AN ABSTRACT OF THE THESIS OF

Ryan Skeele for the degree of Master of Science in Robotics presented on June 13, 2016.

Title: Planning Under Uncertainty for Unmanned Aerial Vehicles

Abstract approved: _____

Geoffrey Hollinger

Unmanned aerial vehicle (UAV) technology has grown out of traditional research and military applications and has captivated the commercial and consumer markets, showing the ability to perform a spectrum of autonomous functions. This technology has the capability of saving lives in search and rescue, fighting wildfires in environmental monitoring, and delivering time dependent medicine in package delivery. These examples demonstrate the potential impact this technology will have on our society. However, it is evident how sensitive UAVs are to the uncertainty of the physical world. In order to properly achieve the full potential of UAVs in these markets, robust and efficient planning algorithms are needed. This thesis addresses the challenge of planning under uncertainty for UAVs. We develop a suite of algorithms that are robust to changes in the environment and build on the key areas of research needed for utilizing UAVs in a commercial setting. Throughout this research three main components emerged: mon-

itoring targets in dynamic environments, exploration with unreliable communication, and risk-aware path planning.

We use a realistic fire simulation to test persistent monitoring in an uncertain environment. The fire is generated using the standard program for modeling wildfire, FARSITE. This model was used to validate a weighted-greedy approach to monitoring clustered points of interest (POIs) over traditional methods of tracking a fire front. We implemented the algorithm on a commercial UAV to demonstrate the deployment capability.

Dynamic monitoring has limited potential if if coordinated planning is fallible to uncertainty in the world. Uncertain communication can cause critical failures in coordinated planning algorithms. We develop a method for coordinated exploration of a multi-UAV team with unreliable communication and limited battery life. Our results show that the proposed algorithm, which leverages meeting, sacrificing, and relaying behavior, increases the percentage of the environment explored over a frontier-based exploration strategy by up to 18%. We test on teams of up to 8 simulated UAVs and 2 real UAVs able to cope with communication loss and still report improved gains. We demonstrate this work with a pair of custom UAVs in an indoor office environment.

We introduce a novel approach to incorporating and addressing uncertainty in planning problems. The proposed Risk-Aware Graph Search (RAGS) algorithm combines traditional deterministic search techniques with risk-aware planning. RAGS is able to trade off the number of future path options, as well as the mean and variance of the associated path cost distributions to make online edge traversal decisions that minimize the risk of executing a high-cost path. The algorithm is compared against existing graph

search techniques on a set of graphs with randomly assigned edge costs, as well as over a set of graphs with transition costs generated from satellite imagery data. In all cases, RAGS is shown to reduce the probability of executing high-cost paths over $A^*$, $D^*$ and a greedy planning approach.

High level planning algorithms can be brittle in dynamic conditions where the environment is not modeled perfectly. In developing planners for uncertainty we ensure UAVs will be able to operate in conditions outside the scope of prior techniques. We address the need for robustness in robotic monitoring, coordination, and path planning tasks. Each of the three methods introduced were tested in simulated and real environments, and the results show improvement over traditional algorithms.

# Planning Under Uncertainty for Unmanned Aerial Vehicles

by

Ryan Skeele

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Presented June 13, 2016
Commencement June 2017

Master of Science thesis of Ryan Skeele presented on June 13, 2016.

APPROVED:

---

Major Professor, representing Robotics

---

Head of the School of Mechanical, Industrial and Manufacturing Engineering

---

Dean of the Graduate School

I understand that my thesis will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my thesis to any reader upon request.

---

Ryan Skeele, Author

# ACKNOWLEDGEMENTS

The author would like to acknowledge key participants in the research presented: advisor, Dr. Geoff Hollinger; friend and colleague, Post-Doc Jen Jen Chung, as well as Carrie Rebhuhn, and Kyle Cesare who all contributed to the work presented in this thesis. Jen Jen and Carrie both contributed to the risk-aware planning algorithm, Jen Jen in particular was crucial to the work. Kyle was a my partner for the indoor exploration project and it would have been impossible without him. I'd specifically like to thank Geoff for providing me a learning opportunity and the ability to mature as an engineer and research enthusiast. I'd like to thanks my friends and family for supporting me through this part of my life, even when I floundered or lost motivation. I've have had countless hours of support from my peers in the robotics program. My committee members, Kagan Tumer and Bill Smart, provided me valuable guidance and instruction in my time at Oregon State University. Finally, I want to thank OSU for not only supporting and encouraging my passion for robotics but actually developing and funding a robotics program in my time here.

This research has been funded in part by NASA NNX14AI10G, ONR N00014-14-1-0509, and the Department of AirForce FA8651-14-C-0135.

# TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

# LIST OF FIGURES

LIST OF FIGURES (Continued)

# LIST OF TABLES

# LIST OF ALGORITHMS

# Chapter 1: Introduction

Unmanned aerial vehicle (UAV) technology has grown out of traditional research and military applications and has captivated the commercial and consumer markets, showing the ability to perform a spectrum of autonomous functions. This technology has the capability of saving lives in search and rescue, fighting wildfires in environmental monitoring, and delivering time dependent medicine in package delivery. These examples demonstrate the potential impact this technology will have on our society. However, it is evident how sensitive UAVs are to the uncertainty of the physical world. In order to properly achieve the full potential of UAVs in these markets, robust and efficient planning algorithms are needed.

As humans, we are are able to adapt to our physical environment with speed and ease. Our natural adaptability to reason out planning decisions is a skill we often take for granted. In our day to day lives, our world often surprises us, and yet we are able to adapt and recover. One of the biggest challenges UAV technology faces is dealing with the randomness and uncertainty involved with navigating the physical world. The ability of UAVs to reliably perform tasks is largely dependent on either a structured environment, or a robust planner. In robotics, achieving the ability to withstand adverse conditions comparable to humans has proven difficult. It is challenging for a robot to make planning decisions robust to the variability of the physical world. Roboticists have turned to using agile robots and planners to react quickly enough, or structuring

the environment to the capabilities of the robot.

We propose to bridge gaps in existing traditional robotic planning methods and the robustness required to operate in the unpredictable physical world. It is our hope that this work will help bring robots out of research laboratories and into positions of dependability in industry.

## 1.1 Thesis Statement

Prior research has provided a solid foundation of planning algorithms for coordination tasks, environmental monitoring, and especially path planning. While this body of work is comprehensive there exists a gap in extending standard methods towards considering the uncertainty of the physical world. This thesis addresses the challenge of planning under uncertainty for UAVs. We develop a suite of algorithms that are robust to changes in the environment and build on the key areas of research needed for utilizing UAVs in a commercial setting. Throughout this research three main components emerged: monitoring targets in dynamic environments, exploration with unreliable communication, and risk-aware path planning.

Robotic monitoring has become a main research topic in recent years, and caused robots to play a more integral role in collecting environmental data. Some monitoring tasks have proven difficult to automate due to challenging environments. For instance, one of the main issues in combating wildfires is monitoring the progression of the fire as it spreads. Live fire frontier monitoring by UAVs can help make rapid decisions regarding resource allocation and fire management. However, moving targets make

it arduous to use traditional monitoring techniques. Monitoring planners capable of tracking dynamic points of interest make for more robust planners to a world that is constantly changing. We use a realistic fire simulation to test persistent monitoring in an uncertain environment. The fire is generated using the standard program for modeling wildfire, FARSITE. This model was used to validate a weighted-greedy approach to monitoring clustered points of interest (POIs) over traditional methods of tracking a fire front. We implemented the algorithm on a commercial UAV to demonstrate the deployment capability.

The operating time of UAVs are typically limited by the capability of their batteries. This makes it challenging to complete complex exploration tasks without the use of multiple UAVs. However, obstacle dense environments can be restrictive to wireless propagation and make it difficult to communicate between UAVs. Coordinated planning improves exploration efficiency and motivates the need for adaptive, heterogeneous behavior. Previously, unreliable communication has caused critical failures for coordination planners. While recent work allows for intermittent connectivity [1], we address the problem where the environment is unknown and reconnectivity can't be planned for. Unreliable communication for coordination is demonstrated in the exploration domain, and shown to provide significant improvements over planners not able to share information. Our multi-robot coordination algorithm provides sophisticated meet, relay and sacrifice behavior to adjust for limitations on communication and battery life. We showed this behavior in simulation and with a team of two UAVs capable of multi-robot exploration in indoor environments. The contributions of this work enhance proficiency in managing unreliable communication for coordination.

Graph theory is an expanding field, and graph search algorithms have proven to be extremely useful tools in optimization, computer science, and robotics. Path planning has an extensive volume of literature associated with it, largely in graph search. Discrete graph search planners like A* and sampling-based methods like RRT* have emerged as standards in the field of robotics. The planners operate on known state and transition costs, which are sometimes difficult to obtain in the physical world. We build on this work by developing a planner over probabilistic representations of transition costs. By introducing this simple extension, high-level planners may provide robust paths to changing environmental conditions. The main novelty of this work is the introduction of a nonmyopic graph search algorithm for risk-aware planning. Risk-Aware Graph Search (RAGS) is a mechanism for deciding when to be conservative and when to be aggressive. With edge costs modeled as normal distributions, we can derive a principled way of leveraging information further down a path. The result is a lower probability that the executed path results in a high cost.

The main contributions of this thesis pertaining to robust planning decisions in unknown, uncertain, and dynamic environments are as follows:

- Improving monitoring capabilities of dynamic frontiers by leveraging the realistic fire simulator, FARSITE, and demonstrating the Weighted-Greedy algorithm on a UAV.

- Introducing a state machine for coordinating with unreliable communication, and simulating on teams of up to 8 UAVs. In addition, demonstrating on a team of 2 custom UAVs.

- Introduce risk-aware graph search (RAGS), a dynamic planning algorithm that plans over uncertain transition costs and demonstrating over satellite imagery for low flying UAVs in obstacle dense environments.

## 1.2 Thesis Summary

The following chapter, Chapter 2, introduces the fundamentals of normal (Gaussian) distributions, data clustering, and graph search. A review of the literature on conventional approaches to planning shows that prior work does not have methods for dealing with uncertainty. A history in coordination, monitoring, and path planning algorithms are presented. Chapter 3 explores robotic monitoring of dynamic frontiers. This section expands on how the uncertain environment can cause challenges in identifying regions to monitor. The problem domain is motivated and the related research is examined. A simulated wildfire is used to test the algorithm in addition to field experiments with a UAV. Chapter 4 investigates multi-robot coordination and presents a method for dealing with uncertain communication, specifically for the exploration domain. State of the art research is presented, and the problem is formally introduced. A solution is developed and tested in simulation and experiments. Chapter 5 introduces the planning problem with uncertain transitions. This chapter develops a more generalized framework for modeling and planning under uncertainty. We give theoretical analysis of the algorithm as well as simulated random testing and experiments on real satellite imagery. Lastly, Chapter 6 summarizes the contribution of each chapter and identifies directions for future research.

# Chapter 2: Background

"Uncertainty arises if the robot lacks critical information for carrying out its task. It arises from five different factors: environment, sensors, robots, models, and computation" Thrun et al. [2].

## 2.1 Basic Theory

This thesis builds on a range of work that includes: graph search, environmental monitoring, and probabilistic planning. This chapter develops the reader's understanding of probability in robotics. The beginning of this chapter introduces basic probability theories, while the latter section surveys the state of the art robotic planning algorithms using these theories.

### 2.1.1 Normal (Gaussian) Distributions

Techniques in this thesis use estimation and decision making in continuous spaces. In probability theory a random variable can characterize a continuous set of values to represent the space. A continuous random variable can be modeled by a probability density function (PDF). Probability density functions come in many different shapes (Normal, Poisson, Uniform, etc.), with the most common being normal (Guassian) distributions.

Normal distributions are common continuous probability models used to represent

real-valued random variables, for example: sensor measurements. The normal (Gaussian) distribution is the most common continuous probability model and random variable $x$ is defined by a mean $\mu$ and variance $\sigma^2$ in the following equation:

$$p(x) = (2\pi\sigma^2)^{-\frac{1}{2}} \exp\left\{-\frac{(x-\mu)^2}{2\sigma^2}\right\} \tag{2.1}$$

This is often represented by $\mathcal{N}(\mu, \sigma^2)$ and will show up many times later in this thesis.

### 2.1.2 Clustering

Uncertainty is represented and managed in many different ways. Just as a Gaussian function can represent an independent random variable, so can clustering be used to represent groupings of noisy data.

Clustering is the process of organizing objects based on their similarities. Grouping divides objects into smaller, more representational groups. An object is more similar to the objects in its own group as opposed to other groups. There are many types of clustering techniques; however, in this work, centroid-based clustering is used.

In centroid-based clustering the clusters are represented by a central vector, which may not necessarily be a member of the data set. K-means clustering [3] is an example of centroid-based clustering and uses a fixed number of clusters (K). The algorithm optimizes the squared distances from the centroid to each object in the cluster.

### 2.1.3 Graph Search

A graph is used to model a pairwise relationship between objects and is made up of vertices (nodes) and edges. A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is made up of a set of vertices $\mathcal{V}$ connected by a set of edges $\mathcal{E}$. In robotics graphs are often used to model states of a robot in the world. Searching through the possible states from a source to a goal is the traditional method for generating a plan. A generic form of a graph search algorithm is presented in Algorithm 1. The difference between graph search algorithms is the method by which the next node to expand is selected.

**Algorithm 1** Generic Graph Search (Graph, Start, Goal)

```
open_set ← []
closed_set ← []
next_node ← start
while next_node isn't goal do
    closed_set += next_node
    open_set += successors of next_node, not in closed_set
    next_node ← select from open_set
    remove next_node from open_set
return next_node
```

## 2.2 Standard Approaches

The focus of this work is to expand traditional planning methods to account for uncertainty in the environment. The following sections will introduce prior work relevant to deploying robots in the physical world, and the most prevalent techniques used today. Additionally, each section will contain a review of recent work in incorporating uncertainty into the planners. The three main topics include coordination planners, monitoring

algorithms, and path planners.

### 2.2.1 Coordination Planners

Coordination of multiple robots was initially studied in the 1980's [4] and since expanded into famous examples like Amazon's autonomous warehouse and the world robotic soccer competition RoboCup. Research directions have expanded into task planning, motion planning, exploration, and communication.

Multi-robot coordination methods can be separated into two general categories: (1) decentralized approaches and (2) centralized approaches. Centralized planners require a single global controller that gives plans to each agent. While these planners are generally more efficient, they often fail if communication is not guaranteed or if the planners' knowledge of the environment is incorrect. In contrast to centralized approaches, decentralized planners scale better and are robust to uncertain communication. However, a decentralized planner can be suboptimal with respect to the global objective if the agents only consider their local knowledge. This limited knowledge can lead to varying communication schemes between decentralized planners to coordinate tasks.

Communication can be categorized into explicit or implicit communication. Each communication model has different coordination challenges. Explicit communication allows each robot to exchange information directly with other robots. In contrast implicit communication shares information through the environment. As an example of explicit coordination, Hollinger and Sing design a periodic connectivity framework for search, survey and cover problems [1].

### 2.2.2 Monitoring Algorithms

Work on autonomous information gathering began with early work in sequential hypothesis testing [5], which focused on determining which experiments could efficiently classify the characteristics of an unknown. This line of research developed into more general approaches and evolved into the field of active perception [6]. Similar insights led to using optimization techniques on robotic information gathering problems. It is important to note that persistent monitoring is different than the coverage problem because it manages surveillance over a changing environment where each target must be visited infinitely often and cannot be fully covered by the monitoring agents.

Traditional monitoring algorithms began with examining the sweep coverage problem as seen in [7] and [8]. Sweep coverage is different than static coverage in that each point of interest in sweep coverage is time-variant as long as a coverage period is guaranteed. Fortunately, research has been directed towards minimizing the uncertainty of the environment monitored. Specifically, Cassandras et al. developed algorithms for minimizing long-term information uncertainty [9, 10], which is critical to robotic monitoring autonomy.

### 2.2.3 Path Planners

Planning is finding a set of actions that transitions from an initial state to a final goal state. States are the configuration of the agent in the world, while transformation between each configuration is the action of an agent. There is a cost correlated with each transition, the cost may be energy, time, distance, or some other metric. A planner finds

a path (series of actions) between the initial and desired goal states. This path can be *optimal*, where the cumulative sum of all costs is the lowest over all possible paths. A planner is considered optimal if it will always find the lowest cost path. In addition, a planner that always finds a path in finite time (if it exists) is considered to be *complete*.

There are several approaches to finding paths given some representation of the environment. There are two types of environmental representations: discrete and continuous [2]. In discrete space, path planners are deterministic and divide the environment into an interval-based graph structure. By using a discrete representation the dimensionality of the planning problem is reduced. Not only does this simplify the planning problem, it also allows the quality of the solution to be bounded by the level of discretization.

In continuous space, sampling-based planners have emerged as the most commonly used planners [11]. In sampling-based path planners the environment is randomly sampled and a path is incrementally refined. Informed methods of sampling and connecting the paths, like in *RRT**, can guarantee optimality as well.

For additional reading on heuristic-based path planners Ferguson et al. [12] provide a clear and concise overview. An in depth review of sampling-based methods is presented by LaValle [11], and a summarized review of these algorithms will be presented in the following few sections.

#### 2.2.3.1 Discrete Space Planners

In 1959 Dijkstra's search algorithm for finding the shortest path between nodes on a graph was published [13]. The basic premise of Dijkstra's algorithm is to expand the

search by visiting the next closest neighbor from the source until reaching the goal. The search is over nodes on a graph with edges of fixed and known costs connecting the neighboring nodes.

A simple extension to this is directing the search towards the final desired state by using a heuristic. A heuristic is a metric for roughly estimating the quality of a guess, it is in no way guaranteed to be accurate [14]. A heuristic is considered admissible if it never overestimates the cost from any state to the goal, this is important because it guarantees that the final path is optimal. By using a heuristic in combination with Dijkstra's algorithm the search will be guided towards states closer to the desired state, saving time and computing power by potentially limiting the number of states explored. This algorithm is one of the most popular search algorithms, known as $A*$ [15]. $A*$ can efficiently find an optimal path between two discrete states as long as the heuristic is admissible, and there are known transition costs between each state. A notable limitation to this algorithm is planning over changing transition costs.

