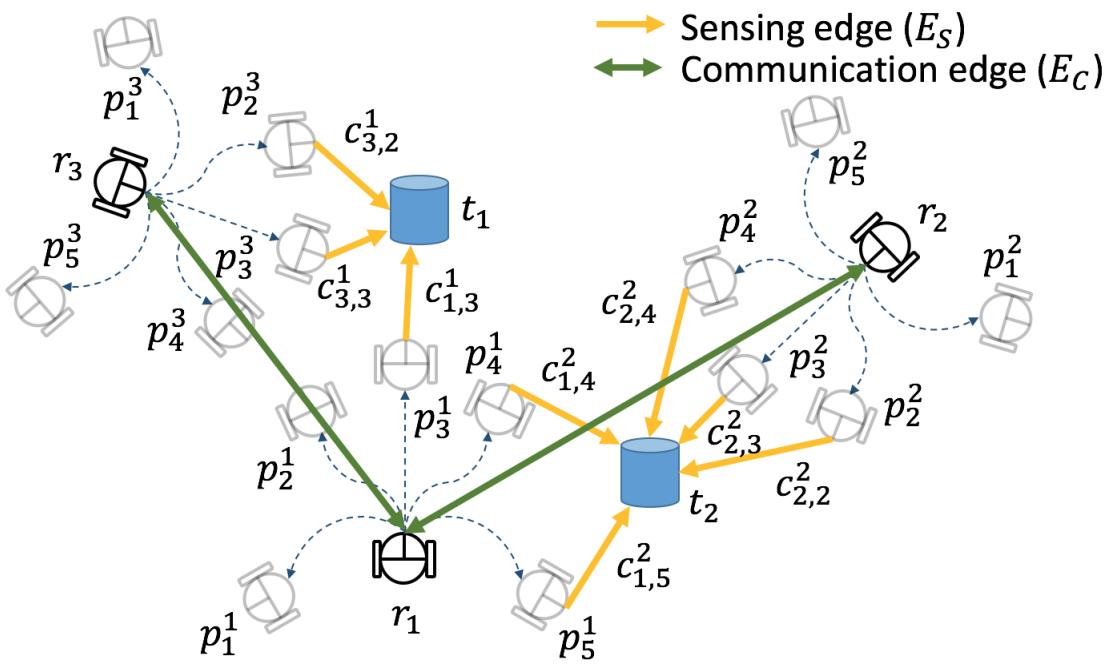


Figure 3.1: Description of multi-robot task allocation for multi-target tracking. In this example, each robot has five motion primitives to choose from at each time step.

We assume that each robot has a number of motion primitives to choose from. A motion primitive is a local trajectory obtained by applying a sequence of actions [92]. A motion primitive can track a target if the target is in the FOV of the robot. The set of targets tracked by different motion primitives may be different. The assignment of targets to robots is therefore coupled with the selection of motion primitives for each robot. Our goal is to assign motion primitives to the robots so as to track the most number of targets or maximize the quality of tracking. We term this as the distributed Simultaneous Action and Target Assignment (SATA) problem.

This problem can be viewed as the dual of the set cover problem, known as the maximum (weighted) cover [93]. Every motion primitive covers some subset of the targets. Therefore, we would like to pick motion primitives that maximize the number (or weight) of covered targets. However, we have the additional constraint that only one motion primitive per robot can be chosen at each step. This implies that the relationship between a robot and the corresponding motion primitives turns out to be a packing problem [93] where only one motion primitive can be "packed" per robot. The combination of the two aforementioned problems is called a Mixed Packing and Covering Problem (MPCP) [94].

We study two versions of the problem. The first version can be formulated as a (sub)modular maximization problem subject to a partition matroid constraint [95]. A sequential greedy algorithm, where the robots take turns to greedily choose motion primitives, is known to yield a 2–approximation for this problem [27]. We evaluate the empirical performance of this

algorithm by comparing it with a centralized (globally optimal solution). The drawback of the sequential greedy algorithm is that it requires at least as many communication rounds as the number of robots. This may be too slow in practice. Consequently, we study a second version of the problem for which we present a *local* algorithm whose performance degrades gracefully (and provably) as a function of the number of communication rounds.

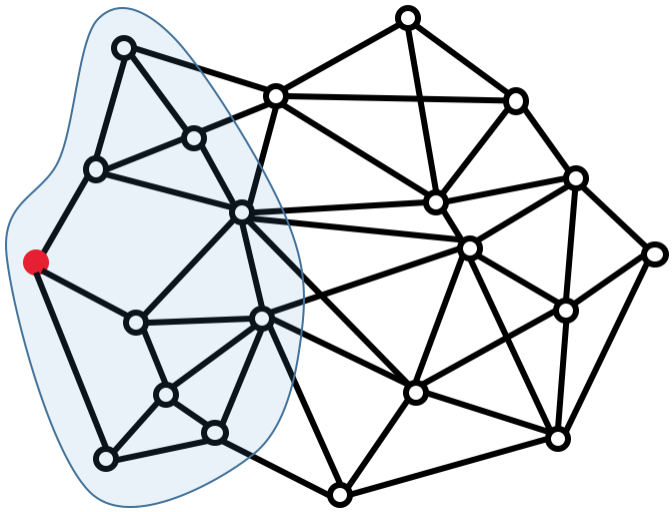


Figure 3.2: Communication graph. The blue shaded region indicates a radius–2 neighborhood of the red solid node. The red solid node may be unaware of the entire communication graph topology. A local algorithm that works for the red solid node only requires local information of nodes in the blue shaded region. The same local algorithm runs on all the nodes and ensures bounded approximation guarantees on the global optimality.

A local algorithm [93] is a constant-time distributed algorithm that is independent of the size of a network. This enables a robot only to depend on local inputs in a fixed-radius neighborhood of robots (Figure 3.2). The robot does not need to know information beyond its local neighborhood, thereby achieving better scalability.

Floréen *et al.* [1] proposed a local algorithm to solve MPCP using max-min/min-max Linear Programming (LP) in a distributed manner. We show how to leverage this algorithm to solve SATA. This algorithm has a bounded communication complexity unlike typical distributed algorithms. Specifically, the algorithm yields a $\mathcal{O}\left((1+\epsilon)(1+1/h)\right)$ approximation to the globally optimal solution in $\mathcal{O}(h \log 1/\epsilon)$ synchronous communication rounds where h and ϵ are input parameters.[1] We verify the theoretical results through empirical evaluation.

The contributions of this chapter are as follows:

1. We present two versions of the SATA problem.

[1] An algorithm is called a $\mathcal{O}(x)$ approximation to a maximization problem if it guarantees a solution whose value is at least $\frac{c}{x}$ of the optimal value, where c is some constant.

2. We show how to use the greedy algorithm and adapt the local algorithm for solving the two versions of the SATA problem.
3. We perform empirical comparisons of the proposed algorithm with baseline centralized solutions.
4. We demonstrate the applicability of the proposed algorithm through Gazebo simulations.

The rest of the chapter is organized as follows. We begin by introducing the related work in Section 3.2. We describe the problem setup in Section 3.3. Our proposed distributed algorithms are presented in Section 3.4. We present results from representative simulations in Section 3.5 before concluding with a discussion of future work in Section 3.6.

3.2 Related Work

A number of algorithms have been designed to improve multi-robot coordination under limited bandwidth [96, 97, 98, 99, 100] and under communication range constraints [101, 102, 103]. This includes algorithms that enforce connectivity constraints [104, 105], explicitly trigger when to communicate [106, 107, 108] and operate when connectivity is intermittent [109, 110]. In this section, we focus on work that is most closely related to the SATA problem and local algorithms.

3.2.1 Multi-Robot Target Tracking

There have been many studies on cooperative target tracking in both control and robotics communities. We highlight some of the recent related work in this section. For a more comprehensive overview of multi-robot multi-target tracking, see the recent surveys [111, 112].

Charrow *et al.* [113] proposed approximate representations of the belief to design a control policy for multiple robots to track one mobile target. The proposed scheme, however, requires a centralized approach. Yu *et al.* [114] worked on an auction-based decentralized algorithm for cooperative path planning to track a moving target. Ahmad *et al.* [115] presented a unified method of localizing robots and tracking a target that is scalable with respect to the number of robots. Zhou and Roumeliotis [116] developed an algorithm that finds an optimal trajectory of multiple robots for the active target tracking problem. Capitan *et al.* [117] proposed a decentralized cooperative multi-robot algorithm using auctioned partially observable Markov decision processes. The performance of decentralized data fusion under

limited communication was successfully shown but theoretical bounds on communication rounds were not covered. Moreover, theoretical properties presented in the above references considered single target tracking, which may not necessarily hold in the case of tracking multiple targets in a distributed fashion.

Pimenta *et al.* [118] adopted Voronoi partitioning to develop a distributed multi-target tracking algorithm. However, their objective was to cover an environment coupled with multi-target tracking. Banfi *et al.* [119] addressed the *fairness* issue for cooperative multi-robot multi-target tracking, which is achieving balanced coverage among different targets. One of the problems that we define in Section 3.3 (*i.e.*, Problem 1) has a similar motivation. However, unlike the algorithm in Banfi *et al.* [119], we are able to give a global performance guarantee. Xu *et al.* [120] presented a decentralized algorithm that jointly solves the problem of assigning robots to targets and positioning robots using mixed-integer nonlinear programming. While they proved the complexity in terms of computational time and communication (*i.e.*, the amount of data needed to be communicated), the solution quality was only evaluated empirically. Instead, we bound the solution quality as a function of the communication rounds. Furthermore, our formulation takes as input a set of discrete actions (*i.e.*, motion primitives) that the robot must choose from, unlike the previous work.

We study a problem similar to the one termed as Cooperative Multi-robot Observation of Multiple Moving Targets (CMOMMT) proposed by Parker and Emmons [87]. The objective in CMOMMT is to maximize the collective time of observing targets. Parker [2] developed a distributed algorithm for CMOMMT that computes a local force vector to find a direction vector for each robot. We empirically compare this algorithm with our proposed one and report the results in Section 3.5. Kolling and Carpin [89] studied the behavioral CMOMMT that added a new mode (*i.e.*, help) to the conventional track and search modes of CMOMMT. The help mode asks other robots to track a target if the target escapes from the FOV of some robot. Although our work does not allow mode changes, previous works regarding CMOMMT did not provide theoretical optimality guarantees and did not explicitly consider scenarios where the communication bandwidth is limited. Refer to Section IV(C) of Reference [111] for a more detailed summary of CMOMMT.

In our prior work [27], we addressed the problem of selecting trajectories for robots that can track the maximum number of targets using a team of robots. However, no bound on the number of communication rounds was presented, possibly resulting in all-to-all communication in the worst case. Instead, in this work, we introduce a new version of the problem and also explicitly bound the amount of communication required for target assignment.

3.2.2 Multi-Robot Task Assignment

Multi-robot task assignment can be formulated as a discrete combinatorial optimization problem. The work by Gerkey and Matarić [121] and the more recent work by Korsah *et al.* [122] contain detailed survey of this problem. There exists distributed algorithms with

provable guarantees for different versions of this problem [123, 124, 125]. There also exists various multi-robot deployment strategies for task assignment under communication constraints. These constraints include limited available information [126], limited communication flows [127], and connectivity requirement [128]. See the survey papers [129, 130] on these results. Ny *et al.* [127] studied a formulation with a similar communication constraint as ours. However, their formulation assumed that the robots know which targets to track. In this chapter, we tackle the challenge of simultaneously assigning robots to targets by choosing motion primitives with limited communication bandwidth which might degrade task performance when there are unreliable communication links and communication delays.

Turpin *et al.* [131] proposed a distributed algorithm that assigns robots to goal locations while generating collision-free trajectories. Morgan *et al.* [132] solved the assignment problem by using distributed auctions and generating collision-free trajectories by using sequential convex programming. Bandyopadhyay *et al.* [133] adopted the Eulerian framework for both swarm formation control and assignment. However, these works may not be suitable for target tracking applications as the targets were assumed to be static. For more survey results about SATA, see the work by Chung *et al.* [134]. Recently, Otte *et al.* [135] investigated the effect of communication quality on auction-based multi-robot task assignment. None of the above works, however, analyzed the effect of communication rounds on the solution quality, as is the focus of our work.

3.2.3 Local Algorithms

A local algorithm [136, 137, 138] is a distributed algorithm that is guaranteed to achieve desired objective in a finite (typically, fixed) amount of time. The typical approach is to find approximate solutions with provable (and global) performance guarantees while ensuring a bound on the communication complexity that is independent of the number of vertices in the graph. Local algorithms have been proposed for a number of graph-theoretic problems. These include, graph matching [139], vertex cover [140, 141], dominating set [142], and set cover [143]. Suomela [93] gives a broad survey of local algorithms. We build on this work and adapt a local algorithm for solving SATA.

3.3 Problem Description

Consider a scenario where multiple robots are tracking multiple mobile targets. Robots can observe targets within their FOV and predict the future states of targets. Based on predicted target states, robots decide where to move (*i.e.*, by selecting a motion primitive) in order to keep track of targets. By discretizing time, the problem becomes one of combinatorial optimizations — choose the next position of robots based on the predicted position of the targets. Thus, we solve the SATA problem at each time step.

We define sets, R and T, to denote the collection of robot and target labels respectively: $R = \{1, ..., i, ..., |R|\}$ for robot labels and $T = \{1, ..., j, ..., |T|\}$ for target labels. Let r and t denote the set of robot states and *predicted* target states, respectively. In this chapter, states are given by the positions of the robots and the targets in 2- or 3-dimensional space. However, the algorithms presented in this chapter can be used for more complex states (*e.g.*, 6 degree-of-freedom pose). Here, $r(k) = \{\mathbf{r}_1(k), ..., \mathbf{r}_i(k), ..., \mathbf{r}_{|R|}(k)\}$ denotes the state of the robots at time k. $t(k) = \{\mathbf{t}_1(k), ..., \mathbf{t}_j(k), ..., \mathbf{t}_{|T|}(k)\}$ denotes the state of the targets at the next time step (*i.e.*, at time $k+1$) predicted at time k. We assume that the targets can be uniquely detected and multiple robots know if they are observing the same target. Therefore, no data association is required. Each robot independently obtains the predicted states, $t(k)$, by fusing its own noisy sensor measurements using, for example, a Kalman filter.

We define the labels of available motion primitives for the i-th robot as $P^i = \{1, ..., m, ..., |P^i|\}$. These labels correspond to a set of motion primitive states of the i-th robot at time k given by: $p^i(k) = \{\mathbf{p}^i_1(k), ..., \mathbf{p}^i_m(k), ..., \mathbf{p}^i_{|P^i|}(k)\}$. Note that the term *motion primitives* in this chapter represents the future state of a robot at the next time step (*i.e.*, at time $k+1$) computed at time k. We compute a set of the motion primitives a priori by discretizing the continuous control input space. This can be done by various methods such as uniform random sampling or biased sampling based on predicted target states. However, once a set of the motion primitives is obtained, the rest of the proposed algorithms (in Section 3.4) remain the same.

We define $\mathcal{RS}(\mathbf{p}^i_m(k))$ to be the set of targets that can be observed by the m-th motion primitive of i-th robot at time k. Specifically, the j-th target is said to be observable by the m-th motion primitive of a robot i, iff $\mathbf{t}_j(k) \in \mathcal{RS}(\mathbf{p}^i_m(k))$. It should be noted that only targets that were observed by robot i at time $k-1$ are candidates to be considered for time k because unobserved targets at time $k-1$ cannot be predicted by the robot i. Note also that since $\mathcal{RS}$ is a set function, we can model complex FOV and sensing range constraints that are not necessarily restricted to 2D.

We make the following assumptions.[2]

Assumption 1. ***(Communication Range).*** *If two robots have a motion primitive that can observe the same target, then these robots can communicate with each other. This implies if there exists a target j such that $\boldsymbol{t}_j(k) \in \mathcal{RS}(\boldsymbol{p}^i_m(k))$ and $\boldsymbol{t}_j(k) \in \mathcal{RS}(\boldsymbol{p}^l_m(k))$, then i-th and l-th robots can communicate with each other.*

Assumption 2. ***(Synchronous Communication).*** *All the robots have synchronous clocks leading to synchronous rounds of communication.*

From Assumption 1, neighboring robots can share their local information with each other when they observe the same targets. For example, robots can use techniques such as covariance intersection [144] to merge their individual predictions of the target's state into a joint

[2]After these assumptions, we omit the time index (*i.e.*, k) for notational convenience.

prediction T. This can be achieved in one round of communication when each robot simply broadcasts its own estimate of all the targets within its FOV. Note that a robot does not need to know the prediction for all the targets but only the ones that are within the FOV of one of its motion primitives. In this sense, a communication graph $\mathcal{G}_C = (R, E_C)$ can be created from a sensing graph $\mathcal{G}_S = (P \cup T, E_S)$ at each time, where E_C and E_S denote edges among robots and edges between targets and motion primitives, respectively.

As shown in Figure 3.1, each robot is able to compute feasible motion primitives of its own and detect multiple unique targets within the FOV. Then, the objective of the proposed problem is to choose one of the motion primitives for each robot, yielding either the best quality of tracking or the maximum number of targets tracked by the robots, depending on the application. One possible quality of tracking can be measured by the summation of all distances between selected primitives and the observed targets.

Let x_m^i be the binary variable which represents the i-th robot selecting the m-th motion primitive. That is, $x_m^i = 1$ if a motion primitive m is selected by a robot i and 0 otherwise.[3] Since each robot can choose only one motion primitive, we have:

$$\sum_{m \in P^i} x_m^i \leq 1 \quad \forall i \in R. \tag{3.1}$$

Our objective is to find x_m^i. We propose two following problems.

Problem 1 (BOTTLENECK)**.** *The objective is to select primitives such that we maximize the minimum tracking quality:*

$$\underset{x_m^i}{\text{argmax}} \quad \min_{j \in T} \left(\sum_{i \in R} \sum_{m \in P^i} c_{i,m}^j x_m^i \right), \tag{3.2}$$

subject to the constraints in Equation (3.1). Here, $c_{i,m}^j$ denotes weights on sensing edges E_S between m-th motion primitive of i-th robot and j-th target.

Here, $c_{i,m}^j$ can represent the tracking quality given by, for example, the inverse of the distance between m-th motion primitive of i-th robot and j-th target. Alternatively, $c_{i,m}$ can be binary (1 when the m-th motion primitive of robot i sees target j and 0 otherwise) making the objective function equal to maximizing the minimum number of targets tracked.

We term this as the BOTTLENECK version of SATA. In the BOTTLENECK version, multiple robots may be assigned to the same target. We also define a WINNER- TAKESALL variant of SATA where only one robot is assigned to a target.

We define additional binary decision variable, y_i^j. y_i^j represents the i-th robot assigned to track the j-th target. We have, $y_i^j = 1$ if i-th robot is assigned to j-th target and 0 otherwise.

[3] If all $x_m^i = 0$ for a robot i, then it can choose any motion primitives since the objective value will remain the same.

Since we restrict only one robot to be assigned to the target (unlike BOTTLENECK), we have:

$$\sum_{i \in R} y_i^j \leq 1 \quad \forall j \in T. \tag{3.3}$$

Problem 2 (WINNERTAKESALL)**.** *The objective is to maximize the total quality of tracking given by,*

$$\underset{x_m^i, y_i^j}{\operatorname{argmax}} \sum_{j \in T} \left(\sum_{i \in R} y_i^j \left(\sum_{m \in P^i} c_{i,m}^j x_m^i \right) \right), \tag{3.4}$$

subject to the constraints in Equations (3.1) and (3.3).

Both versions of the SATA problem are NP-Hard [145]. The WINNERTAKESALL version can be optimally solved using a Quadratic Mixed Integer Linear Programming (QMILP) in the centralized setting.[4] Our main contributions are to show how to solve both problems in a distributed manner: an LP-relaxation of the BOTTLENECK variant using a local algorithm; and the WINNERTAKESALL variant using a greedy algorithm. The following theorems summarize the main contributions of our work.

Theorem 1. *Let $\triangle_R \geq 2$ be the maximum number of motion primitives per robot and $\triangle_T \geq 2$ be the maximum number of motion primitives that can see a target. There exists a local algorithm that finds an $\triangle_R(1+\epsilon)(1+1/h)(1-1/\triangle_T)$ approximation in $\mathcal{O}(h \log 1/\epsilon)$ synchronous communication rounds for the LP-relaxation of the* BOTTLENECK *version of SATA problem, where h and $\epsilon > 0$ are parameters.*

The proof follows directly from the existence of the local algorithm described in the next section. We show how the local algorithm for MPCP can be modified to solve SATA by means of a linear relaxation.

Theorem 2. *There exists a 2–approximation greedy algorithm for the* WINNERTAKESALL *version of the SATA problem for any $\epsilon > 0$ in polynomial time.*

This directly follows from the fact that the problem is a modular maximization problem subject to a partition matroid constraint [95]. The algorithms are described in the next section.

3.4 Distributed Algorithms

We begin by describing the local algorithm that solves the BOTTLENECK version of SATA.

[4]Note that Problem 2 can also be converted into a simpler Mixed Integer Linear Programming (MILP) by linearizing the product of the binary variables in Equation (3.4), which is not covered in this chapter.

3.4.1 Local Algorithm

In this section, we show how to solve the BOTTLENECK version of the SATA problem using a local algorithm. We adapt the local algorithm for solving max-min LPs given by Floréen *et al.* [1] to solve the SATA problem in a distributed manner.

Consider the tripartite, weighted, and undirected graph, $\mathcal{G} = (R \cup P \cup T, E)$ shown in Figure 3.3. Each edge $e \in E$ is either $e = (\mathbf{r}_i, \mathbf{p}_m^i)$ with weight 1 or $e = (\mathbf{t}_j, \mathbf{p}_m^i)$ with weight $c_{i,m}^j$. The maximum degree among robot nodes $\mathbf{r}_i \in r$ is denoted by $\triangle_R$ and among target nodes $\mathbf{t}_j \in t$ is $\triangle_T$. Each motion primitive $\mathbf{p}_m^i \in p^i$ is associated with a variable x_m^i. The upper two layers of $\mathcal{G}$ in Figure 3.3 are related with a packing problem (Equation (3.4)). The lower two layers are related with the covering problem.

Lemma 1. *The* BOTTLENECK *version (Equation (3.2)) can be rewritten as a linear relaxation of ILP:*

$$\begin{aligned} &\textit{maximize} \quad w \\ &\textit{subject to} \quad \sum_{m \in P^i} x_m^i \leq 1 \quad \forall i \in R \\ &\qquad\qquad \sum_{i \in R} \sum_{m \in P^i} c_{i,m}^j x_m^i \geq w \quad \forall j \in T \\ &\qquad\qquad\qquad x_m^i \geq 0 \quad \forall m \in P^i. \end{aligned} \tag{3.5}$$

The proof is given in Appendix A.1.

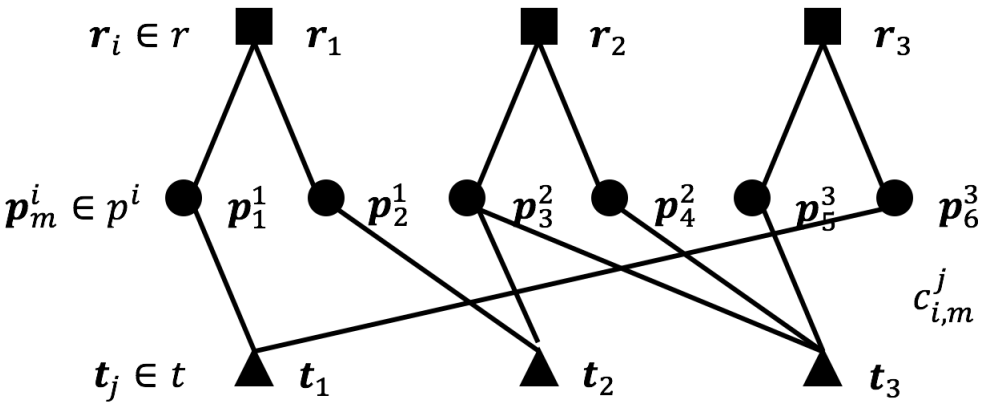


Figure 3.3: One instance of a graph for MPCP when there are three robot nodes, six motion primitive nodes and three target nodes.

Floréen *et al.* [1] presented a local algorithm to solve MPCP in Equation (3.5) in a distributed fashion. They presented both positive and negative results for MPCP. We show how to adopt this algorithm for solving the BOTTLENECK version of SATA.

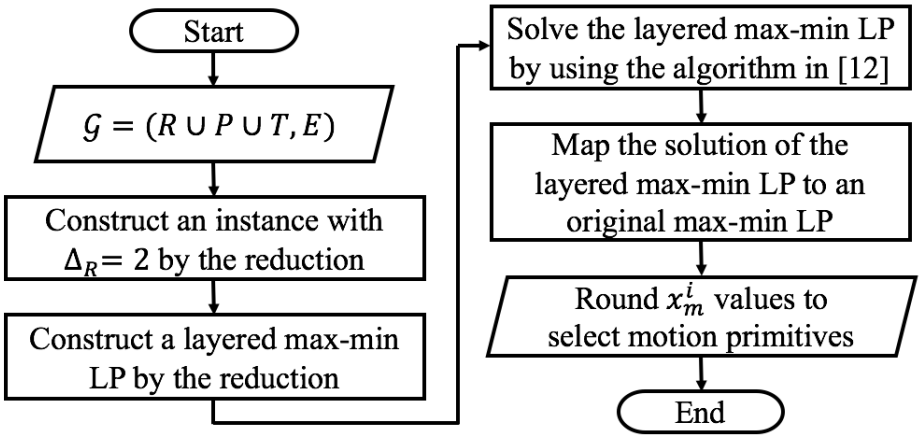


Figure 3.4: Flowchart of the proposed local algorithm.

An overview of our algorithm is given in Figure 3.4. We describe the main steps in the following.

Local Algorithm from Reference [1]

The local algorithm in Reference [1] requires $\triangle_R = 2$. However, they also present a simple local technique to split nodes in the original graph with $\triangle_R > 2$ into multiple nodes making $\triangle_R = 2$. Then, a *layered* max-min LP is constructed with h layers, as shown in Figure 3.5. h is a user-defined parameter that allows to trade-off computational time with optimality. If the number of layers is set to h, then it means that a robot can communicate with another robot that is no more than h communication edges (*i.e.*, hops) away. The layered graph breaks the symmetry that inherently exists in the original graph. This layered mechanism is specifically designed for solving MPCP and is covered in depth in Section 4 of Reference [1]. We omit the details in this chapter due to limited space and redirect the readers to Section 4 of Reference [1] for the construction of the layered graph.

They proposed a recursive algorithm to compute a solution of the layered max-min LP. The solution for the original max-min LP can be obtained by mapping from the solution of the layered one. The obtained solution corresponds to values of x_m^i. They proved that the resulting algorithm gives a constant-factor approximation ratio.

Theorem 3. *There exists local approximation algorithms for max-min and min-max LPs with the approximation ratio* $\triangle_R(1+\epsilon)(1+1/h)(1-1/\triangle_T)$ *for any* $\triangle_R \geq 2$, $\triangle_T \geq 2$, *and* $\epsilon > 0$, *where* h *denotes the number of layers.*

Proof. Please refer to Corollary 4.7 from Reference [1] for a proof. □

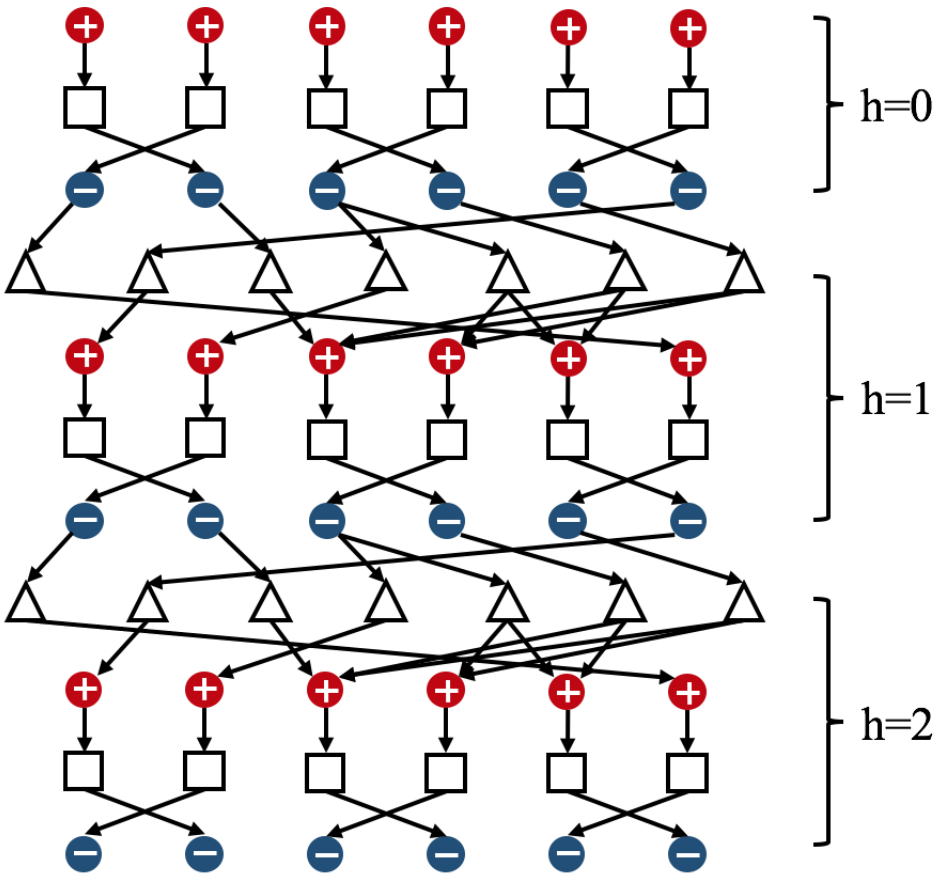


Figure 3.5: Graph of the layered max-min LP with $h = 2$ that is obtained from the original graph of Figure 3.3 after applying the local algorithm. The details for constructing a layered graph are given in Section 4 of Reference [1]. Each motion primitive $\mathbf{p}_m^i \in p^i$ is colored either red or blue to break the symmetry of the original graph. Squares, circles, and triangles represent robot nodes, motion primitive nodes, and target nodes, respectively, corresponding to Figure 3.3.