Dynamic $A*$ ($D*$) [16], and the more efficient version $D*$ Lite [17], were introduced as adaptable versions of $A*$. These planners are able to work with uncertain, changing graphs. The $D*$ Lite algorithm behaves like $A*$ but saves graph information after a plan is found. The saved information reduces computational complexity if replanning is needed due to unknown obstacles, or changing edge costs in the environment.

#### 2.2.3.2 Sampling-Based Planners

An alternative approach to planning, are sampling-based techniques. The major draw to sampling-based planners is their ability to efficiently create plans in high-dimensional spaces and operate in continuous space. These algorithms sample the configuration space (C-Space) of the planning problem, where C-Space is the collision free region accessible by the robot. The generic form of a sampling-based planning algorithm is described in Figure 2.1.

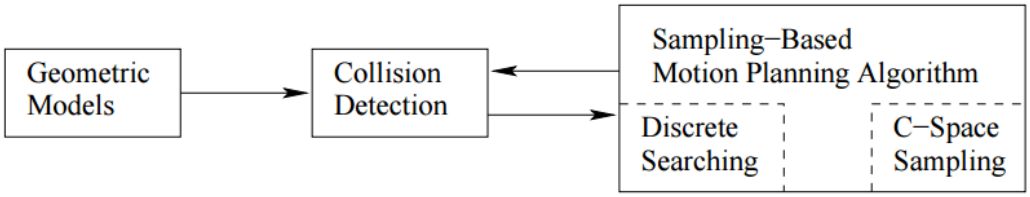



Figure 2.1: sampling-based planning philosophy diagram demonstrating the separation of sampling/searching and collision detection. Image taken from Planning Algorithms, LaValle [11].

One of the first sampling-based algorithms is known as probabilistic roadmap (*PRM*). A probabilistic roadmap (*PRM*) randomly samples the configuration space, connects neighboring points, tests for collisions, and finds a path from the start state to the goal state [18]. This method is probabilistically complete, meaning it will find a solution if one exists, as long as it can sample the free space infinitely. An asymptotically optimal version, known as PRM*, uses a variable connection radius that scales $\log(n)/n$ where $n$ is the number of samples.

Similar to *PRM*s, the rapidly exploring random trees (*RRT*) method builds a connected graph by sampling the space, however this graph is connected after each sampling. As each point is sampled the graph is extended a fixed distance towards the point

then checked for collision. When the goal state can be connected to the graph, a path is found [19]. *RRT** is the asymptotically optimal version of *RRT*, where connections are updated to new nodes if they result in a shorter path to the node than the previous connections [20]. Many different sampling methods exist presenting interesting and efficient variants of these base algorithms.

All these sampling-based planners make assumptions that the robot is operating on a known state, and transition functions. To address the problem of motion planning in the presence of state uncertainty, Bry and Roy introduced a novel sampling-based motion planning method [21]. The Rapidly-exploring Random Belief Tree algorithm ensures a bounded probability of collision by balancing reduction of uncertainty with low cost paths. This addresses half of the problem, the other half is the uncertainty of representing the environment.

## 2.3 Summary

The concepts overviewed in this chapter provide a platform of understanding for research in planning under uncertainty. Each subsequent chapter will explore a different gap in the current research, and bring concepts introduced in this chapter to address those challenges. First, a significant weakness in monitoring research is how to effectively monitor moving point of interest. We will build on current research by investigating how to effectively monitor these dynamic points, specifically along a fire frontier. Second, using multiple vehicles can provide more coverage, better efficiency, and add robustness to the system, however an unaddressed challenge to using multiple robots is

dealing with unreliable communication. We will cover this gap with a communication framework, specifically for the challenging indoor exploration domain. Third, when these systems are planning paths through uncertain environments and terrain the paths are often unreliable and slow. A clear gap in planning research is incorporating path uncertainty. We will build on current path planning research by using traditional methods of planning while incorporating uncertain traversal cost into the planner.

# Chapter 3: UAV Planning for Dynamic Wildfires

One of the benefits of UAVs is that they provide sensing capabilities other ground based platforms lack. These robots are being implemented in various domains, but specifically show promise in applications hazardous for humans. In the future, aerial monitoring and inspection tasks will likely be a primary sector in commercial markets. Real world applications for monitoring are likely not all stationary targets. We begin by investigating the challenge of monitoring dynamic points of interest. We specifically study the uncertainty around monitoring a wildfire frontier. Studying wildfires has an obvious benefit when considering the human cost spent combating them.

One of the main issues in combating wildfires is monitoring the progression of the fire over time [22]. Live fire frontier monitoring can help produce quicker decisions and result in better resource allocation and fire management [23]. During wildfires, the information available to the Incident Commander (IC) is critical. Current methods of tracking a fire involve a human pilot flying several miles away from the fire and verbally reporting to the IC what trends they see in the fire. Satellite imaging is also available but is often rendered useless by smoke. In 2012, there were a total of 67,774 fires, destroying 9.3 million acres, and costing over $1.9 billion to suppress in the U.S. alone [24].

This chapter develops different hotspot monitoring algorithms to gather important information for the Incident Commander (IC) managing the wildfire. This research

The work from this chapter was published in Field and Service Robotics 2015.

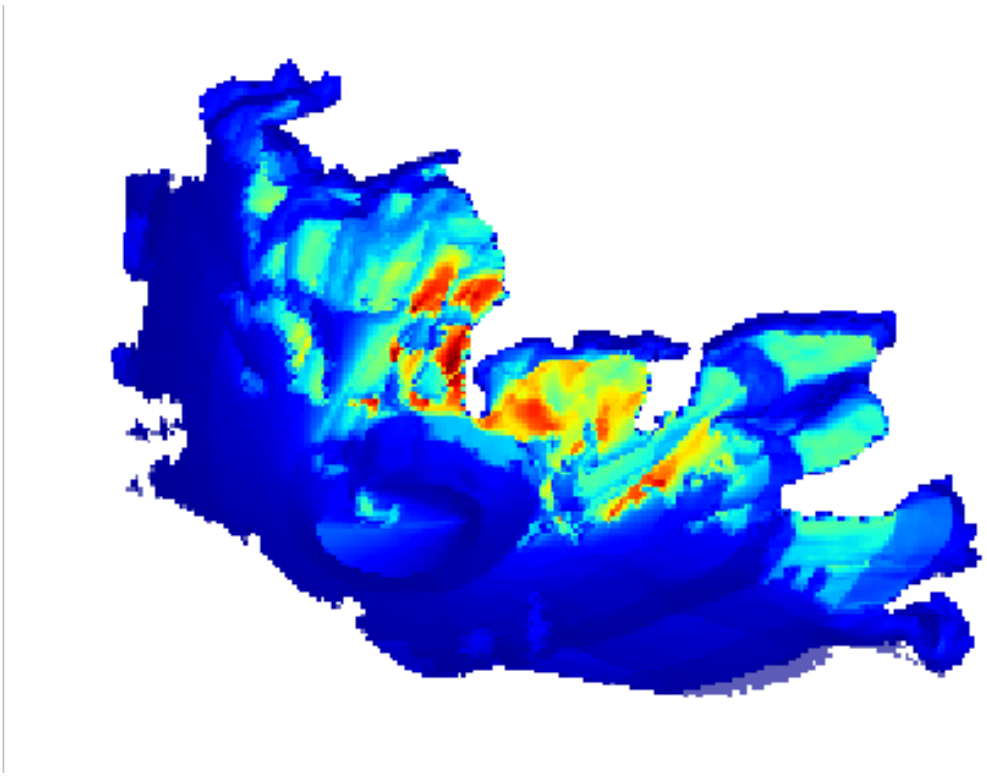


Figure 3.1: Wildfire simulation example (red areas correspond to hotter areas of the fire). We propose a weighted-greedy algorithm for optimizing the monitoring trajectories of aerial vehicles in wildfire scenarios.

aims to improve a UAV's effectiveness in gathering valuable information for the IC. To simulate wildfires, a program developed by the Department of Agriculture and Forest Service is used. FARSITE is a free program used by the U.S. Forest Service, National Park Service, and more specifically ICs, to predict the fire's behavior using data on the topography, weather, wind, moisture, and fuel [25]. FARSITE exports various characteristics of the fire; while our simulation (FLAME) uses fireline intensity data (BTU/ft/s). Other fire metrics like flame length, and rate of spread are also incorporated. See figure 3.1 for a fireline intensity map of a simulated fire.

Wildfires are highly unpredictable, acting as a unique dynamic frontier. Dynamic monitoring has been explored [26, 27], but fire frontier monitoring is a largely unexplored domain. Our simulator (FLAME) models a dynamic fire frontier and uses techniques like Mini-Batch K-Means Clustering to achieve a similar problem formulation as related monitoring research.

Tracking the most volatile locations on the fireline will give valuable information for the IC. These hotspots will be intelligently monitored by the UAV, using algorithms for minimizing the time hotspots are left unmonitored.

While wildfires were the chosen domain, this research is not limited to wildfires. Similar application domains with dynamic frontiers include: algae blooms, pollution spills, and military battles [28, 29]. These similar domains can also be analyzed using the techniques developed in this chapter.

The main novelties of this chapter include: (1) a simulation (FLAME) which uses realistic fire modeling software for accurate fire characteristics, (2) a novel fire tracking algorithm which outperforms existing methods, (3) the first investigation into adaptive

monitoring of hotspots along a dynamic frontier, and (4) hardware experiments demonstrating the ability to implement our work with existing technology. Taken together, these contributions provide a new approach to the general problem of monitoring dynamic frontiers.

The remainder of this chapter is organized as follows. First, we establish the current state of related research (Section 3.1). Following that, we overview the problem and assumptions we made during our investigation (Section 3.2). Next, we describe the simulation and a novel approach to frontier monitoring (Section 3.3). Finally, the algorithm is described in detail in (Section 3.4), the simulation results are presented (Section 3.5), and the hardware experiments are discussed (Section 3.6).

## 3.1 Related Work

Robotic systems are becoming more commonly used as mass data-gathering tools by scientists [26, 30, 31]. Robots are already collecting large datasets on environmental change. Algae blooms, pollution, and other climate variables are application domains for persistent monitoring techniques. Persistent/adaptive monitoring in robotics is currently a growing research topic. Prior work has explored different approaches to monitoring stationary and dynamic feature points. While these adaptive sampling techniques focus on optimizing uncertainty levels in static [9, 32, 33] and in dynamic environments [34, 26], prior work often focuses on systems in obstacle-free environments. Some research has examined collision avoidance [35, 36], but adaptive sampling along dynamic frontiers remains an ongoing research problem.

In [37], fire frontier tracking was integrated into a simulation for determining UAV tracking accuracy of the fire perimeter. Our baseline uses the same method of UAVs follow a circular path around the fire. However, our metric is to track the most active parts of the fire. We compare the baseline against our weighted-distance algorithm. Our research presents the first investigation into adaptive monitoring of hotspots along a dynamic frontier. We span the domains of hotspot monitoring and dynamic frontier tracking to evaluate path planning techniques in our FLAME simulator. This line of work allows us to test new algorithms in real-world scenarios.

## 3.2 Problem Formulation

We will now formally introduce the problem domain and the assumptions we made. We will also introduce the metric we use to evaluate our algorithm against the baseline.

We assume that GPS and communication between the IC and the vehicle are always available. This means the UAV can always localize itself and never needs to return to the starting location to transfer collected data. We assume the UAV always has the simulated fire frontier in order to find the hotspot locations. Additionally, the UAV is assumed, for comparison purposes, to have unlimited endurance.

Each hotspot has a corresponding time since last tracked by the UAV and the maximum time its been left untracked ($\phi$) in the past. The sum of $\phi$ of all hotspots was chosen as the metric to evaluate the effectiveness of an algorithm. In this chapter, fireline intensity is used as the crucial information needed by the IC. The intensity is monitored through the clustering into hotspots, directly relating to the goal of providing the IC with

up-to-date information about the fire progression.

$$J(t) = \sum_{i=0}^{hotspots} \phi_i \,, \tag{3.1}$$

The goal is to minimize the metric $J(t)$, which corresponds to timely hotspot monitoring, through an optimized trajectory for the UAV.

## 3.3 FLAME Simulation

We will now explain our simulation and how we developed each of the different components. Figure 3.2 should be used as a reference of the state transitions in the simulation. There are two aspects to the weighted-greedy algorithm, (1) picking which hotspot to go to, and (2) how to get there.

Fire data is generated using FARSITE, the wildfire simulator currently used by ICs during wildfire management [25]. The data is exported in the form of time of arrival, and a measurable characteristic of the fire. In this work we use the fireline intensity at each location. As stated above, the task is to minimize the sum of max time untracked ($\phi$) over all hotspots. At mission start, the UAV must first find the fire and begin identifying the hotspot regions.

Tracking a hotspot is done by calculating the distance between a previous set of hotspots relative to a new set. To determine when a hotspot moved as the fire progressed, a threshold is implemented. If a hotspot is not within the distance threshold of any previous hotspots, it is then classified as a new hotspot. Even after careful tuning, this approach can still lead to some untracked hotspots where the hotspot existence is too

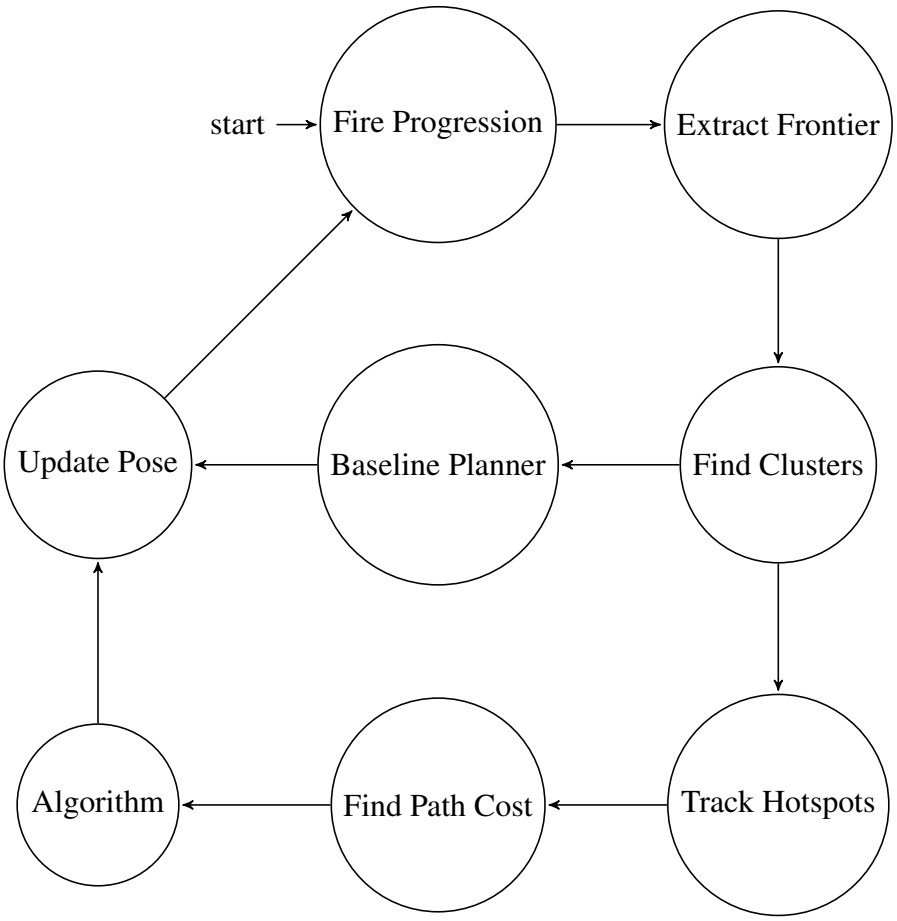



Figure 3.2: State diagram of FLAME

short for any response by the UAV.

To identify hotspots, all points along the frontier with a fireline intensity above a normalized threshold are parsed using a clustering technique called Mini-Batch K-means [38]. K-means clustering was chosen because it directly relates the number of interest points (how active the fire is) to the number of cluster centers (hotspots). The desirable amount of clusters (K) changes as the fire evolves. We actively determine the K value for adaptive hotspot extraction with the following formula,

$$K = \sqrt{N/2}, \tag{3.2}$$

where K is the number of centers, and N is the number of interest points.

FLAME uses A* path planning for generating paths from the UAV to hotspot locations around the fire. This method works better for estimating path cost compared to a simple Euclidean distance estimate due to the spherical tendency of the fire spread. Other similar methods were explored to increase efficiency, such as Jump Point Search. Jump Point Search gives respectable speed gains in environments with large open spaces, however the UAVs path remained mostly along the fire frontier. Methods like wall-following could provide faster simulation times, but lack expandability to more complex frontiers, and provide less accurate path costs. Due to the shape of the fire, any benefits of these alternatives were determined to be inconsequential. It was therefore determined to use the A* search algorithm as the UAV's path planner.

A cost map is passed to the A* algorithm, and is generated by applying a blur to the map of the fire up to that point in time and assigning a high cost to areas within the fire.

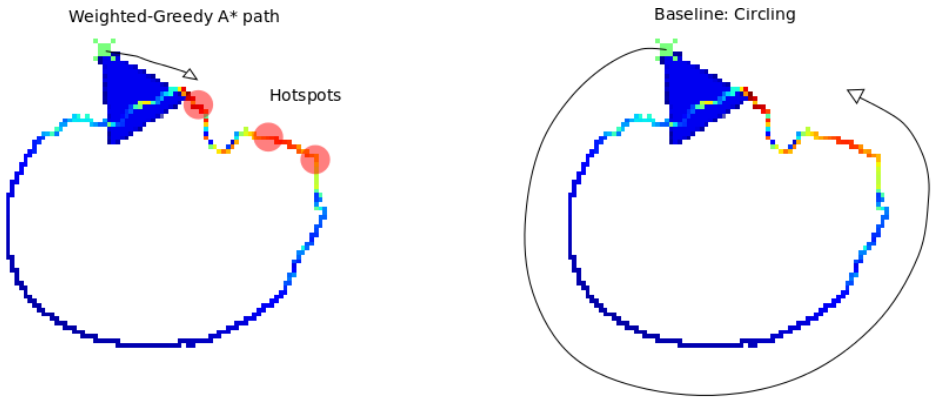
This helps ensure the path generated for the UAV is not within dangerous proximity of the fire, but can still be navigated close enough to monitor the hotspots. The algorithms were tested over seven different fires generated in FARSITE. The baseline and proposed weighted-greedy algorithm are described in pseudocode in Algorithms 2 and 3.

## 3.4 Algorithms

The proposed algorithm is evaluated against a baseline in the following tests. The following sections will describe each algorithm and how it was implemented in the FLAME simulation. The first monitoring technique described is used as the baseline comparison. It exemplifies current tactics utilized in real world wild fire monitoring, and prior research into UAV fire monitoring [37]. This is compared to our proposed approach, a weighted-greedy algorithm that moves to the hotspot that has remained untracked the longest with a tunable parameter weighting the distance of the hotspot from the UAV. Figure 3.3 should be used as a reference for the difference between the two algorithms behaviors.

### 3.4.1 Baseline

The baseline model travels parallel to the dynamic fire frontier. Calculating a 90 degree transformation of the vector from the UAVs current location to the nearest point on the fire frontier gives the travel vector of the UAV. Maximum and minimum distance thresholds are imposed on the UAV so it can then move along the frontier monitoring hotspots while maintaining a safe distance from the fire. We use this as a baseline




(a) UAV monitoring the fire using proposed algorithm identifies and tracks the most important part of the fire.

(b) UAV monitoring the fire by constantly circling will continue regardless of the state of the fire.

Figure 3.3:

comparison based on the work of [37].

### 3.4.2 Weighted-Greedy

The weighted-greedy algorithm checks the untracked time of every live hotspot, calculates the distance to it, and targets the hotspot with the highest score. Unlike the baseline, the weighted-greedy algorithm makes target decisions based on the current state of the hotspots.

This is done using the following formula where $\mathcal{H}$ is the target hotspot, $\mathcal{T}$ is the untracked time of each hotspot and $\mathcal{C}$ is the path cost to each hotspot:

$$\mathcal{H} = \underset{h}{\operatorname{argmin}}\, \mathcal{T}_h - \alpha * \mathcal{C}_h \tag{3.3}$$

The proposed algorithm accounts for the distance to each hotspot when choosing

**Algorithm 2** Baseline Algorithm

Inputs: UAV_Location, frontier
**for** all points in frontier **do**
    points.distance = $\sqrt{(\text{points.x - UAV\_location.x})^2 + (\text{points.y - UAV\_location.y})^2}$
closest_point = min(points.distance)
vector_to_nearest = ([UAV_location.x - closest.point.x], [UAV_location.y - closest.point.y])
normalized_vector = vector_to_nearest / distance_to_nearest
**if** dist_to_nearest $>$ max_distance_to_fire **then**
    travel_vector = vector_to_nearest
**else if** dist_to_nearest $<$ min_distance_to_fire **then**
    travel_vector = -vector_to_nearest
**else**
    travel_vector = (-vector_to_nearest.x, vector_to_nearest.y)
path = travel_vector

**Algorithm 3** Weighted Algorithm

Inputs: hotspots{location, time_untracked}, $\alpha$, UAV_location
**for** all h in hotspots **do**
    h.path, h.path_cost = ASTAR(h.location, UAV_location)
    h.score = h.time_untracked $- \alpha *$ path_cost(h)
    **if** hotspot.score $>$ target_hotspot.score **then**
        target_hotspot = h
path = target_hotspot.path

the targeted hotspot. The weighting factor $\alpha$ is a parameter evaluated in figures 3.4, 3.5, and 3.6. The use of a weighting factor addresses some sub-optimality of using just a greedy algorithm. The weighting parameter helps intelligently pick a hotspot that may not be the longest untracked but is closer to the vehicle. A greedy algorithm will immediately move towards the hotspot with the longest time left untracked, disregarding any nearby hotspots that may not have been untracked for nearly quite as long.

## 3.5 Results

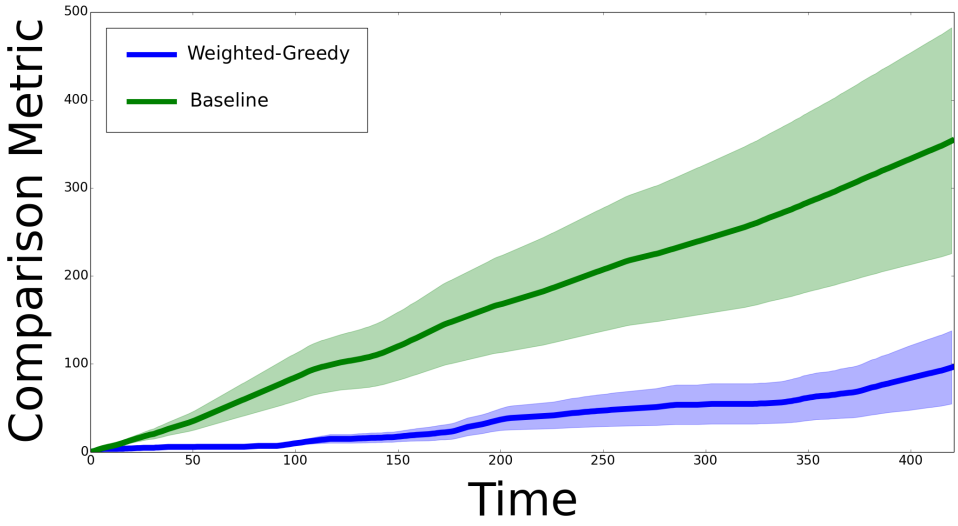
Using our FLAME simulator, we can compare our proposed weighted-greedy approach with traditional methods of monitoring wildfires or dynamic frontiers. The simulation was run on an Intel i7-4702HQ processor with 8 Gb of RAM. The UAV's decision and planning methods took an average of .74 seconds to complete. This is fast enough for a UAV to implement in the field (see section 3.6).

In comparison to the baseline, the weighted algorithm provided substantial improvement over the course of the trials. The plots in Figures 3.4, 3.5, and 3.6 show the performance of the two monitoring algorithms performance with different parameter settings. As previously discussed, the weighting parameter ($\alpha$) is multiplied by path cost to the hotspot location. The hotspot cutoff $\beta$ is the normalized threshold for a spot along the fire to be considered an interest point. This directly affects the total number of hotspots. Tests run with a lower $\beta$ will generate a higher number of hotspots for the UAV to track. Time on the x axis begins at first cluster appearance during the simulation. The y axis shows the results of the comparison metric $J(t)$.

The averaged score over the seven fires are depicted as the bold lines. Around each line, the standard error of the mean is represented by the shading. Figure 3.5 shows the simulation results with a hotspot threshold $\beta$ = .35. The plot shows the results with a corresponding $\alpha$ value of .5. Our proposed algorithm performs significantly better than currently used approach.

In figure 3.4 the simulation is run with a $\beta$ equal to .25. The $\beta$ value (.25) is the lowest used and figure 3.4 shows the performance of both algorithms in an environment

with the corresponding large set of hotspots.



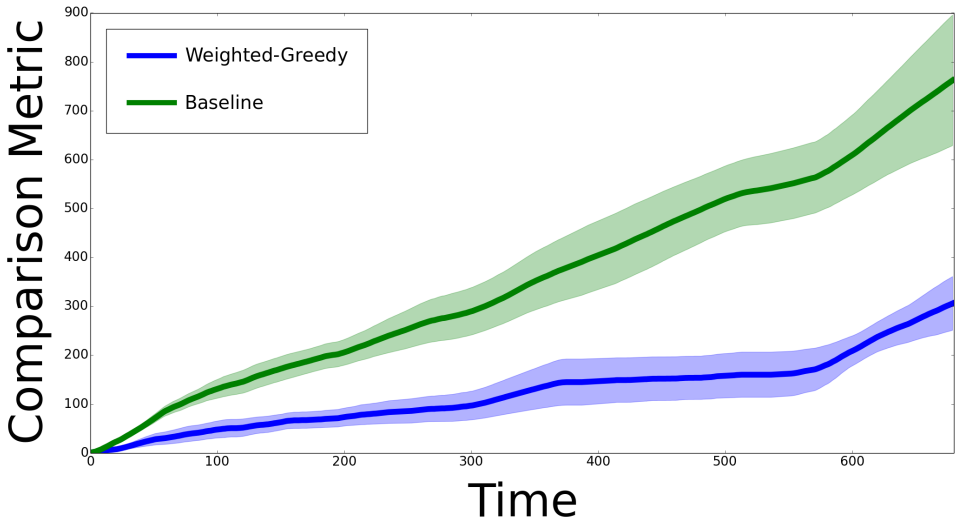

(a) Weighting parameter $\alpha$ = .5 Hotspot Threshold $\beta$ = .25

Figure 3.4: Wildfire simulation, where the comparison metric is J(t) or the sum of max time untracked of all hotspots. Lower is better. The weighting parameter $\alpha$ is set at .5. The normalized threshold $\beta$, for a spot along the fire to be considered an interest point, is set to .25. A lower $\beta$ corresponds to more hotspot locations. Error bars are one SEM.

Figure 3.6 depicts the simulation results with a hotspot threshold $\beta$ at .45. This trial uses the highest $\beta$ (fewest number of hotspots), and shows the performance with $\alpha$ value at .5. The standard error of the mean (SEM) for both algorithms is significantly higher in this test environment.

The results demonstrate the weighted-greedy algorithm's ability to outperform the baseline in environments with only few clusters, or many clusters. In all cases presented here the proposed algorithm showed significant improvement over traditional wildfire monitoring methods. Our algorithm better tracks the dynamic regions of a dynamic frontier, providing valuable data to the IC.

An interesting characteristic of the frontier monitoring is that it may be simplified



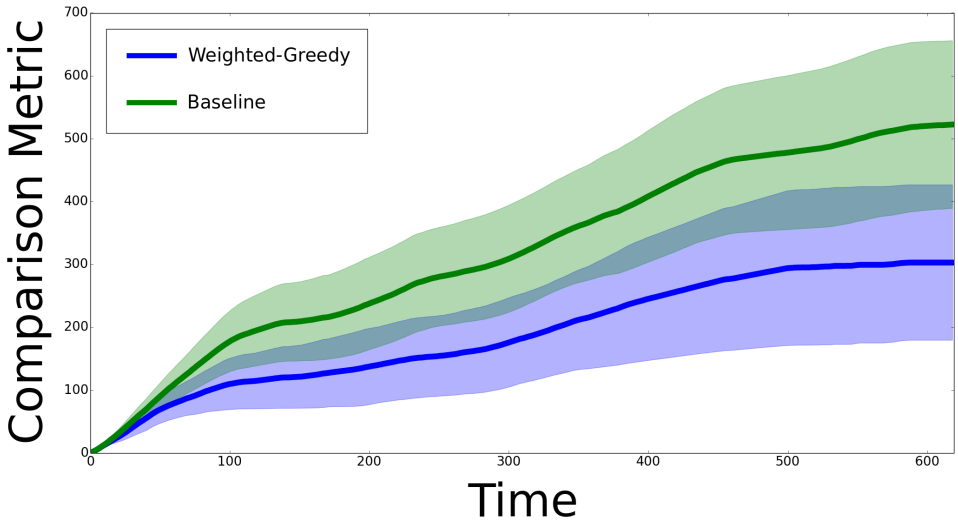

(a) Weighting parameter $\alpha$ = .5 Hotspot Threshold $\beta$ = .35

Figure 3.5: Wildfire simulation, where the comparison metric is J(t) or the sum of max time untracked of all hotspots. Lower is better. The weighting parameter $\alpha$ is set at .5. The normalized threshold $\beta$, for a spot along the fire to be considered an interest point, is set to .35. A lower $\beta$ corresponds to more hotspot locations. Error bars are one SEM.

into a 1-dimensional problem. Each timestep the UAV must decide between two options, if it wishes to move clockwise or counter-clockwise. Further investigation into leveraging this characteristic may be worthwhile.

## 3.6 Hardware Experiments

To demonstrate the feasibility of the proposed algorithm, we implemented the algorithm on a live test. To test on hardware we set up the FLAME simulation as a ground station that acted as live satellite data would for a real fire. The algorithm then sent a live stream of coordinates to a UAV to monitor the fire. While a real fire was not used for purpose of this test (for safety reasons), we are able to demonstrate that a UAV can effectively



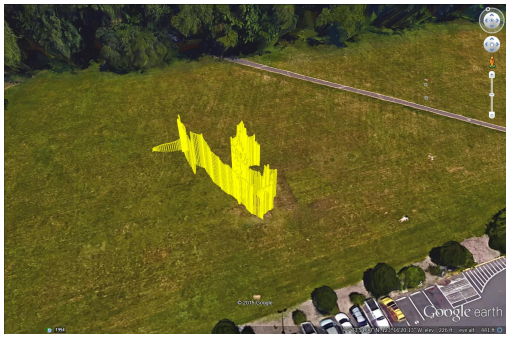

(a) Weighting parameter $\alpha$ = .5 Hotspot Threshold $\beta$ = .45

Figure 3.6: Wildfire simulation, where the comparison metric is J(t) or the sum of max time untracked of all hotspots. Lower is better. The weighting parameter $\alpha$ is set at .5. The normalized threshold $\beta$, for a spot along the fire to be considered an interest point, is set to .45. A lower $\beta$ corresponds to more hotspot locations. Error bars are one SEM.

perform these tasks.

We converted FLAME into a ROS package to use the MAVROS plugins [39]. MAVROS acted as a communication bridge between FLAME and the flight controller on the UAV. This allowed us to update the UAV's path in time with the simulation. We used a tethered IRIS+ quadcopter as the platform for these experiments, see Figure 3.7. We ran the experiment for over 10 minutes, about half the max flight time of the vehicle. The experiment was performed outdoors in about a 60ft x 60ft area. The simulation coordinates were scaled and transformed to GPS degrees to support sending waypoints. We present the path of the vehicle around the fire in figure 3.8.

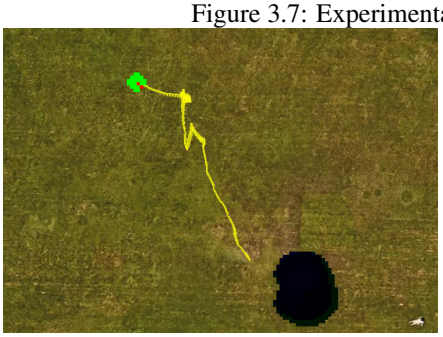
The UAV successfully followed the trajectories generated in the simulation to the best locations along the fire to monitor it as it spread. This illustrates our ability to be-



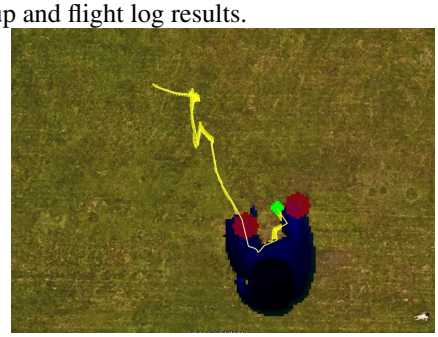

(a) A 3d view of our flight log in Google Earth showing the flight distance and trajectories.

(b) Experiment set up, computer running live simulation and IRIS+ flying autonomously (tethered).

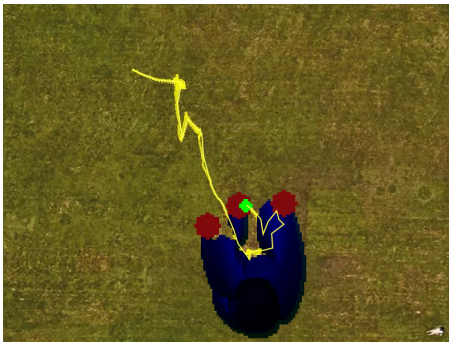
Figure 3.7: Experimental setup and flight log results.



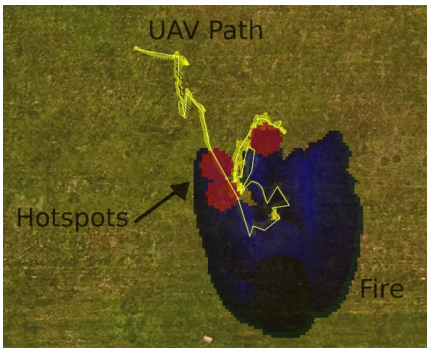
(a) The UAV first starts off a safe distance away from the fire and must travel to the frontier.



(b) Upon reaching the frontier and identifying a hotspot the UAV stays outside the burn area as it grows.



(c) The UAV moves from one hotspot to another to reduce the untracked time.




(d) Final fire size and flight log of our field experiments.

Figure 3.8: Four images demonstrating the algorithm path planning during field tests.

gin introducing robotic monitoring into these dynamic monitoring situations and gather valuable data from it.

## 3.7 Discussion

In this chapter, we have introduced FLAME, a simulation developed for testing monitoring techniques on a dynamic frontier, or more specifically a wildfire. The two algorithms tested in the simulation have demonstrated that there is significant benefit in a weighted-greedy approach for selecting hotpot waypoints over the baseline method of flying around the fire frontier. Using Mini-Batch K-Means Clustering for identifying hotspots, our proposed weighted-greedy algorithm optimized for $J(t)$, the sum of max untracked time of all hotspots. Three different normalized hotspot thresholds ($\beta$) (.25, .35, .45) were used. Results showed the weighted-greedy algorithm with significant improvements over the baseline.

These algorithms depend on global knowledge of the fire, or more specifically where the hotspots are. Future work will include implementing a probabilistic model of hotspot locations and studying the exploration/exploitation trade-off for tracking and updating the model. In this chapter we assume the UAVs have unlimited flight time. However, the cost of flight with limited endurance is an important factor. Additionally, hotspots are not all equal, and things such as risk to critical areas will need to be considered. Continuation of the project will also focus on implementation of multiple UAVs and the introduction of common fire monitoring challenges, including smoke and adverse weather conditions.