Note that each node in the layered graph carries out its local computation (details of the local computation for solving SATA are included in Reference [1]). Each node also receives and sends information from and to neighbors at each synchronous communication round. Constructing the layered graph is done in a local fashion without requiring any single robot to know the entire graph.

Realization of Local Algorithm for SATA

To apply the local algorithm of Section 3.4.1 to a distributed SATA problem, each node and edge in a layered graph must be realized at each time step (*i.e.*, generating a graph shown in Figure 3.5 which becomes the input to the local algorithm [1]). In our case, the only computational units are the robots. Nodes that correspond to motion primitives, $\mathbf{p}_m^i \in p^i$, can be realized by the corresponding robot $\mathbf{r}_i \in r$. Moreover, nodes corresponding to the targets must also be realized by robots. A target j is realized by a robot i satisfying $\mathbf{t}_j \in \mathcal{RS}(\mathbf{p}_m^i)$. If there are multiple robots whose motion primitives can sense the target (by Assumption 1), they can arbitrarily decide which amongst them realizes the target nodes in a constant number of communication rounds.

After applying the local algorithm of Section 3.4.1 to robots, each robot obtains x_m^i on corresponding $\mathbf{p}_m^i$ at each time. However, due to the LP relaxation, x_m^i will not necessarily be binary, as in Equation (3.1). For each robot we set the highest x_m^i equal to one and all others as zero. We shortly show that the resulting solution after rounding is still close to optimal in practice. Furthermore, increasing the parameter h finds solutions that are close to binary.

The following pseudo-code explains the overall scheme of each robot for a distributed SATA. We solve the SATA problem at each time step.

Algorithm 2: Local algorithm

```
for r_i(k) ∈ r(k) do
    p^i(k) ←ComputeMotionPrimitives(r_i(k)).
    Find targets that can be sensed by m-th motion primitive of i-th robot (p_m^i(k)).
    Construct a h-hop communication graph.
    Apply local algorithm [1].
    x̂_m^i ← Rounding(x_m^i).
    p_m^{i*}(k) ← Motion primitive with x̂_m^i = 1.
    ApplyAction(p_m^{i*}(k)).
    k ← k + 1.
end
```

Advantages of the Local Algorithm

It is possible that there are some robots that are isolated from the others. That is, the communication graph or the layered graph may be disconnected. However, each component of the graph can run the local algorithm independently without affecting the solution quality. Furthermore, if a robot is disconnected from the rest, then it can take a greedy approach as described in Reference [27] before they reach any other robots to communicate.

The algorithm also allows for the number of robots and targets to change over time. Since each robot determines its neighbors at each time step, any new robots or targets will be identified and become part of the time-varying local layered graphs. The robots can also be anonymous (as long as they can break the symmetry to determine which robot, amongst a set, will realize the target node, when multiple robots can observe the same target).

The number of layers, h, directly affects the solution quality and can be set by the user. Increasing h results in better solutions at the expense of more communication. $h = 0$ is equivalent to the greedy approach where no robots communicate with each other.

$\mathbf{p}_m^i$	x_m^i	$h = 2$	$h = 10$	$h = 30$
$\mathbf{p}_1^1$	$x_1^1 =$	0.5000	0.5000	0.5000
$\mathbf{p}_2^1$	$x_2^1 =$	0.5000	0.5000	0.5000
$\mathbf{p}_3^2$	$x_3^2 =$	0.6667	0.7591	0.7855
$\mathbf{p}_4^2$	$x_4^2 =$	0.3333	0.2409	0.2145
$\mathbf{p}_5^3$	$x_5^3 =$	0.3333	0.2409	0.2145
$\mathbf{p}_6^3$	$x_6^3 =$	0.6667	0.7591	0.7855

Table 3.1: Solution returned by the local algorithm for the example shown in Figure 3.3, with all edges' weights set to 1, as a function of h.

The above table shows the result of applying the local algorithm to the graph in Figure 3.3 when all edge weights were set to 1. Three different values for h were tested: 2, 10, and 30. In all cases, $\mathbf{p}_3^2$ and $\mathbf{p}_6^3$ have larger values of x_p than other nodes. Thus, the robot 2 ($\mathbf{r}_2$) and the robot 3 ($\mathbf{r}_3$) will select the motion primitive 3 ($\mathbf{p}_3^2$) and the motion primitive 6 ($\mathbf{p}_6^3$), respectively, after employing a rounding technique to x_p's.

As the number of layers increases, the more distinct the x_p^i values returned by the algorithm. Another interesting observation is that robot 1 has the same equal value on both motion primitives of its own no matter how many number of layers are used. This is because all the targets are already observed by robots 2 and 3 with higher values.

3.4.2 Greedy Algorithm

The greedy algorithm requires a specific ordering of the robots given in advance. The first robot greedily chooses a motion primitive that can maximize the number of targets being observed. Those observed targets are removed from the consideration. Then, the second robot makes its choice; this repeats for the rest of robots. If the communication graph is disconnected and forms more than one connected component, the greedy algorithm can independently be applied to each connected component without modifying the algorithm. Note again that the greedy algorithm is for the WinnerTakesAll version of SATA.

Algorithm 3: Greedy algorithm

Input : Order of robots R.

1 Initialize $w(\mathbf{t}_j) = 0 \ \forall j \in T$.
2 **for** $i \in R$ **do**
3 **for** $m \in P^i$ **do**
4 Compute $c^j_{i,m} \ \forall j \in T$.
5 $w'(\mathbf{p}^i_m) = \sum_j \max\{w(\mathbf{t}_j), c^j_{i,m}\}$.
6 **end**
7 Determine $x^i_m = \text{argmax}\, w'(\mathbf{p}^i_m) \ \forall m \in P^i$.
8 Update $w(\mathbf{t}_j) = \max\{w(\mathbf{t}_j), c^j_{i,m}\} \ \forall j \in T$.
9 **end**
10 $y^j_i \leftarrow 0 \ \forall i \in R, \ j \in T$.
11 **for** $j \in T$ **do**
12 $\mathbf{r}^*_i \leftarrow \text{argmax}_{i \in R} \sum_m c^j_{i,m} x^i_m$.
13 $y^j_{i^*} \leftarrow 1$.
14 **end**

As shown in Algorithm 3, the greedy algorithm runs in $|R|$ communication rounds at each time step. We define two additional functions: $w(\mathbf{t}_j)$ gives a quality of tracking for j-th target; and $w'(\mathbf{p}^i_m)$ gives the sum of quality of tracking over all feasible targets using m-th motion primitive of i-th robot. If, for example, $c^j_{i,m}$ is used as a distance metric, the max ensures that the quality of tracking for j-th target is only given by the distance of the nearest robot/primitive. That is, even if multiple primitives can track the same target j, when counting the quality we only care about the closest one. The total quality will then be the sum of qualities for each target.

The objective in Line 5 in Algorithm 3 appears, at first sight, to be different than that given in Equation (3.4). The following lemma, however, proves that the two objectives are equivalent.

Lemma 2. *Greedy algorithm of Algorithm 3 gives a feasible solution for the* WinnerTakesAll *version of SA- TA.*

The proof is given in Appendix A.2. Since the objective in Line 5 in Algorithm 3 is submodular, the resulting algorithm yields a 2–approximation to WinnerTakes- All [95].

The greedy algorithm can perform arbitrarily worse than the optimal solution if it is applied to the Bottlene- ck version of the problem. In Appendix A.3, we show an example where the greedy yields an arbitrarily bad solution for the Bottleneck version.

A centralized-equivalent approach is one where the robots all broadcast their local information until some robot has received information from all others. This robot can obtain a centralized solution to the problem. A centralized-equivalent approach for a complete $\mathcal{G}_C$ runs in 2 communication rounds for receiving and sending data to neighbors. However, the greedy algorithm and local algorithm have $|R|$ and $h\log(1/\epsilon)$ communication rounds, respectively, for a complete $\mathcal{G}_C$. Note that $h \ll |R|$ for most practical cases.

3.5 Simulations

We carried out four types of simulations to verify the efficacy of the proposed algorithms under the condition that the amount of time required for communication is limited. First, we compare the performance of the greedy and local algorithms with centralized, optimal solutions. Second, we study the effect of varying the parameters (*i.e.*, the number of layers) for the local algorithm. Third, we describe how to implement the algorithms for sequential planning over multiple horizons and evaluate their performance over time. Last, we compare the greedy algorithm with a state-of-the-art distributed tracking algorithm.

3.5.1 Comparisons with Centralized Solutions

We performed comparison studies to verify the performance of the proposed algorithms. We compared the greedy solution with the optimal, centralized QMILP solution as well as a random algorithm as a baseline for the WinnerTakesAll version. We compared the local algorithm's solution with the optimal ILP solution as well as the LP with rounding for Bottleneck. For these comparisons, we assumed that there are only two primitives to choose from (making the random algorithm a powerful baseline). We later analyzed the algorithms with more primitives. We used TOMLAB [146] to solve the QMILP and ILP problems. The toolbox works with MATLAB and uses IBM's CPLEX optimizer in the background. On a laptop with processor configuration of Intel Core i7-5500U CPU @ 2.40GHz x 4 and 16 GB of memory the maximum time to solve was around 3 seconds on a case with 150 targets. Most of our cases were solved in less than 2 seconds.

We randomly generated graphs similar to Figure 3.3 for the comparison. To control the topology of the randomly generated graphs, we defined $\phi : \mathcal{G}_S \to \mathbb{R}$ to be the percentage of targets that are detected by a motion primitive. We denote the average degree of edges by

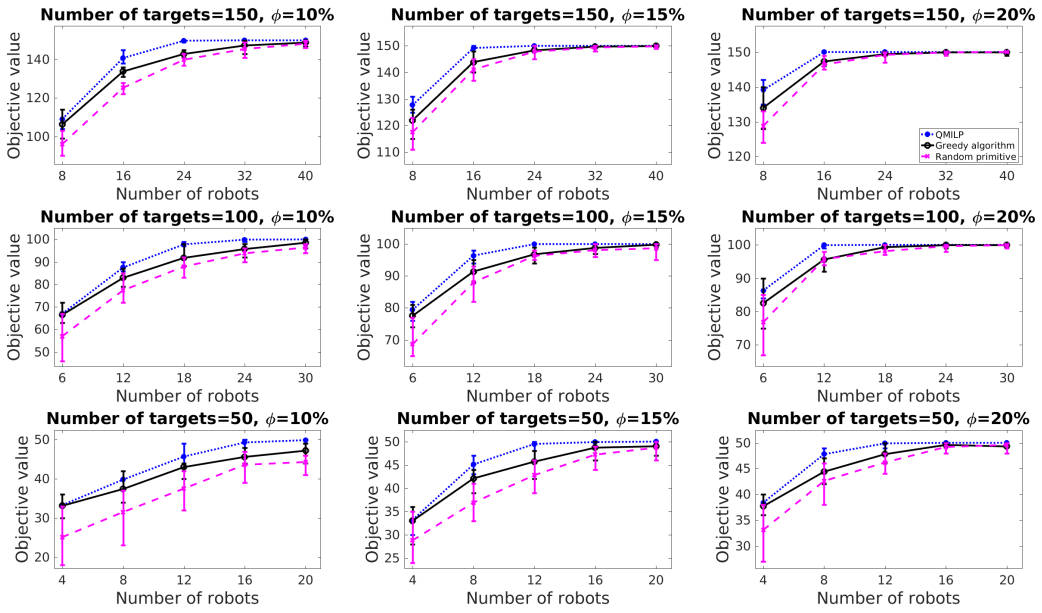


Figure 3.6: Showing the comparative results of QMILP, greedy algorithm, and randomly choosing a motion primitive for WINNERTAKESALL. To generate the graphs, we varied number of robots, total number of targets, and $\phi(\mathcal{G}_S)$. We ran 100 trials for each case.

$d_{avg}(\cdot)$. Therefore:

$$\phi(\mathcal{G}_S) := \frac{d_{avg}(T)}{\sum_{i=1}^{|R|} |P^i|} \times 100 = \frac{|E_S|}{\sum_{i=1}^{|R|} |P^i||T|} \times 100. \tag{3.6}$$

We started with the upper half of the graph, connecting each robot to its two motion primitives. Then, we iterated through each of motion primitive and randomly chose a target node to create an edge. Next, we iterated through target nodes and randomly chose a motion primitive to create an edge. We also added random edges to connect disconnected components (to keep the implementation simpler). We repeated this in order to get the required graph. If we needed to increase the degree of target nodes in the graph, we created new edges to random primitives till we achieved the desired $\phi(\mathcal{G}_S)$. We generated cases by varying $\phi(\mathcal{G}_S)$, number of targets, and number of robots using the method described above. Here, the tracking quality was defined as the number of targets, *i.e.*, $c^j_{i,m} \in \{0, 1\}$ for all cases.

The comparative simulation results for WINNERTAKES- ALL are presented in Figure 3.6. The plots show minimum, maximum, and average of the targets covered by the greedy algorithm and QMILP running 100 random instances for every setting of the parameters. We also show the number of targets covered when choosing motion primitives randomly as a baseline. We observe that the greedy algorithm performs comparatively to the optimal algorithm, and is

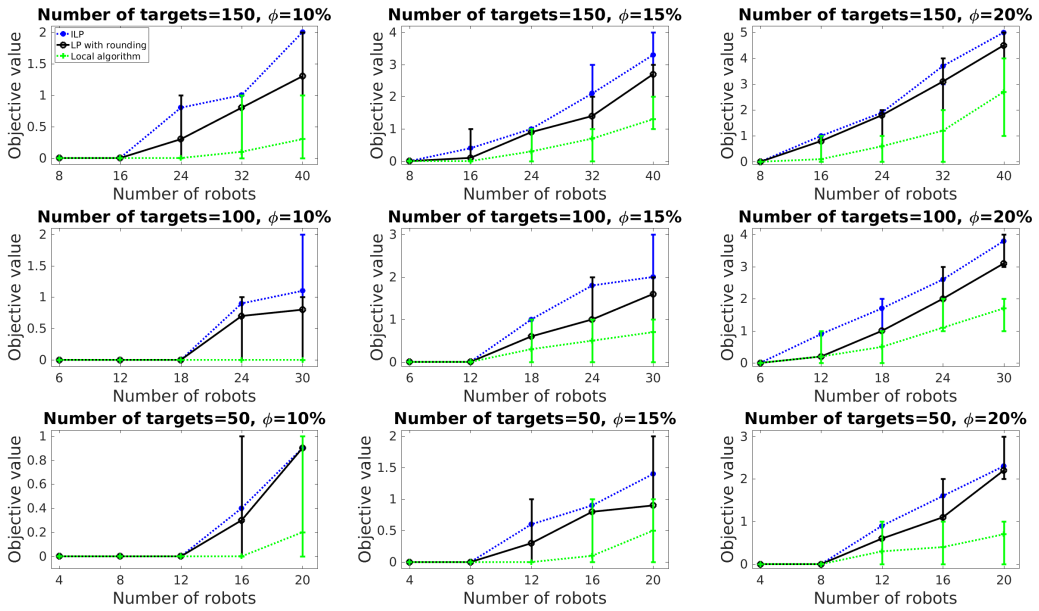
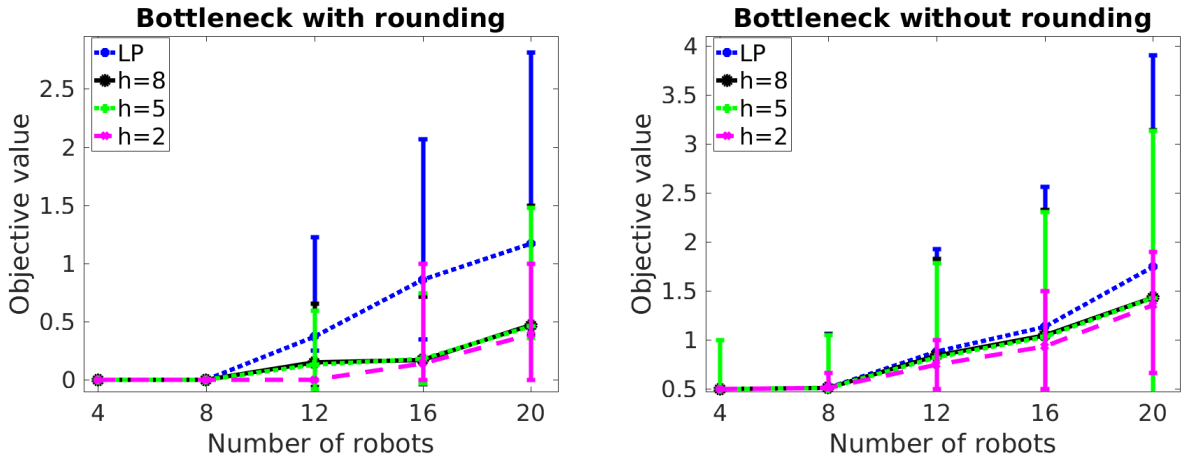


Figure 3.7: Comparison simulation for the BOTTLENECK version of the ILP, LP with rounding, local algorithm and randomly choosing a motion primitive. We set h to 2 in the local algorithm, for all cases. Each case was obtained from 100 trials.

always better than the baseline. In all the figures, $\Delta_R = 2$, making random a relatively powerful baseline. The difference between the greedy algorithm and the baseline becomes smaller as $\phi(\mathcal{G}_S)$ increases. This could be because of the fact that the baseline saturates at the maximum objective value with fewer robots as $\phi(\mathcal{G}_S)$ increases. As $\phi(\mathcal{G}_S)$, number of targets, and number of robots increase, the performance of the greedy algorithm also improves.

Figure 3.7 shows the comparison results for BOTTLENE- CK where the objective values were computed from the w term of Equation (3.5). As the proposed local algorithm is a linear relaxation of the ILP formulation, we compared the local solution with the optimal ILP solution. Note that both the ILP and LP with rounding are centralized methods. If the solution value is 0, this means that at least one target is not covered by any selected motion primitives. The specific configuration of input motion primitives and target states is such that no matter what motion primitives are chosen, at least one target will be left uncovered. This means that the bottleneck objective (*i.e.*, the optimal value of ILP) is 0. If the mean value is larger than 0, this implies that all targets are covered by at least one motion primitive on average. The ILP and LP with rounding outperform the local algorithm in all cases. Nevertheless, we find that the local algorithm performs comparably to the centralized methods (and far better than the theoretical bound).

3.5.2 Effect of h for the Local Algorithm



(a) LP with rounding.

(b) LP without rounding.

Figure 3.8: Analysis of varying the number of layers (h) for the local algorithm. The number of targets used is 50 and $\phi(\mathcal{G}_S) = 15\%$. We ran 100 trials for each case.

We analyzed the performance of the local algorithm for different number of layers (*i.e.*, h), as shown in Figure 3.8. The LP value (without rounding) is the upper bound on the optimal solution. We observed how much the rounding sacrifices by comparing the LP with and without the rounding. In the case where h was set to 5 and 8 for both with and without the rounding, there is no evident difference between them. This implies that h should not necessarily be large as it does not contribute to the solution quality much (as also seen in Theorem 1). In other words, the local algorithm does not require a large number of communication hops among robots, which is a powerful feature of the local algorithm.

3.5.3 Multi-robot Multi-target Tracking over Time

The greedy and local algorithms find the motion primitives to be applied over a single horizon. In order to track over time, the SATA problem will need to be solved repeatedly at each time step. In this section, we describe how to address this and other challenges associated with a practical implementation. We demonstrate a realistic scenario of cooperative multi-target tracking in the Gazebo simulator using ROS (Figure 3.9). A bounded environment consists of dynamic targets that move in a straight line and change their heading direction randomly after a certain period. The motion model is not known to the robots.

Greedy Algorithm. We implemented the greedy algorithm to solve the WinnerTakesAll variant in a fully distributed fashion. There was no centralized algorithm and each robot

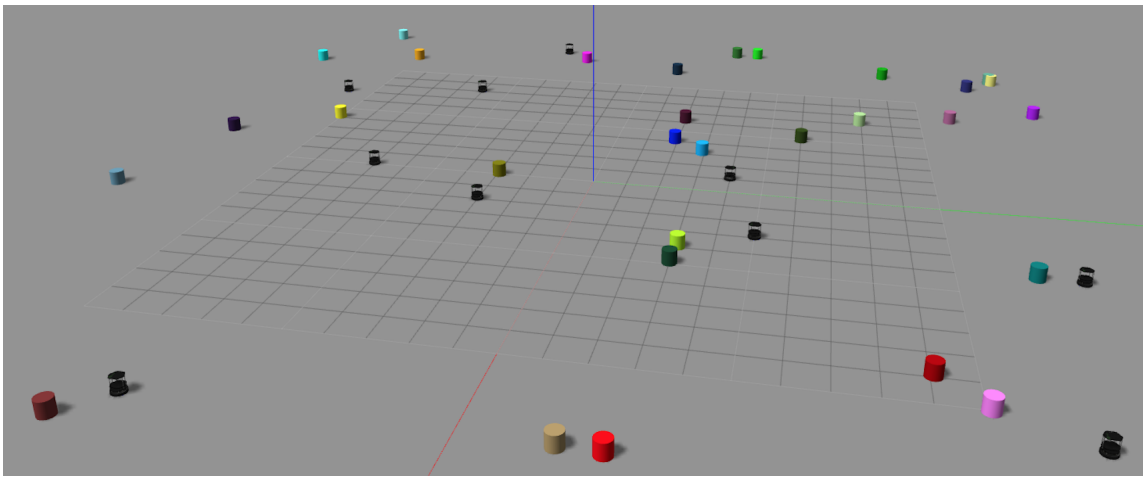


Figure 3.9: Gazebo simulator showing ten robots tracking thirty randomly moving targets. We set the sensing and communication ranges to $5m$ and $10m$, respectively.

was a separate ROS node that only had access to the local information. Each robot had its local estimator that estimated the state of targets within its FOV. We simulated proximity-limited communication range such that only robots that can see the same target can exchange messages with each other.

A sketch for the implementation of the greedy algorithm is as follows. Each robot has a local timer which is synchronized with the others. Each robot also knows its own ID which is also the order in which the sequential decisions are made. We partition the planning horizon into two periods. In the first *selection* period, the robots choose their own primitives sequentially using the greedy algorithm. In the second *execution* period, the robots apply their motion primitives and obtain measurements of the target.

In the selection period, a robot waits for the predecessor robots (of lower IDs) to make their selections. Every robot knows when it is its turn to select a motion primitive (since the order is fixed). Before its turn, a robot simply keeps track of the most recent $w(\mathbf{t}_j)$ vector received from a predecessor robot within communication range. During its turn, the robot chooses its motion primitive using the greedy algorithm, and updates the $w(\mathbf{t}_j)$ vector based on its choice. It then broadcasts this updated vector to the neighbors, and waits for the selection period to end. Then, each robot applies its selected motion primitive till the end of the horizon. The process repeats after each planning horizon. The selection period can be preceded by a sensor fusion period, where the robots can execute, for example, the covariance intersection algorithm [144].

For simulations we set the selection and execution periods times to $0.2|R|s$ and $6s$, respectively, where $|R|$ is the number of robots. Each robot made its choice after $0.2s$ within the selection period. Each robot had a precomputed library of 21 motion primitives including staying in place. It should be noted that our algorithms do not require a motion primitive of

stay in place. Each robot had a disk-shaped FOV. The sensing and communication ranges were set to $5m$ and $10m$, respectively. We tested both the inverse of the distance and the number of targets as tracking quality (which defines $c^j_{i,m}$).

We carried out simulations using ten robots tracking thirty moving targets, as shown in Figure 3.9. Initial positions of robots and targets were randomly chosen in a $30 \times 30m$ square environment. It may be possible that some targets were outside the FOV of any robots in the beginning.

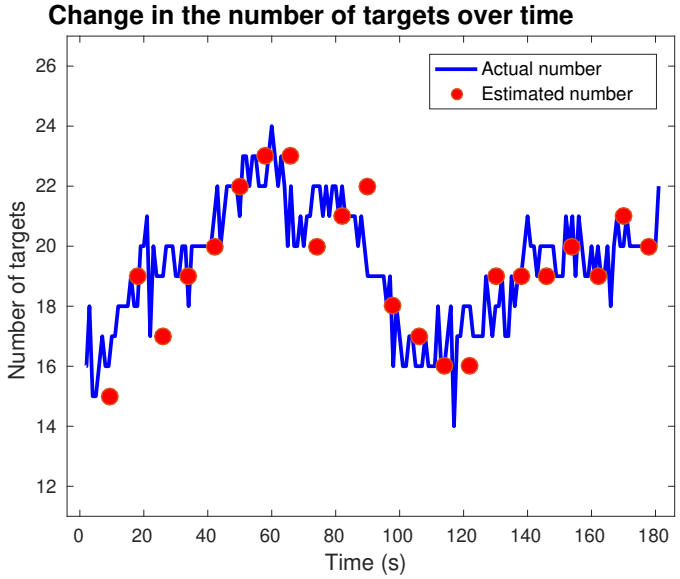
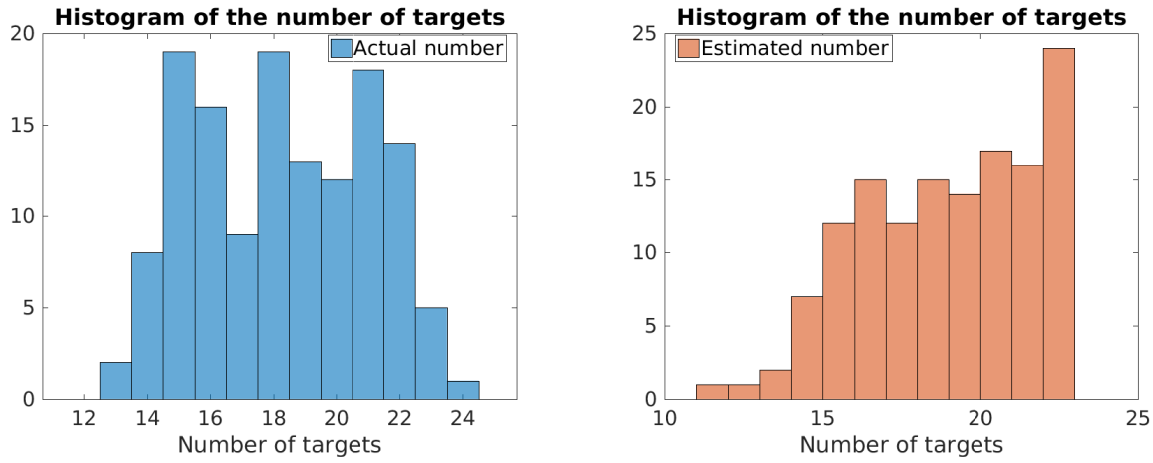


Figure 3.10: Change in the number of targets over time when ten robots are tracking thirty moving targets.

Figure 3.10 shows the change in the number of targets over time from a randomly generated instance where the objective was to track the most number of targets. We show both the estimated number of targets and the actual number of targets. The estimated number is the value of the solution found at the end of the selection period (obtained every $8s$). This is based on the predicted trajectory of the targets.[5] The actual number of targets was found by counting the target that is within the FOV of any robots during the execution period. Figure 3.11 shows the histogram of the actual and estimated number of targets for 10 trials, each lasting three minutes.

Figures 3.12 and 3.13 show the corresponding plots when the objective was to maximize the total quality of tracking (inverse distance to the targets). Here, we saw that the estimated and the actual values differed much more than the previous case. We conjectured that this was due to the fact that the uncertainty in the motion model of the robots, targets,

[5]Although we model linear motion for the targets, more sophisticated models for the prediction of target states can also be employed.



(a) Actual number of targets.

(b) Estimated number of targets.

Figure 3.11: Histogram of the number of targets.