# Chapter 4: Coordinated Exploration under Unreliable Communication

In the previous chapter, we only considered a single UAV in the planning problem; now, we will build on that work by planning with multiple UAVs. Coordinating more than one UAV involves a stream of communication. The reduced cost and increased reliability of autonomous vehicles and sensing technology has made it possible to field multi-robot teams capable of mapping unknown environments. High-impact applications include urban search and rescue, military reconnaissance, and underground mine rescue operations. However, with unreliable communication coordination algorithms often break down. We used the challenge of coordinated exploration to address unreliable communication in unknown environments.

The algorithm presented here is applicable across multiple domains, but is primarily motivated by the use of UAVs to explore indoor environments (e.g., buildings, caves, and mines). In such environments, clutter and rubble often make the use of ground vehicles impractical, and fixed-wing aircraft are not appropriate because of tight spaces and obstacles. Small autonomous rotorcraft (e.g., quadcopters) are capable of highly-agile operation within confined spaces and are not impeded by ground obstacles [40]. As such, they are uniquely suited for the previously noted exploration tasks.

A key limitation of small autonomous rotorcraft is that they typically have limited battery life (often on the order of tens of minutes) [41]. This limitation restricts their

The work presented in this chapter was published in the International Conference on Robotics and Automation 2015.

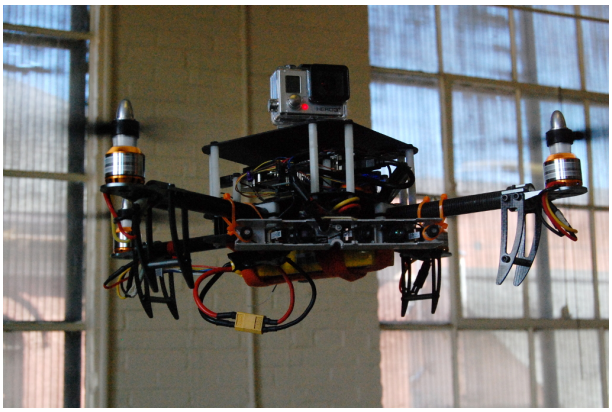


Figure 4.1: Unmanned aerial vehicle (UAV) flying in an indoor environment. Our proposed algorithm allows UAVs to "sacrifice" themselves by continuing to explore even when they do not have sufficient battery for returning to the base station. The resulting information is then transmitted back to the base station through the use of relay UAVs.

operating lifespan and makes it challenging to complete complex exploration missions in a limited amount of time. The use of multiple rotorcraft is one method of overcoming this limitation, their combined flight time may be sufficient to complete the mission. However, many indoor environments create restrictions on communication between vehicles (e.g., WiFi range and signal strength limitations), which makes it difficult to communicate between team members. Finally, these scenarios of interest do not allow for instrumenting the environment with wireless access points or cameras as a means of improving communication or localization. The combination of these factors motivates the design of multi-robot coordination techniques that allow for adaptive, heterogeneous behavior to improve the exploration efficiency of the team.

The proposed algorithm utilizes four states – explore, meet, sacrifice, and relay – to improve exploration efficiency in communication-based coordination scenarios. When in the explore state, a robot uses an existing frontier-based exploration algorithm to generate it's goals. When in the meet state, a robot returns to a previous location in

attempt to rendezvous with team members. Subsequently, in the sacrifice state, a robot continues exploring even when it no longer has sufficient battery life to return to the base station. Once the robot has nearly exhausted its battery, it may land to act as a communication relay.

The proposed algorithm is the first to provide heterogeneous exploration capabilities with limitations on communication and battery life. The key novelties of this chapter are: (1) a multi-robot coordination algorithm that provides sophisticated relay and sacrifice behavior to adjust for limitations on communication and battery life, and (2) the design and implementation of a team of low-cost autonomous rotorcraft capable of multi-robot exploration in indoor environments. The contributions of this work enhance proficiency in managing unreliable communication for coordination.

The remainder of this chapter is organized as follows. A discussion of related work in multi-robot exploration is shown in Section 4.1, highlighting the need for an adaptive, heterogeneous approach. The next section will formulate the multi-robot exploration problem (Section 4.2) and present the proposed coordination algorithm (Section 4.3). Following this, simulations demonstrating the benefit of our proposed approach versus a frontier-based exploration algorithm are presented (Section 4.4). We demonstrate an implementation on a team of custom-built autonomous quadcopters (Section 4.5) and finally, we conclude and discuss avenues for future work (Section 4.6).

## 4.1 Related Work

The research in robot coordination in this chapter focuses on the difficulties of real-world environment mapping with autonomous robots. There is often unreliable communication between robots, making coordination difficult. While previous research has focused on exploration algorithms [42, 43] and maintaining mesh networks [44, 45], we focus on coordinating with unreliable communication. Additionally, power limitations cause difficulty when implementing previous research on physical systems [41]. Finite power limits the usefulness of exploration robots in large environments. Our research addresses these limitations in developing an optimized control algorithm.

Prior work has examined a number of algorithms for coordinating robots to explore environments, including using stochastic differential equation solutions [46]. Path planning for aerial vehicles has also been considered in conjunction with state estimation [47, 48]. Coordination algorithms have been applied to autonomous ground vehicles in real-world environments [49]. However, these prior works do not consider heterogeneous behaviors that allow for improved efficiency with unreliable communication and limited battery life. Our work is designed to bridge this gap.

The proposed algorithm is compared to a frontier-based exploration baseline [42]. While there is extensive research in exploration algorithms, frontier-based exploration was chosen because it allows for fully distributed operation. Prior work has shown that frontier-based algorithms perform competitively with alternative approaches for indoor exploration tasks [50]. Notably, the proposed algorithm may use any exploratory algorithm, such as market-based approaches [51], as its core baseline (see Section 4.3).

Previous research [42] requires each robot to return home for collection of its map of the environment. This constraint wastes precious energy since robots are returning through explored areas. A key contribution of this work is the ability to build on a core exploration algorithm by adding heterogeneous relay and sacrifice behaviors to improve the efficiency of operation.

## 4.2 Problem Formulation

We are given the task of coordinating a team of robots to navigate an unknown environment and maximize exploration for their given battery life. We assume the robots are able to communicate intermittently, limited by obstacles or range. We also assume that sacrificing robots for a gain in exploration is acceptable. The robots only have communication with the base station when they are in range. To be useful, at least one robot must return to the base station at a pre-specified location.

More formally, we are given $K$ robots with limited battery life. We denote the battery life for robot $k$ as $B_k$. A robot may stop moving (land) to conserve battery. The goal is to explore the maximal area of a bounded unknown environment and then relay that information back to a base station. We assume that the environment is 2.5D, in that the area that must be mapped is 2D, but the robots may fly some distance above the ground.

Our assumptions on the robots and environment closely follow those described in [42]. We assume a bounded planar workspace $\mathcal{W} \subseteq \mathbb{R}^2$. The workspace is divided into free regions $\mathcal{W}_{free}$ and obstacle regions $\mathcal{W}_{obs}$. The partition of the workspace into obstacle

and free is initially unknown to the team. The areas with unknown and known partition status to a robot $k$ at time $t$ are denoted as $\mathcal{W}^k_{unknown}(t)$ and $\mathcal{W}^k_{known}(t)$, respectively. The goal is to reveal the maximal subset $\mathcal{W}_{known}$ and relay that information back to a base station.

Each robot is modeled as a disk of radius $p$, whose configuration $q_k$ is described by its Cartesian center. The particular kinematics of the robots are not considered in order to focus on the coordination algorithm. Each robot is path controllable (i.e., it can follow any path in its configuration space with arbitrary accuracy). The robots are equipped with an omnidirectional sensor which allows them to explore (and map) the environment around them.

The optimization problem can be stated as follows:

$$\mathcal{P} = \arg\max_{\mathcal{P}\in\Psi} A_r(\mathcal{P}) \text{ s.t. } |\mathcal{P}_k| < B_K \ \forall k, \tag{4.1}$$

where $\Psi$ is the space of possible team paths, $A_r(\mathcal{P})$ is the area explored and communicated back to the base station by a set of trajectories $\mathcal{P}$, and $|\mathcal{P}_k|$ is the battery consumed by a trajectory $\mathcal{P}_k$ for robot $k$.

## 4.3 Coordination Algorithm

First, we assign the robots sequential identification numbers. We define four possible states for an individual robot: explore, meet, sacrifice, and relay. The current state is dependent on the robot's ID number, remaining battery life $\beta$, battery required to return home $B$, time since the last meeting with any other robot $t$, and predicted distances from

other robots.

All robots begin in the exploration state. This state uses a frontier-based exploration strategy, as described in [42, 43], though any algorithm could be substituted. The basic idea is for the robots to build their own local maps and extract "frontiers" between the explored space and the unknown space. The heuristic of moving to the closest frontier is used here, which has been shown to provide competitive performance with other heuristics [50]. The robots share their maps and merge them using existing map merging techniques (see Section 4.5). If two robots move towards frontiers that are near each other, we impose a conflict resolution step where the robot with the higher priority ID number takes the frontier, and the other robot chooses the next-closest frontier. Collision avoidance is handled in a similar manner where the robot with the lower priority stops or lands and allows the higher priority robot to pass. This approach allows for fully distributed operation, correlating with our thesis objective of robust planning under uncertainty.

After a constant time $T$ (set as a parameter), a robot will transition into the meet state. In this state, a robot will attempt to travel into communication range of another robot to transmit map information and to update its internal state of other robots' locations. If it is able to meet with a robot, it will negotiate relay and sacrifice roles. The robot with the lower ID of the two marks itself as a relay and the higher ID robot marks itself as the sacrifice. These flags will be used when the robots decide to enter the relay or sacrifice state. Any meeting with another robot occurring at a later time will overwrite these flags.

If a robot is not a relay, it will transition into the sacrifice state when it determines

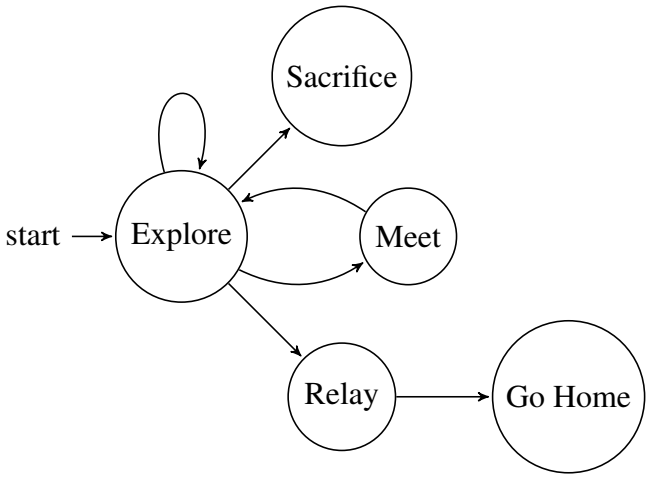



Figure 4.2: State diagram for the proposed adaptive exploration algorithm. The explore, meet, sacrifice, and relay states allow for heterogeneous behavior that adjusts for limitations on communication and battery life.

it has just enough battery to make it back to the last position it sighted its relay. It will then travel to that location in an attempt to meet with the relay. Should it not find its relay at the predicted position, it will revert to an exploration mode in a final attempt to locate the robot. If a robot is a relay, it will eventually reach a point at which it has only enough battery to travel to the base station. At this point, the robot will land (or go into a hibernation state) to conserve battery while it waits for its sacrificial robot to locate it. Once the relay is found by the sacrificial robot, it will travel back to the base station.

These state changes are made independently of other robots' states, which leads to a fully distributed architecture. Algorithm 4 gives a summary of the proposed algorithm, and Figure 4.2 shows a state diagram for the explore, meet, sacrifice, and relay states.

**Algorithm 4** Exploration coordination algorithm

```
1: Inputs: current battery β, time between meetings T
2: t ← 0
3: while β > 0 do
4:     B ← battery required to go home
5:     Transmit with nearby robots
6:     if β ≤ B then
7:         if is relay then
8:             Land until sacrifice robot meets
9:         else
10:            Travel to last meeting location
11:    else if t ≥ T then
12:        Travel to last meeting location p
13:        if robot found then
14:            Update relay and sacrifice flags
15:        else if at p then
16:            Revert to exploration
17:        t ← 0
18:    else
19:        Explore local frontiers
20:    t ← t + 1
```

## 4.4 Simulated Results

Simulations were used to test the proposed coordination algorithm. Simulations were run over two maps (shown in Figure 4.3) with varying battery levels and starting positions. The quadcopters are capable of moving at a speed of 1 m/s in simulation, and the parameter $T$ was empirically set to one minute.

We compared the proposed algorithm to a baseline frontier-based exploration approach [42]. The frontier-based exploration approach is equivalent to remaining in the "explore" state and then returning to the base station once battery is low. Thus, the simu-

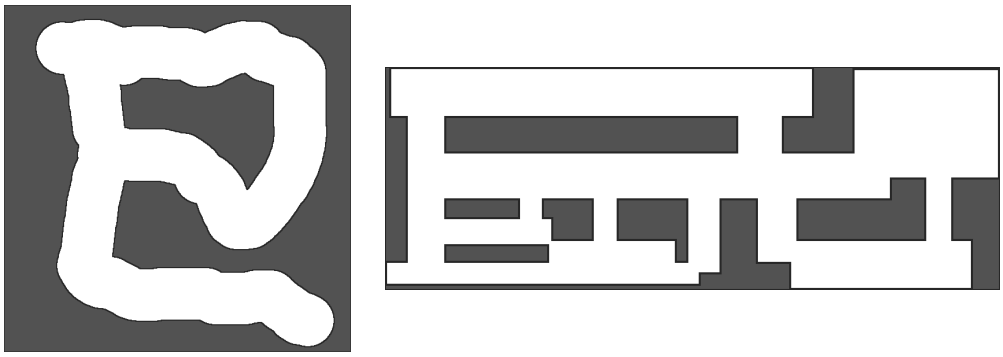
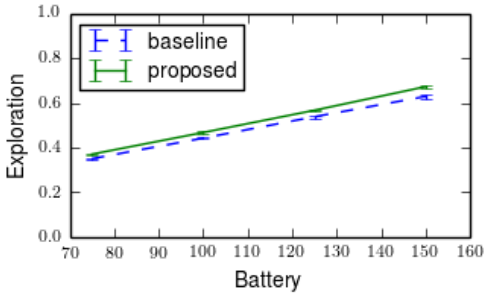


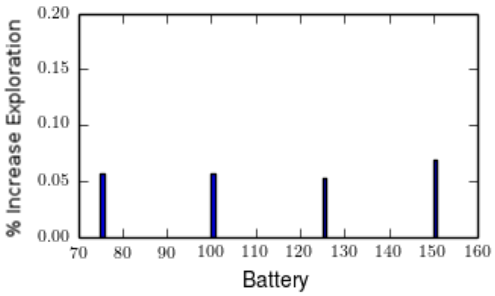
Figure 4.3: **Left:** Simple environment map used for multi-robot exploration trials. The environment is approximately 50 m $\times$ 50 m. **Right:** More complex environment map used for multi-robot exploration trials (120 m $\times$ 40 m).

lations demonstrate the improvement from utilizing the meet, sacrifice, and relay behaviors. Figure 4.4 shows the results of these simulations demonstrating that the proposed algorithm is able to explore a greater percentage of the map than the baseline algorithm under varying conditions. The improvement of the proposed algorithm ranges from 5% to 18%. We note that in all cases the proposed algorithm provides some improvement over the frontier-based exploration baseline.

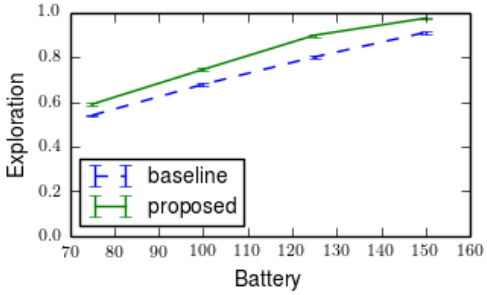
In the complex environment, the improvement from the sacrifice, meet, and relay behavior increases as the number of robots increases. This result is expected since increasing team sizes means that more robots can be sacrificed for additional exploration. We show the results for a large team (8 robots) in the simple environment, which provides substantial benefit over the frontier-based exploration baseline. These improvements demonstrate that the proposed algorithm provides advancement in both cluttered environments (e.g., office buildings) and more open environments (e.g., caves). Leveraging shared information allows coordinated planners to outperform independent algorithms through the simple extension of being robust to communication dropout.



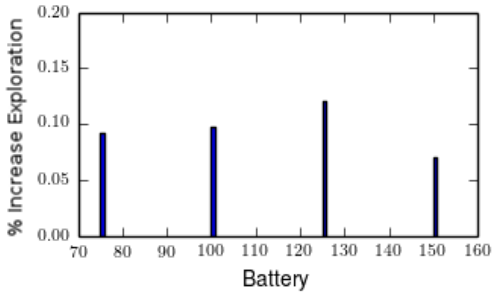

(a) Complex Environment, 2 robots



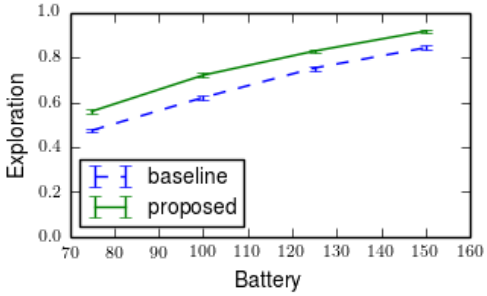

(b) Complex Environment, 2 robots



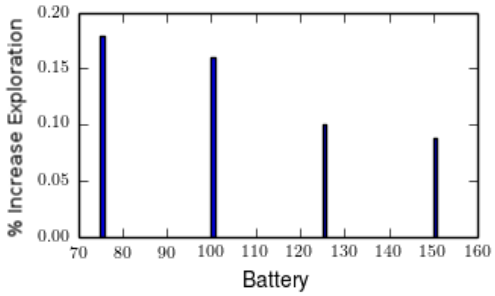

(c) Complex Environment, 4 robots




(d) Complex Environment, 4 robots




(e) Simple Environment, 8 robots




(f) Simple Environment, 8 robots

Figure 4.4: Simulated algorithm performance. Each data point is an average of 200 simulation runs with random starting points. Error bars are one standard error of the mean.

## 4.5 Experiments on Autonomous Quadcopters

The proposed algorithm was demonstrated in an office environment using two autonomous quadcopters. The system used to test our robot coordination algorithm is described below. Design requirements are identified as follows. All processing must be performed onboard the system, and the system must be capable of full autonomy once in the air. Each robot must be able to communicate and coordinate with additional robots. We also design for low cost (less than $1,500 per robot). Existing robots are either substantially more expensive or require an instrumented environment [41].

The system must have sufficient computing power to run vision and planning algorithms in real-time. A Gigabyte Brix i7-4500 was chosen to meet our power, weight, and computational performance requirements. A PX4FMU was used for flight stabilization, and a PX4Flow camera for optical flow measurements. An Asus Xtion acted as the main onboard camera, and provided RGB and depth images.

### 4.5.1 Software Architecture

The software architecture is split into two systems: the high-level processing running on the onboard computer and the low-level flight stabilization on the flight control board. The onboard computer runs the Robot Operating System (ROS) [39] on top of Ubuntu 13.10. The OpenNI ROS package interfaces directly with the Xtion camera to publish RGB and depth information, which is processed by the RGBDSLAMv2 package. The latter package localizes the vehicle in its environment and produces a point cloud map of the area. To improve the usability of this map, the point cloud is passed to the OctoMap

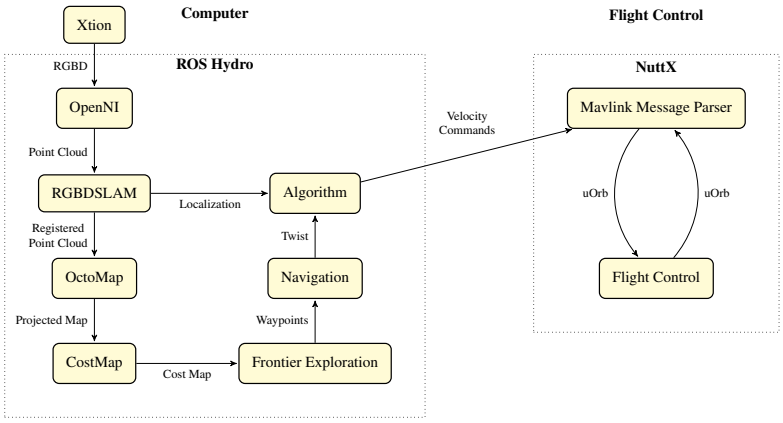



Figure 4.5: Software architecture for the custom-built autonomous quadcopter system.

package to produce a 3D probabilistic occupancy tree. Then the ROS navigation stack is used for 2D path planning and frontier exploration.

The flight control board accepts local velocity commands from the onboard computer over the MAVLink protocol. It then interprets these commands based on its internal velocity estimate provided by the flow camera and inertial measurement unit. Beyond this point, the flight control software is treated as a black box.

### 4.5.2 Experiments

We successfully demonstrated the key components of our proposed algorithm on a team of two quadcopters. A sacrifice-relay hand-off was implemented using one sacrificial quadcopter (denoted as SQ) to perform frontier based exploration in a new room while a relay quadcopter (denoted as RQ) traveled to a meeting location and then returned with data of the explored room. A colored OctoMap of the passage that RQ mapped is shown on the left in Figure 4.6, and the additional explored area mapped by SQ is shown on

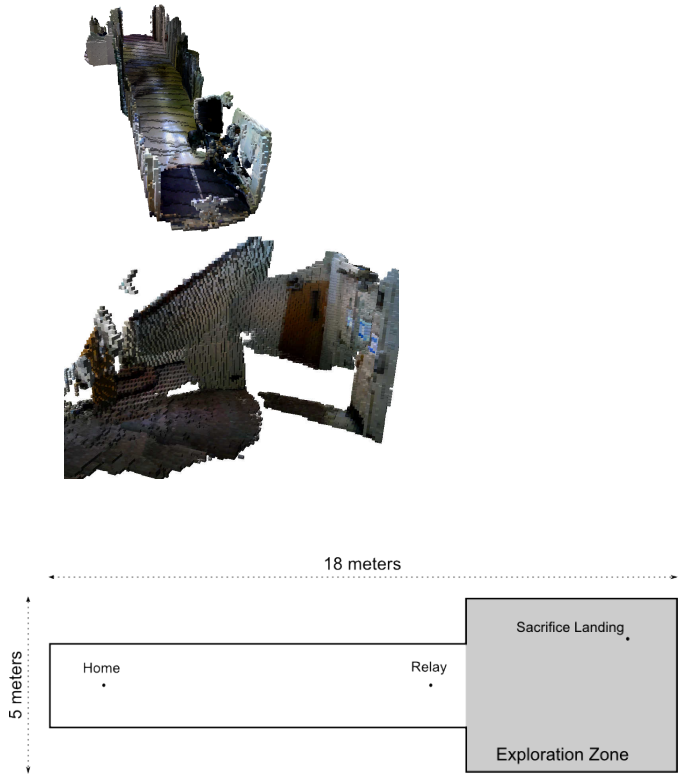



Figure 4.6: **Left:** Colored OctoMap from relay quadcopter, leading to unexplored area. **Right:** Colored OctoMap from sacrificial quadcopter showing additional explored area. **Bottom:** An overview of the testing area.

the right.

The experiment proceeded as follows: (1) both RQ and SQ started at the same home base location; (2) SQ and RQ were both moved manually (due to safety considerations in the experiment space) to the entrance of the unexplored area where RQ landed as a relay; (3) SQ was flown to a stable altitude and switched to autonomous frontier-based exploration until it registered that it was nearly out of battery; (4) SQ landed and transmitted its map data to RQ, (5) RQ was returned to the home base with the gained map information from both SQ and RQ. This proof-of-concept demonstration shows

successful exploration and mapping capabilities on the quadcopter hardware as well as a successful hand-off of information between UAVs acting in the relay and sacrifice roles. Additional hardware experiments are shown in this chapter's supplemental video attachment.

## 4.6 Discussion

This chapter has demonstrated coordinated exploration using a multi-UAV team with unreliable communication and limited battery life. Our results show that the proposed algorithm, which leverages meeting, sacrificing, and relaying behavior, increases the percentage of the environment explored. By capitalizing on relay behavior, the mapped environment returned to the base station was shown to be more complete than a standard frontier-based exploration strategy. This was also demonstrated with a team of two autonomous quadcopters, which were designed to be low-cost, fully autonomous, and capable of operating without pre-installed infrastructure.

It is important to note that the mapping capabilities of the quadcopters use off-the-shelf algorithms available in the ROS package. It may be beneficial to improve the mapping capabilities using more sophisticated distributed smoothing and mapping approaches [52]. Additional future work lies in examining the effect of parameter selection in various types of environments (e.g., tunnels, caves, mines, etc.). Finally, this schema can be readily scaled up to larger team sizes, vast teams could be used to achieve exploration in larger-scale environments.

# Chapter 5: Risk-Aware Graph Search

"Risk: A state of uncertainty where some possible outcomes have an undesired effect or significant loss," Hubbard [53].

While the previous chapter dealt with unreliable communication, the underlying path planning problem is still fragile to the environmental model. When planning in unstructured environments there is a greater need for fast, reliable path planning methods capable of operating under uncertainty. Planning under uncertainty allows for robustness when faced with unknown and partially known environments. We introduce a method, Risk-Aware Graph Search (RAGS), for finding paths through graphs with uncertain edge costs. Our method bridges the gap between traditional search methods [15], [17] and risk-aware planning [54, 55].

Graph theory is an expanding field, and graph search algorithms have proven to be extremely useful tools over a variety of domains. Traditionally, graphs are composed of nodes and edges, with a node representing some state and an edge representing the transition between states. Effectively searching through a graph with known edges has been extensively researched and applies across disciplines in robotics, computer science, and optimization. We aim to expand the capabilities of graph search algorithms by allowing for uncertainty in traversal costs, while avoiding the blowup in computation from

The work presented in this chapter was published in the International Conference on Robotics and Automation 2015, Workshop: Beyond Geometric Constraints.

more expressive frameworks (e.g., POMDPs). Our novel approach searches over uncertain edge costs with known distributions. This formulation allows for computationally efficient methods of reducing the risk of traversing paths with high cost.

Traditional graph search methods (such as $A*$ and $D*$) search over deterministic costs [15], [17] when traversing a graph. Often the traversal between states can be the biggest unknown. For instance, imagine that you are going from your house to the grocery store, you know which intersections you will drive through but without knowing future traffic conditions, you cannot know how long it will take to go down each street. Assuming known traversal costs ignores valuable information, which could help you avoid delays. There have been several recent developments in risk-aware planning based on this intuition [54, 55], but much work involving uncertainty-based planning remains strictly concerned with the belief state of the robot's location [21], [56].

The main novelty of this chapter is the introduction of a nonmyopic graph search algorithm for risk-aware planning. RAGS is an online planning mechanism that incorporates live feedback for deciding when to be conservative and when to be aggressive. With edge costs modeled as normal distributions, we can derive a principled way of leveraging information further down a path. The result is a lower probability of the executing a path with a high cost.

We compare our method to $A*$, $D*$, and a greedy approach. The algorithms are compared by searching from start to goal through a number of random connected graphs. The results show that RAGS performs similarly to $A*$ in terms of mean path cost, but it substantially reduces the number of high cost runs compared to all other tested methods. These results confirm the effectiveness of a risk-aware planning approach.

In addition to testing over random graphs, we designed a validation using satellite imagery. By applying a series of filters to satellite data, we are able to extract potential obstacles that may impede a robot moving through the map. To deal with the imprecise nature of obstacle extraction, we can utilize the benefits of RAGS to find paths that are less likely to be substantially delayed. We show examples of different cases in which RAGS finds preferable paths. The algorithm is run over 64 satellite images taken from a broad set of landscapes. The resulting path costs over the image database confirms the benefit of the RAGS algorithm in a real-world planning domain.

The remainder of this chapter is organized as follows. First we review related work and research in probabilistic planners (Section 5.1). We then introduce the path search problem with uncertain traversal costs (Section 5.2). We next derive a method for quantifying path risk (Section 5.3) and present a way to reduce the search space by removing dominated paths (Section 5.4). In Section 5.5 we discuss RAGS and show comparisons to existing search algorithms. We then highlight the utility of RAGS by demonstrating its effectiveness for planning through various terrain captured from satellite imagery (Section 5.6). Finally, we draw conclusions and propose future directions for this line of research (Section 5.8).

## 5.1 Related Work

### 5.1.1 Optimal Path Planning

Optimal path planning is described as the process of generating a series of waypoints from a starting location to a desired goal, typically under constraints and a metric. Constraints are often environmental restrictions, such as avoiding obstacles or respecting torque limits. Metrics are usually some form of cost function like minimum time or energy. It is common to discretize the search space to create a discrete planning problem. Heuristic search techniques have become popular for this kind of deterministic planning [15], [17]. On the other hand, sampling-based planning algorithms have become widely used for their ability to solve problems in high-dimensional continuous state spaces [57]. Many of these classical path planning algorithms are discussed in [11], and [58]. These algorithms are well suited to problems with deterministic actions and a well-defined search space, but are not well adapted to dealing with planning under uncertainty.

Some of the most-used graph search algorithms include breadth-first-search, depth-first-search, $A*$ and Dijkstra's [11]. These search methods have bounds on time and space complexity. Dijkstra's and $A*$, are both complete and optimal, which have led to their wide use across domains. However, when dealing with nondeterministic graphs, these guarantees are no longer applicable. Unfortunately this makes them difficult to incorporate into a stochastic environment without making restrictive assumptions or constraining the available information [59].

### 5.1.2 Belief Space Planning

There are a number of graph structures that do incorporate uncertainty, such as Partially Observable Markov Decision Processes (POMDP) [60], and belief trees [21]. Each of these take a different approach from the risk graphs used in this chapter. While the POMDP framework is known to be general enough for a large set of sequential decision processes, the problem can quickly become intractable for a large search space. POMDPs can be used to represent the same environmental uncertainties but with our framework we can avoid these computational challenges. We incorporate the uncertainty in the edge cost to represent the unknowns of transitioning from one state to another. This creates a less expressive framework than POMDPs, but allows for efficient planning and determination of paths that have high likelihood of yielding a low cost.

In belief trees the nodes represent belief of the systems state, similar to POMDPs. Bry and Roy [21] perform planning over possible trajectories of a robot's uncertain state. By bounding the probability of collision in the belief tree the planner can search for a safe trajectory for the robot to navigate with uncertain localization. This probabilistic planning benefits from a collapsed search space, making it more tractable than the POMDP framework.

### 5.1.3 Planning Under Uncertainty

Planning under uncertainty is a challenging problem in the field of robotics. In fact, an underlying assumption in many existing planning algorithms is that the search space is not stochastic. By taking this approach researchers have developed many powerful

algorithms like *RRT** [20] that perform efficient point to point planning; however, these do not account for uncertainties in the search space. Recent work is now exploring ways of incorporating uncertainty back into planners in field robotics applications [54, 61].

The importance of probability in robotics is the ability for performance to degrade gracefully when encountering uncertainty. Notable work has been done on incorporating uncertainty from sensors into the state estimation of the system, [56], [62]. Chaves et al, [63] explore risk aversion in belief space planning by incorporating measurement uncertainty into an active SLAM framework. While in [64] the authors minimize the path length with respect to a lower bound on the probability of success. This is similar to work by [65] which instead bounds the average risk. These algorithms define reasonable ways of assigning a value or function for trading off between risk and a primary search objective like distance. Significant effort has gone into developing methods for dealing with multiple independent heuristics in a search space [66]. In our work, we derive a formula for reducing the risk of a path based on the uncertainty of the traversal costs themselves.

## 5.2 Problem Formulation

We now formulate the path search problem with uncertain edge costs and introduce notation that will be used throughout the chapter. General nomenclature is provided in Table 5.1.

Consider a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where the cost of traversing edge $E \in \mathcal{E}$ is drawn from a normal distribution $\mathcal{N}(\mu_E, \sigma_E^2)$. The cost of a path $\mathcal{P} \subset \mathcal{E}$ is the sum of the edge

TABLE 5.1: NOMENCLATURE

| | |
|---|---|
| $\mathcal{A}, \mathcal{B}, \cdots$ | sets of random variables $\{A_i\}, \{B_j\}, \cdots$ |
| $A_i$ | random variable drawn from $\mathcal{N}\{\mu_{A_i}, \sigma^2_{A_i}\}$ |
| $i$ | iterator from 1 to $m$ |
| $j$ | iterator from 1 to $n$ |
| $k$ | iterator from 1 to $n$, $k \neq j$ |
| $m$ | size of $\mathcal{A}$ |
| $n$ | size of $\mathcal{B}$ |
| $c_{A_i}$ | sample of random variable $A_i$ |
| $c_{A_{min}}$ | $\min_{A_i \in \mathcal{A}} c_{A_i}$ |

traversal costs, each of which are normally distributed. Thus the total path cost is drawn from $\mathcal{N}(\mu_{\mathcal{P}}, \sigma^2_{\mathcal{P}})$, where $\mu_{\mathcal{P}} = \sum_{\mathcal{P}} \mu_E$ and $\sigma^2_{\mathcal{P}} = \sum_{\mathcal{P}} \sigma^2_E$.

Given any pair of start and goal vertices in the graph, $V_s, V_g \in \mathcal{V}$, the task is to traverse the graph along the path of least risk. More precisely, the path of least risk retains the highest probability of traversing the least-cost path to goal as each vertex transition is executed from $V_s$ to $V_g$.

The decision at each vertex can be formulated by considering the next available transitions and their associated path sets. For example, say edge connections exist between the current vertex $V_t$ and each of the vertices in the set $\mathcal{V}_{t+1} = \{A, B, C, \cdots\}$; furthermore, $m$ acyclic paths exist from vertex $A$ to $V_g$, while $n$ acyclic paths exist from vertex $B$ to $V_g$, etc. Let $\mathcal{A}$ be the set of random variables $A_i$, $i = \{1, \cdots, m\}$, where $A_i \sim \mathcal{N}\left(\mu_{A_i}, \sigma^2_{A_i}\right)$ represents the cost distribution of path $i$ from vertex $A$ to $V_g$, see Figure 5.1. Let $c_{A_i}$ be a sample of the random variable $A_i$ and let $c_{A_0}$ be a sample of the random variable representing the cost of transitioning between $V_t$ and $A$, then the

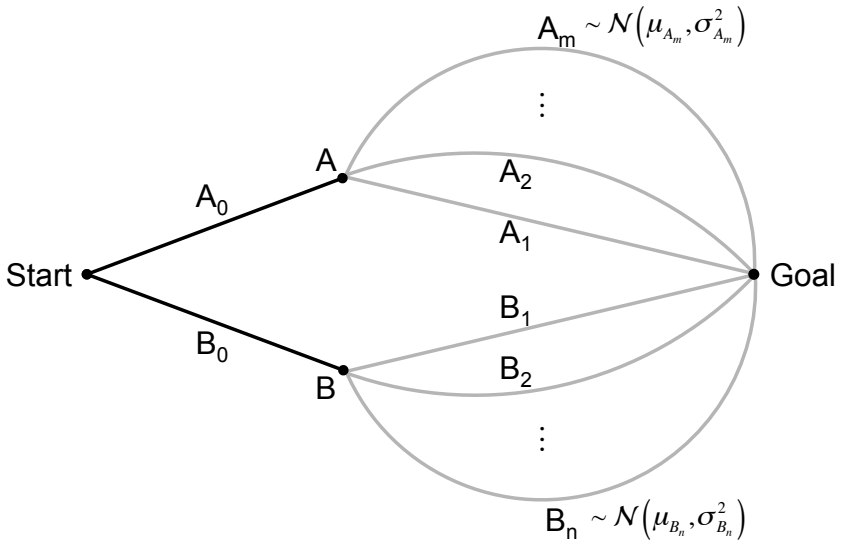



Figure 5.1: Setup of pairwise comparison for sequential look-ahead planning. Given the path cost distributions, we can directly compute the probability that traveling from Start to Goal through A will ultimately yield a cheaper path than traveling via B.

lowest-cost path from $V_t$ to $A$ and onwards to $V_g$ has cost:

$$c_{A_{min}} = c_{A_0} + \min_{A_i \in \mathcal{A}} c_{A_i}. \tag{5.1}$$

Similar statements can be made for path sets $\mathcal{B}, \mathcal{C}, \cdots$. To traverse the path of least-risk, each edge transition must select the next vertex $V \in \mathcal{V}_{t+1}$ such that the following probability holds,

$$P\left(c_{V_{min}} < c_{(\mathcal{V}_{t+1} \setminus V)_{min}}\right) \geq 0.5. \tag{5.2}$$

We note here that since the path cost distributions are derived from the same graph, then pairwise comparisons between the available path sets obey a total ordering of pref-

erence. That is, given

$$P\left(c_{A_{min}} < c_{B_{min}}\right) = \gamma,$$
$$P\left(c_{A_{min}} < c_{C_{min}}\right) = \epsilon,$$

then,

$$\gamma \leq \epsilon \Rightarrow P\left(c_{B_{min}} < c_{C_{min}}\right) \geq 0.5,$$

with equality iff $\gamma = \epsilon$. Thus $|\mathcal{V}_{t+1}| - 1$ pairwise comparisons are needed to solve for the next vertex $V$ using (5.2).