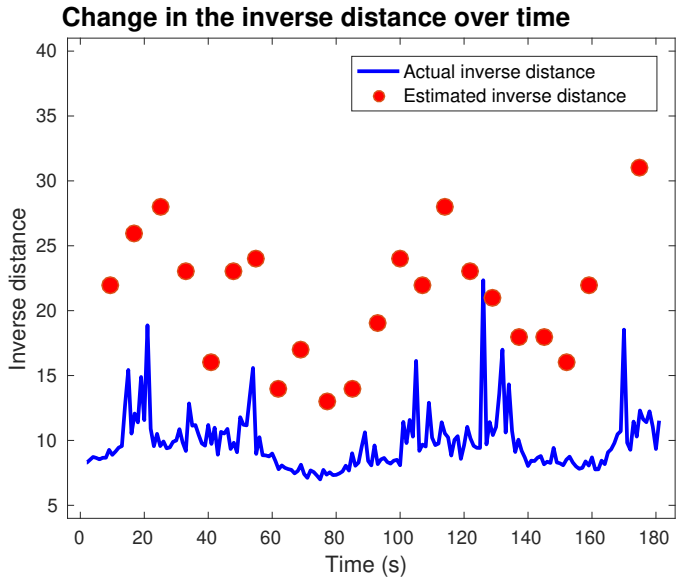
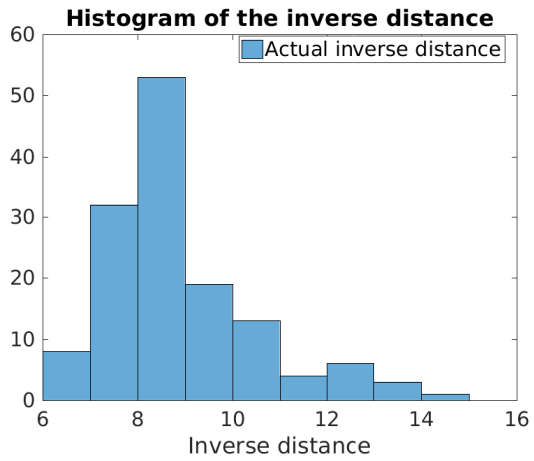


Figure 3.12: Change in the inverse of the distance over time when ten robots are tracking thirty moving targets.



(a) Actual inverse distance. (b) Estimated inverse distance.

Figure 3.13: Histogram of the inverse distance.

and measurements had a larger effect on the actual quality of tracking as compared to the number of targets tracked. For instance, even if the actual state of the target deviates from the predicted state, it is still likely that the target will be in the FOV. However, the actual distance between the robot and the target may be much larger than estimated.

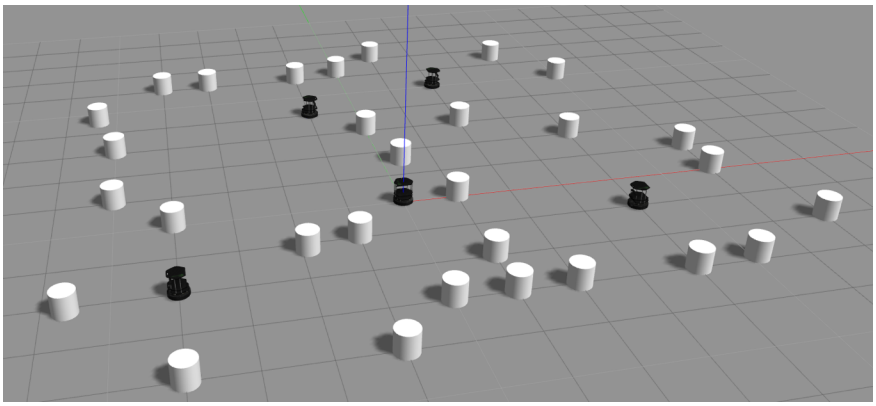


Figure 3.14: Snapshot of the Gazebo simulator that shows when five robots are tracking thirty stationary and moving targets. The sensing and communication ranges were set to $3m$ and $6m$, respectively.

Local Algorithm. We also implemented the proposed local algorithm as shown in Figure 3.14. Five mobile robots were deployed to track thirty targets (a subset of which were mobile) with a FOV of $3m$ on the xy plane. For each robot two motion primitives were used: one was to remain in the same position and the other one was randomly generated between $-30°$ and $30°$ of the robot's heading traveling randomly up to $1m$.

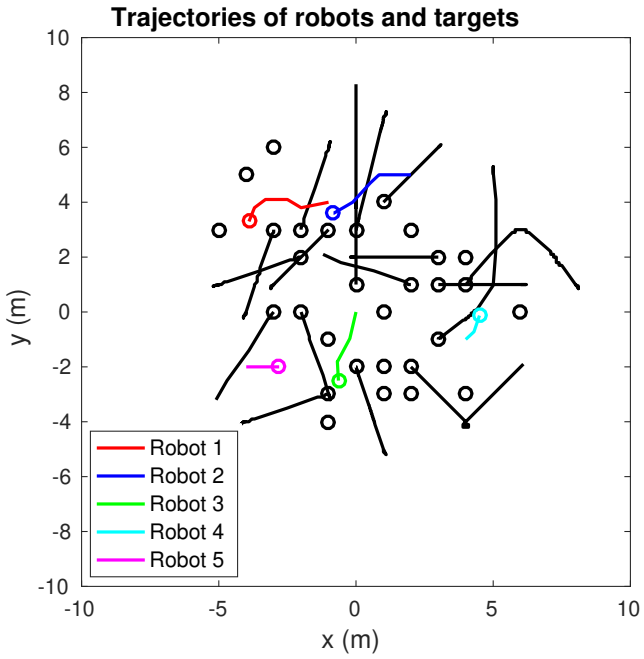
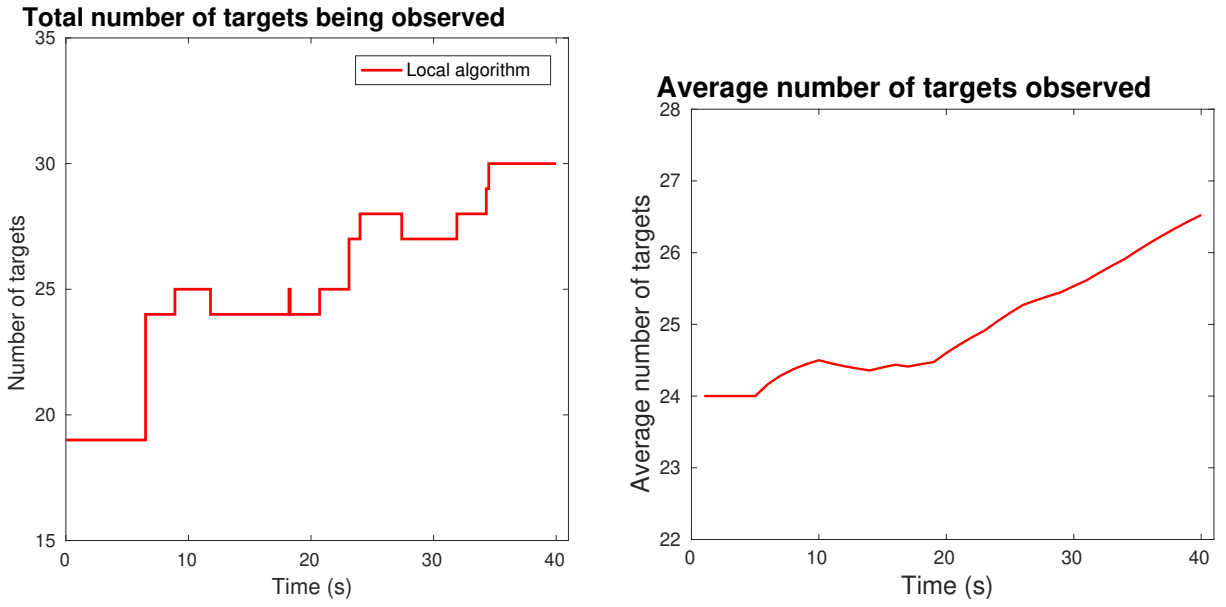


Figure 3.15: Plot of trajectories of robots and targets applying the local algorithm to the simulation given in Figure 3.14. Black lines represent trajectories of thirty targets. ∘ denotes the end position of trajectories. The algorithm was performed for 40 seconds.



(a) Total number of targets observed.

(b) Average number of targets observed over time.

Figure 3.16: Change in the total and average number of targets being observed by any robots over time.

The objective of this simulation is to show the performance of the proposed algorithm for the BOTTLENECK version. At each time step, the local algorithm was employed to choose motion primitives that maximize the minimum number of targets being observed by any robots.

Figure 3.15 shows the resultant trajectories of robots and targets obtained from the simulation. Figure 3.16 presents the (total/average) number of targets tracked by the local algorithm for a specific instance. Although the local algorithm has a sub-optimal performance guarantee, we observe that in practice, it performs comparably to the optimal path.

3.5.4 Comparison of the Greedy Algorithm with Other CMOMMT Algorithm

We compared the greedy algorithm with an algorithm proposed by Parker [2] following the CMOMMT approach. This algorithm addresses the same objective as the WINNERTAKESALL. Parker's algorithm computes a local force vector for all robots (attraction by nearby targets and repulsion by other nearby robots). It does not require any central unit to determine their motion primitives and considers limited sensing and communication ranges, similar to this chapter. Parker's algorithm determines the moving direction of robots by using the local force vector and moves the robots along this direction until they meet the available maneuverability at each time step. However, no theoretic guarantee with respect to the optimal solution was provided by this algorithm.

We created an environment of $200 \times 200m$ square for comparison using MATLAB. The robots can move $10m$ per time step while the targets can move $5m$ per time step and randomly changed their direction every 25 time steps. If the targets met the boundary of the environment, they picked a random direction that kept them within the environment. In each instance, robots and targets were randomly located initially. The sensing and communication ranges were set to $40m$ and $80m$, respectively.

We empirically studied two cases: the first is to evaluate the objective value of the proposed greedy algorithm and Parker's algorithm for the same problem instance at a given time step; and the second is to apply the two algorithms over 200 time steps starting from the same configuration.

When both algorithms were applied to the same problem setup (Figure 3.17(a)), the objective values for both algorithms increased as the number of targets increased. Nevertheless, the greedy algorithm outperformed Parker's algorithm. This can be attributed to the fact that Parker's algorithm computes the local force vector based on a heuristic (get closer to the targets) but does not explicitly optimize the objective function of WINNERTAKESALL. In Figures 3.17(b) and 3.17(c), similar results can be seen when both algorithms generate different trajectories for robots after 200 time steps. The comparison measure used in Figure 3.17(c) is the average of the objective value over time, first proposed by Parker [2]. These empirical simulations show the superior performance of the greedy algorithm over the existing method.

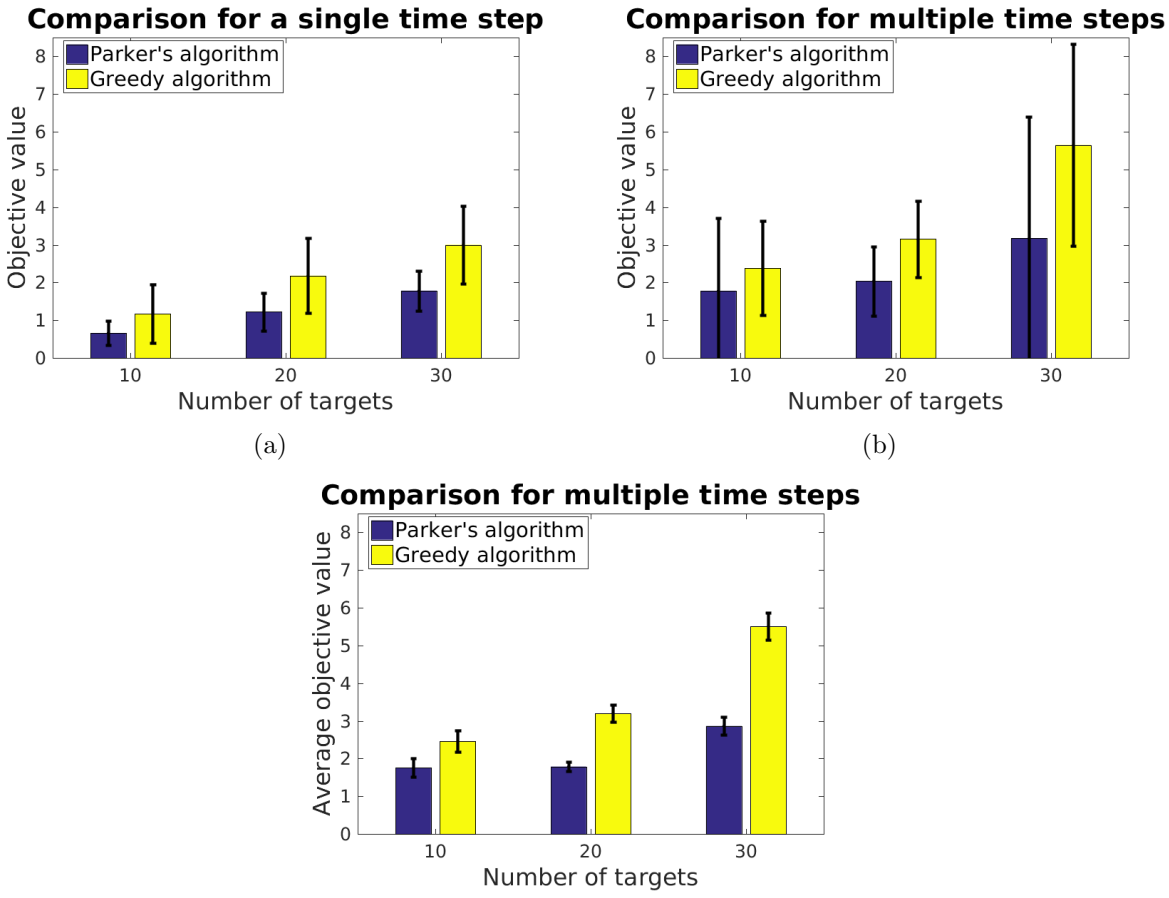


Figure 3.17: Comparison with the Parker's algorithm [2]. (a) 200 instances were run. (b) 200 time steps were run. (c) 200 time steps were run to compare the metric proposed by Parker [2]. We used 10 robots for all cases. We ran 10 trials for (b) and (c). Bar graphs show the mean and standard deviation for different number of targets (10, 20 and 30 targets).

In summary, we find that our algorithms perform comparably with centralized, optimal algorithms and outperform the baseline algorithm. We also find that greedy algorithm has better performance than the decentralized algorithm from Parker [2]. In theory, the performance bound for the local algorithm worsens as h, the amount of communication available, decreases. However, in practice, we find that the local algorithm does not require a large number of layers to yield good performance, which reduces the computational and communication burden.

3.6 Conclusion

This chapter gives a new approach to solve the multi-robot multi-target assignment problem using greedy and local algorithms. Our work is motivated by scenarios where the robots would like to reduce their communication to solve the given assignment problem while at the same time maintaining some guarantees of tracking. We used powerful local communication framework employed by Floréen *et al.* [1] to leverage an algorithm that can trade-off the optimality with communication complexity. We empirically evaluated this algorithm and compared it with the baseline greedy strategies.

Our immediate future work is to expand the scope of the problem to solve both versions of SATA over multiple horizons. In principle, we can replace each motion primitive input with a longer horizon trajectory and plan for multiple time steps (say, H time steps). However, this comes at the expense of increased number of trajectories $\left|P^i\right|^H$ to choose from which will result in increased computational time. Furthermore, planning for a longer horizon will require prediction of targets' states far in the future which can lead to poorer tracking performance. We are also working on implementing the resulting algorithms on actual aerial robotic systems to carry out real-world experimentation.

Chapter 4

Coordinated Online Exploration of a Translating Plume

4.1 Introduction

We investigate the problem of exploring and mapping flows of an unknown hazardous agent in aquatic environments using a team of autonomous aerial robots. Our overall vision is to develop algorithms for enabling a team of robots to assist emergency responders in disaster scenarios, such as dispersal of oil aerosols and radioactive particulates in the environment. Previous work has shown the value of using Unmanned Surface Vehicles (USVs) for monitoring and sampling spatiotemporal plumes in aquatic environments [147, 148]. However, USVs can only provide a narrow (local) view of the plumes. Detecting a hazardous agent in the environment may require a USV to cover a large portion of the aquatic system, which may take a considerable amount of time. Furthermore, even after detecting the threat, teams of USVs may not be able to keep up with the rapidly spreading plume. Thus, emergency responders may not be informed of the full extent of the hazards, limiting their ability to respond quickly and effectively. This motivates the use of Unmanned Aerial Vehicles (UAVs) which can provide a wider (regional) picture and that can coordinate with USVs for more targeted deployments.

The problem of exploring an unknown 2D environment is a well-studied one in the robotics [149, 150, 151, 152] and computational geometry [153, 154, 155, 156] communities. The problem considered in this chapter differs from these works in two critical ways. First, we consider the case where the plume is not static but is instead translating with a given velocity. As a result, the performance of the algorithm depends on the relative speeds of the robots and the plume. Second, in our setup the robots are not restricted to stay inside (or over) the

Figure 4.1: An aerial robot (UAV) conducting the plume exploration in an abandoned quarry near Blacksburg, Virginia.

plume all the time. The robots can fly over locations that are not part of the plume, thereby allowing them to "shortcut" from one part of the plume to the other. Contrast this with conventional 2D exploration problems, where the robots are restricted to stay within the boundary of the environment. Because the robots do not know the shape of the plume a priori, they may not be able to take a "shortcut" even if one exists. As a result, the robots may end up taking a longer path, resulting in a poorer performance. Nevertheless, we present an algorithm that is *competitive* with respect to the optimal algorithm.

We use the notion of *competitive ratio* [157] to analyze the performance of our algorithm. The competitive ratio for an online algorithm is defined as the largest (*i.e.*, worst-case input) ratio of the time taken by the online algorithm to the time taken by an optimal offline algorithm. The offline algorithm is one which knows the shape of the 2D plume a priori. We seek algorithms that have a low (preferably, constant) competitive ratio. Our main result is a constant competitive ratio for exploring a translating plume for a fixed number of robots. The constant depends on the relative speeds of the plume and the robots.

We require the robots to ensure that all points of the plume are within the Field-Of-View (FOV) of at least one of the aerial robots along their paths. The objective is to minimize the time required for all the robots to explore the plume and return back to the starting position. Our algorithm builds on the one presented by Higashikawa *et al.* [156] for exploring an unknown binary tree. We show how to reduce the problem of exploring the plume to that of exploring a binary tree. We first start with the simpler scenario where the plume is modeled as a 2D grid and then generalize it to the case where the plume boundary is any smooth (formally defined in Section 4.3) 2D curve. For both cases, we show that our algorithm yields a constant-competitive ratio.

We validate our algorithm through simulations that quantify the performance as a function of the size of the plume, the number of robots, and the relative speeds of the plume and the robots. We also conduct a proof-of-concept field experiment using a UAV with a downwards-facing camera to explore and map a stationary region of interest (runway). We discuss how to implement the algorithm in a practical setting and discuss challenges associated with noisy measurements.

A preliminary version of this chapter was presented in [3]. This version improves upon [3] with a more expansive literature survey, a more detailed explanation on the proposed algorithm, and new simulation results and proof-of-concept experiments, including a description of how to implement the proposed algorithm using a robot with a downward-facing camera.

The rest of the chapter is organized as follows. We begin by introducing the related work in Section 4.2. We describe the problem setup in Section 4.3. Our proposed algorithm for a grid-based map is presented in Section 4.4. We then extend this to arbitrarily-shaped shape in Section 4.5. We present results from representative simulations in Section 4.6 and field experiments in Section 4.7, respectively, before concluding with a discussion of future work in Section 4.8.

4.2 Related Work

Environmental monitoring has extensively been studied in robotics due to its practical applications. Some of highlighted tasks include precision agriculture [10, 158], wildlife habitat monitoring [159, 8, 160] and atmospheric plume tracking [161, 162, 163]. For survey results, see reference [148]. The area coverage and exploration are crucial for environmental monitoring as a given environment must be explored by robots in order to detect a target of interest. Galceran and Carreras [164] listed coverage path planning algorithms that can be used for different sensing and motion models. In case of plume tracking, the aim is to explore and map a plume by robots with limited sensing capability.

The objective of online exploration [150, 151, 152] is to explore and map a region without having prior knowledge on the size and the shape of the region. Corah and Michael [165] studied informative exploration in which the objective is to maximize the information gathered along the planned trajectories. They developed a near-optimal distributed algorithm for multi-robot exploration, which approximates the well-known sequential greedy assignment [166]. Plonski *et al.* [160] proposed algorithms for tracking radio-tagged invasive fish using USVs and ground robots. They proved competitive ratios for navigating an environment containing an unknown obstacle and energy-efficient solar exploration. Hitz *et al.* [12] focused on localizing interesting areas in an unknown environment using level set estimation to monitor hazardous cyanobacteria blooms in lakes. The objective in these works was not to completely map an unknown environment (which is the case in this chapter) but to maximize information gain, track and localize targets of interest.

Sim and Little [167] proposed a vision-based exploration and mapping solution for a single robot. Cesare *et al.* [168] developed a multi-robot exploration algorithm for heterogeneous robots with limited communication and battery-life constraints. However, these approaches do not guarantee complete coverage.

Bender *et al.* [169] and Das *et al.* [170] addressed the problem of dealing with unlabelled (*i.e.*, anonymous) vertices when exploring an unknown graph. The former defined a pebble that can identify a vertex and found the number of pebbles required to map an unknown environment. While the former considered the case of a single robot, the latter proposed a distributed version, allowing multiple robots to start from different vertices, and proved upper bounds on the time complexity of their algorithm. Their algorithms, however, do not yield a competitive ratio used as a performance measure in this chapter.

When a plume region (or any monitoring object) can be represented by a grid polygon, there exists literature which explores a polygonal region not only completely but also competitively with respect to the optimal trajectory. This can be categorized into *lawn mowing* and *milling* where the former allows a robot to move outside the boundary of a polygon whereas the latter does not (see Figure 4.2). Icking and Kamphans [171] proposed a strategy of generating a competitive tour for online milling which may contain holes. Icking [153] showed $\frac{4}{3}$–competitive algorithm for online milling without considering holes. The algorithms presented by Arkin *et al.* [172] have $(3+\epsilon)$–approximation for offline lawn mowing and 2.5–approximation for offline milling. Kolenderska *et al.* [154] developed an online milling algorithm of a grid polygon without holes that has a competitive ratio of $\frac{5}{4}$. However, aforementioned works did not take into account a multi-agent perspective. Although Arya *et al.* [173] presented an approximation algorithm for milling where multiple robots can be deployed, their algorithm solves an offline problem. In this work, we pose an online milling version for multiple robots, taking into account their limited FOV.

Previous works in computational geometry assumed specific properties of the region under exploration to ease the analysis. We restrict the plumes to satisfy a specific notion of fatness (defined in the next section). Stappen and Overmars [174] used the notion of k–fatness in motion planning with obstacles — the smaller the value of k, the fatter the obstacle. Efrat [175] defined a (α, β)–covered object if each angle of a triangle fully inside the object is at least α and each edge of this triangle is at least β multiplied by the diameter of the object are satisfied. Aloupis *et al.* [176] adopted the same notation of the fatness for the application of triangulating and guarding polygons. Lee *et al.* [177] used a similar fatness for a triangulation of a planar region for multi-robot coverage. These approaches exploited the fatness to prove the space complexity of their algorithms. In this work, we also define the fatness for proving the competitive ratio for arbitrary plume shape.

When multiple robots are considered, most of works [155, 149, 156, 178, 179, 180] have studied a tree-based exploration by employing a recursive Depth-First Search (DFS). In these works, the environment to be explored was assumed to be a tree. Fraigniaud *et al.* [155] proposed a tree exploration algorithm using R robots that is $\mathcal{O}(\frac{R}{\log R})$–competitive. In their work,

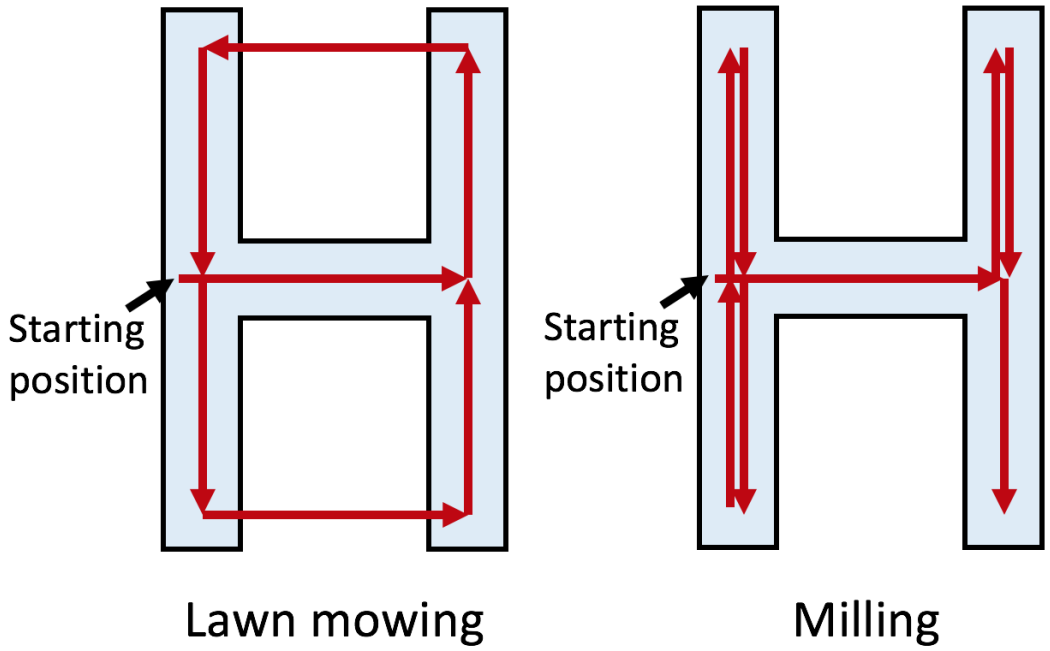


Figure 4.2: Example of different trajectories obtained by applying lawn mowing and milling approaches to the H-shaped plume. Whereas lawn mowing allows the robot to move outside the plume, milling restricts the robot to stay inside the plume.

each robot was allowed to observe the incident edges but not the adjacent vertices. Brass *et al.* [149] used the same sensing model and improved the competitive ratio of Fraigniaud *et al.* [155] to $2|E|/R + \mathcal{O}((R+r)^{R-1})$, where $|E|$ and r denote the number of edges and radius of the graph, respectively. Dynia *et al.* [179] improved the lower bound proposed by Fraigniaud *et al.* [155] of $2 - \frac{1}{R}$ to $\Omega(\frac{\log R}{\log\log R})$.

Megow *et al.* [181] showed that the competitive ratio of a single-robot DFS is $2(2+\epsilon)(1+2/\epsilon)$, where ϵ is a fixed positive parameter, when applied to general graphs. Higashikawa *et al.* [156] presented a $\frac{R+\lfloor\log R\rfloor}{1+\lfloor\log R\rfloor}$–competitive algorithm for exploring a binary tree with R robots. Das *et al.* [182] presented an algorithm for minimizing the number of robots given limited energy E for each robot.

Preshant *et al.* [180] showed that the competitive ratio remains largely the same, $\frac{2(\sqrt{2}R+\log R)}{1+\log R}$, where the environment was an orthogonal polygon[1] but was modeled as a tree. We build on this and generalize this to the case where the environment boundary is not necessarily orthogonal. In fact, it can be curved and may contain holes as well. Furthermore, we show how to adapt this algorithm to the case where the environment itself is translating.

To share information among multiple robots, global or local communication can be used. Das *et al.* [170] and Brass *et al.* [149] introduced bookkeeping devices to write local informa-

[1] An orthogonal polygon is one in which the edges are aligned with either the X or Y axes.

tion on the vertex so that other robots can read this information when they visit the same vertex later. Lee *et al.* [177] proposed distributed online exploration algorithms assuming a fully connected network. In Higashikawa *et al.* [156], robots can communicate with each other when they meet at the same vertex. We adopt the same model.

4.3 Problem Description

We consider the problem of mapping a slowly translating plume (Definition 1) using a team with R robots. The size and the shape of the plume are not known to the robots a priori. We use $P \in \mathbb{R}^2$ to denote the 2D plume. Let $int(P)$ be the interior of P and ∂P be the boundary of P.

We assume the plume is translating on the plane at zero height and that the aerial robots fly at a fixed altitude. Each robot has a downwards-facing camera that yields a square footprint on the plane containing the plume. That is the FOV is a square. Without loss of generality, we assume that the side length of the square FOV is 1 in this work.

We consider the plume whose size is at least as large as the FOV of the robots. Specifically, we require the plume to satisfy the following assumption.