## 5.3 Quantifying Path Risk

The pairwise comparison $P\left(c_{A_{min}} < c_{B_{min}}\right)$ describes the probability that the lowest-cost path in the set $\mathcal{A}$ is cheaper than the lowest-cost path in the set $\mathcal{B}$. This probability can be expanded to,

$$P\left(c_{A_{min}} < c_{B_{min}}\right) = \int_{-\infty}^{\infty} P\left(c_{B_{min}} = x\right) \cdot P\left(c_{A_{min}} < x\right) \mathrm{d}x. \tag{5.3}$$

We can now express each term in the integral according to the path cost distributions of each respective set. Let,

$$f(x, \mathcal{A}) = P\left(c_{A_{min}} < x\right),$$
$$g(x, \mathcal{B}) = P\left(c_{B_{min}} = x\right).$$

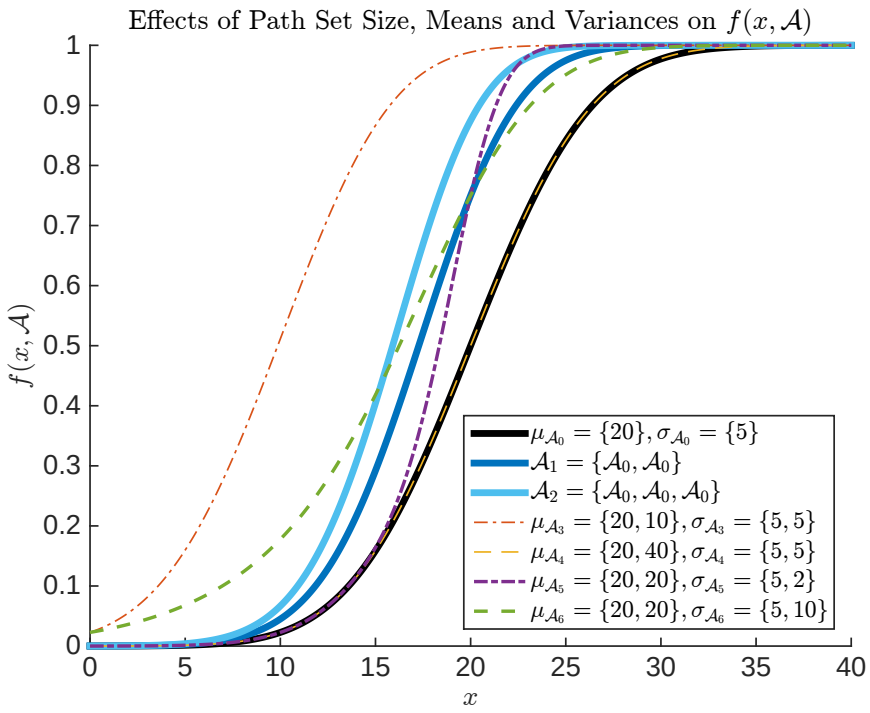



Figure 5.2: The plot of $f(x, \mathcal{A})$ for seven path sets. $\mathcal{A}_0$ has a single path with mean cost 20 and variance $5^2$, while $\mathcal{A}_1$ and $\mathcal{A}_2$ have two and three times as many paths (of equal means and variances) as $\mathcal{A}_0$, respectively. Each of $\mathcal{A}_3 : \mathcal{A}_6$ have two paths, one which is identical to the path in $\mathcal{A}_0$, and a second with lower mean cost, higher mean cost, lower variance, or higher variance, respectively.

Since each of the path costs are drawn from normal distributions, then

$$f(x,\mathcal{A}) = 1 - \prod_{i=1}^{m} \tfrac{1}{2}\mathrm{erfc}\left(d(A_i)\right), \tag{5.4}$$

$$g(x,\mathcal{B}) = \sum_{j=1}^{n} \left[ \tfrac{1}{\sqrt{2\pi}\sigma_{B_j}} \exp\left(-d(B_j)^2\right) \cdot \prod_{\substack{k=1 \\ k\neq j}}^{n} \tfrac{1}{2}\mathrm{erfc}\left(d(B_k)\right) \right], \tag{5.5}$$

where $d(\cdot) = \frac{x-\mu_\cdot}{\sqrt{2}\sigma_\cdot}$.

As an aside, note that $f'(x,\cdot) = g(x,\cdot)$. Thus, using integration by parts, we can show that

$$P\left(c_{A_{min}} < c_{B_{min}}\right) = 1 - P\left(c_{B_{min}} < c_{A_{min}}\right),$$

confirming that these two events are indeed complementary.

Equations (5.4) and (5.5) give some intuitive insight into the calculation of path risk. This formulation performs a trade-off between the number of available paths in each set and the quality of the paths in each set, the latter represented by the means and variances of the path cost distributions. For example, $f(x,\mathcal{A}_0)$ calculates the probability that the best path in $\mathcal{A}$ has a path cost less than $x$, the plot of $f(x,\mathcal{A})$ is shown in Figure 5.2 for $\mu_{\mathcal{A}_0} = \{20\}$ and $\sigma_{\mathcal{A}_0} = \{5\}$ as well as six other path set variants. From (5.4), we note that as $m \to \infty$, $f(x,\mathcal{A}) \to 1$, $\forall x \in (-\infty,\infty)$ and this is shown in the plots of $f(x,\mathcal{A}_{1,2})$. Path set $\mathcal{A}_1$ has twice as many paths (of equal means and variances) as $\mathcal{A}_0$, while $\mathcal{A}_2$ has three times as many. The curves show a trend towards an earlier and more rapid transition to 1 as $m$ increases; however it is also apparent that adding more paths results in diminishing returns in terms of driving $f(x,\mathcal{A}) \to 1$, $\forall x \in (-\infty,\infty)$.

Other trends can be observed when adding paths of varying path cost distributions to the path set. $\mathcal{A}_3$ includes a second path with lower mean and equivalent variance to $\mathcal{A}_0$. The corresponding $f(x, \mathcal{A}_3)$ is shifted significantly to the left of $f(x, \mathcal{A}_0)$, causing the transition towards $1$ to occur much earlier. On the other hand, the addition of a second path with higher mean and equivalent variance results in almost no change to the curve, as shown by the overlap between $f(x, \mathcal{A}_4)$ and $f(x, \mathcal{A}_0)$. Adding a path with equivalent mean but a lower variance results in the plot of $f(x, \mathcal{A}_5)$, which shows a much more rapid transition to $1$ compared to either $f(x, \mathcal{A}_0)$ or $f(x, \mathcal{A}_1)$. However, the addition of a path with equal mean and higher variance increases the overall uncertainty associated with the path set, as shown by the shallower gradation in the probability curve. Furthermore, this means that although the transition towards $1$ begins at lower values of $x$, as $x \to \infty$, $f(x, \mathcal{A}_6) \to 1$ more slowly than for $f(x, \mathcal{A}_1)$ or $f(x, \mathcal{A}_5)$.

This analysis motivates a bounding approach that compares the values of path cost means and variances to retain only the most relevant paths for the calculation of (5.3). Bounding the path set is especially desirable since the computation of each pairwise comparison has complexity $\mathcal{O}(n^2 m)$, where $n$ and $m$ are the sizes of the path sets under consideration.

## 5.4 Non-Dominated Path Set

Existing graph search algorithms such as A* use a total ordering of vertices based on the calculated cost-to-arrive to find a single optimal path between defined start and goal vertices. However, an implementation of RAGS requires two sweeps of the graph, since

an initial pass is required to gather information regarding the cost-to-go at each node for quantifying the path risk at execution. If all possible acyclic paths between start and goal are to be considered, then this initial pass is exponential in the average branching factor of the search tree. In an effort to reduce this computation, we introduce a partial ordering condition based on the path cost means and variances to sort the priority queue and terminate the search. This bounds the set of resultant paths to be considered during execution by accepting only those that exhibit desirable path cost mean and variance characteristics.

As discussed in Section 5.3, the addition of paths with higher mean costs to the existing path set results in little improvement in the overall risk of committing to that set. Similarly, adding paths with higher variances on the path cost results in a slower convergence of $f(x, \mathcal{A})$ to 1 as well as greater uncertainty regarding the true path cost outcomes for the set. Thus, an intuitive bounding condition is to only accept paths with lower path cost means and/or lower path cost variances. Furthermore, the same condition can be applied to the cost-to-arrive distributions during search to provide a partial ordering for the node expansions.

In practice, the partial ordering considers whether a path is *dominated* by an existing path, either in the open or closed set. For any two paths, $A$ and $B$ on the graph $\mathcal{G}$,

$$A < B \leftrightarrow (\mu_A < \mu_B) \wedge \left(\sigma_A^2 < \sigma_B^2\right). \tag{5.6}$$

That is, if both the mean and variance of the cost of path $B$ are greater than those of path $A$, then path $A$ is said to dominate path $B$. This is a similar method to the one described

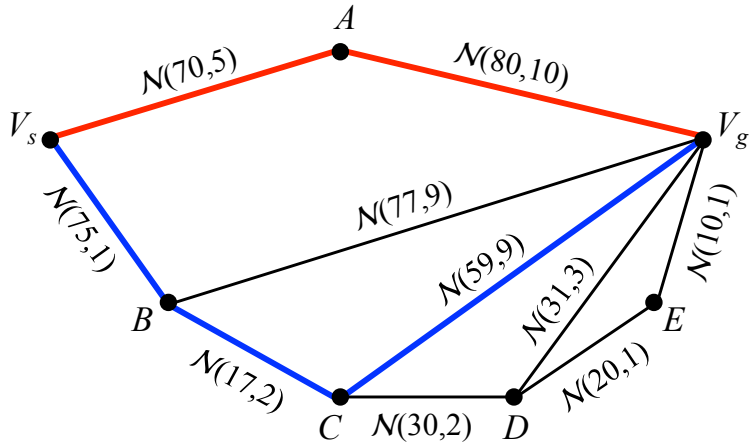



Figure 5.3: Comparison of the executed RAGS path vs. the A* path, which is planned based on the edge cost means only. The edge cost distributions are displayed next to each edge. The RAGS path $\{V_s, B, C, V_g\}$ is in blue, while the A* path $\{V_s, A, V_g\}$ is shown in red. RAGS trades off the slightly higher mean path costs associated with traversing to $B$ against the fewer number of available path options of traversing to $A$. The risk associated with each consecutive traversal decision was $[41.85\%, 29.62\%, 47.16\%]$.

in [21]. Using (5.6) to sort the priority queue guarantees that all non-dominated nodes are expanded before any dominated nodes are considered, and only non-dominated paths to the goal vertex are accepted. This also allows the search to terminate once all paths in the open set are dominated by paths in the accepted set. Thus the set of accepted paths from the initial search through the graph is referred to as the *non-dominated path set*.

## 5.5 RAGS Dynamic Execution

Given the non-dominated path set, path execution can occur by conducting edge transitions at each node to select the path set of least risk according to (5.2). A benefit of this construction is that local updates to the path cost information can be dynamically incorporated into the path execution step by adjusting the input values to (5.3). For ex-

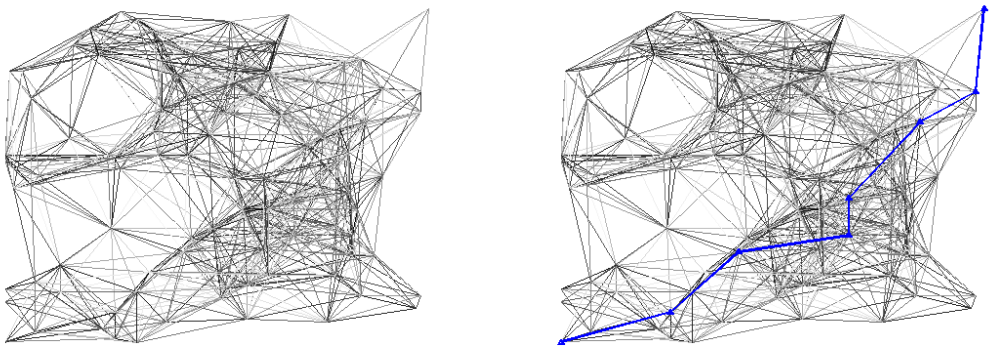
**Algorithm 5** RISK-AWARE GRAPH SEARCH

```
    // INITIAL SWEEP
 1: Initialise open, closed ← ∅
 2: V_s ← start, V_g ← goal
 3: N.append(V_s)
 4: open.push(N)                                  ▷ Place the start node in the open set
 5: while open ≠ ∅ do
 6:     N_0 ← open.pop()                          ▷ Current search node
 7:     V_{t+1} ← getNeighbors(N, G)
 8:     for V in V_{t+1} do
 9:         N ← N_0.append(V)
10:         if newNeighbor(V) ∧ nonDom(N, closed) then
11:             if V = V_g then
12:                 closed.push(N)
13:             else
14:                 open.push(N)
15:     if ¬nonDom(open.top(), closed) then
16:         break

    // PATH EXECUTION
17: G_ND ← closed                                ▷ Directed graph formed by
18: N ← ∅                                            non-dominated path set
19: V_0 ← V_s
20: while V_0 ≠ V_g do
21:     N.append(V_0)
22:     V_{t+1} ← getNeighbors(N, G_ND)
23:     V_ordered ← ComparePathSets(V_{t+1}, G_ND)   ▷ From (5.3)
24:     V_0 = V_ordered.pop()
```

ample, in the comparative experiments discussed in the following sections, the true edge transition cost becomes available for all neighbouring edges to the current node. This information is included into (5.3) by directly substituting the known value of $c_{V_0}$ into (5.1). The pseudocode for the complete RAGS algorithm is provided in Algorithm 5.

A simple example comparing RAGS to an A* search over the mean edge costs is



(a) (b)

Figure 5.4: A sample search graph is shown in (a). The mean cost is the sum of the Euclidean distance plus a random additional cost. Edge variances are represented using grayscale on the graph. The darker the edges the less variance there is on the cost. The executed RAGS path shown in (b) demonstrates the ability of the algorithm to account for both the path cost distributions as well as the available path options to goal. Not only does it favor traversing edges with balanced mean costs and variances, it also maintains a high number of path options towards the goal.

shown in Figure 5.3. Five possible paths between $V_s$ and $V_g$ exist with path cost distributions shown in Table 5.2 below:

TABLE 5.2: PATH COST DISTRIBUTIONS

| Path | $\mu$ | $\sigma^2$ |
|---|---|---|
| $V_s, A, V_g$ | 150 | 15 |
| $V_s, B, V_g$ | 152 | 10 |
| $V_s, B, C, V_g$ | 151 | 12 |
| $V_s, B, C, D, V_g$ | 153 | 8 |
| $V_s, B, C, D, E, V_g$ | 154 | 7 |

The path via $A$ has the lowest path cost mean and is selected according to the A* criterion. However, taking into account all available paths and their cost distributions allows RAGS to calculate the risk associated with committing to any edge traversal. For exam-

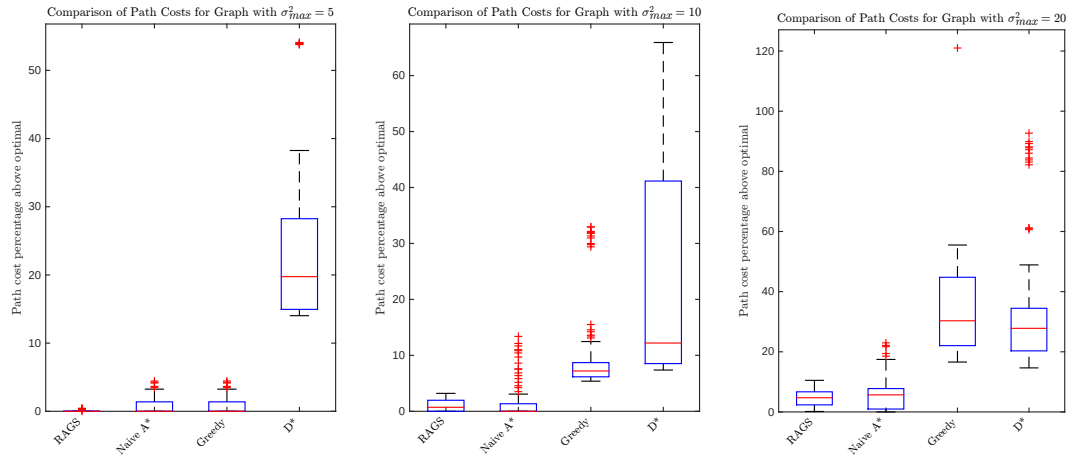



Figure 5.5: Path search results over 100 samples of three graphs with uncertain edge costs; edge cost variances are drawn uniformly between 0 and $\{5, 10, 20\}$ for the three graphs (a)-(c), respectively. The difference in cost of the executed path and the true optimal path as a percentage of the optimal path cost is shown. Note that the true optimal path cost can only be calculated in hindsight.

ple, when deciding between traversing to $A$ or $B$, the risk associated with choosing $B$ is in fact lower (41.85% vs. 58.15%). This is due to the larger number of path options that remain available by following this route, as well as the lower uncertainty associated with those paths. The final path executed by RAGS is shown in blue in Figure 5.3, the set of visited vertices are $\{V_s, B, C, V_g\}$ and the associated risk at each consecutive traversal decision was $[41.85\%, 29.62\%, 47.16\%]$.