Definition 1 (Fat Plumes)**.** *For any $p' \in \partial P$, let $p \in int(P)$ be a point on the normal to ∂P at p' such that p is at a distance of $\frac{\sqrt{2}}{2}$ from p'. Let $B(p)$ be an open ball of radius $\frac{\sqrt{2}}{2}$, i.e., $B(p) = \{q \mid \|p - q\|_2 < \frac{\sqrt{2}}{2}\}$ where $q \in \mathbb{R}^2$. We say that the plume P is* fat *if $B(p)$ lies completely inside $int(P)$ for all $p' \in \partial P$.*

Figure 4.3 shows an example of a plume that is *fat*. This definition disallows plumes that have a *width* less than that of the FOV of the robot. Note, however, we still allow the plume to contain one or more holes.

We assume that the plume translates with a fixed speed of S_p in a fixed direction, both of which are known to the robots.[2] The speed and the moving direction of the plume can be determined from the flow of the water which can be found from the environmental conditions such as wind and ocean current models [183].

We assume that all robots move at a speed of $S_r > S_p$. We further assume that all robots can communicate with each other at all times and thus, restrict our attention to centralized algorithms.

We focus on the mapping problem in this chapter. Therefore, we assume that all robots start at the same location where they first observe the plume. We seek tours for each robot that explore the plume and return back to this starting location.

[2]This is equivalent to the rigid-body translation of P.

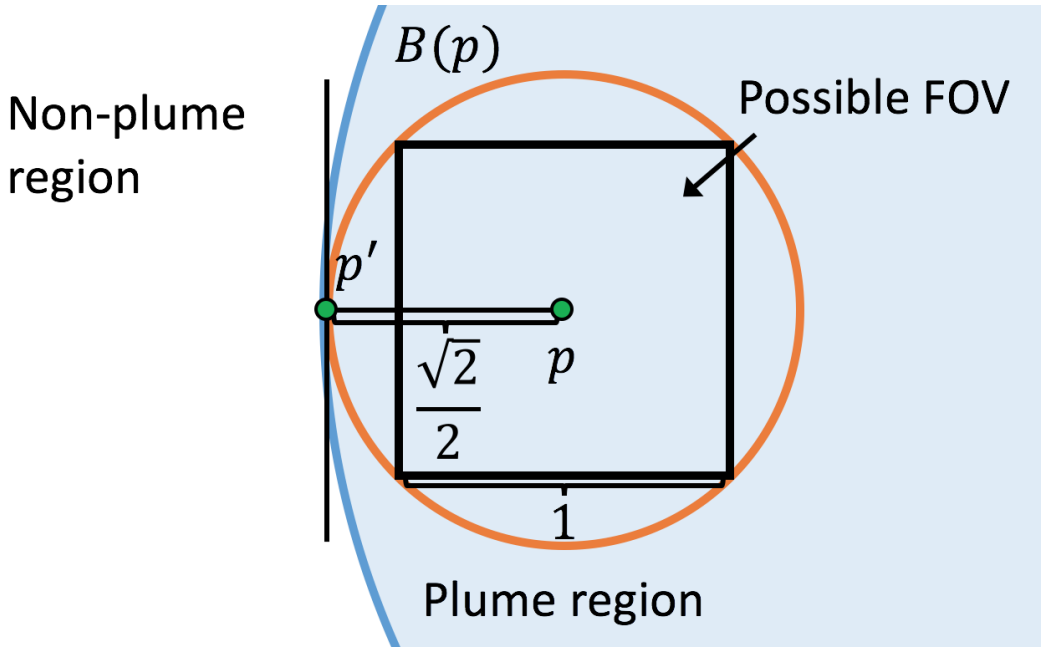


Figure 4.3: We restrict our attention to plumes that are *fat* (Definition 1).

Problem 3 (Multi-Robot Exploration of Translating Plume)**.** *Find a tour for all the robots that minimizes the exploration time such that every point in the plume is visible from the FOV of at least one robot's tour. All tours must return to the same starting position. The exploration time is given by the time when the last robot returns to the starting position.*

The proposed problem is an online exploration problem. The objective function is the exploration time which is the time of the longest tour. In the next section, we present an algorithm that is based on recursive DFS which is competitive with respect to the optimal solution.

4.4 Plume Exploration over a Grid Map

In this section, we present our main algorithm. We first solve a simpler version of Problem 3 where the plume is approximated as a grid map. We then use this result to solve Problem 3 by relaxing the grid approximation afterwards. Our algorithm is based on the recursive DFS that models the plume under exploration as a tree. We first show that our strategy is competitive for the grid map case and then analyze the effect of approximating an arbitrary plume shape with a grid.

4.4.1 Recursive DFS Algorithm for a Grid Map

In this section, we assume that the plume is represented as a grid map [184]. The environment is modeled as a collection of cells, each of which is a square of unit side length. Each cell is connected to four of its neighbors. The plume P is just a collection of C cells that form one connected set (if a cell $c \in P$ is part of the plume, then one of its four neighbors must also be a part of the plume when $C > 1$).

The problem of exploring the plume is then simplified to that of exploring a grid map and identify the cells that belong in P. Since we assume that the FOV is also a unit square, a robot may obtain an image by positioning itself at the center of a cell. By analyzing the pixels on the boundary of the image, the robot can then determine if any of the four neighboring cells are also part of the plume or not.

We model P as a tree and propose a recursive DFS algorithm based on the tree exploration algorithm given by Higashikawa *et al.* [156]. Higashikawa *et al.* [156] developed a recursive DFS algorithm for exploring a binary tree. In our case, the grid graph to be explored is not necessarily a tree (it may contain cycles). Regardless, we show that modeling the underlying graph as a binary tree still leads to an algorithm with a constant competitive ratio.

The root of the tree is the cell corresponding to the starting position of the robots. Upon visiting a cell, the robots can identify if one or more of the four neighboring cells also contain the plume. The neighboring cells that contain the plume are added as children of the present cell in the tree unless those cells have been previously added to the tree. This condition prevents cycles.

The number of neighboring cells when a robot visits a new cell can be at most three. Therefore, the resulting tree may not be binary. However, by introducing a dummy edge of length 0 and a dummy vertex, we can convert the tree into a binary tree without loss of generality.[3]

Each neighboring plume cell determined by the sensing model becomes one of candidate cells that robots can choose from as the next vertex to visit. The goal becomes to visit all $C - 1$ cells (that correspond to the plume cells but excluding the starting cell) at least once by one of the robots.

If $R = 1$, then our algorithm becomes conventional recursive DFS for a single robot. However, in the multi-robot case, as the robots build the tree, we split the robots as equally as possible and assign them to explore the children vertices.

We define three states for each vertex in the tree: *unexplored* if the vertex is not visited by any robots; *under exploration* if the vertex is visited by any robots but the leaf vertex connected from the vertex is not visited by any robots; and *explored* if the vertex as well as the leaf vertex in the same branch are visited by any robots. When robots decide which vertex to move among neighboring cells of a plume region, they do not consider *explored*

[3]This step is included in Line 15 of Algorithm 4.

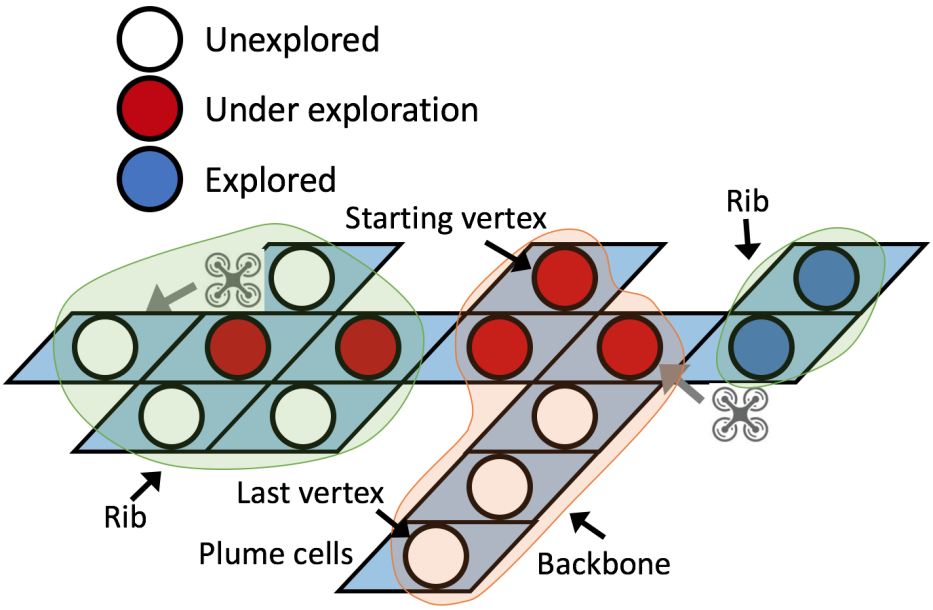
vertices but vertices that are either *unexplored* or *under exploration*. This is because having *explored* vertex means that the offspring of it must have also been explored by any robots (see Figure 4.4).

The details are given in Algorithm 4.[4] All vertices are marked as *unexplored* state in the beginning. Each robot runs Algorithm 4 whenever it reaches a vertex. The algorithm can be implemented to a single robot independently with respect to other robots as long as they can share the state information of vertices. The algorithm terminates when all robots return to the starting vertex and all vertices are marked as *explored*.

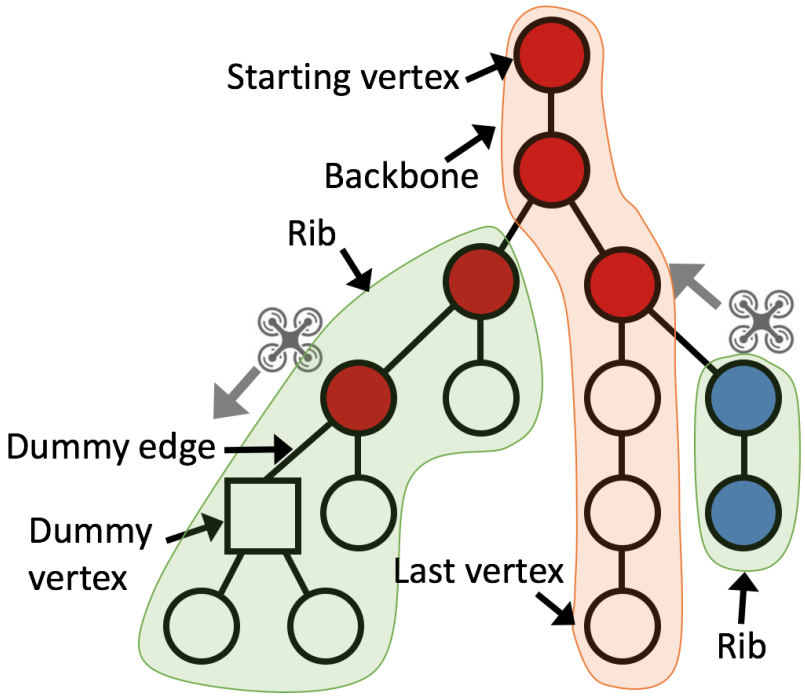
Algorithm 4: Multi-Robot Recursive DFS

```
1  Observe N(v) to determine whether neighboring cells are plume cells or non-plume cells.
2  if |N(v)|=0 then
3  |  Mark v as explored.
4  |  Move back to the parent vertex (→next vertex) and directly jump to Line 24.
5  end
6  Communicate with robots to update the state of N(v), i.e., unexplored, under exploration,
     and explored.
7  N(v) ← N(v)\{explored vertices}.
8  if v' ∈ N(v) is under exploration then
9  |  if moving to v' generates a cycle in the tree then
10 |  |  N(v) ← N(v)\{v'}.
11 |  end
12 end
13 if |N(v)| > 1 then
14 |  if |N(v)| > 2 then
15 |  |  Add a dummy edge of length 0 and a dummy vertex in order to keep the tree as a
   |  |    binary tree.
16 |  end
17 |  Split robots into two children as equally as possible.
18 |  Move to one of two children (→next vertex) and mark v as under exploration.
19 else if |N(v)| = 1 then
20 |  Move to the child (→next vertex) and mark v as under exploration.
21 else if |N(v)| = 0 then
22 |  Move back to the parent vertex (→next vertex).
23 end
24 v ←the next vertex.
```

[4] In the algorithm, we use $N(v_i)$ to denote the neighborhood of the i-th vertex such that $N(v_i) = \{v_j \in V | (v_j, v_i) \in E\}$.



(a) Two robots exploring the grid map.



(b) Tree generated from the recursive DFS.

Figure 4.4: Description of tree components. The binary tree consists of a backbone and a finite number of ribs. Each vertex is marked as one of *unexplored*, *under exploration* or *explored*.

4.4.2 Theoretical Analysis

In this section we analyze the proposed Algorithm 4. We start with the upper bound analysis. We then show the lower bound for optimal algorithm, followed by the competitive analysis for the case of grid approximation.

Upper Bound Analysis To analyze the cost of the proposed algorithm, we adapt the reward[5] collecting rule proposed by Higashikawa *et al.* [156] to the case of a translating plume. Note that this rule is not required for implementing the algorithm, but only for analyzing the competitive ratio.

Higashikawa *et al.* [156] define the concept of a backbone and a rib in a tree (shown in Figure 4.4). The backbone is a path that starts from the root vertex and ends at one of the leaf vertices. The rib is a subtree generated by discarding the backbone and edges incident with the backbone from the original tree.

Let $l(e)$ be the length of an edge e. The length of an edge e is 0 if e is dummy edge or 1 otherwise. $L = \sum_{e \in E} l(e)$ be the sum of the total length of all edges in the tree. Note that $L = C - 1$ where the plume consisting of C cells is represented by the tree structure.

Higashikawa *et al.* [156] define two reward functions, each with a total reward of $l(e)$, on every rib edge e and $1 + \lfloor \log R \rfloor$ reward functions, again each with a total reward of $l(e)$, on every backbone edge e. The rewards are collected continuously by the robots following the rules described next:

(1) Only one robot in a group traversing a rib edge in the forward direction for the first time collects a reward. (2) Only one robot in a group traversing a rib edge in the backward direction for the first time collects a reward. (3) Each of the $1 + \lfloor \log R \rfloor$ robots traversing a backbone edge in the forward direction for the first time collects a reward. (4) Only one robot in a group traversing a backbone edge after the first group collects a reward.

Let t_{last} be the time when the last robot reaches a leaf vertex in the tree. Higashikawa *et al.* [156] show that the total sum of rewards collected by all the robots is at least $(1 + \lfloor \log R \rfloor) t_{last}$. This assumes that the robots move at unit speed and the tree is static. In our case, the tree (actually, plume) is moving with a speed of S_p and the robots are moving with a speed of S_r. The reward collection rule does not change in this case. What changes is the total sum of rewards collected in t_{last} time. In our case, the total sum of rewards collected by all the robots will be at least $(S_r - S_p)(1 + \lfloor \log R \rfloor) t_{last}$. The term $(S_r - S_p)$ comes from the lower bound on the relative speeds of the robots and the plume.

Higashikawa *et al.* [156] also show that the total possible reward that the robots can collect is at most $2(L - d_{max}) + (1 + \lfloor \log R \rfloor) d_{max}$. Here d_{max} denotes the distance of the farthest

[5]We use the term *reward function* to replace the term *token* defined in reference [156].

vertex in the tree from the root. Therefore, we have:

$$(S_r - S_p)(1 + \lfloor \log R \rfloor)t_{last} \leq 2(L - d_{max}) + (1 + \lfloor \log R \rfloor)d_{max}. \tag{4.1}$$

We denote the time taken by the proposed algorithm by ALG. We are now ready to state the upper bound on ALG.

Lemma 3 (Upper Bound for Multi-Robot Recursive DFS)**.**

$$\texttt{ALG} \leq \frac{2(C + d_{max}\lfloor \log R \rfloor)}{(S_r - S_p)(1 + \lfloor \log R \rfloor)}. \tag{4.2}$$

Proof. ALG can be upper bounded as follows:

$$\texttt{ALG} \leq t_{last} + \frac{d_{max}}{S_r - S_p}, \tag{4.3}$$

where $\frac{d_{max}}{S_r - S_p}$ is the time taken to traverse the longest length of the backbone when the robot and the plume move away from each other. By using Equation (4.1), we have:

$$t_{last} + \frac{d_{max}}{S_r - S_p} \leq \frac{2L + (\lfloor \log R \rfloor - 1)d_{max}}{(S_r - S_p)(1 + \lfloor \log R \rfloor)} + \frac{d_{max}}{S_r - S_p}, \tag{4.4}$$

$$= \frac{2(C - 1 + d_{max}\lfloor \log R \rfloor)}{(S_r - S_p)(1 + \lfloor \log R \rfloor)}, \tag{4.5}$$

$$\leq \frac{2(C + d_{max}\lfloor \log R \rfloor)}{(S_r - S_p)(1 + \lfloor \log R \rfloor)}. \tag{4.6}$$

Equation (4.5) is obtained by the fact that $L = C - 1$. Removing a negative term from Equation (4.5) completes the proof as Equation (4.6).

□

Corollary 1 (Special Cases)**.** *Upper bounds for the following special cases can be derived from Lemma 3, such as Multi-Robot Static Plume (MRSP), Single Robot Translating Plume (SRTP), and Single Robot Static Plume (SRSP).*

MRSP	**SRTP**	**SRSP**
$\texttt{ALG} \leq \frac{2(C+d_{max}\lfloor \log R \rfloor)}{S_r(1+\lfloor \log R \rfloor)}$	$\texttt{ALG} \leq \frac{2C}{S_r - S_p}$	$\texttt{ALG} \leq \frac{2C}{S_r}$

Table 4.1: Upper bounds of special cases.

Note that the upper bound for MRSP becomes the result from Higashikawa et al. [156] if $S_r = 1$. Also, the upper bound for SRSP is equivalent to Icking et al. [171] if $S_r = 1$.

Proof. Please refer to Appendix A.4. □

Lower Bound Analysis We study the lower bound for the optimal algorithm in order to obtain a competitive ratio. Let $\texttt{OPT}_g^1$ be the time taken by the optimal algorithm to explore a grid map when using a single robot. The lower bound can be constructed as:

$$\texttt{OPT}_g^1 \geq \frac{C-1}{S_r + S_p}. \tag{4.7}$$

We use $\texttt{OPT}_g^R$ to represent the time taken by the optimal algorithm over any grid polygon of a plume region using R robots. Then, the following lemma gives the lower bound for $\texttt{OPT}_g^R$.

Lemma 4 (Lower Bound for Optimal Algorithm)**.**

$$\mathit{OPT}_g^R \geq \frac{C-1}{(S_r + S_p)R}. \tag{4.8}$$

Proof. Please refer to Appendix A.5. □

Theorem 4 (Competitive Ratio over the Grid Polygon)**.** *The competitive ratio of Algorithm 4 for a grid map is:*

$$\begin{aligned} \mathit{ALG} \leq & \frac{2(S_r + S_p)(R + \lfloor \log R \rfloor)}{(S_r - S_p)(1 + \lfloor \log R \rfloor)} \mathit{OPT}_g^R \\ & + \frac{2}{(S_r - S_p)(1 + \lfloor \log R \rfloor)}. \end{aligned} \tag{4.9}$$

Proof. Substituting Equation (4.8) into Equation (4.2) gives:

$$\texttt{ALG} \leq \frac{2((S_r + S_p)R\texttt{OPT}_g^R + 1 + d_{max}\lfloor \log R \rfloor)}{(S_r - S_p)(1 + \lfloor \log R \rfloor)}. \tag{4.10}$$

Since $\frac{d_{max}}{S_r + S_p} \leq \texttt{OPT}_g^R$, it follows:

$$\texttt{ALG} \leq \frac{2(S_r + S_p)(R + \lfloor \log R \rfloor)\texttt{OPT}_g^R + 2}{(S_r - S_p)(1 + \lfloor \log R \rfloor)}. \tag{4.11}$$

□

4.5 Plume Exploration over an Arbitrary Plume Shape

The presented results so far are for a grid map approximation of the plume. In this section, we will relate the bounds obtained for the grid map case to the case of arbitrarily-shaped

plumes. Specifically, we will extend Lemma 3 to apply to a plume region that may have an arbitrary shape.

The algorithm for exploring the plume remains the same. We will still construct a tree that represents a grid map of the plume. The main difference here is that in the previous analysis, we assumed that the boundary of the plume matched the boundary of a grid map exactly. This will no longer hold. Instead, we will explore a grid map that is an *outer* approximation of the plume (Figure 4.5).

We define C_{out}^{ALG} and C_{in}^{ALG} to denote the number of cells in the outer and inner grid approximation by our algorithm, respectively. The outer grid map completely contains the plume whereas the inner grid map lies completely inside the plume. Therefore, the term C in the upper bound (Lemma 3) will now be replaced by C_{out}^{ALG}. However, the C term in the lower bound (Lemma 4) cannot be replaced by C_{in}^{ALG}. This is because C_{in}^{ALG} is defined by the grid imposed by our algorithm. It may be possible to have another grid map (of the same unit side length) that is oriented and/or translated such that it contains fewer than C_{in}^{ALG} cells in the interior. We will first find the relationship between C_{out}^{ALG} and C_{in}^{ALG}. Then, we will relate C_{in}^{ALG} to C_{in}^{BEST} which is the best grid that contains the fewest number of cells completely inside the plume.

By a slight abuse of notation, we interchangeably use C_{out}^{ALG} and C_{in}^{ALG} to also denote the corresponding set of cells (along with denoting the number of cells in the set).

Lemma 5 (Grid Approximation of Arbitrary Plume Shape)**.** *The upper bound on C_{out}^{ALG} for a* fat *polygon (from Definition 1) is given by:*

$$C_{out}^{ALG} \leq 3C_{in}^{ALG} + 6. \tag{4.12}$$

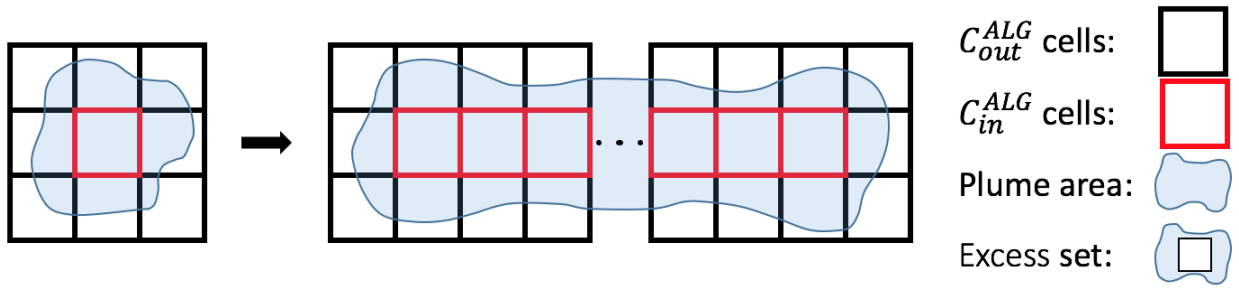


Figure 4.5: Row formation of C_{in}^{ALG} cells as the number of cells changes from 1 to a finite number.

Proof. To prove the lemma, we define an `Excess` set that contains all cells, $p \in P \backslash C_{in}^{ALG}$. That is, `Excess` set contains all cells in C_{out}^{ALG} but not in C_{in}^{ALG}. Therefore, the size of the `Excess` set is equal to $C_{out}^{ALG} - C_{in}^{ALG}$. We prove the lemma in three steps.

`Excess` is maximum if all cells in C_{in}^{ALG} form a convex polygon. If there is a reflex vertex[6] in C_{in}^{ALG}, then the reflex vertex cell does not contribute any cell to the `Excess` set that is already contributed by one of the neighbors of the reflex vertex. Since C_{in}^{ALG} is a grid map, the only convex shape possible is a rectangle.

If the width of the rectangle is equal to one, then each cell in C_{in}^{ALG} contributes two cells that are in the `Excess` set (one above and one below) in addition to three more cells on either end point. This is shown in Figure 4.5. Therefore, the size of `Excess` set is equal to $2C_{in}^{ALG} + 6$.

If the width of the rectangle is more than 1, then each cell will contribute at most one addition cell in the `Excess` set. Therefore, the size of the `Excess` set is less than or equal to $C_{in}^{ALG} + 8$.

The width of the rectangle cannot be less than 1; it violates the *fat* condition for the polygon. Therefore, the maximum possible value for the size of the `Excess` set is $2C_{in}^{ALG} + 6$. By substituting `Excess` with $C_{out}^{ALG} - C_{in}^{ALG}$, we have Equation (4.12). □

The grid corresponding to C_{out}^{ALG} and C_{in}^{ALG} is generated by the proposed algorithm. It is possible that there exists some other grid which has fewer than C_{in}^{ALG} cells completely contained within the plume. It may not be possible to generate this "best" grid due to the nature of online exploration. Nevertheless, we analyze how the relationship between C_{in}^{ALG} and C_{in}^{BEST}. We define C_{in}^{BEST} to denote the fewest number of cells in the inner grid approximation that is completely contained in the plume (and adding any other cell to C_{in}^{BEST} would not allow C_{in}^{BEST} to be completely inside the plume). The relationship is given by:

Lemma 6 (Best Possible Grid-Approximation)**.**

$$C_{in}^{ALG} \leq 6C_{in}^{BEST}. \tag{4.13}$$

Proof. To prove this relationship, it is sufficient to consider any grid approximation (generated by any algorithm) with respect to the best grid approximation.

Figure 4.6 shows a part of grid cells generated by any grid approximation. Each number in the figure corresponds to a different side of grid cells. Let C_{in}^{BEST} be a single grid cell generated from the best grid approximation that overlaps with the central cell $(4, 6, 7, 9)$ without loss of generality. Our observation is that the number of crossings is equal to the number of cells in C_{in}^{ALG} that C_{in}^{BEST} overlaps.

We prove that 7 crossings are impossible. In order to cross more than four edges, C_{in}^{BEST} has to cross all of the (4, 6, 7, 9) edges. In addition, it must cross three of $(1, 2, 3, 5, 8, 10, 11, 12)$ edges. Let us consider the case when edge (1) is crossed. The other cases are symmetric. If edge (1) is crossed, then crossing (5, 8, 10, 11, 12) is impossible since these edges are

[6] A vertex is called the reflex vertex if the external angle formed by two edges incident to this vertex is less than 180°.

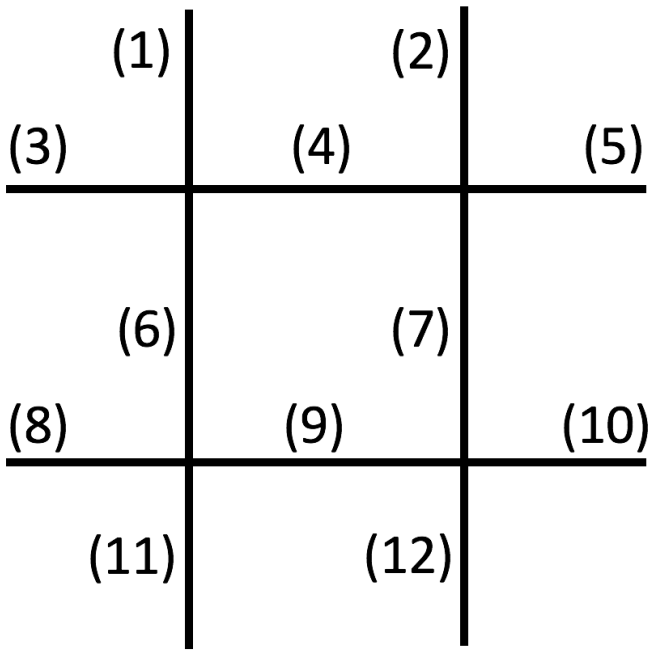


Figure 4.6: A part of grid cell from any grid approximation. Unique number is assigned to a different side of grid cells.

more than a unit distance apart. Only (2) and (3) edges are only available edges to cross more. However, if C_{in}^{BEST} crosses both (2) and (3) edges, the side length of C_{in}^{BEST} becomes greater than 1. Therefore, C_{in}^{BEST} cannot cross more than seven edges. Therefore, $C_{in}^{ALG} \leq 6C_{in}^{BEST}$. □

Finally we give our main result as follows:

Theorem 5 (Competitive Ratio for Arbitrary Plume Shape)**.** *Let* OPT^R *be the time taken by the optimal algorithm over any arbitrary plume shape using* R *robots.*

$$\begin{aligned} \mathit{ALG} \leq & \frac{2(S_r + S_p)(18R + \lfloor \log R \rfloor)}{(S_r - S_p)(1 + \lfloor \log R \rfloor)} \mathit{OPT}^R \\ & + \frac{48}{(S_r - S_p)(1 + \lfloor \log R \rfloor)}. \end{aligned} \tag{4.14}$$

Proof. Although $\mathtt{OPT}^R$ is the cost for any arbitrary plume shape, we can still lower bound this using C_{in}^{BEST} (similar to Lemma 4) as:

$$\mathtt{OPT}^R \geq \frac{C_{in}^{BEST} - 1}{(S_r + S_p)R}. \tag{4.15}$$

Let $(S_r - S_p)(1 + \lfloor \log R \rfloor)$ be $\mathcal{M}$. We can obtain the following inequalities from Lemmas 3, 5, and 6 as follows.

$$\texttt{ALG} \leq \alpha C_{out}^{ALG} + a \leq \beta C_{in}^{ALG} + b \leq \gamma C_{in}^{BEST} + b, \tag{4.16}$$

where $\alpha = \frac{2}{\mathcal{M}}$, $a = \frac{2d_{max}\lfloor \log R \rfloor}{\mathcal{M}}$, $\beta = \frac{6}{\mathcal{M}}$, $b = \frac{12+2d_{max}\lfloor \log R \rfloor}{\mathcal{M}}$, and $\gamma = \frac{36}{\mathcal{M}}$.
Substituting Equation (4.15) into the last inequality of Equation (4.16) and using $\frac{d_{max}}{S_r+S_p} \leq \texttt{OPT}^R$, we have:

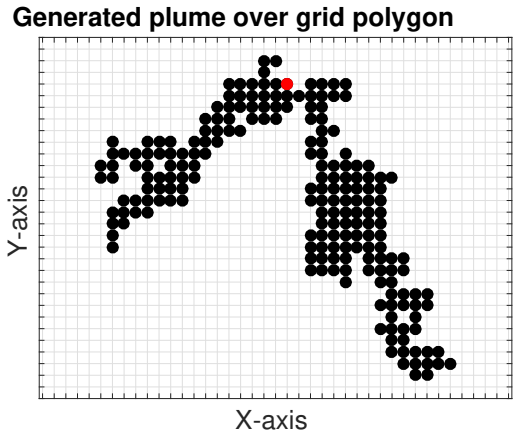
$$\begin{aligned} \texttt{ALG} &\leq \frac{36(S_r + S_p)R}{\mathcal{M}}\texttt{OPT}^R + \frac{48 + 2d_{max}\lfloor \log R \rfloor}{\mathcal{M}}, \\ &\leq \frac{2(S_r + S_p)(18R + \lfloor \log R \rfloor)}{\mathcal{M}}\texttt{OPT}^R + \frac{48}{\mathcal{M}}. \end{aligned} \tag{4.17}$$

□

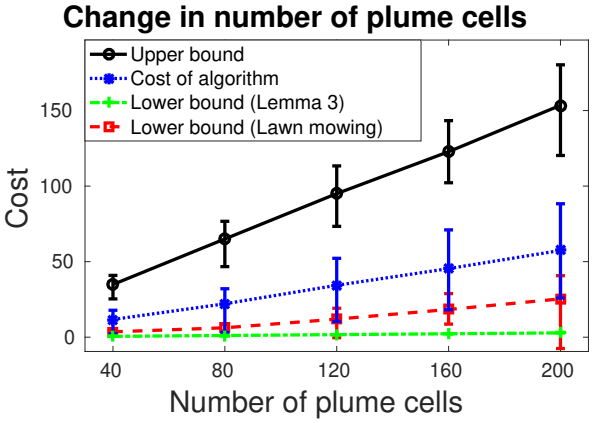
4.6 Simulation

We empirically evaluated our algorithm using MATLAB simulations. Specifically, we verified the performance of the proposed recursive DFS for the grid map approximation of the plume (Theorem 4).