### 5.5.1 Comparison to Existing Search Algorithms

We compared RAGS against a *naïve A** implementation, a *greedy* approach, and *D**. During path execution the true costs of immediate neighboring edges became observ-

able. *Naïve A** finds and executes the lowest-cost path based on the mean edge costs and does not perform any replanning. The greedy search is performed over the set of non-dominated paths and selects the cheapest edge to traverse at each step, while *D** initially plans over the mean costs and updates the plan at each step given the new edge cost information.

The search algorithms were tested on a set of graphs generated with a uniform random distribution of $100$ vertices over a space $100 \times 100$ units in size, and connected according to the PRM* radius [20]. Edge costs were represented by normal distributions with mean equal to the Euclidean distance between vertices plus an additional cost drawn from a uniform random distribution over $[0, 100]$. The variance of each distribution was drawn from a uniform distribution over $[0, \sigma^2_{max}]$, where $\sigma^2_{max} = \{5, 10, 20\}$ for the three separate sets of experiments. Note that a minimum cost of the Euclidean distance was enforced in the following experiments. The start vertex was defined at $V_s = (0, 0)$, with the goal at $V_g = (100, 100)$. Figure 5.4a gives an example of a randomly generated graph, with edge variance shown in greyscale (darker lines having a smaller variance). In Figure 5.4b the RAGS path is shown in blue.

In Figure 5.5 we show the results for $100$ trials, where each trial drew new edge costs from the same distribution as described above. *Naïve A** performs well in the mean but is prone to substantial outliers of more expensive paths, especially as the edge uncertainty increases due to any edges with high variance along the path. RAGS is able to mitigate against such outliers by choosing safer routes with lower variances and

RAGS does not require the true neighboring edge costs to formulate a path. However, like *D**, it is able to incorporate this information into its path execution and this is demonstrated in the following results.

more path options, demonstrating the benefit of risk-aware planning. The performance of greedy search decays as the edge cost variances increase, this trend can be attributed to the fact that the greedy algorithm performs search over the non-dominated path set, which becomes less representative of the true optimal path as the variances in the edge costs increase. At low cost variances the non-dominated path set is a tighter representation of the optimal path set but this bound becomes looser as cost variances increase. Due to the myopic nature of its planning, greedy is fallible to arriving at vertices with few path alternatives that all turn out to have high costs. RAGS does not suffer from the same performance decay since it accounts for the full path-to-goal cost distribution at each decision instance. *D** exhibits similar myopia to greedy. However, since it does not restrict its search to the non-dominated path set, its performance does not decay dramatically as cost variances increase.

TABLE 5.3: PATH SEARCH AND EXECUTION TIME (S)

| $\sigma^2_{max}$ | RAGS | A* | Greedy | D* |
|---|---|---|---|---|
| 5 | $2.06 \pm 0.85$ | $0.42 \pm 0.18$ | $0.81 \pm 0.41$ | $1.51 \pm 0.46$ |
| 10 | $2.53 \pm 1.51$ | $0.37 \pm 0.13$ | $0.69 \pm 0.34$ | $1.46 \pm 0.45$ |
| 20 | $2.92 \pm 1.99$ | $0.36 \pm 0.11$ | $0.70 \pm 0.23$ | $1.48 \pm 0.53$ |

The computation time averaged over $100$ samples of $20$ different graphs is presented in Table 5.3. Graph search and execution was calculated on a 2.1GHz Intel Core i7 laptop. As expected, both *A** and greedy planning execute significantly faster than either RAGS or *D**. The initial *A** search is faster than any of the other tested algorithms since it returns only a single path which it then executes. Similarly, the greedy execution-time

decisions require only the comparison of immediately neighboring edge costs. The main increase in computation time over $A*$ is due to the initial sweep for the non-dominated path set. Even so, this overhead is comparatively small when considering the size of the non-dominated set. On average, this set contained $69$ paths and took $0.13\text{s}$ longer to compute than the $A*$ path.

Aside from the initial search, the dynamic replanning of $D*$ is another contributor to its planning time since it is triggered whenever a change is detected in the graph. In this set of trials, new edge information is available at every step and so the graph search must be constantly updated. In general, however, $D*$ is shown to search and execute faster than RAGS in these trials, with variation in computation time driven by the differing number of steps taken to reach the goal.

As discussed in Section 5.3, the computation time of RAGS is heavily influenced by the pairwise risk comparison at each execution step and thus is sensitive to the branching factor of the non-dominated path set. Although this set is pruned as edge transitions are executed, causing path traversal decisions to increasingly become faster to compute, the RAGS execution decision (5.3) actively attempts to maintain a large set of future path options. Despite this, these computation times suggest that real-time implementation on board a platform is feasible, and future work will investigate methods for improving the computational speeds of the algorithm.

## 5.6 Satellite Data Experimental Setup

We also apply RAGS to a real world domain using satellite data. In robotic path planning there is often prior information available of the environment, but this information is not necessarily reliable. An example of this is low resolution overhead satellite imagery, for estimating terrain traversability. In these trials, we use available satellite images, along with some filtering, to extract potential obstacles for a ground robot or a low-flying UAV. To convert the imagery into a useful mapping of obstacles, we perform a series of filters to provide a correspondence between obstacle density and pixel intensity. The satellite images are first converted to grayscale and are then blurred using a Gaussian filter. We then increase the contrast of the image. Finally, we erode and then reconstruct the image to better identify trees and obstacles.

After the filtering process the images provide a rough estimate of obstacles that could force the vehicle to slow down or fly around. In Figures 5.6-5.8, the brighter the pixel, the more likely there is to be an obstacle. The satellite information is too pixelated to provide fully reliable information, but we can use the imagery as an estimate of the obstacles in the environment. To do this we calculate the mean and variance of the pixel intensity values over an edge and use these to characterize the edge cost distributions. Similar to the previous comparison trials, the mean intensity is added to the Euclidean distance to provide a spatial scaling. Using the same method as before, we randomly sample the space to generate a connected graph. The edges are assigned distributions, and then RAGS searches through the graph for a path from the top left start vertex to the bottom right goal vertex. Actual values of the edge costs are drawn from the distribution

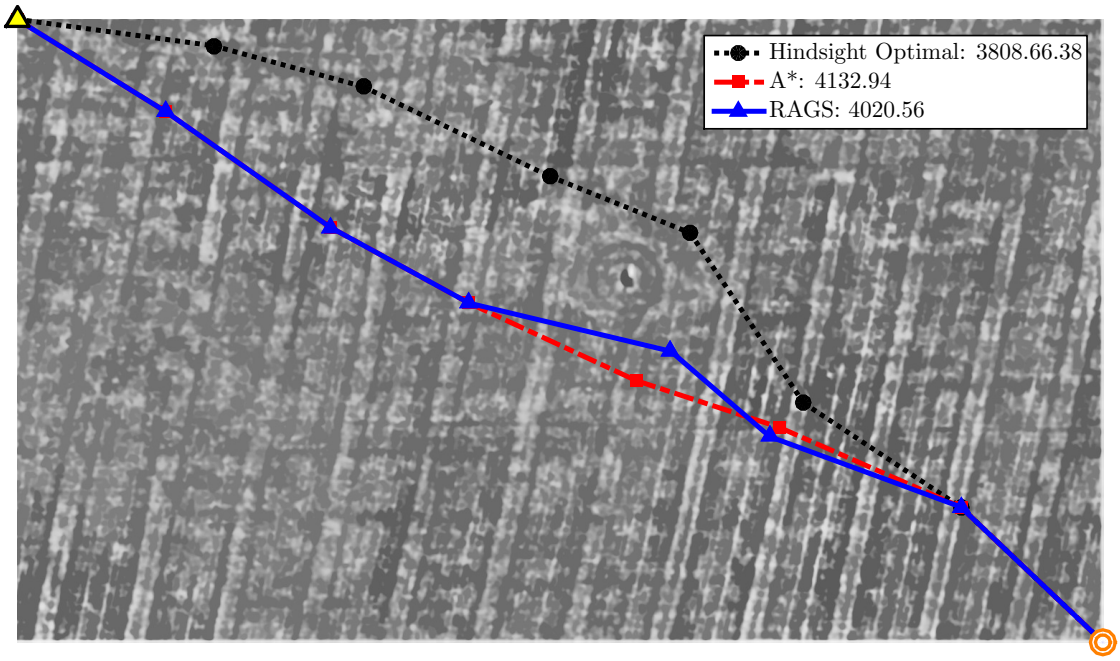



Figure 5.6: Paths of RAGS, $A*$, and the global optimal (known only in hindsight) are shown in the figure with the final path costs. The goal is to traverse from the top left start vertex to the bottom right goal vertex. The empty field is a test case showing that both algorithms plan straight through as expected in uniform, obstacle-free environment.

as the simulated robot moves through the terrain.

## 5.7 Results

Using satellite imagery in three distinct scenarios we tested the algorithm against $A*$. Figures 5.6, 5.7, and 5.8, give visual reference to the benefits of RAGS. In Figure 5.6 the paths through an empty field are straight from start (yellow triangle) to goal (orange circles). As expected there is little variance in edge costs, and the trajectories for RAGS and $A*$ are quite similar. Analyzing the paths found in Figure 5.7 is more interesting. Here we see the benefit of RAGS in obstacle-dense environments. The path from start to

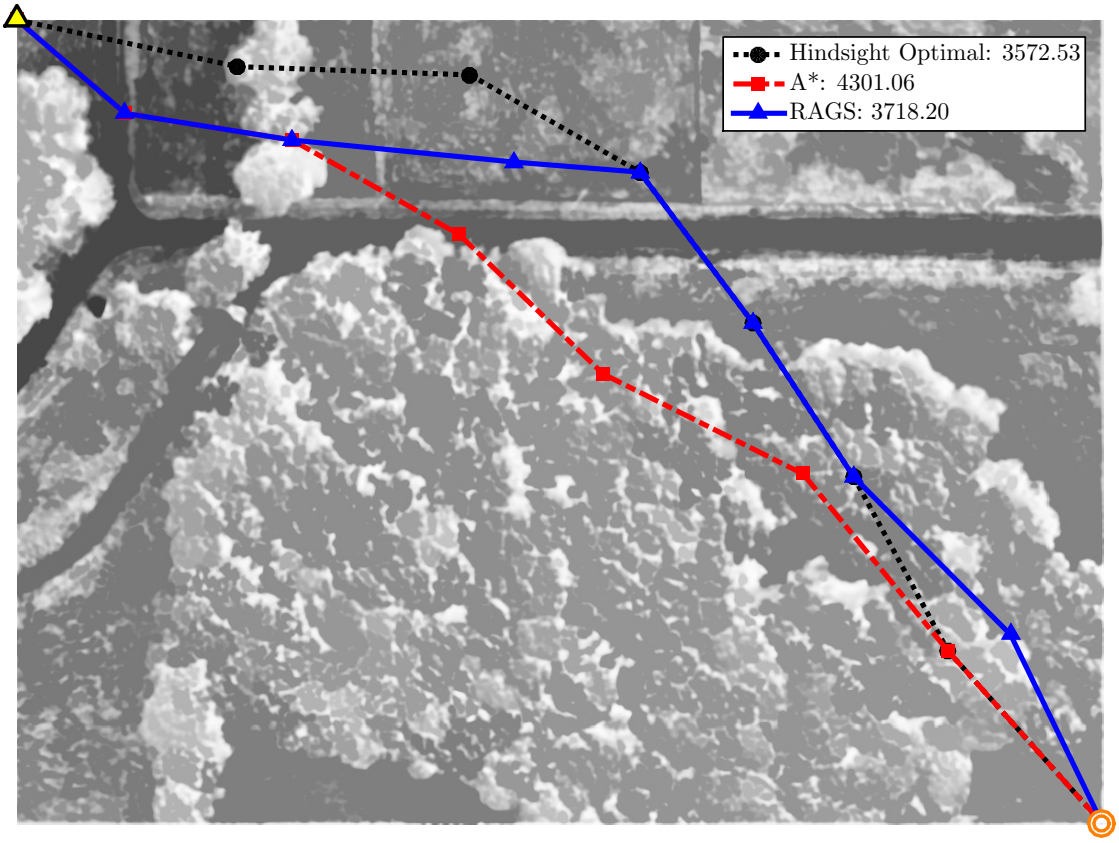



Figure 5.7: A path is planned through a densely cluttered environment. Here we can see RAGS plans around the main tree cluster, where there are more low cost paths, before traveling through a less cluttered region to the goal. RAGS takes advantage of the wide open region instead of looking for a single cheaper narrow track that may exist through a cluster of trees.

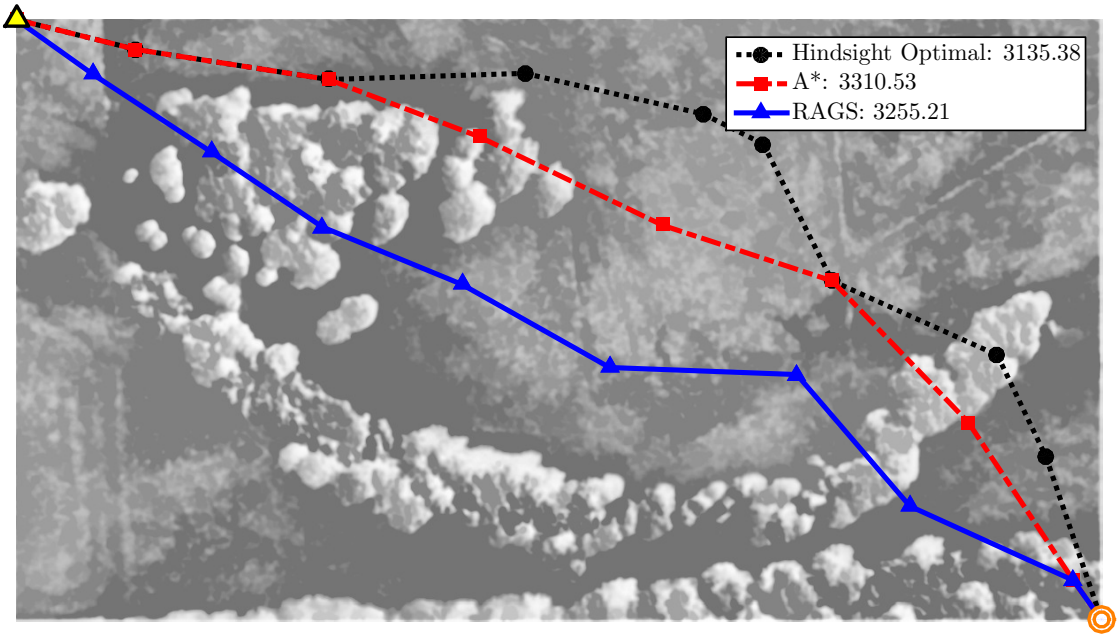



Figure 5.8: The RAGS path is shown traveling straight through the cluster of trees and then taking advantage of the direct route through the wide open space. *A** finds a path that does not navigate completely around the trees and so incurs additional traversal costs on top of taking a less direct route. The costs after execution show that RAGS is actually a cheaper path due to balancing the risk of finding a path through the scattered trees that connects to a more direct path to goal. This demonstrates the benefit of RAGS, knowing when to take risks and when to act conservatively.

goal is blocked by a large cluster of semi-permeable forest. $A*$ executes a path through the center of the cluster that has a low cost in the mean but does not allow for easy deviations if the path is found to be untraversable. On the other hand, the path executed by RAGS demonstrates the nonmyopic nature of this algorithm. RAGS selects a path that travels part way around the cluster to minimize the portion of the path within the dense section of the forest. Values are drawn from the edge distributions to calculate what would have been the optimal path in hindsight. The optimal path (known only after execution) is shown in black, and we can see that it follows a similar trajectory as RAGS.

The final test case can be seen in Figure 5.8, where a small cluster of trees stands between a direct path from the start to the goal. In blue, RAGS plans a path through the trees that is able to take advantage of the clearing in the center. In this example, RAGS is able to assess the risk of taking the shorter, more direct route through the trees and compare this to the expected cost of traveling around the cluster. In comparison, the $A*$ solution finds a less direct route to the goal, however it does not navigate completely around the trees and so incurs additional traversal costs on top of taking a longer path.

Experiments were run over a set of $64$ satellite images such as the ones shown above. The images are of fields with trees of varying tree densities and may also contain houses or other built structures. Images were captured at different resolutions as well as at different altitudes. The majority of the data have tree clusters scattered around the image to provide interesting path planning dilemmas. The compiled results are shown in a box plot of percent above the hindsight optimal, see Figure 5.9.