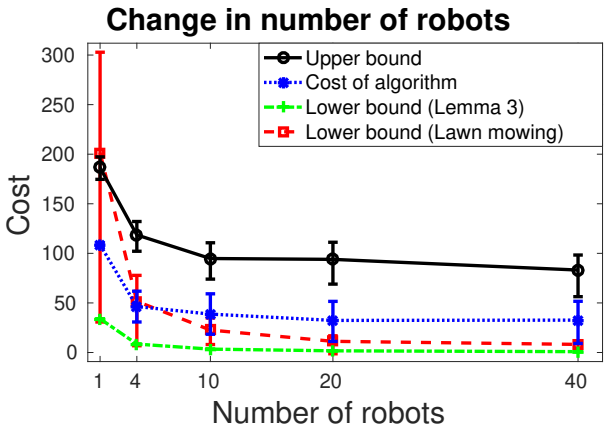
We randomly generated a set of plume grid maps. We randomly chose one of four directions (*i.e.*, north, south, east, and west) for the direction of translation of the plume. The assumption that the moving direction of the plume is known a priori enables robots to align the axis of the grid map with that direction although robots still do not know about best approximation for grid map. Figure 4.7 (a) shows an example of the generated plume that consists of 200 cells. We measured the cost of our algorithm as well as the upper and lower bounds by changing the number of plume cells, the number of robots, and the speed ratio between the robot and the plume. The lower bound in our analysis is given in Lemma 4. In deriving the lower bound, we assume that the robots always travel opposite to the plume, thereby yielding the lowest possible time. In practice, the robots will not always travel in this best possible direction. Therefore, we find another lower bound using a baseline lawn mower algorithm. We assume that this baseline algorithm knows the tightest rectangle that is guaranteed to contain the plume. The axis of the rectangle is aligned with the direction of translation of the plume. Given R robots, we split this rectangle into R smaller ones. We can split this rectangle either along its length or its breadth. The exploration time will be different in each case. We find the time required to cover each smaller rectangle using a lawn mower strategy in both cases and take the lower one. We also ignore the time required for the robots to go from the starting position to the smaller rectangles. This is clearly a lower bound for any online strategy that does not know the size of the plume. Nevertheless, we find that the exploration time for our algorithm is comparable to this lower bound.



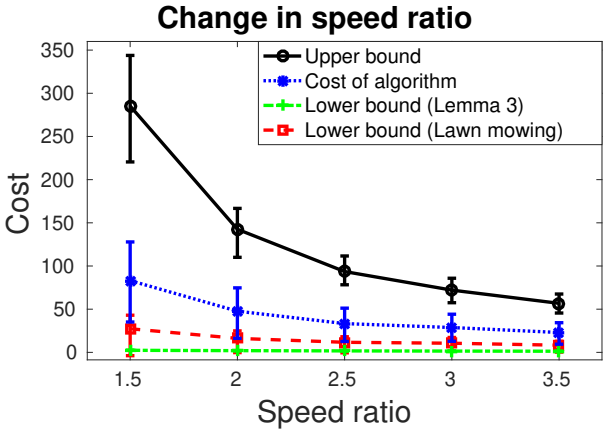
(a) Example of the randomly generated plume over grid cells. The red dot represents the starting vertex for robots.



(b) Plot of the cost when changing the number of plume cells.



(c) Plot of the cost when changing the number of robots.



(d) Plot of the cost when changing the speed ratio between the robot and the plume.

Figure 4.7: Simulation results. We fixed the number of plume cells, the number of robots, the speed ratio as 120, 20, 2.5, respectively, when the corresponding variable was not a subject to be changed. We ran 100 trials for each case. Each case is plotted as mean, maximum and minimum values from 100 trials.

Each case was obtained from 100 trials (see Figures 4.7 (b–d)). Figure 4.7 (b) shows that the expected exploration time for all cases is proportional to the number of plume cells. The difference between the maximum and minimum costs also becomes larger as more plume cells are to be explored. Figure 4.7 (c) plots the exploration time when changing the number of robots. The exploration time of our algorithm and the lower bound decrease as the number of robots increases. Unlike these, the upper bound does not show a steady decreasing tendency because randomly generated plume cells affect d_{max}. Figure 4.7 (d) shows the exploration time when changing the speed ratio between the robot and the plume, *i.e.*, $\frac{S_r}{S_p}$. The exploration time for our algorithm and the upper bound decrease as the speed ratio increases.

The simulation results verify the theoretical upper and lower bounds determined by our analysis. In addition, they demonstrate that the practical performance of our algorithm is better than that indicated by the upper bounds.

4.7 Field Experiment

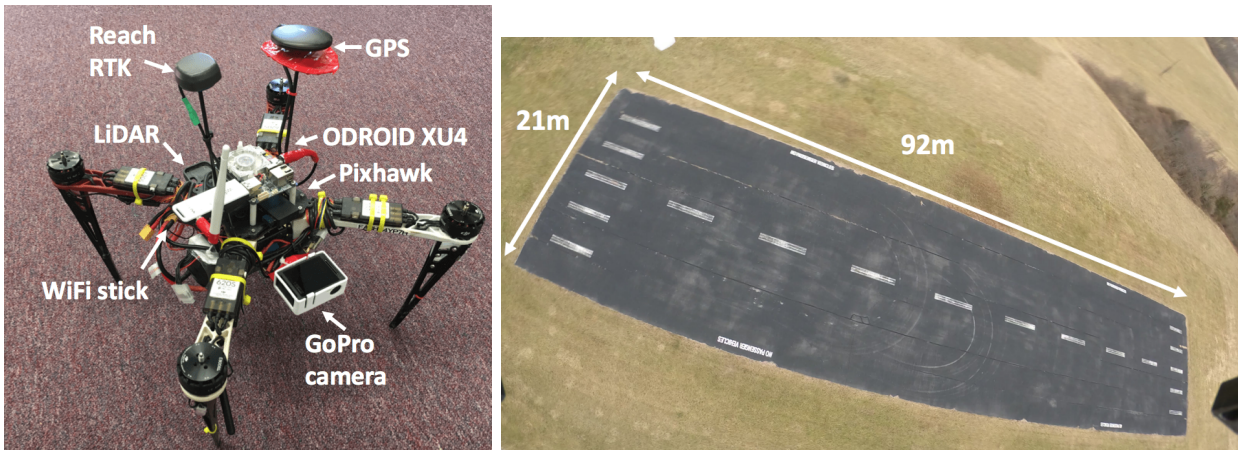
In this section, we conduct proof-of-concept experiments using a single UAV equipped with a downward-facing camera to monitor a stationary region of unknown size and shape. In a practical implementation, there are a number of design choices that must be made (*e.g.*, what altitude to fly? how to convert the camera images into cell measurements? how to deal with erroneous sensor measurements?). In this section, we answer these questions in the context of our system.

Our environment to be mapped is a $92m \times 21m$ long runway which serves as a proxy of the plume. Figure 4.8 shows hardware details of the UAV and the snapshot of the environment. The UAV has ODROID-XU4 single-board computer which runs Ubuntu 16.04 with ROS Kinetic [79]. The onboard software controls the UAV, communicates with GoPro HERO4 camera over WiFi to read sensor information, and detects the runway.

Our planning strategy consists of two modes: (1) lawn mowing and (2) the single-robot version of Algorithm 4. The input to the lawn-mowing mode is a bounding box that is guaranteed to contain at least some part of the runway. In the lawn-mowing mode, the UAV sweeps the bounding box. As soon as the UAV observes a part of the runway, the UAV switches to the recursive DFS algorithm.

Once in the recursive DFS mode, we discretize the environment into grid cells. The grid is aligned with the UTM coordinates [185]. The origin of the grid is placed at the starting location of the UAV. Note that the grid is not aligned with the runway (Figure 4.8 (b)) and the location, shape, and size of the runway are not given to the UAV a priori.

Each cell is of size $4m \times 4m$. We use the onboard GoPro images to determine if a cell corresponds to the runway. Each image is divided into 3×3 regions (Figure 4.9). The size



(a) UAV platform. (b) Runway ($92m \times 21m$) to be explored.

Figure 4.8: Experimental setting.

of a grid map cell ($4m \times 4m$) corresponds to the footprint of the center region in the image when flying at an altitude of $12m$. The center and the top, left, bottom, and right regions in the image (refer Figure 4.9) are used to determine if the current cell and its four neighbors in the grid map contain the runway or not.

Each image is first converted into grayscale and thresholded (at the intensity value of 150). Then, the thresholded image is dilated (7 times) so that the gaps in the grass region are filled in (refer the left image in Figure 4.9). Consequently, the entire grass region becomes filled with white pixels and the runway region mostly comprises black pixels. Then the percentage of black pixels in the individual regions is calculated, for each highlighted neighbouring region in the figure. Finally, we produce a binary classification result based on if this percentage is above or below a threshold.

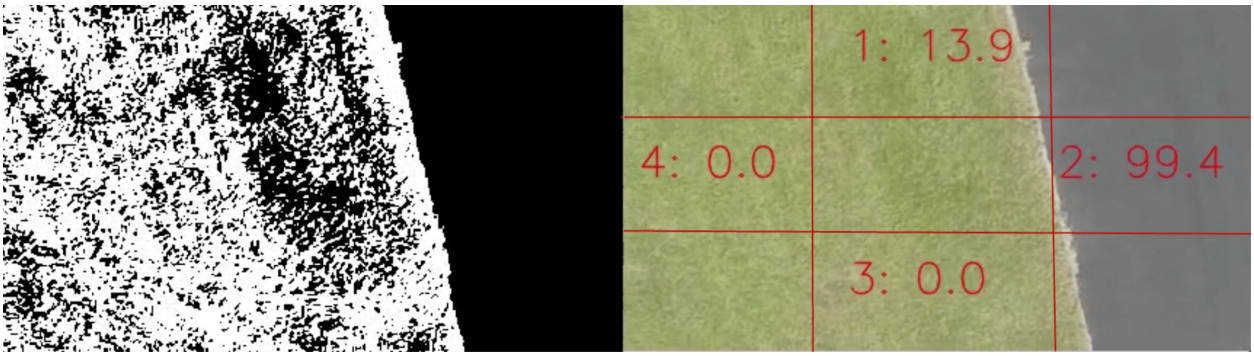


Figure 4.9: Example of our sensing model using a single image that contains both the runway and the grass region. The left image is the thresholded image. The right image shows the detection result indicating the percentage of black pixel values printed on the grid cells (colored in red in four neighboring cells).

Our classifier may not give a correct detection result due to a number of reasons. First, it may produce false positives and false negatives. Second, the detection result is sensitive to the intensity threshold value we set. Third, even if the UAV is on top of the current cell, the camera may point to a wrong cell due to pitch and roll used to counter wind disturbance and imperfect flight controller. Lastly, the change in the sunlight condition might produce noisy measurements. Therefore, instead of relying on a single image, we use five images per cell. If three of these images mark a cell as containing the runway, we treat it as a positive detection. When mapping a region, we care more about the completeness of exploration rather than the efficiency. Therefore, we make the following conservative design choices. If a cell is marked as containing the runway, it will never be reversed later on, even if a future measurement taken from some other location yields the opposite detection. On the other hand, if a cell is marked as not containing the runway and a subsequent measurement suggests otherwise, we will mark it as a positive runway detection.

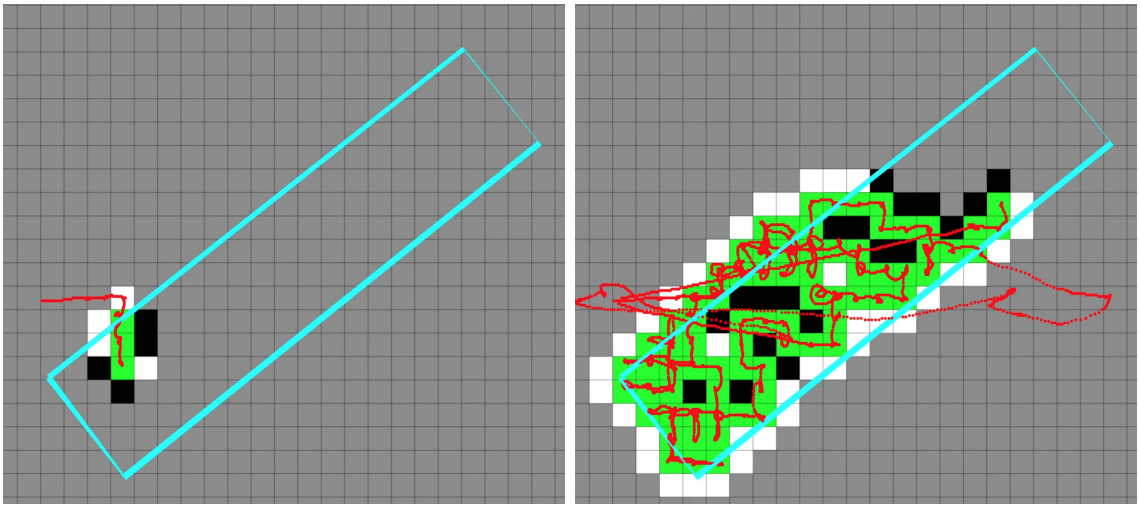
The measured flight time of the UAV with a single battery is approximately 12 minutes, which is not enough for finishing the exploration task. Therefore, we designed our software in a way to keep track of its previous computation even after replacing the battery. To do that, the software stores the information of the generated tree (*i.e.*, the status of vertices and the tree structure) and feeds that to the next flight in order for the UAV to start from the vertex visited last in the previous flight.

Figure 4.10 shows the result of the runway exploration experiment. We flew the UAV at a nominal speed of $1m/s$ and the ambient wind speed was approximately $3.6m/s$. The total flight time was 1 hour and 13 minutes and consisted of six battery replacements. The final tree contained 143 vertices.

The final grid map overlaid on Google Earth is shown in Figure 4.11. The percentage of the area of intersection between the ground-truth and the final map, normalized by the area of their union, was 75.7%.

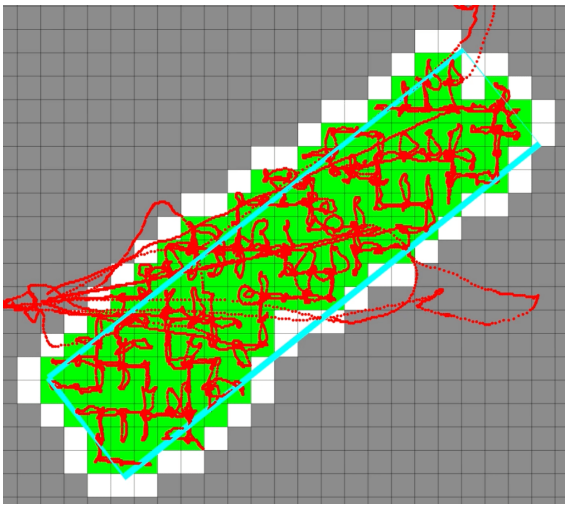
We also compute the false-positive and false-negative detection rates. Out of the total 483 detections, 27 were false positives and 53 were false negatives. Note that all but two cells that gave false-negative detections, eventually gave a positive detection. As a result, the final map (Figure 4.11) has only two cells that are incorrectly mapped as not being part of the runway. The cells that are just outside the boundary, however, are incorrectly mapped as being part of the runway. This is likely because of our conservative exploration choice of marking a cell as containing the runway even if a single detection is positive.

In summary, we present how our algorithm handles real-world issues and can be implemented on actual robots. We show the efficacy of the proposed scheme through field experiments.



(a) Snapshot at time 2 minutes.

(b) Snapshot at time 26 minutes and 40 seconds.



(c) Snapshot at time 1 hour and 13 minutes.

Figure 4.10: Experimental result. The blue square line represents the ground truth of the runway to be explored that is unknown initially. The red line denotes the trajectory of the UAV. The size of each cell is $4m \times 4m$ and gray, white, black and green colors represent unknown, non-runway, unexplored runway and explored cells, respectively. The total flight time taken to completely explore the entire runway was 1 hour and 13 minutes, consisting of six battery replacements. The video is available at: `https://youtu.be/RK_sMXXjtIo`.

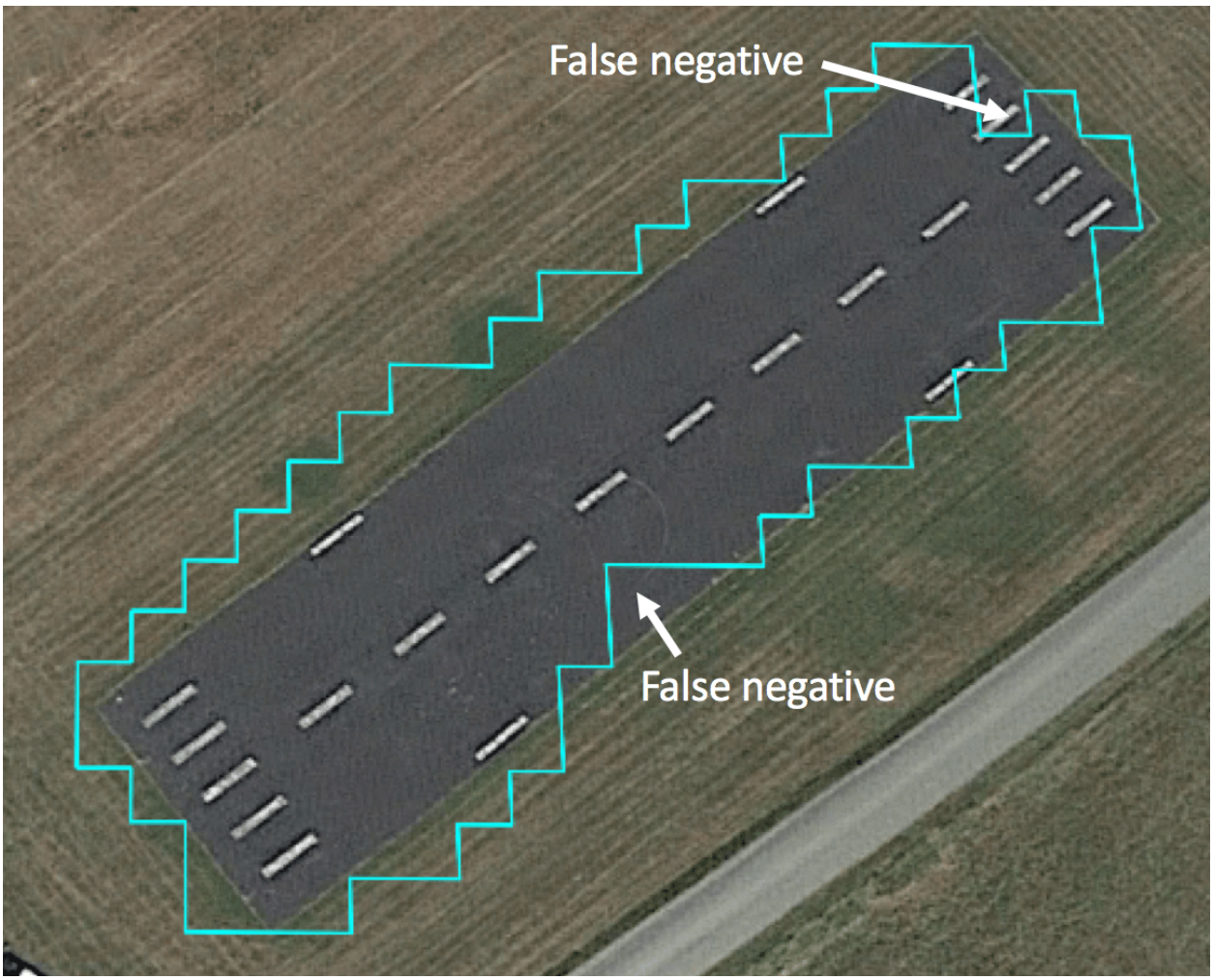


Figure 4.11: Resultant boundary of the runway region (*i.e.*, the boundary of the grid map) plotted on Google Earth colored in blue.

4.8 Conclusion

We propose a recursive DFS algorithm for a team of aerial robots to explore a translating plume without knowing its shape and size. We present two approaches for the given problem where the first approximates the plume to map to the grid whereas the second considers any arbitrary shape of plume as long as it is *fat*. Both approaches are competitive with respect to the optimal algorithm. We demonstrate the performance of our algorithm through proof-of-concept deployment to map a stationary plume.

One of the practical concerns not modeled by our system is that robots, especially UAVs, have limited battery lifetime. As such, we would like to devise algorithms that can map the plume subject to the limited battery lifetime constraint. In particular, in our formulation, we restrict the UAV to fly at a fixed altitude. However, one may be able to extend the coverage range by flying at higher altitudes. An interesting and relevant extension of this work would be to plan in 3D space as opposed to just 2D.

Chapter 5

Hotspot Identification in Limited Time

5.1 Introduction

Robots are tasked with monitoring unknown environments but their limited sensing capabilities restrict them from observing the entire environment at once. It is thus of importance to actively explore the environment and learn the underlying characteristics of the environment.

Given the limited resources (e.g., operation time and fuel), the robot must carefully choose its actions so as to better estimate and predict states of the environment.

This work is motivated by one such problem of monitoring hazardous plumes of pollutants released in water bodies, such as oil spills. The overarching project [186] is a collaboration with microbiologists interested in studying the transport of aerosolized pollutants from water bodies [187]. Figure 5.1 demonstrates our team of an Unmanned Aerial Vehicle (UAV) and Unmanned Surface Vehicle (USV) cooperatively monitoring a lake. The UAV observes regions in the lake with a downward-facing camera using which it can map out the concentration of the (visible) hazardous agent. To analyze the characteristics of toxic particulates just mapping it with UAVs is not enough, instead we need physical samples that can be analyzed ex situ. The UAV can direct the USV to the location with the highest concentration. The USV can then collect physical specimen at that location. Thus, the UAV acts as an explorer whereas the USV acts as a sampler. In this chapter, our focus is on planning strategies for the UAV to find the location with the highest concentration in the limited battery lifetime.

Figure 5.1: A UAV exploring the environment to search for the plume in a lake [3].

While the aforementioned application motivates our work, the problem we study is general enough to apply to many settings where UAVs with downward-facing cameras can be used for finding hotspots of unknown, spatially varying function. There exists many practical applications in the environmental monitoring literature, such as precision agriculture [10], wildlife habitat monitoring [7], plume tracking [3], where such a problem arises.

Due to noisy measurements, the more number of measurements from one sensing location, the more accurate the estimate. However, spending too much time at one location is not beneficial since the UAV has a limited budget. This is the famous exploration-exploitation dilemma which is studied under the Multi-Armed Bandit (MAB) setting (we introduce several variants of MAB in Section 5.2).

Our problem poses two major challenges that cannot be handled by existing MAB approaches. First, as the UAV can change its flight altitude (or, equivalently, the size of the camera footprint), we need different ways of evaluating the performance for sensing locations at different altitudes. Second, there is no clear interpretation between the camera image and measurements, and moreover, we need to define how good a sensing location is with respect to the other sensing location.

Our approach is based on the Gaussian Process Upper Confidence Bound (GP-UCB) algorithm proposed by Srinivas *et al.* [188] that adopts the Gaussian Process (GP) to resolve spatially-correlated sensing locations. Our algorithm extends this to deal with the above challenges, to be applicable in a 3D environment. We then propose several heuristic planning strategies to qualitatively evaluate the proposed algorithm.

The contributions of this chapter are as follows: (1) We propose an algorithm adapted from the MAB framework for environmental monitoring in a 3D environment with a camera sensor

whose noise depends on the altitude. (2) We empirically compare the performance of the proposed heuristic strategies through extensive simulations and real-world data.

Our validation is based on real-world data we gathered from the field using a UAV. We empirically show the performance of several planning algorithms through simulations. These results demonstrate the effectiveness of the future variance that we propose in the MAB framework which reflects various flight altitudes of the UAV.

The rest of this chapter is organized as follows. We begin by introducing the related work in Section 5.2. We describe the problem setup in Section 5.3. Our proposed algorithm is presented in Section 5.4. We present results from simulations in Section 5.6 before concluding with a discussion of future work in Section 5.7.

5.2 Related Work

There have been various algorithms proposed for environmental monitoring. In this section, we briefly introduce some of recent works. For survey results, see [148].

Offline algorithms compute a trajectory for the robot before operation. Such coverage planning algorithms [164] need to know the environment a priori. Either a predefined trajectory (*e.g.*, boustrophedon path [189]) or tree search algorithms, such as depth-first search, can be utilized to completely cover the environment.

If only partial information about the environment is given, the robot must be able to adaptively cope with unexpected situations and uncertain environment online. Many decision-making challenges arise from this context. For environmental monitoring, the robot has to decide what samples to collect if samples are available only for specific time [190, 191], and where to take measurements constrained to a limited budget [3].

Information gathering is also an important task for learning the environment. In particular, entropy reduction [192, 193], or mutual information maximization [194] are two popular information metrics. However, these approaches are not suitable for identifying hotspots in the environment as they are interested in exploring the environment rather than exploiting the current knowledge of learned information.

Bayesian optimization [195] addresses this issue by balancing between exploration and exploitation. Previous works regarding MAB problems proposed algorithms to identify a near-optimal arm within a given budget. They also proved some guarantees on the regret bound (*i.e.*, a quantity implying how suboptimal reward the robot obtains).

There exist variants of MAB which consider switching costs. These variants can be useful in robotics because a traveling cost is a bottleneck for robots. Reverdy *et al.* [196] employed a block allocation scheme that restrains the robot from switching to other arm frequently. Guha and Munagala [197] developed an algorithm and related with the orienteering

problem [198] to minimize traveling costs while maximizing collected rewards. However, their algorithm cannot handle spatially correlated arms. Their approach and Audibert and Bubeck [199] used a terminal metric for evaluating their algorithms, which is considered in this chapter.

5.3 Problem Description

Let $\mathcal{E} \subseteq \mathbb{R}^2$ represent the 2D environment that contains the contamination. The intensity of the contamination varies within $\mathcal{E}$ but we assume that the intensity does not change during the course of the robot deployment.

The goal of this chapter is to find a point in the environment (denoted by $\mathbf{x}^{OPT} \in \mathcal{E}$) that has the highest intensity of contamination. We deploy a UAV to explore the environment, constrained by limited time allowed for searching for this particular point. We denote the total allowed time by budget B. We compute the time spent as the combination of sensing time T_S and traveling time T_T. While T_S is a fixed constant value, T_T is the time measured for traveling from one location to the next, assuming that the UAV moves with unit speed. The highest contamination point inferred by the UAV will then be visited by the USV to physically collect a sample to be analyzed.

The UAV is mounted with a downward-facing camera to observe the contamination in the environment. We denote the state of the UAV at timestep k by $\mathbf{v}(k) \in \mathbb{R}^3$. At every sensing location, the UAV collects an image of the environment from a limited FOV camera sensor.

Let $f(\cdot)$ be the true intensity function. Each image gives a noisy observation of the true intensity function. We assume that the UAV can use image processing and other estimation techniques to fuse the data obtained from all images and form an estimate of $f(\mathbf{x})$. Based on its estimate, the UAV must determine, $\mathbf{x}^{ALG}$, its own estimate of the point of highest intensity at the end of its path. We wish to find strategies that will minimize $f(\mathbf{x}^{OPT}) - f(\mathbf{x}^{ALG})$. Since the UAV has a limited budget B, however, the number of images the UAV can collect is also limited. Thus, sensing locations must be carefully chosen as the information on the unknown intensity is revealed online.

Without learning the unknown intensity, the UAV cannot identify the highest contamination point. Thus, the UAV explores unvisited regions to gather more information while carefully spending its given budget.

To sum up, we propose the following problem.

Problem 4. *(**Hotspot Identification**)*

Let $f(x)$ be the true intensity function and $\mathbf{x}^{OPT}$ be the location where $f(\cdot)$ achieves the global maxima. Given the starting position of the UAV, $\mathbf{v}(0)$, find a strategy that produces as output: (1) a sequence of sensing locations, $\mathbf{v}(1), \mathbf{v}(2), ..., \mathbf{v}(k)$; and (2) a point, $\mathbf{x}^{ALG} \in \mathcal{E}$, to

minimize $f(\mathbf{x}^{OPT}) - f(\mathbf{x}^{ALG})$ *subject to the constraint that* $\sum_{k=1} \left\{ T_T\big(\mathbf{v}(k), \mathbf{v}(k+1)\big) + T_S \right\} \leq B$.

5.4 Algorithm

5.4.1 Arm Locations

We call sensing locations as *arm locations* since that is the standard terminology in the multi-armed bandit literature. We create a 3D grid where each grid location is an arm (Figure 5.2). We denote the arm locations by $\mathrm{A} = \{\mathbf{a}_1, ..., \mathbf{a}_i, ..., \mathbf{a}_N\} \subseteq \mathbb{R}^3$ where N is the total number of arm locations. We place the grid such that every point in the environment will be in the FOV of at least one arm at the lowest altitude.

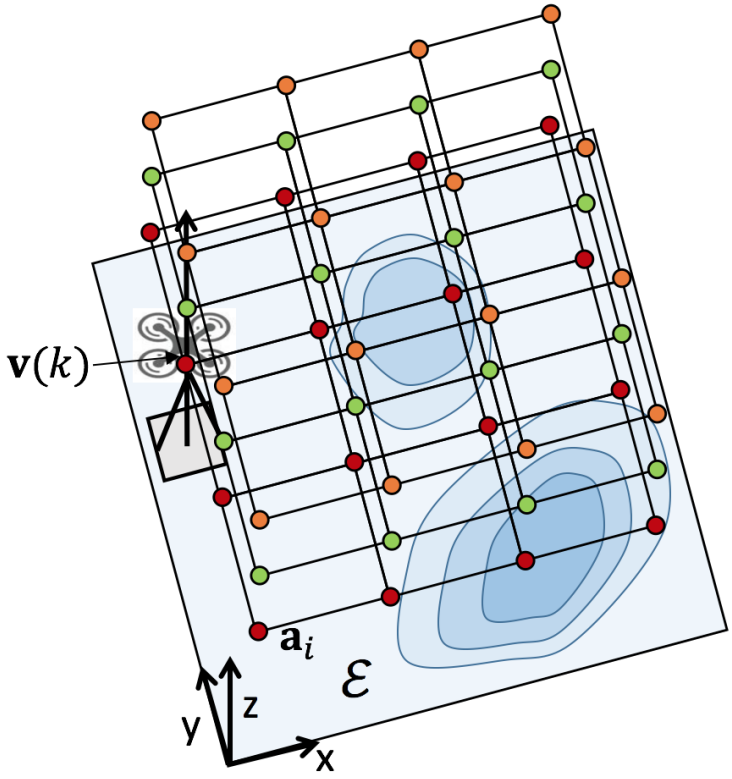


Figure 5.2: Illustration of our problem setting where arms at three different altitudes are placed in a given environment $\mathcal{E}$.

5.4.2 Sensor Model

At each arm location, the UAV obtains a camera image from the downward-facing camera. The camera footprint will be larger from higher altitudes.

A single image measurement yields M pixel measurements. By using the camera projection equations with the known intrinsic, extrinsic parameters and known camera height, we can compute points in $\mathcal{E}$ that corresponds to M measurements [200].

5.4.3 Reward

We assume that there is a function that takes as input an image and produces a noisy estimate of the true intensity function for every pixel. The reward is a function of concentration of the contamination in the area covered by that pixel. In the simulation section, we describe the reward function we use.

We denote the reward value of the j-th measurement by $y_j \in \mathbb{R}_{\geq 0}$. We assume that the robot obtains noisy estimates of the true intensity function:

$$y_j = f(\mathbf{x}_j) + \epsilon, \tag{5.1}$$

where $\epsilon \sim \mathcal{N}(0, \sigma_N^2)$ is additive i.i.d. Gaussian noise. The point $\mathbf{x}_j$ corresponds to the j-th measurement position. Note again that the larger y represents a higher contamination.

At each timestep, the UAV obtains M measurements where one pixel corresponds to one measurement. This measurement is the noisy version of the true intensity function at the location at the center of the footprint of the pixel on the ground. Therefore, at timestep k, we have a collection of kM measurements, which we denote by Y. The corresponding kM measurement locations are denoted by X. Note that these kM locations denoted by X are not the same as the k sensing locations where the k images are obtained from.

Notice that the footprint of a pixel is not a point but an area, as shown in Figure 5.3. The size of the footprint of a pixel gets larger as the altitude of the UAV becomes higher. The larger footprint size would not give an accurate reward value from a particular point $\mathbf{x}$ compared to the smaller footprint size. This means that the measurement noise variance σ_N^2 is proportional to the altitude of the UAV. Instead of using a fixed σ_N^2, we use $\sigma_N^2(\mathbf{a}_i)$ that can vary depending on the arm altitude. We assume that the proportion of $\sigma_N^2(\mathbf{a}_i)$ with respect to the arm altitude is linear.

5.4.4 GP

This subsection discusses about how the UAV proceeds with obtained measurements and maintains its belief of the contamination density distribution in the environment.

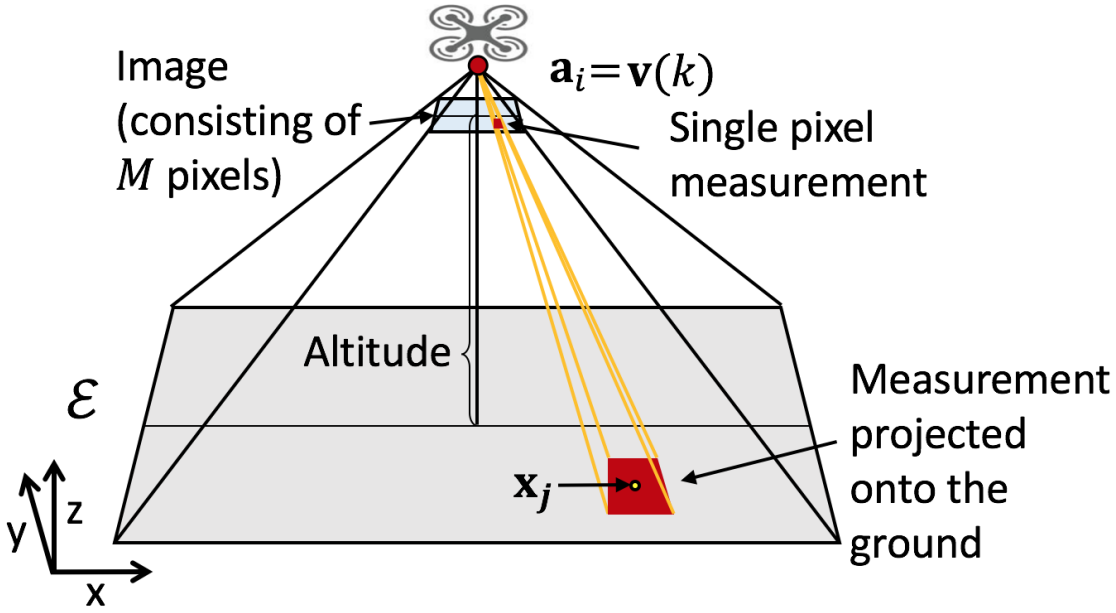


Figure 5.3: Description of how the obtained image can be converted into measurements from an arm location.

We use GP to represent the belief over time (*i.e.*, $f \sim \mathcal{GP}$). Unlike conventional GP, however, measurements we obtain have various noise levels based on which altitude the measurements were observed. This is related with the noise variance $\sigma_N^2(\mathbf{a}_i)$ (one of hyperparameters of the GP), as explained in Section 5.4.2. Therefore, we adapt the GP to our case that can take into account various noise levels.

When we learn the hyperparameters (length-scale, signal variance, and noise variance) offline by using the log marginal likelihood, $\sigma_N^2(\mathbf{a}_i)$ is larger for data obtained from higher altitude than the one from lower altitude. We define the noise variance matrix given by:

$$Q(\mathrm{X}) = \begin{bmatrix} \sigma_N^2(\mathbf{a}_i) & & 0 \\ & \ddots & \\ 0 & & \sigma_N^2(\mathbf{a}_i) \end{bmatrix}, \tag{5.2}$$

which is a diagonal matrix. Each diagonal element of $Q(\mathrm{X})$, $\sigma_N^2(\mathbf{a}_i)$, has a unique value based on the altitude of $\mathbf{a}_i$, but we set $\sigma_N^2(\mathbf{a}_i)$ for arms at the same altitude to have the same value. The dimension of $Q(\mathrm{X})$ corresponds to the total number of measurements accumulated during the flight, *i.e.*, $Q(\mathrm{X}) \subseteq \mathbb{R}^{kM \times kM}$ at timestep k.

Given a 2D grid of the environment, let L_i be the number of grid cells that fall in the camera footprint of arm $\mathbf{a}_i$ which we call test points. We denote the set of L_i test points by $\mathrm{X}_i^* = \{\mathbf{x}_1^*, ..., \mathbf{x}_{L_i}^*\} \subset \mathcal{E}$. As we have N arms, the total number of all test points from all arms becomes $L = \sum_{i=1}^{N} L_i$. The set of all test points from all arms is $\mathrm{X}^* = \cup_{i=1}^{N} \mathrm{X}_i^* =$

$\{\mathbf{x}_1^*, ..., \mathbf{x}_l^*, ..., \mathbf{x}_L^*\} \subset \mathcal{E}$. The prior of the GP then becomes:

$$\begin{bmatrix} \mathrm{Y} \\ f_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathcal{K}(\mathrm{X},\mathrm{X}) + Q(\mathrm{X}) & \mathcal{K}(\mathrm{X},\mathrm{X}^*) \\ \mathcal{K}(\mathrm{X}^*,\mathrm{X}) & \mathcal{K}(\mathrm{X}^*,\mathrm{X}^*) \end{bmatrix}\right), \tag{5.3}$$

where $\mathcal{K}(\cdot,\cdot)$ is a covariance function (or kernel). In this work, we use the squared exponential covariance function [201]. The predictive mean and covariance are:

$$\mu = \mathcal{K}(\mathrm{X}^*,\mathrm{X})\Big(\mathcal{K}(\mathrm{X},\mathrm{X}) + Q(\mathrm{X})\Big)^{-1}\mathrm{Y}, \tag{5.4}$$

$$P = \mathcal{K}(\mathrm{X}^*,\mathrm{X}^*) - \mathcal{K}(\mathrm{X}^*,\mathrm{X})\Big(\mathcal{K}(\mathrm{X},\mathrm{X}) + Q(\mathrm{X})\Big)^{-1}\mathcal{K}(\mathrm{X},\mathrm{X}^*). \tag{5.5}$$

5.4.5 Pseudo-code

The pseudo-code in Algorithm 5 demonstrates the entire steps of hotspot identification. The UAV starting from an initial position $\mathbf{v}(0) = \mathbf{a}_i(0)$ is given a budget B and initializes GP with zero mean. At every timestep k, the UAV updates the GP mean and variance functions (Lines 10 and 13) by using measurements collected up to and including the previous timestep (Line 20). To compute the objective function (Line 17), we calculate the average mean and the average variance (Lines 14 and 15) as the number of test points is not 1. If the UAV spends the time more than B (Line 2), the algorithm terminates and the UAV finds $\mathbf{x}^{ALG}$ from learned GP for the USV to sample from this location.

5.5 Planning Strategies

In this section, we present several heuristic planning strategies that can be applied in Algorithm 5. Our main contribution is the strategy that is presented in Section 5.5.2. However, before we present our strategy, we explain the standard GP-UCB algorithm and then present the two changes we make to the algorithm (future variance and dynamic windows).

We present the objective function (Line 17 of Algorithm 5) that is decision-making of the UAV to decide which arm to visit next. The weighted combination of mean and variance is the conventional functional form in MAB, however, β values proposed by the UCB algorithms as well as the GP-UCB algorithm [188] cannot directly be applied to our problem according to challenges explained in Section 5.1. Thus, we propose the following exponential functional form for β:

$$\beta = \gamma e^{(\lambda k)}, \tag{5.6}$$

where γ and λ are hyperparameters that control the decreasing rate or the increasing rate of β. We tune them offline through simulations.

Algorithm 5: GP-UCB with Future Variance

Input : Initial position of the UAV: $\mathbf{v}(0) = \mathbf{a}_i(0) \in$A, GP prior: $\mu(0) = 0$, $\sigma(0)$, $\mathcal{K}$, budget: B, arm set: $\mathrm{A} = \{\mathbf{a}_1, ..., \mathbf{a}_i, ..., \mathbf{a}_N\}$, test set: $\mathrm{X}^* = \{\mathbf{x}_1^*, ..., \mathbf{x}_l^*, ..., \mathbf{x}_L^*\}$, measurement set: $\mathrm{X} = \emptyset$, reward value set: $\mathrm{Y} = \emptyset$.

for $k = 1, 2, ...$ **do**
  if $T_T\big(\mathbf{v}(k), \mathbf{v}(k\text{-}1)\big) + T_S k > B$ **then**
    $\mathbf{x}^{ALG} = \mathrm{argmax}_{\mathbf{x}}\, \mu$.
    `TerminateExploration`$\Big(\mathbf{x}^{ALG}\Big)$.
  end
  if $k = 1$ **then**
    Choose a nearest arm $\mathbf{a}_i(1) = \mathbf{v}(1)$.
  end
  else
    $\mu = \mathcal{K}(\mathrm{X}^*, \mathrm{X})\Big(\mathcal{K}(\mathrm{X}, \mathrm{X}) + Q(\mathrm{X})\Big)^{-1}\mathrm{Y}$.
    for $i = 1, 2, ..., N$ **do**
      $\mathrm{X}' = \mathrm{X} \cup \mathbf{x}_{I_i}^*$.
      $P|\mathbf{a}_i = \mathcal{K}(\mathbf{x}_{I_i}^*, \mathbf{x}_{I_i}^*) - \mathcal{K}(\mathbf{x}_{I_i}^*, \mathrm{X}')\Big(\mathcal{K}(\mathrm{X}', \mathrm{X}') + Q(\mathrm{X}')\Big)^{-1}\mathcal{K}(\mathrm{X}', \mathbf{x}_{I_i}^*)$ where $P \ni \sigma_l^2\ \forall l \in \mathrm{I}_i$.
      $\overline{\mu}_i = \big(\sum_{l \in \mathrm{I}_i} \mu_l\big)/L_i$.
      $\overline{\sigma}_i^2 = \big(\sum_{l \in \mathrm{I}_i} \sigma_l^2\big)/L_i^2$.
    end
    $\mathbf{a}_i(k) = \mathrm{argmax}_i \Big\{\overline{\mu}_i + \beta\overline{\sigma}_i\Big\}$.
  end
  $\mathbf{a}_i(k) \leftarrow$ `MoveUAV`$\Big(\mathbf{v}(k\text{-}1)\Big)$.
  $\big\{\mathrm{X}, \mathrm{Y}\big\} \leftarrow \big\{\mathrm{X}, \mathrm{Y}\big\} \cup$ `GetMeasurements`$\Big(\mathbf{a}_i(k)\Big)$.
end

5.5.1 GP-UCB

GP-UCB we use is originated from Srinivas *et al.* [188] but extended to take into account our problem setting. Moreover, with Equations (5.4) and (5.5), GP-UCB can handle various noise levels for collected measurements. Algorithm 5 without having Line 12 is this GP-UCB version. Taking out Line 12 implies that the variance of test points is estimated without worrying about where the current measurements are observed. Since the flight altitude of the UAV affects the sensing credibility, we tackle this in the next strategy.

5.5.2 GP-UCB with Future Variance

Although the predictive covariance P in Equation (5.5) considers various noise levels for the training points (*i.e.*, accumulated measurements), this aspect is not addressed for predicting the covariance at test points in Equation (5.5). That is, we predict the covariance at a test point as if we were to observe the test point from the altitude of the corresponding arm. We achieve this by adopting conditional variance, *i.e.*, $P|\mathrm{A}$.

Let I_i be the index set containing indices of test points from X^* which can be observed from arm $\mathbf{a}_i$. Thus, the set of test points by arm i is represented by $\mathbf{x}^*_{I_i}$. We decompose the conditional variance $P|\mathrm{A}$ into N conditional variances, *i.e.*, $P|\mathbf{a}_i$ for each arm. Then, the test points and the training points become $\mathbf{x}^*_{I_i}$ and $\mathrm{X}' = \mathrm{X} \cup \mathbf{x}^*_{I_i}$, respectively, for arm i. The decomposed conditional variance for the i-th arm can be computed as:

$$\begin{aligned} P|\mathbf{a}_i =& \mathcal{K}(\mathbf{x}^*_{I_i}, \mathbf{x}^*_{I_i}) - \mathcal{K}(\mathbf{x}^*_{I_i}, \mathrm{X}')\Big(\mathcal{K}(\mathrm{X}', \mathrm{X}') + Q(\mathrm{X}')\Big)^{-1} \\ & \times \mathcal{K}(\mathrm{X}', \mathbf{x}^*_{I_i}). \end{aligned} \tag{5.7}$$

We call this planning strategy as GP-UCB with future variance. Similarly, we call the standard GP-UCB as GP-UCB with current variance.

5.5.3 DWA GP-UCB

The previous strategies are not concerned with minimizing the total traveling cost of the UAV. If an algorithm makes the UAV keep moving from one end of the environment to the other end, the UAV cannot gather much information and may output a low intensity point due to a budget B.

We employ the Dynamic Window Approach (DWA) where the 3D window is defined centered on the current position of the UAV. The UAV is only allowed to visit near-neighboring arms to observe at each timestep. To do that in Algorithm 5, the UAV considers neighboring arms (*i.e.*, a subset of A) with respect to the current UAV position $\mathbf{v}(k)$ from Line 10 to Line 17 to decide which arm to visit next.

5.6 Simulations

We implemented Monte Carlo simulations using MATLAB to verify the performance of Algorithm 5. We randomly generated 40 environments (20×20 square meters, having the same maximum intensity value (*i.e.*, 50) so as to have the same $f(\mathbf{x}^{OPT})$ for all cases) and ran 10 cases for each environment. We set three altitudes for the UAV to fly at, *i.e.*, 10, 40, and 70 meters. Each image consists of 9 pixels and the camera footprint size at the lowest altitude is 1×1 square meters. We considered the total budget of 100 in all cases.

We conducted a comparison analysis of planning strategies in Section 5.5 with baseline algorithms. We then compared 3D planning with 2D planning where the UAV was allowed to fly at a fixed altitude. We also tested how the performance of algorithms varies depending on the amount of budget.

We define two performance metrics that are the percent of how close $f(\mathbf{x}^{ALG})$ is to $f(\mathbf{x}^{OPT})$ (from perspective of a point) and that of how close $\sum_{\mathbf{a}_i^{ALG}} f(\mathbf{x})$ is to $\sum_{\mathbf{a}_i^{OPT}} f(\mathbf{x})$ (from perspective of an arm). The point performance metric is relevant when a USV will go and collect a physical sample at ($\mathbf{x}^{OPT}$. The arm performance metric is relevant when, instead of a USV, a UAV will go and collect a final image measurement at the best arm location.

5.6.1 Comparison Analysis

We evaluated 8 heuristic planning algorithms where we also consider the decreasing rate and the increasing rate of β and compared with 10 other baselines (see the details of all algorithms in the caption of Table 5.1).

From randomly generated environments, we tune the hyperparameters (γ and λ) of β (Equation (5.6)) offline through simulations for both the increasing rate and the decreasing rate. We use the following exponential functional forms:

- $\beta = 1.5e^{-0.05k}$ for GP-UCB with current variance of decreasing β.
- $\beta = 10e^{-0.05k}$ for GP-UCB with future variance of decreasing β.
- $\beta = -0.5e^{-0.05k} + 0.5$ for GP-UCB with current variance of increasing β.
- $\beta = -10e^{-0.05k} + 10$ for GP-UCB with future variance of increasing β.

	Point	Arm
CV (β--)	$47.65 \pm 4.52\%$	$46.31 \pm 4.51\%$
FV (β--)	$57.40 \pm 4.83\%$	$57.49 \pm 5.17\%$
CV (β++)	$49.39 \pm 4.81\%$	$50.62 \pm 4.83\%$
FV (β++)	$59.46 \pm 5.07\%$	$58.49 \pm 5.14\%$
DCV (β--)	$62.94 \pm 4.89\%$	$63.53 \pm 4.82\%$
DFV (β--)	$71.28 \pm 4.82\%$	$71.96 \pm 4.59\%$
DCV (β++)	$61.49 \pm 4.72\%$	$62.19 \pm 4.59\%$
DFV (β++)	$\mathbf{71.60 \pm 4.73\%}$	$\mathbf{74.51 \pm 4.52\%}$
Block UCL	$35.13 \pm 1.84\%$	$11.91 \pm 0.45\%$
BH	$35.86 \pm 3.16\%$	$3.40 \pm 0.00\%$
BM	$34.81 \pm 2.74\%$	$3.40 \pm 0.00\%$
BL	$28.73 \pm 2.83\%$	$3.40 \pm 0.00\%$
D2DH (β--)	$65.81 \pm 4.81\%$	$69.58 \pm 4.52\%$
D2DH (β++)	$66.93 \pm 4.70\%$	$72.12 \pm 4.50\%$
D2DM (β--)	$63.89 \pm 5.00\%$	$69.08 \pm 3.93\%$
D2DM (β++)	$69.23 \pm 4.71\%$	$71.22 \pm 4.12\%$
D2DL (β--)	$44.48 \pm 4.67\%$	$45.88 \pm 4.60\%$
D2DL (β++)	$47.56 \pm 5.00\%$	$48.31 \pm 4.91\%$

Table 5.1: Comparison with eight heuristic planning strategies, two baseline algorithms, and 2D exploration algorithms. The values represent the percents of how close the best estimated value is to the true best value with one-sigma error. Acronyms in this table are: CV (Current Variance), FV (Future Variance), DCV (DWA Current Variance), DFV (DWA Future Variance), BH (Boustrophedon at the Highest altitude), BM (Boustrophedon at the Middle altitude), BL (Boustrophedon at the Lowest altitude), D2DH (DWA 2D exploration at the Highest altitude), D2DM (DWA 2D exploration at the Middle altitude), and D2DL (DWA 2D exploration at the Lowest altitude). β++ and β-- imply β with the increasing rate and the decreasing rate, respectively.

Comparison with baseline algorithms

We introduce two baseline algorithms that can also be applied to our problem: (1) the boustrophedon algorithm [189] and (2) the block Upper Credible Limit (UCL) algorithm [196]. The boustrophedon algorithm is coverage planning where the entire environment is completely covered by the camera footprint of the UAV but gathered information online is not exploited. The block UCL algorithm is a Bayesian approach that addresses the exploration-exploitation dilemma. The unique feature of this algorithm is that it not only maximizes the expected reward obtained within a budget, but it also minimizes the number of switching to other arms, which is related to minimizing a traveling cost.

From Table 5.1, the proposed algorithms outperform baseline algorithms in all cases. For the

case of boustrophedon algorithms, the budget of 100 was not sufficient to cover the entire environment and consequently ended up with poor performance. Due to the allocation scheme used by the block UCL, the algorithm requires sensing at an arm a large number of times before moving to the next arm. This resulted in exploring only a small portion of the environment.

Comparison with 2D planning

We compared with 2D planning algorithms where we fixed the altitude at which the UAV can fly. It can be seen from Table 5.1 that flying at a specific altitude is not beneficial in comparison with 3D exploration.

Effect of the amount of budget

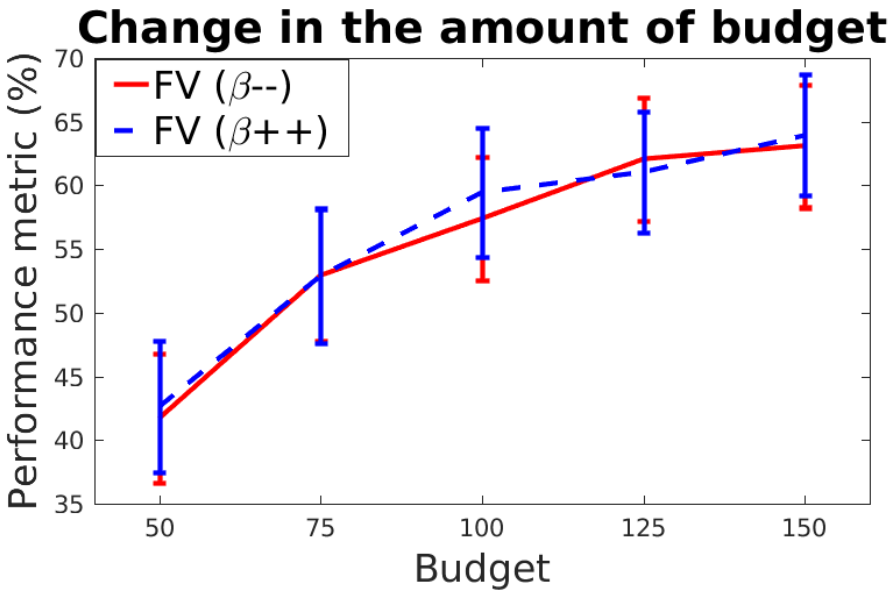


Figure 5.4: Relationship between the budget and the performance metric for a point in case of GP-UCB with current and future variances.

We tested how the performance metric for a point varies with respect to the change in the amount of budget given to the UAV. As shown in Figure 5.4, the performance metric increases logarithmically as the budget increases.

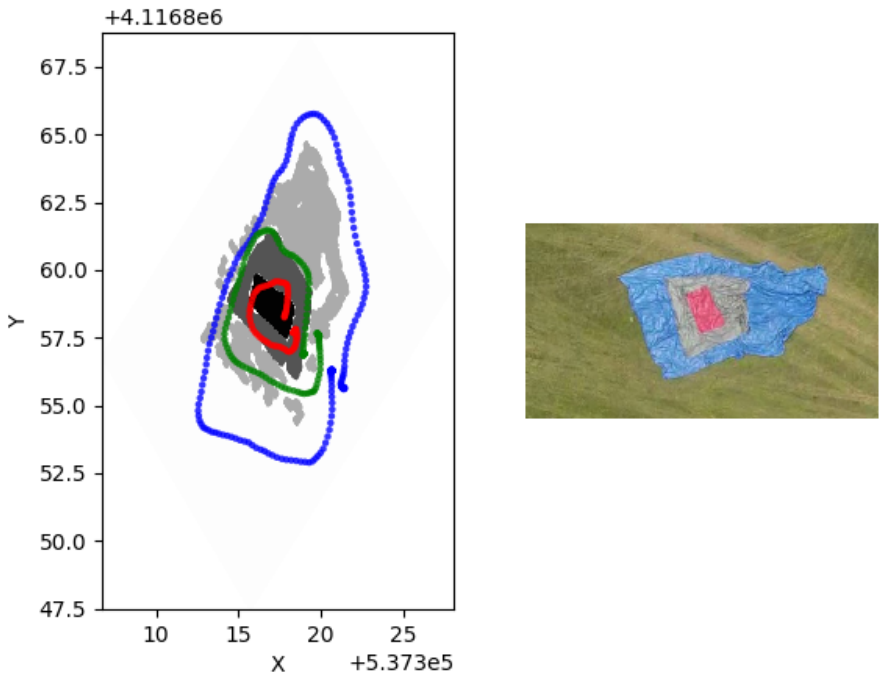
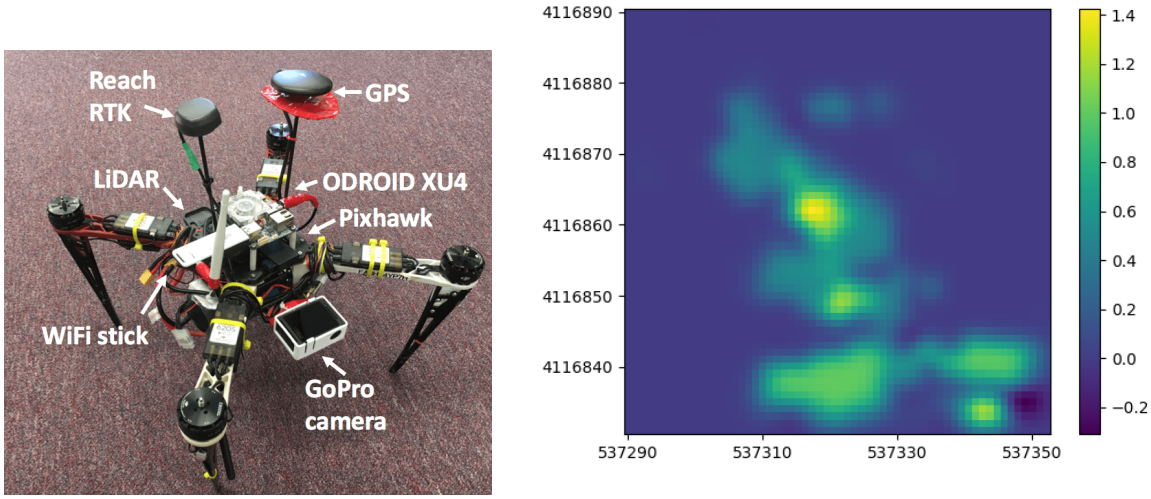


Figure 5.5: Image on the right shows the tarps used as proxy of the regions with different concentrations. Image on the left shows the concentration for the image on the right along with the ground truth boundaries.