From the comparison on the three sets of randomly generated graphs in Section 5.5

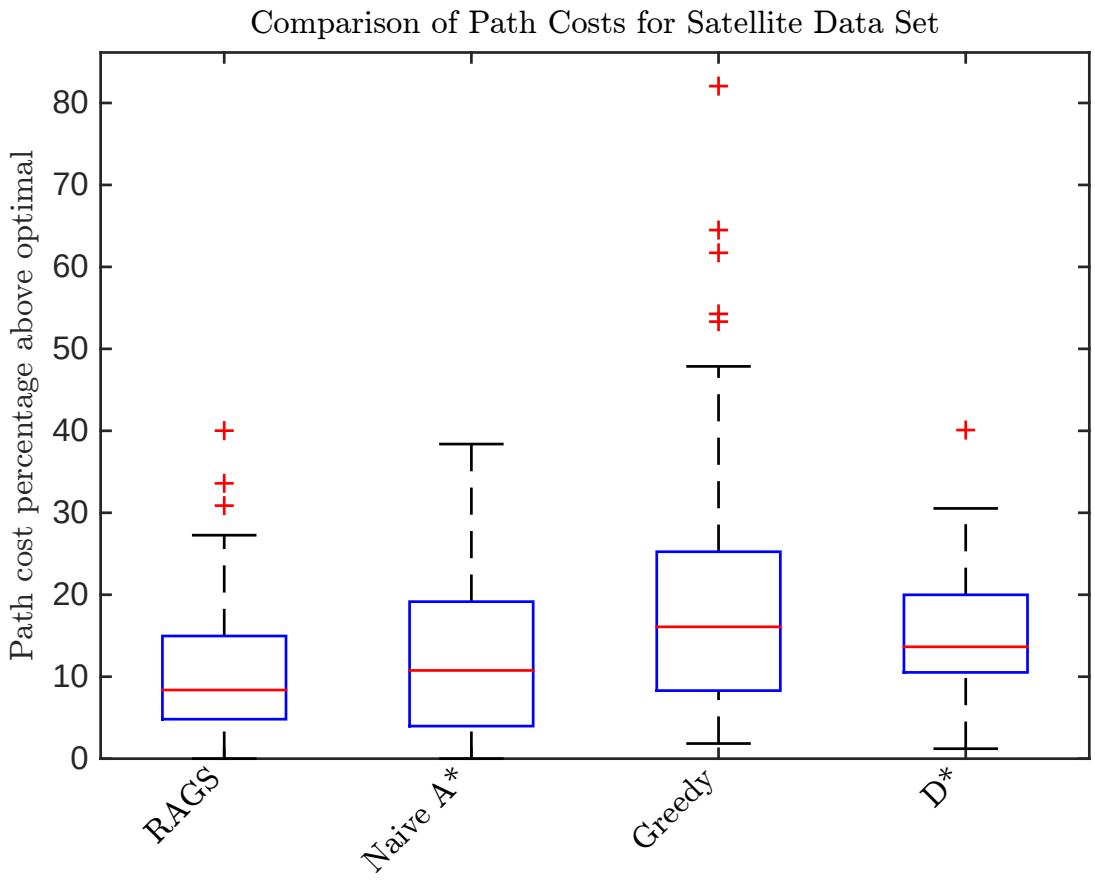



Figure 5.9: Results from satellite data experiment, using 64 images. Box plots represent the path cost percentage above what would have been the optimal path, as calculated in hindsight. The same trends in performance are seen as with the simulated graphs. RAGS accounts for uncertainty in traversing the world, balancing risk of travel time (cost) to outperform comparative algorithms.

we showed that as edge variance increases so does the relative performance of RAGS. This is accounted for by the fact that the other planning algorithms are merely searching over the single heuristic of mean traversal cost. If variance is low then this can be enough to solve for a path that is close to the optimal solution. However, Figure 5.9 reveals that real world data sets can contain significant noise and it is valuable to account for that variability during planning.

## 5.8 Discussion

In this chapter we have introduced a novel approach to incorporating and addressing uncertainty in planning problems. The proposed RAGS algorithm combines traditional deterministic search techniques with risk-aware planning. RAGS is able to trade off the number of future path options, as well as the mean and variance of the associated path cost distributions to make online edge traversal decisions that minimize the risk of executing a high-cost path. The algorithm was compared against existing graph search techniques on a set of graphs with randomly assigned edge costs, as well as over a set of graphs with traversability costs generated from satellite imagery data. In all cases, RAGS was shown to reduce the probability of executing high-cost paths over $A^*$, $D^*$ and a greedy planning approach.

Our next step will be to implement the RAGS algorithm in an information optimization domain. In this case, edge cost probability distributions will represent environmental information, and paths will be generated based on the amount of information the vehicle may collect in different parts of the map. Additionally, we will investigate methods to reduce the computational complexity and memory requirements of the algorithm for implementation on a robotic platform.

# Chapter 6: Conclusions and Future Directions

In this thesis we build on key areas of research to incorporate uncertainty when operating in the physical world. We demonstrate the robustness of each algorithm in both simulated and real environments. A summary of the work done in this thesis follows:

- Fire fighting is challenging, dangerous, and unpredictable work. The ability for UAVs to monitor the fire frontier could mitigate some of these challenges, assist firefighters in their invaluable work, and potentially save lives. Using clustering techniques allowed us to identify regions of interest in noisy real world data. Adaptive monitoring to the changing world is a challenging problem, and beyond filtering the variability modeling, it will be fundamental to future monitoring work. The work was demonstrated on a UAV flying over a simulated fire using the fire simulation, FARSITE, a standard in the community.

- The robust communication method showed gains up to 18% over traditional frontier-based exploration in percent map explored. In the typical exploration problem, if UAVs are coordinating their movements they are limited to staying within communication range or must operate as completely independent agents. Beyond algorithmic improvements, we also demonstrated the capability of deploying systems in the physical world. We implement our algorithm on a team of two custom UAVs.

- This thesis also introduces RAGS, a risk-aware dynamic planning algorithm that adds uncertainty into transition costs associated with graph search algorithms. RAGS was shown to reduce the probability of executing high-cost paths over $A^*$, $D^*$ and a greedy planning approach. This work was simulated over satellite imagery for path planning a low flying UAV through obstacle dense environments.

## 6.1 Future Work

While there have been many compelling examples of robots operating successfully in the unstructured world, there is still a great deal of work left to be done. Each proposed algorithm presented in this thesis has its own list of potential improvements.

Monitoring a wildfire frontier requires tracking the fire fronts changes. One method is targeting the most dynamic regions of the frontier, which due to their dynamic nature are difficult to plan for. This approach can be improved with better models of how the regions move. A possible approach is to use a Recursive Bayesian Estimation to track these points of interest. Recursive Bayesian Estimation is a standard approach for continuously estimating a likely state: Probabilistic state estimation of dynamic POIs will allow for more informed monitoring.

Another possible improvement is to model the entire frontier. For the dynamic frontier domain, modeling with a Gaussian process would improve the frontier prediction. Gaussian processes are used in a variety of research applications to solve regression problems. A Gaussian process model of a wildfire would provide a continuous model of uncertainty along the frontier. This would provide a robust method of tracking un-

certainty and planning for monitoring regions of high uncertainty. Frontier monitoring would benefit from further research on incorporating a Gaussian process model.

In a multi-UAV coordination domain, communication restrictions are both environment and system dependent. The proposed coordination state machine would be improved with optimized parameter selections based on environmental conditions. Parameters such as time between meetings, and battery percent to enter sacrifice-relay mode need generalized approaches that are generated from information of the system and environment. When exploring an unknown area there is often geometric information about the environment that can be used to infer when communication may be re-established. This idea of guessing geometry of unknown areas could be extended to modeling communication. Coordinated exploration would benefit from further research on incorporating inference techniques.

The risk planner, RAGS, could be improved with a faster integration approximator for concatenating child paths. Linear programming has been used to approximate normal distributions with some success and would be worth exploring. This may provide a computational speed up but will not address the algorithm's memory requirements. The risk-aware planning algorithm would adapt well to the informative path planning problem (IPP). Information gathering paths could be planned by using a similar algorithm to RAGS and an estimated reward map with mean and variance. The algorithm would have to optimize over uncertainty of information gained of all paths less than a set travel budget. The approach introduced for developing RAGS could guide design of an algorithm for informative path planning and warrants further research.

The algorithmic contributions in this thesis have given researchers a strong founda-

tion to continue analysis of uncertainty in path planning for UAVs. Leveraging these results, we achieve robust planning algorithms capable of operating in the dynamic, uncertain physical world.

# Bibliography

[1] G. Hollinger and S. Singh, "Multirobot coordination with periodic connectivity: Theory and experiments," *IEEE Trans. Robotics*, vol. 28, no. 4, pp. 967–973, 2012.

[2] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. MIT press, 2005.

[3] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, pp. 281–297, 1967.

[4] L. E. Parker, "Current research in multirobot systems," *Artificial Life and Robotics*, vol. 7, no. 1-2, pp. 1–5, 2003.

[5] A. Wald, "Sequential tests of statistical hypotheses," *The Annals of Mathematical Statistics*, vol. 16, no. 2, pp. 117–186, 1945.

[6] R. Bajcsy, "Active perception," *Proceedings of the IEEE*, vol. 76, pp. 966–1005, Aug 1988.

[7] I. Rekleitis, V. Lee-Shue, A. P. New, and H. Choset, "Limited communication, multi-robot team based coverage," in *IEEE International Conference on Robotics and Automation.*, vol. 4, pp. 3462–3468, IEEE, 2004.

[8] M. Li, W. Cheng, K. Liu, Y. He, X. Li, and X. Liao, "Sweep coverage with mobile sensors," *IEEE Transactions on Mobile Computing*, vol. 10, pp. 1534–1545, Nov 2011.

[9] C. Cassandras, X. C. Ding, and X. Lin, "An optimal control approach for the persistent monitoring problem," in *50th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC)*, pp. 2907–2912, Dec 2011.

[10] C. Cassandras, X. Lin, and X. Ding, "An optimal control approach to the multi-agent persistent monitoring problem," *IEEE Transactions on Automatic Control*, vol. 58, pp. 947–961, April 2013.

[11] S. M. LaValle, *Planning Algorithms*. Cambridge University Press, 2006.

[12] D. Ferguson, M. Likhachev, and A. Stentz, "A guide to heuristic-based path planning," in *Proceedings of the International Workshop on Planning Under Uncertainty for Autonomous Systems, International Conference on Automated Planning and Scheduling (ICAPS)*, pp. 9–18, 2005.

[13] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[14] S. Russell and P. Norvig, "A modern approach," *Artificial Intelligence.*, vol. 25, p. 27, 1995.

[15] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, pp. 100–107, July 1968.

[16] A. Stentz and I. C. Mellon, "Optimal and efficient path planning for unknown and dynamic environments," *International Journal of Robotics and Automation*, vol. 10, pp. 89–100, 1993.

[17] S. Koenig and M. Likhachev, "Fast replanning for navigation in unknown terrain," *IEEE Transactions on Robotics*, vol. 21, pp. 354–363, June 2005.

[18] L. E. Kavraki, P. Švestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[19] S. M. LaValle, "Rapidly-exploring random trees a new tool for path planning," 1998.

[20] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[21] A. Bry and N. Roy, "Rapidly-exploring random belief trees for motion planning under uncertainty," in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 723–730, 2011.

[22] R. Kremens, A. Seema, A. Fordham, D. Luisi, B. Nordgren, S. VanGorden, and A. Vodacek, "Networked, autonomous field-deployable fire sensors," *Proceedings of the International Wildland Fire Safety Summit*, 2001.

[23] C. E. Koulas, "Extracting wildfire characteristics using hyperspectral, lidar, and thermal ir remote sensing systems," in *SPIE Defense, Security, and Sensing*, pp. 72983Q–72983Q, 2009.

[24] N. I. F. Center, "Federal fire fighting costs," 2015. Accessed: 2014-09-26.

[25] "Farsite: Fire, fuel and smoke," 2014. Accessed: 2014-09-27.

[26] R. N. Smith, M. Schwager, S. L. Smith, B. H. Jones, D. Rus, and G. S. Sukhatme, "Persistent ocean monitoring with underwater gliders: Adapting sampling resolution," *Journal of Field Robotics*, vol. 28, no. 5, pp. 714–741, 2011.

[27] A. L. Bertozzi, M. Kemp, and D. Marthaler, "Determining environmental boundaries: Asynchronous communication and physical scales," in *Cooperative Control*, pp. 25–42, Springer, 2005.

[28] D. Marthaler and A. L. Bertozzi, "Tracking environmental level sets with autonomous vehicles," in *Recent Developments in Cooperative Control and Optimization*, pp. 317–332, Springer, 2004.

[29] S. Susca, F. Bullo, and S. Martínez, "Monitoring environmental boundaries with a robotic sensor network," *IEEE Transactions on Control Systems Technology*, vol. 16, no. 2, pp. 288–296, 2008.

[30] 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.

[31] G. Hollinger, S. Choudhary, P. Qarabaqi, C. Murphy, U. Mitra, G. Sukhatme, M. Stojanovic, H. Singh, and F. Hover, "Underwater data collection using robotic sensor networks," *IEEE Journal on Selected Areas in Communications*, vol. 30, no. 5, pp. 899–911, 2012.

[32] X. Lan and M. Schwager, "Planning periodic persistent monitoring trajectories for sensing robots in gaussian random fields," in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2415–2420, May 2013.

[33] S. L. Smith and D. Rus, "Multi-robot monitoring in dynamic environments with guaranteed currency of observations," in *2010 49th IEEE Conference on Decision and Control (CDC)*, pp. 514–521, IEEE, 2010.

[34] S. L. Smith, M. Schwager, and D. Rus, "Persistent robotic tasks: Monitoring and sweeping in changing environments," *IEEE Transactions on Robotics*, vol. 28, no. 2, 2012.

[35] D. E. Soltero, S. Smith, and D. Rus, "Collision avoidance for persistent monitoring in multi-robot systems with intersecting trajectories," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3645–3652, Sept 2011.

[36] G. A. Hollinger and G. Sukhatme, "Sampling-based motion planning for robotic information gathering.," in *Robotics: Science and Systems*, 2013.

[37] D. W. Casbeer, R. Beard, T. McLain, S.-M. Li, and R. K. Mehra, "Forest fire monitoring with multiple small UAVs," in *Proceedings of the American Control Conference*, pp. 3530–3535, IEEE, 2005.

[38] S. Lloyd, "Least squares quantization in PCM," *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, 1982.

[39] 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, 2009.

[40] S. Shen, N. Michael, and V. Kumar, "Obtaining liftoff indoors: Autonomous navigation in confined indoor environments," *IEEE Robotics and Automation Mag.*, vol. 20, no. 4, pp. 40–48, 2013.

[41] N. Michael, S. Shen, K. Mohta, Y. Mulgaonkar, V. Kumar, K. Nagatani, Y. Okada, S. Kiribayashi, K. Otake, E. T. K. Yoshida, K. Ohno, and S. Tadokoro, "Collaborative mapping of an earthquake-damaged building via ground and aerial robots," *Journal of Field Robotics*, vol. 29, no. 5, pp. 832–841, 2012.

[42] A. Franchi, L. Freda, G. Oriolo, and M. Vendittelli, "The sensor-based random graph method for cooperative robot exploration," *IEEE/ASME Trans. Mechatronics*, vol. 14, pp. 163–175, 2009.

[43] B. Yamauchi, "Frontier-based exploration using multiple robots," in *Proc. Int. Conference on Autonomous Agents*, pp. 47–53, 1998.

[44] J. Fink, A. Ribeiro, and V. Kumar, "Robust control for mobility and wireless communication in cyber-physical systems with application to robot teams," *Proc. IEEE*, vol. 100, no. 1, pp. 164–178, 2012.

[45] B. J. Thibodeau, A. H. Fagg, and B. N. Levine, "Signal strength coordination for cooperative mapping," tech. rep., DTIC Document, 2005.

[46] S. Shen, N. Michael, and V. Kumar, "A stochastic differential equation-based exploration algorithm for autonomous indoor 3D exploration with a micro-aerial vehicle," *Int. Journal Robotics Research*, vol. 31, no. 12, pp. 1431–1444, 2012.

[47] J. Tisdale, Z. Kim, and J. K. Hedrick, "Autonomous path planning and estimation using UAVs," *IEEE Robotics and Automation Mag.*, June 2009.

[48] S. Shen and N. Michael, "State estimation for indoor and outdoor operation with a micro-aerial vehicle," in *Proc. Int. Symp. Experimental Robotics*, (Quebec City, Canada), 2012.

[49] C. Stachniss, *Coordinated multi-robot exploration*, vol. 55, pp. 43–71. Springer, 2009.

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

[51] N. Kalra, *A Market-Based Framework for Tightly-Coupled Planned Coordination in Multirobot Teams*. PhD thesis, Robotics Institute, Carnegie Mellon Univ., 2006.

[52] A. Cunningham, V. Indelman, and F. Dellaert, "DDF-SAM 2.0: Consistent distributed smoothing and mapping," in *Proc. IEEE Int. Conf. Robotics and Automation*, pp. 5200–5207, 2013.

[53] D. Hubbard, *How to Measure Anything: Finding the Value of Intangibles in Business*. Wiley, 2014.

[54] G. A. Hollinger, A. A. Pereira, J. Binney, T. Somers, and G. S. Sukhatme, "Learning uncertainty in ocean current predictions for safe and reliable navigation of underwater vehicles," *Journal of Field Robotics*, vol. 33, no. 1, pp. 47–66, 2016.

[55] L. Murphy and P. Newman, "Risky planning on probabilistic costmaps for path planning in outdoor environments," *IEEE Transactions on Robotics*, vol. 29, no. 2, pp. 445–457, 2013.

[56] H. Kurniawati, D. Hsu, and W. S. Lee, "SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces," in *Robotics: Science and Systems*, 2008.

[57] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *The International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.

[58] J.-C. Latombe, *Robot Motion Planning*, vol. 124. Springer Science & Business Media, 2012.

[59] C. Rebhuhn, R. Skeele, J. J. Chung, G. A. Hollinger, and K. Tumer, "Learning to trick cost-based planners into cooperative behavior," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4627–4633, 2015.

[60] G. E. Monahan, "State of the Art—-A survey of partially observable Markov decision processes: Theory, models, and algorithms," *Management Science*, vol. 28, no. 1, pp. 1–16, 1982.

[61] J. Bohren, R. B. Rusu, E. G. Jones, E. Marder-Eppstein, C. Pantofaru, M. Wise, L. Mösenlechner, W. Meeussen, and S. Holzer, "Towards autonomous robotic butlers: Lessons learned with the PR2," in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5568–5575, 2011.

[62] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Journal of Fluids Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[63] S. M. Chaves, J. M. Walls, E. Galceran, and R. M. Eustice, "Risk aversion in belief-space planning under measurement acquisition uncertainty," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2079–2086, 2015.

[64] W. Sun, S. Patil, and R. Alterovitz, "High-frequency replanning under uncertainty using parallel sampling-based motion planning," *IEEE Transactions on Robotics*, vol. 31, no. 1, pp. 104–116, 2015.

[65] S. Feyzabadi and S. Carpin, "Risk-aware path planning using hierarchical constrained Markov Decision Processes," in *IEEE International Conference on Automation Science and Engineering (CASE)*, pp. 297–303, 2014.

[66] S. Aine, S. Swaminathan, V. Narayanan, V. Hwang, and M. Likhachev, "Multi-heuristic A*," *The International Journal of Robotics Research*, vol. 35, no. 1–3, pp. 224–243, 2016.