(a) The UAV platform. (b) The resultant GP learned by the UAV from real-world data.

Figure 5.6: Simulation results using real-world data.

5.6.2 Real-world Data

In our experimental setup we used the UAV (Figure 5.6) with a single onboard-PC which has Linux 16.04 and ROS kinetic [79] installed. It is equipped with a GPS sensor, a compass and a downward-facing camera sensor (GoPro Hero4), which can communicate with the UAV over WiFi. We used three tarps of blue, gray, and red colors as a proxy of regions with different concentration of toxicants. We collected our real-world data by flying the UAV at 3 different altitudes of 10, 20, and 30 meters in a boustrophedon pattern (3.25×4.73, 6.51×9.45, 9.76×14.12 square meters, respectively, in the camera footprint size). We gathered 346 images spanned over these altitudes. Figure 5.5 shows the ground truth. For each image in our dataset, we assigned the concentration value $f(\mathbf{x})$ for each color on a per-pixel basis. We have three discrete concentration values of 3, 2, and 1 associated with HSV color ranges of the red, gray, and blue tarps, respectively. To generate the concentration matrix, we created masks for each of the aforementioned colors. These masks are morphologically processed for smoothening, and depending on which mask a particular pixel belongs to, it is assigned a concentration value. Left image in Figure 5.5 shows the visualization of the concentration values in Universal Transverse Mercator (UTM) coordinates. The darker region represents higher coordinate value.

In simulation the original resolution of the image 240×432 was reduced to 19×34. Thus, the number of pixel measurements from an image (*i.e.*, M) was 646. The camera footprint at an arm depends on the yaw of the UAV as well as $\mathbf{v}$. To use the concentration matrix with GP-UCB, we do the mapping from image coordinates to real-world coordinates. To estimate UTM coordinates for each pixel, we make three main assumptions. (1) The principal point

Camera Parameters	Focal Length	Sensor Height	Sensor Width
Value	$6.21mm$	$4.04mm$	$5.87mm$

Table 5.2: Intrinsic camera parameters obtained from camera calibration.

of the camera aligns with the image center. (2) Consequently, the UTM northing and easting of the UAV can be projected directly onto the image center. (3) Roll and pitch errors of the UAV are negligible. To estimate each pixel position of a given image in UTM coordinates, we rotate the frame of reference to make it align with the direction of UTM coordinate axis at that point. We achieve this by rotating each pixel, hence the entire image in a direction opposite to the UAV heading. Consequently, in the rotated frame of reference the UAV heading aligned with the north direction and hence the UTM coordinate axis. This makes it possible to use the geometric formula for translation in x- and y-axis along with UTM coordinates of the image center to interpolate the UTM location of each pixel.

Once we have the rotated frame of reference and the UTM coordinates of the image center, C, the translation values T_x and T_y in x- and y-direction, respectively, can be used to compute the UTM coordinates of the pixel P [200] as follows:

$$T_x = \frac{Altitude(mm) \times P_x - C_x(px) \times Sensor\ Breadth(mm)}{f(mm) \times Image\ Breadth(mm)} \tag{5.8}$$

$$T_y = \frac{Altitude(mm) \times P_y - C_y(px) \times Sensor\ Height(mm)}{f(mm) \times Image\ Height(mm)} \tag{5.9}$$

$$P_{northing} = C_{northing} \pm T_x \tag{5.10}$$

$$P_{easting} = C_{easting} \pm T_y \tag{5.11}$$

The intrinsic parameters of the camera sensor are obtained from camera calibration (Table 5.2). It can be seen from Figure 5.5 that our estimates of the UTM locations aligns well with the ground-truth boundaries.

Figure 5.6 (b) shows the resultant GP by applying the proposed algorithm to real-world data. $\mathbf{x}^{OPT}$ is $(537317, 4116858)$ and $\mathbf{x}^{ALG}$ is $(537317.78, 4116858.29)$. The performance metric we obtained is 100% as both $f(\mathbf{x}^{OPT})$ and $f(\mathbf{x}^{ALG})$ are 3.

5.7 Conclusion

In this chapter, we propose the exploration algorithm that can find a hotspot in an unknown environment in limited time. We show how the UAV equipped with a downward-facing camera can be adopted to MAB and present simulation results to verify the performance of the algorithm.

Immediate future work would be to conduct more rigorous real-world experiments using the proposed scheme. Also, since we may suffer from explosively increasing number of measurements, we may need to use sparse GP. Analyzing the regret bound or suboptimality with respect to the optimal reward would be promising.

Chapter 6

Conclusion

6.1 Summary of Work

In this dissertation, we investigate problems arising when deploying robots for hazardous environmental monitoring. In particular, we propose two types of hazards that may exist in the environment (*i.e.*, individual targets and plumes) that require different monitoring strategies. Search and tracking are important for finding individual targets that might be hidden or cluttered in the environment, whereas for plumes, exploration and mapping are key tasks to learn the boundary or the unknown density of the plume.

We address various challenges of dealing with these types of hazards and develop algorithms that can overcome the proposed challenges as follows.

Tracking of an unknown and varying number of hazardous targets

In Chapter 2, we tackle the problem of tracking an unknown and varying number of individual hazardous targets, especially when deploying a UAV equipped with a limited FOV sensor. Our proposed algorithm can not only estimate states of an unknown and varying number of targets, but also handle erroneous sensing results and an unknown motion model for targets. We demonstrate the performance of our algorithm through proof-of-concept field experiments using a single UAV.

Coordinated multi-target tracking under limited bandwidth

In Chapter 3, we study the problem of tracking individual hazardous targets using multiple robots with limited communication ranges where only a short amount of communication time is allowed for each communication round. We design two distributed algorithms that can guarantee a global tracking performance in a bounded number of communication rounds. We validate our distributed algorithms via an exhaustive comparison analysis with centralized algorithms and Gazebo simulations.

Coordinated online exploration of a translating plume

In Chapter 4, we address the problem of exploring a translating plume whose size and shape are not given to robots a priori. In this work, we consider a binary sensing model that outputs either the plume region or the non-plume region. We develop an online mapping algorithm for multiple UAVs which yields a competitive ratio of mission time with respect to the optimal offline solution. We show how our algorithm can be implemented and operates from real-world field experiments.

Hotspot identification in limited time

In Chapter 5, we study the problem of finding the best sampling location in limited time from a field of unknown contamination density by deploying a UAV in 3D environment. Our contribution is to show how the MAB framework can be adopted to our problem, addressing the issues of obtaining multiple observations from each sensing location and various noise levels on observations depending on which altitude they are obtained. We empirically evaluate the performance of our algorithm using real-world data.

In summary, the major contributions of this thesis are to propose algorithms with provable guarantees that overcome hardware limitations (*e.g.*, imperfect sensing and limited bandwidth/communication range), limited resources (*e.g.*, operation time and fuel), and uncertain and partial information of a given environment. We show an extensive simulation analysis and real-world experiment results using actual robots (UAVs) in the field to evaluate the proposed algorithms.

6.2 Discussion and Future Research Areas

Although our algorithms are not a universal solution that addresses all the challenges we mentioned above at once, we hope the contributions we propose would extend the current state-of-the-art techniques to a more powerful method that can be implemented on actual robots to carry out hazardous environmental monitoring tasks. We conclude the thesis by

introducing some future research areas that are not covered in this work but could help multi-robot environmental monitoring.

Heterogeneous team of robots

The employment of robots having different maneuverability, sensing ability, and capability would be valuable for hazardous environmental monitoring as shortcomings of one type of robots may be complemented by other type of robots. For example, in case of monitoring hazards in lakes, while the UAV can explore the environment faster than the USV, the USV have an access to water samples that are difficult to be obtained by the UAV. Moreover, the UAV can guide the USV to a desired region more effectively using its wide range of FOV. The USV can also inform the UAV about its learned characteristic of hazardous agents so as to update a prior knowledge of hazards of the UAV. Bringing the concept of heterogeneity into environmental monitoring tasks would give a large flexibility in the choice of options and ease various challenges that the current state-of-the-art algorithms cannot handle.

Irrevocable sample selection

As most hazardous plume varies both spatially and temporally, available samples from the current location of a robot might have different characteristics when visited later. Thus, it is essential to make a correct decision on the collection of samples, and otherwise the robot may miss the best possible sample that might not exist anymore. Irrevocable sample selection is an important problem in environmental monitoring, and therefore, it needs to be integrated in the proposed framework in this thesis.

Risk-averse planning

In some scenarios, hazardous agents are harmful not only to humans, but also to robots if hazards are fire or some materials that could possibly damage the robotic platforms. In such cases, robots must plan their trajectories in a way that they avoid the critical harmful regions but are still able to monitor hazards. This aspect of the objective must be included in the presented algorithms for risk-averse planning so that the robots can complete their mission before breakdown.

Bibliography

[1] P. Floréen, M. Hassinen, J. Kaasinen, P. Kaski, T. Musto, and J. Suomela, "Local approximability of max-min and min-max linear programs," *Theory of Computing Systems*, vol. 49, no. 4, pp. 672–697, 2011.

[2] L. E. Parker, "Distributed algorithms for multi-robot observation of multiple moving targets," *Autonomous robots*, vol. 12, no. 3, pp. 231–255, 2002.

[3] Y. Sung, D. Dixit, and P. Tokekar, "Online multi-robot exploration of a translating plume: Competitive algorithm and experiments," *arXiv preprint arXiv:1811.02769*, 2019.

[4] "Flickr fukushima explosion." `https://www.flickr.com/photos/vizpix/5529038135`. Accessed: 2018-11-29.

[5] "Flickr deepwater horizon oil spill." `https://www.flickr.com/photos/kk/4710168879`. Accessed: 2018-11-29.

[6] "Pxhere surveillance camera." `https://pxhere.com/en/photo/661437`. Accessed: 2018-11-29.

[7] O. M. Cliff, R. Fitch, S. Sukkarieh, D. Saunders, and R. Heinsohn, "Online localization of radio-tagged wildlife with an autonomous aerial robot system.," in *Robotics: Science and Systems*, 2015.

[8] P. Tokekar, E. Branson, J. Vander Hook, and V. Isler, "Tracking aquatic invaders: Autonomous robots for monitoring invasive fish," *IEEE Robotics & Automation Magazine*, vol. 20, no. 3, pp. 33–41, 2013.

[9] P. P. Neumann, S. Asadi, A. J. Lilienthal, M. Bartholmai, and J. H. Schiller, "Autonomous gas-sensitive microdrone: Wind vector estimation and gas distribution mapping," *IEEE robotics & automation magazine*, vol. 19, no. 1, pp. 50–61, 2012.

[10] P. Tokekar, J. Vander Hook, D. Mulla, and V. Isler, "Sensor planning for a symbiotic uav and ugv system for precision agriculture," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1498–1511, 2016.

[11] M. Duarte, J. Gomes, V. Costa, T. Rodrigues, F. Silva, V. Lobo, M. M. Marques, S. M. Oliveira, and A. L. Christensen, "Application of swarm robotics systems to marine environmental monitoring," in *OCEANS 2016-Shanghai*, pp. 1–8, IEEE, 2016.

[12] G. Hitz, A. Gotovos, M.-É. Garneau, C. Pradalier, A. Krause, R. Y. Siegwart, *et al.*, "Fully autonomous focused exploration for robotic environmental monitoring," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pp. 2658–2664, IEEE, 2014.

[13] G. Ferri, M. V. Jakuba, A. Mondini, V. Mattoli, B. Mazzolai, D. R. Yoerger, and P. Dario, "Mapping multiple gas/odor sources in an uncontrolled indoor environment using a bayesian occupancy grid mapping based method," *Robotics and Autonomous Systems*, vol. 59, no. 11, pp. 988–1000, 2011.

[14] M. T. Lazarescu, "Design and field test of a wsn platform prototype for long-term environmental monitoring," *Sensors*, vol. 15, no. 4, pp. 9481–9518, 2015.

[15] J.-H. Kim, Y. Sung, and B. Y. Lattimer, "Bayesian estimation based real-time fire-heading in smoke-filled indoor environments using thermal imagery," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5231–5236, IEEE, 2017.

[16] Y. Sung and P. Tokekar, "Algorithm for searching and tracking an unknown and varying number of mobile targets using a limited fov sensor," in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pp. 6246–6252, IEEE, 2017.

[17] Y. Sung and P. Tokekar, "Gm-phd filter for searching and tracking an unknown number of targets with a mobile sensor with limited fov," *arXiv preprint arXiv:1812.09636*, 2018.

[18] Y. Sung, A. K. Budhiraja, R. K. Williams, and P. Tokekar, "Distributed simultaneous action and target assignment for multi-robot multi-target tracking," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1–9, IEEE, 2018.

[19] Y. Sung, A. K. Budhiraja, R. K. Williams, and P. Tokekar, "Distributed assignment with limited communication for multi-robot multi-target tracking," *Autonomous Robots*, pp. 1–17, 2019.

[20] Y. Sung and P. Tokekar, "A competitive algorithm for online multi-robot exploration of a translating plume," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 3391–3397, IEEE, 2019.

[21] Y. Sung, D. Dixit, and P. Tokekar, "Environmental hotspot identification in limited time with a uav equipped with a downward-facing camera," *arXiv preprint arXiv:1909.08483*, 2020.

[22] T. Furukawa, F. Bourgault, B. Lavis, and H. F. Durrant-Whyte, "Recursive bayesian search-and-tracking using coordinated uavs for lost targets," in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pp. 2521–2526, IEEE, 2006.

[23] Y. Sung and T. Furukawa, "Information measure for the optimal control of target searching via the grid-based method," in *Information Fusion (FUSION), 2016 19th International Conference on*, pp. 2075–2080, IEEE, 2016.

[24] J. Casper and R. R. Murphy, "Human-robot interactions during the robot-assisted urban search and rescue response at the world trade center," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 33, no. 3, pp. 367–385, 2003.

[25] B. Rao, H. F. Durrant-Whyte, and J. Sheen, "A fully decentralized multi-sensor system for tracking and surveillance," *The International Journal of Robotics Research*, vol. 12, no. 1, pp. 20–44, 1993.

[26] P. Dames, P. Tokekar, and V. Kumar, "Detecting, localizing, and tracking an unknown number of moving targets using a team of mobile robots," *The International Journal of Robotics Research*, vol. 36, no. 13-14, pp. 1540–1553, 2017.

[27] P. Tokekar, V. Isler, and A. Franchi, "Multi-target visual tracking with aerial robots," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3067–3072, IEEE, 2014.

[28] M. Dunbabin and L. Marques, "Robots for environmental monitoring: Significant advancements and applications," *IEEE Robotics and Automation Magazine*, vol. 19, pp. 24 –39, Mar 2012.

[29] V. Isler, N. Noori, P. Plonski, A. Renzaglia, P. Tokekar, and J. Vander Hook, "Finding and tracking targets in the wild: Algorithms and field deployments," in *International Symposium on Safety, Security, and Rescue Robotics*, 2015. to appear.

[30] T. Furukawa, L. C. Mak, H. Durrant-Whyte, and R. Madhavan, "Autonomous bayesian search and tracking, and its experimental validation," *Advanced Robotics*, vol. 26, no. 5-6, pp. 461–485, 2012.

[31] P. Skoglar, U. Orguner, D. Törnqvist, and F. Gustafsson, "Road target search and tracking with gimballed vision sensor on an unmanned aerial vehicle," *Remote sensing*, vol. 4, no. 7, pp. 2076–2111, 2012.

[32] J. Tisdale, A. Ryan, Z. Kim, D. Tornqvist, and J. K. Hedrick, "A multiple uav system for vision-based search and localization," in *2008 American Control Conference*, pp. 1985–1990, IEEE, 2008.

[33] B.-N. Vo and W.-K. Ma, "The gaussian mixture probability hypothesis density filter," *IEEE Transactions on signal processing*, vol. 54, no. 11, pp. 4091–4104, 2006.

[34] C. E. Rasmussen and H. Nickisch, "Gaussian processes for machine learning (gpml) toolbox," *The Journal of Machine Learning Research*, vol. 11, pp. 3011–3015, 2010.

[35] J. Joseph, F. Doshi-Velez, A. S. Huang, and N. Roy, "A bayesian nonparametric approach to modeling motion patterns," *Autonomous Robots*, vol. 31, no. 4, pp. 383–400, 2011.

[36] C. Knabe, R. Griffin, J. Burton, G. Cantor-Cooke, L. Dantanarayana, G. Day, O. Ebeling-Koning, E. Hahn, M. Hopkins, J. Neal, *et al.*, "Team valors escher: A novel electromechanical biped for the darpa robotics challenge," in *The DARPA Robotics Challenge Finals: Humanoid Robots To The Rescue*, pp. 583–629, Springer, 2018.

[37] R. R. Murphy, S. Tadokoro, D. Nardi, A. Jacoff, P. Fiorini, H. Choset, and A. M. Erkmen, "Search and rescue robotics," in *Springer Handbook of Robotics*, pp. 1151–1173, Springer, 2008.

[38] M. Bernard, K. Kondak, I. Maza, and A. Ollero, "Autonomous transportation and deployment with aerial robots for search and rescue missions," *Journal of Field Robotics*, vol. 28, no. 6, pp. 914–931, 2011.

[39] V. Kumar, D. Rus, and S. Singh, "Robot and sensor networks for first responders," *IEEE Pervasive computing*, vol. 3, no. 4, pp. 24–33, 2004.

[40] S. Tadokoro, *Rescue Robotics: DDT Project on Robots and Systems for Urban Search and Rescue*. Springer Science & Business Media, 2009.

[41] F. Körner, R. Speck, A. H. Göktogan, and S. Sukkarieh, "Autonomous airborne wildlife tracking using radio signal strength," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pp. 107–112, IEEE, 2010.

[42] H. Bayram, K. Doddapaneni, N. Stefas, and V. Isler, "Active localization of vhf collared animals with aerial robots," in *Automation Science and Engineering (CASE), 2016 IEEE International Conference on*, pp. 934–939, IEEE, 2016.

[43] B. Kartal, J. Godoy, I. Karamouzas, and S. J. Guy, "Stochastic tree search with useful cycles for patrolling problems," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pp. 1289–1294, IEEE, 2015.

[44] S. Radmard and E. A. Croft, "Active target search for high dimensional robotic systems," *Autonomous Robots*, pp. 1–18, 2015.

[45] B. Mobedi and G. Nejat, "3-d active sensing in time-critical urban search and rescue missions," *IEEE/ASME transactions on mechatronics*, vol. 17, no. 6, pp. 1111–1119, 2012.

[46] T. Tomic, K. Schmid, P. Lutz, A. Domel, M. Kassecker, E. Mair, I. L. Grixa, F. Ruess, M. Suppa, and D. Burschka, "Toward a fully autonomous uav: Research platform for indoor and outdoor urban search and rescue," *IEEE robotics & automation magazine*, vol. 19, no. 3, pp. 46–56, 2012.

[47] L. M. Miller, Y. Silverman, M. A. MacIver, and T. D. Murphey, "Ergodic exploration of distributed information," *IEEE Transactions on Robotics*, vol. 32, no. 1, pp. 36–52, 2016.

[48] T. H. Chung and J. W. Burdick, "A decision-making framework for control strategies in probabilistic search," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pp. 4386–4393, IEEE, 2007.

[49] A. Ryan and J. K. Hedrick, "Particle filter based information-theoretic active sensing," *Robotics and Autonomous Systems*, vol. 58, no. 5, pp. 574–584, 2010.

[50] G. Hollinger, S. Singh, J. Djugash, and A. Kehagias, "Efficient multi-robot search for a moving target," *The International Journal of Robotics Research*, vol. 28, no. 2, pp. 201–219, 2009.

[51] T. H. Chung, G. A. Hollinger, and V. Isler, "Search and pursuit-evasion in mobile robotics," *Autonomous Robots*, vol. 31, no. 4, pp. 299–316, 2011.

[52] Y. Bar-Shalom, F. Daum, and J. Huang, "The probabilistic data association filter," *IEEE Control Systems*, vol. 29, no. 6, pp. 82–100, 2009.

[53] S. S. Blackman, "Multiple hypothesis tracking for multiple target tracking," *IEEE Aerospace and Electronic Systems Magazine*, vol. 19, no. 1, pp. 5–18, 2004.

[54] Y. Sung and W. Chung, "Hierarchical sample-based joint probabilistic data association filter for following human legs using a mobile robot in a cluttered environment," *IEEE Transactions on Human-Machine Systems*, vol. 46, no. 3, pp. 340–349, 2016.

[55] L. L. Wong, L. P. Kaelbling, and T. Lozano-Pérez, "Data association for semantic world modeling from partial views," *The International Journal of Robotics Research*, p. 0278364914559754, 2015.

[56] A. Ibarguren, I. Maurtua, M. Pérez, and B. Sierra, "Multiple target tracking based on particle filtering for safety in industrial robotic cells," *Robotics and Autonomous Systems*, vol. 72, pp. 105–113, 2015.

[57] S. Thrun, W. Burgard, D. Fox, *et al.*, *Probabilistic robotics*, vol. 1. MIT press Cambridge, 2005.

[58] R. P. Mahler, "Multitarget bayes filtering via first-order multitarget moments," *IEEE Transactions on Aerospace and Electronic systems*, vol. 39, no. 4, pp. 1152–1178, 2003.

[59] J. Mullane, B.-N. Vo, M. D. Adams, and B.-T. Vo, "A random-finite-set approach to bayesian slam," *IEEE Transactions on Robotics*, vol. 27, no. 2, pp. 268–282, 2011.

[60] P. M. Dames, M. Schwager, D. Rus, and V. Kumar, "Active magnetic anomaly detection using multiple micro aerial vehicles.," *IEEE Robotics and Automation Letters*, vol. 1, no. 1, pp. 153–160, 2016.

[61] P. M. Dames, "Distributed multi-target search and tracking using the phd filter," *Autonomous Robots*, 2019.

[62] C. Ouyang, H.-B. Ji, and Y. Tian, "Improved gaussian mixture cphd tracker for multitarget tracking," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 49, no. 2, pp. 1177–1191, 2013.

[63] R. P. Mahler, *Statistical multisource-multitarget information fusion*. Artech House, Inc., 2007.

[64] R. Mahler, "Phd filters of higher order in target number," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 43, no. 4, pp. 1523–1543, 2007.

[65] B.-T. Vo, B.-N. Vo, and A. Cantoni, "Analytic implementations of the cardinalized probability hypothesis density filter," *IEEE Transactions on Signal Processing*, vol. 55, no. 7, pp. 3553–3567, 2007.

[66] A. Wasik, P. U. Lima, and A. Martinoli, "A robust localization system for multi-robot formations based on an extension of a gaussian mixture probability hypothesis density filter," *Autonomous Robots*, pp. 1–20, 2019.

[67] L. Lin, Y. Bar-Shalom, and T. Kirubarajan, "Data association combined with the probability hypothesis density filter for multitarget tracking," in *Defense and Security*, pp. 464–475, International Society for Optics and Photonics, 2004.

[68] K. Panta, B.-N. Vo, and S. Singh, "Novel data association schemes for the probability hypothesis density filter," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 43, no. 2, pp. 556–570, 2007.

[69] K. Panta, D. E. Clark, and B.-N. Vo, "Data association and track management for the gaussian mixture probability hypothesis density filter," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 45, no. 3, pp. 1003–1016, 2009.

[70] J. F. C. Kingman, *Poisson processes*. Wiley Online Library, 1993.

[71] B.-N. Vo, S. Singh, and A. Doucet, "Sequential monte carlo methods for multitarget filtering with random finite sets," *IEEE Transactions on Aerospace and electronic systems*, vol. 41, no. 4, pp. 1224–1245, 2005.

[72] M. Garzón, J. Valente, J. J. Roldán, L. Cancar, A. Barrientos, and J. Del Cerro, "A multirobot system for distributed area coverage and signal searching in large outdoor scenarios," *Journal of Field Robotics*, 2015.

[73] C.-Y. Kim, D. Song, Y. Xu, J. Yi, and X. Wu, "Cooperative search of multiple unknown transient radio sources using multiple paired mobile robots," *IEEE Transactions on Robotics*, vol. 30, no. 5, pp. 1161–1173, 2014.

[74] R. De Maesschalck, D. Jouan-Rimbaud, and D. L. Massart, "The mahalanobis distance," *Chemometrics and intelligent laboratory systems*, vol. 50, no. 1, pp. 1–18, 2000.

[75] B. Ristic, D. Clark, B.-N. Vo, and B.-T. Vo, "Adaptive target birth intensity for phd and cphd filters," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 48, no. 2, pp. 1656–1668, 2012.

[76] D. E. Clark and J. Bell, "Multi-target state estimation and track continuity for the particle phd filter," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 43, no. 4, 2007.

[77] J. Vander Hook, P. Tokekar, and V. Isler, "Cautious greedy strategy for bearing-based active localization: Experiments and theoretical analysis," in *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 1787–1792, IEEE, 2012.

[78] D. Schuhmacher, B.-T. Vo, and B.-N. Vo, "A consistent metric for performance evaluation of multi-object filters," *IEEE transactions on signal processing*, vol. 56, no. 8, pp. 3447–3457, 2008.

[79] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, p. 5, Kobe, Japan, 2009.

[80] E. Olson, "Apriltag: A robust and flexible visual fiducial system," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 3400–3407, IEEE, 2011.

[81] B. T. Vo, B.-N. Vo, and A. Cantoni, "The cardinality balanced multi-target multi-bernoulli filter and its implementations," *IEEE Transactions on Signal Processing*, vol. 57, no. 2, pp. 409–423, 2009.

[82] B.-T. Vo and B.-N. Vo, "Labeled random finite sets and multi-object conjugate priors," *IEEE Transactions on Signal Processing*, vol. 61, no. 13, pp. 3460–3475, 2013.

[83] B.-N. Vo, B.-T. Vo, and D. Phung, "Labeled random finite sets and the bayes multi-target tracking filter," *IEEE Transactions on Signal Processing*, vol. 62, no. 24, pp. 6554–6567, 2014.

[84] S. Reuter, B.-T. Vo, B.-N. Vo, and K. Dietmayer, "The labeled multi-bernoulli filter.," *IEEE Trans. Signal Processing*, vol. 62, no. 12, pp. 3246–3260, 2014.

[85] B.-N. Vo, B.-T. Vo, and H. G. Hoang, "An efficient implementation of the generalized labeled multi-bernoulli filter," *IEEE Transactions on Signal Processing*, vol. 65, no. 8, pp. 1975–1987, 2017.

[86] G. Best, O. M. Cliff, T. Patten, R. R. Mettu, and R. Fitch, "Dec-mcts: Decentralized planning for multi-robot active perception," *The International Journal of Robotics Research*, p. 0278364918755924, 2018.

[87] L. E. Parker and B. A. Emmons, "Cooperative multi-robot observation of multiple moving targets," in *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on*, vol. 3, pp. 2082–2089, IEEE, 1997.

[88] C. F. Touzet, "Robot awareness in cooperative mobile robot learning," *Autonomous Robots*, vol. 8, no. 1, pp. 87–97, 2000.

[89] A. Kolling and S. Carpin, "Cooperative observation of multiple moving targets: an algorithm and its formalization," *The International Journal of Robotics Research*, vol. 26, no. 9, pp. 935–953, 2007.

[90] W. Hönig and N. Ayanian, "Dynamic multi-target coverage with robotic cameras," in *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pp. 1871–1878, IEEE, 2016.

[91] B. Gharesifard and S. L. Smith, "Distributed submodular maximization with limited information," *IEEE Transactions on Control of Network Systems*, 2017.

[92] T. Howard, M. Pivtoraiko, R. A. Knepper, and A. Kelly, "Model-predictive motion planning: several key developments for autonomous mobile robots," *IEEE Robotics & Automation Magazine*, vol. 21, no. 1, pp. 64–73, 2014.

[93] J. Suomela, "Survey of local algorithms," *ACM Computing Surveys (CSUR)*, vol. 45, no. 2, p. 24, 2013.

[94] N. E. Young, "Sequential and parallel algorithms for mixed packing and covering," in *Foundations of Computer Science, 2001. Proceedings. 42nd IEEE Symposium on*, pp. 538–546, IEEE, 2001.

[95] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, "An analysis of approximations for maximizing submodular set functionsi," *Mathematical Programming*, vol. 14, no. 1, pp. 265–294, 1978.

[96] Z. Yan, N. Jouandeau, and A. A. Cherif, "A survey and analysis of multi-robot coordination," *International Journal of Advanced Robotic Systems*, vol. 10, no. 12, p. 399, 2013.

[97] H. Li, G. Chen, T. Huang, and Z. Dong, "High-performance consensus control in networked systems with limited bandwidth communication and time-varying directed topologies," *IEEE transactions on neural networks and learning systems*, vol. 28, no. 5, pp. 1043–1054, 2017.

[98] M. Otte and N. Correll, "Any-com multi-robot path-planning: Maximizing collaboration for variable bandwidth," in *Distributed Autonomous Robotic Systems*, pp. 161–173, Springer, 2013.

[99] M. Otte and N. Correll, "Dynamic teams of robots as ad hoc distributed computers: reducing the complexity of multi-robot motion planning via subspace selection," *Autonomous Robots*, pp. 1–23, 2018.

[100] A. Kassir, R. Fitch, and S. Sukkarieh, "Communication-efficient motion coordination and data fusion in information gathering teams," in *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pp. 5258–5265, IEEE, 2016.

[101] R. K. Williams and G. S. Sukhatme, "Constrained interaction and coordination in proximity-limited multiagent systems," *IEEE Transactions on Robotics*, vol. 29, no. 4, pp. 930–944, 2013.

[102] J. Vander Hook, P. Tokekar, and V. Isler, "Algorithms for cooperative active localization of static targets with mobile bearing sensors under communication constraints," *IEEE Transactions on Robotics*, vol. 31, no. 4, pp. 864–876, 2015.

[103] Y. Kantaros, M. Thanou, and A. Tzes, "Distributed coverage control for concave areas by a heterogeneous robot–swarm with visibility sensing constraints," *Automatica*, vol. 53, pp. 195–207, 2015.

[104] Y. Kantaros and M. M. Zavlanos, "Distributed intermittent connectivity control of mobile robot networks," *IEEE Transactions on Automatic Control*, vol. 62, no. 7, pp. 3109–3121, 2017.

[105] R. K. Williams, A. Gasparri, G. S. Sukhatme, and G. Ulivi, "Global connectivity control for spatially interacting multi-robot systems with unicycle kinematics," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pp. 1255–1261, IEEE, 2015.

[106] D. V. Dimarogonas, E. Frazzoli, and K. H. Johansson, "Distributed event-triggered control for multi-agent systems," *IEEE Transactions on Automatic Control*, vol. 57, no. 5, pp. 1291–1297, 2012.

[107] L. Zhou and P. Tokekar, "Active target tracking with self-triggered communications in multi-robot teams," *IEEE Transactions on Automation Science and Engineering*, no. 99, pp. 1–12, 2018.

[108] X. Ge and Q.-L. Han, "Distributed formation control of networked multi-agent systems using a dynamic event-triggered communication mechanism," *IEEE Transactions on Industrial Electronics*, vol. 64, no. 10, pp. 8118–8127, 2017.

[109] G. Best, M. Forrai, R. R. Mettu, and R. Fitch, "Planning-aware communication for decentralised multi-robot coordination," in *Proceedings of the International Conference on Robotics and Automation, Brisbane, Australia*, vol. 21, 2018.

[110] M. Guo and M. M. Zavlanos, "Multirobot data gathering under buffer constraints and intermittent communication," *IEEE Transactions on Robotics*, 2018.

[111] A. Khan, B. Rinner, and A. Cavallaro, "Cooperative robots to observe moving targets: Review," *IEEE Transactions on Cybernetics*, 2016.

[112] C. Robin and S. Lacroix, "Multi-robot target detection and tracking: taxonomy and survey," *Autonomous Robots*, vol. 40, no. 4, pp. 729–760, 2016.

[113] B. Charrow, V. Kumar, and N. Michael, "Approximate representations for multi-robot control policies that maximize mutual information," *Autonomous Robots*, vol. 37, no. 4, pp. 383–400, 2014.

[114] H. Yu, K. Meier, M. Argyle, and R. W. Beard, "Cooperative path planning for target tracking in urban environments using unmanned air and ground vehicles," *IEEE/ASME Transactions on Mechatronics*, vol. 20, no. 2, pp. 541–552, 2015.

[115] A. Ahmad, G. Lawless, and P. Lima, "An online scalable approach to unified multirobot cooperative localization and object tracking," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1184–1199, 2017.

[116] K. Zhou, S. I. Roumeliotis, *et al.*, "Multirobot active target tracking with combinations of relative observations," *IEEE Transactions on Robotics*, vol. 27, no. 4, pp. 678–695, 2011.

[117] J. Capitan, M. T. Spaan, L. Merino, and A. Ollero, "Decentralized multi-robot cooperation with auctioned pomdps," *The International Journal of Robotics Research*, vol. 32, no. 6, pp. 650–671, 2013.

[118] L. C. Pimenta, M. Schwager, Q. Lindsey, V. Kumar, D. Rus, R. C. Mesquita, and G. A. Pereira, "Simultaneous coverage and tracking (scat) of moving targets with robot networks," in *Algorithmic foundation of robotics VIII*, pp. 85–99, Springer, 2009.

[119] J. Banfi, J. Guzzi, F. Amigoni, E. F. Flushing, A. Giusti, L. Gambardella, and G. A. Di Caro, "An integer linear programming model for fair multitarget tracking in cooperative multirobot systems," *Autonomous Robots*, pp. 1–16, 2018.

[120] Z. Xu, R. Fitch, J. Underwood, and S. Sukkarieh, "Decentralized coordinated tracking with mixed discrete–continuous decisions," *Journal of Field Robotics*, vol. 30, no. 5, pp. 717–740, 2013.

[121] B. P. Gerkey and M. J. Matarić, "A formal analysis and taxonomy of task allocation in multi-robot systems," *The International Journal of Robotics Research*, vol. 23, no. 9, pp. 939–954, 2004.

[122] G. A. Korsah, A. Stentz, and M. B. Dias, "A comprehensive taxonomy for multi-robot task allocation," *The International Journal of Robotics Research*, vol. 32, no. 12, pp. 1495–1512, 2013.

[123] H.-L. Choi, L. Brunet, and J. P. How, "Consensus-based decentralized auctions for robust task allocation," *IEEE transactions on robotics*, vol. 25, no. 4, pp. 912–926, 2009.

[124] L. Luo, N. Chakraborty, and K. Sycara, "Distributed algorithms for multirobot task assignment with task deadline constraints," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 3, pp. 876–888, 2015.

[125] L. Liu and D. A. Shell, "Assessing optimal assignment under uncertainty: An interval-based algorithm," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 936–953, 2011.

[126] A. Kanakia, B. Touri, and N. Correll, "Modeling multi-robot task allocation with limited information as global game," *Swarm Intelligence*, vol. 10, no. 2, pp. 147–160, 2016.

[127] J. Le Ny, A. Ribeiro, and G. J. Pappas, "Adaptive communication-constrained deployment of unmanned vehicle systems," *IEEE Journal on Selected Areas in Communications*, vol. 30, no. 5, pp. 923–934, 2012.

[128] Y. Kantaros and M. M. Zavlanos, "Global planning for multi-robot communication networks in complex environments.," *IEEE Trans. Robotics*, vol. 32, no. 5, pp. 1045–1061, 2016.

[129] M. M. Zavlanos, M. B. Egerstedt, and G. J. Pappas, "Graph-theoretic connectivity control of mobile robot networks," *Proceedings of the IEEE*, vol. 99, no. 9, pp. 1525–1540, 2011.

[130] X. Ge, F. Yang, and Q.-L. Han, "Distributed networked control systems: A brief overview," *Information Sciences*, vol. 380, pp. 117–131, 2017.

[131] M. Turpin, N. Michael, and V. Kumar, "Capt: Concurrent assignment and planning of trajectories for multiple robots," *The International Journal of Robotics Research*, vol. 33, no. 1, pp. 98–112, 2014.

[132] D. Morgan, G. P. Subramanian, S.-J. Chung, and F. Y. Hadaegh, "Swarm assignment and trajectory optimization using variable-swarm, distributed auction assignment and sequential convex programming," *The International Journal of Robotics Research*, vol. 35, no. 10, pp. 1261–1285, 2016.

[133] S. Bandyopadhyay, S.-J. Chung, and F. Y. Hadaegh, "Probabilistic and distributed control of a large-scale swarm of autonomous agents," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1103–1123, 2017.

[134] S.-J. Chung, A. Paranjape, P. Dames, S. Shen, and V. Kumar, "A Survey on Aerial Swarm Robotics," *IEEE Transactions on Robotics*, 2018.

[135] M. Otte, M. Kuhlman, and D. Sofge, "Multi-robot task allocation with auctions in harsh communication environments," in *Multi-Robot and Multi-Agent Systems (MRS), 2017 International Symposium on*, pp. 32–39, IEEE, 2017.

[136] D. Angluin, "Local and global properties in networks of processors," in *Proceedings of the twelfth annual ACM symposium on Theory of computing*, pp. 82–93, ACM, 1980.

[137] N. Linial, "Locality in distributed graph algorithms," *SIAM Journal on Computing*, vol. 21, no. 1, pp. 193–201, 1992.

[138] M. Naor and L. Stockmeyer, "What can be computed locally?," *SIAM Journal on Computing*, vol. 24, no. 6, pp. 1259–1277, 1995.

[139] M. Hanckowiak, M. Karonski, and A. Panconesi, "On the distributed complexity of computing maximal matchings," *SIAM Journal on Discrete Mathematics*, vol. 15, no. 1, pp. 41–57, 2001.

[140] M. Åstrand, P. Floréen, V. Polishchuk, J. Rybicki, J. Suomela, and J. Uitto, "A local 2-approximation algorithm for the vertex cover problem," in *International Symposium on Distributed Computing*, pp. 191–205, Springer, 2009.

[141] M. Åstrand and J. Suomela, "Fast distributed approximation algorithms for vertex cover and set cover in anonymous networks," in *Proceedings of the twenty-second annual ACM symposium on Parallelism in algorithms and architectures*, pp. 294–302, ACM, 2010.

[142] C. Lenzen and R. Wattenhofer, "Minimum dominating set approximation in graphs of bounded arboricity," in *International Symposium on Distributed Computing*, pp. 510–524, Springer, 2010.

[143] F. Kuhn, T. Moscibroda, and R. Wattenhofer, "The price of being near-sighted," in *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pp. 980–989, Society for Industrial and Applied Mathematics, 2006.

[144] W. Niehsen, "Information fusion based on fast covariance intersection filtering," in *Information Fusion, 2002. Proceedings of the Fifth International Conference on*, vol. 2, pp. 901–904, IEEE, 2002.

[145] V. Vazirani, *Approximation algorithms*. Springer Publishing Company, Incorporated, 2001.

[146] "Tomlab: Optimization environment large-scale optimization in matlab." `http://tomopt.com/docs/quickguide/quickguide006.php`. Accessed: 2017-01-03.

[147] G. W. Podnar, J. M. Dolan, A. Elfes, S. Stancliff, E. Lin, J. C. Hosler, T. J. Ames, J. Moisan, T. A. Moisan, J. Higinbotham, *et al.*, "Operation of robotic science boats using the telesupervised adaptive ocean sensor fleet system," in *2008 IEEE International Conference on Robotics and Automation*, pp. 1061–1068, IEEE, 2008.

[148] M. Dunbabin and L. Marques, "Robots for environmental monitoring: Significant advancements and applications," *IEEE Robotics & Automation Magazine*, vol. 19, no. 1, pp. 24–39, 2012.

[149] P. Brass, F. Cabrera-Mora, A. Gasparri, and J. Xiao, "Multirobot tree and graph exploration," *IEEE Transactions on Robotics*, vol. 27, no. 4, pp. 707–717, 2011.

[150] M. Juliá, A. Gil, and O. Reinoso, "A comparison of path planning strategies for autonomous exploration and mapping of unknown environments," *Autonomous Robots*, vol. 33, no. 4, pp. 427–444, 2012.

[151] S. Nuske, S. Choudhury, S. Jain, A. Chambers, L. Yoder, S. Scherer, L. Chamberlain, H. Cover, and S. Singh, "Autonomous exploration and motion planning for an unmanned aerial vehicle navigating rivers," *Journal of Field Robotics*, vol. 32, no. 8, pp. 1141–1162, 2015.

[152] Y. Girdhar, P. Giguere, and G. Dudek, "Autonomous adaptive exploration using realtime online spatiotemporal topic modeling," *The International Journal of Robotics Research*, vol. 33, no. 4, pp. 645–657, 2014.

[153] C. Icking, T. Kamphans, R. Klein, and E. Langetepe, "Exploring simple grid polygons," in *International Computing and Combinatorics Conference*, pp. 524–533, Springer, 2005.

[154] A. Kolenderska, A. Kosowski, M. Małafiejski, and P. Żyliński, "An improved strategy for exploring a grid polygon," in *International Colloquium on Structural Information and Communication Complexity*, pp. 222–236, Springer, 2009.

[155] P. Fraigniaud, L. Gasieniec, D. R. Kowalski, and A. Pelc, "Collective tree exploration," *Networks: An International Journal*, vol. 48, no. 3, pp. 166–177, 2006.

[156] Y. Higashikawa, N. Katoh, S. Langerman, and S.-i. Tanigawa, "Online graph exploration algorithms for cycles and trees by multiple searchers," *Journal of Combinatorial Optimization*, vol. 28, no. 2, pp. 480–495, 2014.

[157] A. Borodin and R. El-Yaniv, *Online computation and competitive analysis*. cambridge university press, 2005.

[158] J. Das, G. Cross, C. Qu, A. Makineni, P. Tokekar, Y. Mulgaonkar, and V. Kumar, "Devices, systems, and methods for automated monitoring enabling precision agriculture," in *Automation Science and Engineering (CASE), 2015 IEEE International Conference on*, pp. 462–469, IEEE, 2015.

[159] P. Tokekar, D. Bhadauria, A. Studenski, and V. Isler, "A robotic system for monitoring carp in minnesota lakes," *Journal of Field Robotics*, vol. 27, no. 6, pp. 779–789, 2010.

[160] P. A. Plonski, J. Vander Hook, C. Peng, N. Noori, and V. Isler, "Environment exploration in sensing automation for habitat monitoring," *IEEE Transactions on Automation Science and Engineering*, vol. 14, no. 1, pp. 25–38, 2017.

[161] H. Ishida, Y. Wada, and H. Matsukura, "Chemical sensing in robotic applications: A review," *IEEE Sensors Journal*, vol. 12, no. 11, pp. 3163–3173, 2012.

[162] T. Lochmatter, E. A. Göl, I. Navarro, and A. Martinoli, "A plume tracking algorithm based on crosswind formations," in *Distributed Autonomous Robotic Systems*, pp. 91–102, Springer, 2013.

[163] M. Fahad, Y. Guo, B. Bingham, K. Krasnosky, L. Fitzpatrick, and F. A. Sanabria, "Robotic experiments to evaluate ocean plume characteristics and structure," in *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*, pp. 6098–6104, IEEE, 2017.

[164] E. Galceran and M. Carreras, "A survey on coverage path planning for robotics," *Robotics and Autonomous systems*, vol. 61, no. 12, pp. 1258–1276, 2013.

[165] M. Corah and N. Michael, "Distributed matroid-constrained submodular maximization for multi-robot exploration: Theory and practice," *Autonomous Robots*, vol. 43, no. 2, pp. 485–501, 2019.

[166] A. Singh, A. Krause, C. Guestrin, and W. J. Kaiser, "Efficient informative sensing using multiple robots," *Journal of Artificial Intelligence Research*, vol. 34, pp. 707–755, 2009.

[167] R. Sim and J. J. Little, "Autonomous vision-based robotic exploration and mapping using hybrid maps and particle filters," *Image and Vision Computing*, vol. 27, no. 1-2, pp. 167–177, 2009.

[168] K. Cesare, R. Skeele, S.-H. Yoo, Y. Zhang, and G. Hollinger, "Multi-uav exploration with limited communication and battery," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pp. 2230–2235, IEEE, 2015.

[169] M. A. Bender, A. Fernández, D. Ron, A. Sahai, and S. Vadhan, "The power of a pebble: Exploring and mapping directed graphs," *Information and computation*, vol. 176, no. 1, pp. 1–21, 2002.

[170] S. Das, P. Flocchini, S. Kutten, A. Nayak, and N. Santoro, "Map construction of unknown graphs by multiple agents," *Theoretical Computer Science*, vol. 385, no. 1-3, pp. 34–48, 2007.

[171] C. Icking, T. Kamphans, R. Klein, and E. Langetepe, "Exploring an unknown cellular environment.," in *EuroCG*, pp. 140–143, 2000.

[172] E. M. Arkin, S. P. Fekete, and J. S. Mitchell, "Approximation algorithms for lawn mowing and milling," *Computational Geometry*, vol. 17, no. 1-2, pp. 25–50, 2000.

[173] S. Arya, S.-W. Cheng, and D. M. Mount, "Approximation algorithm for multiple-tool milling," *International Journal of Computational Geometry & Applications*, vol. 11, no. 03, pp. 339–372, 2001.

[174] A. F. van der Stappen and M. H. Overmars, "Motion planning amidst fat obstacles," in *Proceedings of the tenth annual symposium on Computational geometry*, pp. 31–40, ACM, 1994.

[175] A. Efrat, "The complexity of the union of (α,β)-covered objects," *SIAM Journal on Computing*, vol. 34, no. 4, pp. 775–787, 2005.

[176] G. Aloupis, P. Bose, V. Dujmović, C. Gray, S. Langerman, and B. Speckmann, "Triangulating and guarding realistic polygons," *Computational Geometry*, vol. 47, no. 2, pp. 296–306, 2014.

[177] S. K. Lee, S. P. Fekete, and J. McLurkin, "Structured triangulation in multi-robot systems: Coverage, patrolling, voronoi partitions, and geodesic centers," *The International Journal of Robotics Research*, vol. 35, no. 10, pp. 1234–1260, 2016.

[178] A. Mahadev, D. Krupke, S. P. Fekete, and A. T. Becker, "Mapping and coverage with a particle swarm controlled by uniform inputs," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1097–1104, IEEE, 2017.

[179] M. Dynia, J. ŁopuszaŃski, and C. Schindelhauer, "Why robots need maps," in *International Colloquium on Structural Information and Communication Complexity*, pp. 41–50, Springer, 2007.

[180] A. Preshant, K. Yu, and P. Tokekar, "A geometric approach for multi-robot exploration in orthogonal polygons," in *Workshop on Algorithmic Foundations of Robotics (WAFR)*, 2016.

[181] N. Megow, K. Mehlhorn, and P. Schweitzer, "Online graph exploration: New results on old and new algorithms," *Theoretical Computer Science*, vol. 463, pp. 62–72, 2012.

[182] S. Das, D. Dereniowski, and C. Karousatou, "Collaborative exploration by energy-constrained mobile robots," in *International Colloquium on Structural Information and Communication Complexity*, pp. 357–369, Springer, 2015.

[183] J. Petrich and K. Subbarao, "On-board wind speed estimation for uavs," in *AIAA Guidance, Navigation, and Control Conference*, p. 6223, 2011.

[184] Z. A. Algfoor, M. S. Sunar, and H. Kolivand, "A comprehensive study on pathfinding techniques for robotics and video games," *International Journal of Computer Games Technology*, vol. 2015, p. 7, 2015.

[185] Wikipedia contributors, "Universal transverse mercator coordinate system — Wikipedia, the free encyclopedia," 2019. [Online; accessed 29-April-2019].

[186] "Nri: Coordinated detection and tracking of hazardous agents with aerial and aquatic robots to inform emergency responders," Oct 2016.

[187] C. Powers, R. Hanlon, and D. Schmale, "Tracking of a fluorescent dye in a freshwater lake with an unmanned surface vehicle and an unmanned aircraft system," *Remote Sensing*, vol. 10, no. 1, p. 81, 2018.

[188] N. Srinivas, A. Krause, S. Kakade, and M. Seeger, "Gaussian process optimization in the bandit setting: no regret and experimental design," in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, pp. 1015–1022, Omnipress, 2010.

[189] H. Choset, "Coverage of known spaces: The boustrophedon cellular decomposition," *Autonomous Robots*, vol. 9, no. 3, pp. 247–253, 2000.

[190] G. Flaspohler, N. Roy, and Y. Girdhar, "Near-optimal irrevocable sample selection for periodic data streams with applications to marine robotics," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1–8, IEEE, 2018.

[191] S. Manjanna, A. Q. Li, R. N. Smith, I. Rekleitis, and G. Dudek, "Heterogeneous multi-robot system for exploration and strategic water sampling," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1–8, IEEE, 2018.

[192] K.-C. Ma, Z. Ma, L. Liu, and G. S. Sukhatme, "Multi-robot informative and adaptive planning for persistent environmental monitoring," in *Distributed Autonomous Robotic Systems*, pp. 285–298, Springer, 2018.

[193] G. A. Hollinger and G. S. Sukhatme, "Sampling-based robotic information gathering algorithms," *The International Journal of Robotics Research*, vol. 33, no. 9, pp. 1271–1287, 2014.

[194] J. Binney, A. Krause, and G. S. Sukhatme, "Optimizing waypoints for monitoring spatiotemporal phenomena," *The International Journal of Robotics Research*, vol. 32, no. 8, pp. 873–888, 2013.

[195] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," in *Advances in neural information processing systems*, pp. 2951–2959, 2012.

[196] P. B. Reverdy, V. Srivastava, and N. E. Leonard, "Modeling human decision making in generalized gaussian multiarmed bandits," *Proceedings of the IEEE*, vol. 102, no. 4, pp. 544–571, 2014.

[197] S. Guha and K. Munagala, "Multi-armed bandits with metric switching costs," in *International Colloquium on Automata, Languages, and Programming*, pp. 496–507, Springer, 2009.

[198] P. Vansteenwegen, W. Souffriau, and D. Van Oudheusden, "The orienteering problem: A survey," *European Journal of Operational Research*, vol. 209, no. 1, pp. 1–10, 2011.

[199] J.-Y. Audibert and S. Bubeck, "Best arm identification in multi-armed bandits," 2010.

[200] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[201] C. E. Rasmussen, "Gaussian processes in machine learning," in *Summer School on Machine Learning*, pp. 63–71, Springer, 2003.

Appendix A

Proofs

A.1 Proof of Lemma 1

Equation (3.5) of a max-min linear program is equivalent to the following max-min problem if the scalar variable w which represents the inner minimization is eliminated:

$$\begin{aligned} \max_{x_m^i} \min_{j \in T} & \left(\sum_{i \in R} \sum_{m \in P^i} c_{i,m}^j x_m^i \right) \\ \text{subject to} & \sum_{m \in P^i} x_m^i \leq 1 \quad \forall i \in R \\ & x_m^i \geq 0 \quad \forall m \in P^i. \end{aligned} \tag{A.1}$$

From Equations (3.5) and (A.1), the following relationship is satisfied:

$$w^* = \min_{j \in T} \left(\sum_{i \in R} \sum_{m \in P^i} c_{i,m}^j {x_m^i}^* \right). \tag{A.2}$$

Since Equation (3.2) does not require x_m^i to be a linear value, Equation (3.2) is equivalent to Equation (3.5) with additional integer constraints.

A.2 Proof of Lemma 2

Considering $c^j_{i,m}$, which is a weight between m-th motion primitive of i-th robot and j-th target on graph $\mathcal{G}_S$, a quality of tracking ($w(\mathbf{t}_j)$) for j-th target can be defined as follows:

$$w(\mathbf{t}_j) \triangleq \max\{c^j_{i,m} \big| x^i_m = 1,\ \forall i \in R, m \in P^i\}. \tag{A.3}$$

Therefore, the sum of quality of tracking over all targets is:

$$\begin{aligned}\sum_{j\in T} w(\mathbf{t}_j) &= \sum_{j\in T} \max\{c^j_{i,m} \big| x^i_m = 1,\ \forall i \in R, m \in P^i\} \\ &= \sum_{j\in T} \Big(\sum_{i\in R} y^j_i \Big(\sum_{m\in P^i} c^j_{i,m} x^i_m \Big) \Big).\end{aligned} \tag{A.4}$$

Equation (A.4) is obtained by taking into account the conditional term of the first equation explicitly. The last equation follows from the property that y^j_i chooses the maximum value of $\sum_{m\in P^i} c^j_{i,m} x^i_m$ among all robots, which is shown in lines 10-14 of Algorithm 3. Therefore, the last equation is equal to the inner term of Equation (3.4).

A.3 Greedy Performs Poorly for the `Bottleneck` Variant

We present an example of instance that shows an arbitrary poor performance of the greedy algorithm when applied to the `Bottleneck` variant. Consider the following case where there are two robots ($\mathbf{r}_i$) having two motion primitives ($\mathbf{p}^i_m$) for each and two targets. The realization of the communication and sensing graphs are as in the following table. The tracking quality in this example corresponds to the number of targets being tracked.

	$\mathbf{p}^1_1$, $\mathbf{p}^2_1$	$\mathbf{p}^1_2$, $\mathbf{p}^2_2$
$\mathbf{r}_1$	$\mathbf{t}_1$	$\varnothing$
$\mathbf{r}_2$	$\varnothing$	$\mathbf{t}_2$

Let's apply the `Bottleneck` version of greedy algorithm to this case. Since the objective of the `Bottleneck` variant is to maximize the minimum tracking quality, the robot 1 ($\mathbf{r}_1$) chooses motion primitive 2 ($\mathbf{p}^1_2$) because choosing motion primitive 1 ($\mathbf{p}^1_1$) gives the value of 1 while choosing motion primitive 2 ($\mathbf{p}^1_2$) gives the value of 0. For the same reason, the

robot 2 ($\mathbf{r}_2$) chooses motion primitive 1 ($\mathbf{p}_1^2$). This gives the total value of 0, whereas the optimal solution is 2 as the first robot and second robot choose motion primitive 1 ($\mathbf{p}_1^1$) and motion primitive 2 ($\mathbf{p}_2^2$), respectively. The similar case is reproducible with a larger number of robots, motion primitives, and targets. Thus, the simple greedy performs arbitrarily badly for the BOTTLENECK variant.

A.4 Proof of Corollaries

The upper bound for MRSP can simply be obtained by plugging $S_p = 0$ into Equation (4.2) of Lemma 3.

The upper bound for SRTP can be derived from the upper bound of MRSP by having $R = 0$. However, we can even tighten the bound by using the following observation: if the robot and the plume move toward each other in one direction, they must move away from each other in order to return to the starting location, and vice versa. Therefore, ALG can be upper bounded as:

$$\texttt{ALG} \leq \frac{C-1}{S_r+S_p} + \frac{C-1}{S_r-S_p}. \tag{A.5}$$

Taking out negative terms from the above equation becomes:

$$\texttt{ALG} \leq \frac{2S_rC}{(S_r+S_p)(S_r-S_p)}, \tag{A.6}$$

which is a tighter bound than $\frac{2C}{S_r-S_p}$. Note that the difference between these bounds is $\frac{S_r}{S_r+S_p}$ that satisfies $\frac{1}{2} < \frac{S_r}{S_r+S_p} \leq 1$ because $S_r > S_p$.

The upper bound for SRSP can be derived by plugging either $R = 1$ and $S_p = 0$ into the upper bound for MRSP or $S_p = 0$ into the upper bound for SRTP.

A.5 Proof of Lemma 4

We claim the following inequalities.

$$\texttt{OPT}^R \leq \texttt{OPT}^1, \tag{A.7}$$

This can be obtained from the fact that the more number of robots are deployed, the shorter time will be taken to explore the entire tree.

Consider a tree consisting of R branches. Then, we claim the following inequality:

$$\texttt{OPT}^1 \leq R\texttt{OPT}^R, \tag{A.8}$$

Since $\texttt{OPT}^R$ is the time for a robot to explore the longest branch in the tree, $R\texttt{OPT}^R$ must be no less than $\texttt{OPT}^1$.

Combining these inequalities and Equation (4.7), we prove Lemma 4.