11-2-2018

Multi-Robot Coordination and Scheduling for Deactivation & Decommissioning

Sebastian A. Zanlongo
Florida International University, szanl001@fiu.edu

FLORIDA INTERNATIONAL UNIVERSITY

Miami, Florida

MULTI-ROBOT COORDINATION AND SCHEDULING FOR DEACTIVATION & DECOMMISSIONING

A dissertation submitted in partial fulfillment of the

requirements for the degree of

DOCTOR OF PHILOSOPHY

in

COMPUTER SCIENCE

by

Sebastián A. Zanlongo

2018

To: Dean John L. Volakis
College of Engineering and Computing

This dissertation, written by Sebastián A. Zanlongo, and entitled Multi-Robot Coordination and Scheduling for Deactivation & Decommissioning, having been approved in respect to style and intellectual content, is referred to you for judgment.

We have read this dissertation and recommend that it be approved.

Date of Defense: November 2, 2018

The dissertation of Sebastián A. Zanlongo is approved.

Florida International University, 2018

DEDICATION

This is dedicated to my father Alex, my mother Miriam, and my brother Nicholas.

ACKNOWLEDGMENTS

The result here would not have been possible without the support of many people. I would not have been able to accomplish this without the unconditional support of my parents, Alex and Miriam. Their encouragement, understanding and patience were critical to my growth and accomplishments.

I sincerely thank my advisor, Dr. Leonardo Bobadilla, who strives to ensure that his students are independent, dedicated researchers, who has mentored me, and trusted me with this work; I would not be where I am today without him. I would also like to express my gratitude to Dr. Yew Teck Tan, for his crucial role in helping me learn new concepts, and for his guidance in taking many of my ideas to a concrete outcome. And special thanks to all of my committee members: Dr. Mark Finlayson, Dr. Monique Ross, Dr. Ning Xie, and Dr. Dwayne McDaniel for their valuable time and feedback to improve my dissertation and my research.

My lab was shared with great students: Triana, Tauhidul, Mahbubur, and Gregory. They not only collaborated on work and provided valuable feedback, but have also become good friends and colleagues.

I am also thankful to Dr. Lagos and my friends and workmates at the Applied Research Center, who have shared in many adventures both academic and recreational.

Finally, I would like to acknowledge the support of Dr. Tamim Sookoor, who mentored my at G2-Inc., Dr. David Mascarenas from Los Alamos National Laboratory, Scott Gladwell from Sandia National Laboratories, and Dr. Timothy Aucott from Savannah River National Laboratory for their roles in helping me broaden my knowledge and explore relevant fields. This dissertation was also supported in part by the Intel/HENAAC Scholars Program, the DHS Scientific Leadership Awards for Minority Serving Institutions Granting Graduate Degrees Funding Opportunity Number: DHS-10-ST-062-003, and the Department of Energy Cooperative Agreement Science and Technology Workforce Development Initiative and facilitating research opportunities under the DOE-FIU Cooperative Agreement DE-EM0000598.

ABSTRACT OF THE DISSERTATION

MULTI-ROBOT COORDINATION AND SCHEDULING FOR DEACTIVATION & DECOMMISSIONING

by

Sebastián A. Zanlongo

Florida International University, 2018

Miami, Florida

Professor Leonardo Bobadilla, Major Professor

Large quantities of high-level radioactive waste were generated during WWII. This waste is being stored in facilities such as double-shell tanks in Washington, and the Waste Isolation Pilot Plant in New Mexico. Due to the dangerous nature of radioactive waste, these facilities must undergo periodic inspections to ensure that leaks are detected quickly. In this work, we provide a set of methodologies to aid in the monitoring and inspection of these hazardous facilities. This allows inspection of dangerous regions without a human operator, and for the inspection of locations where a person would not be physically able to enter. First, we describe a robot equipped with sensors which uses a modified A^* path-planning algorithm to navigate in a complex environment with a tether constraint. This is then augmented with an adaptive informative path planning approach that uses the assimilated sensor data within a Gaussian Process distribution model. The model's predictive outputs are used to adaptively plan the robot's path, to quickly map and localize areas from an unknown field of interest. The work was validated in extensive simulation testing and early hardware tests. Next, we focused on how to assign tasks to a heterogeneous set of robots. Task assignment is done in a manner which allows for task-robot dependencies, prioritization of tasks, collision checking, and more realistic travel estimates among other improvements from the state-of-the-art. Simulation testing of this work shows an increase in the number of tasks which are completed ahead of a deadline. Finally, we consider the case where robots are not able to complete planned tasks fully autonomously and require operator assistance during parts of their planned trajectory. We present a sampling-based methodology for allocating operator attention across multiple robots, or across different parts of a more sophisticated robot. This allows few operators to oversee large numbers of robots, allowing for a more scalable robotic infrastructure. This work was tested in simulation for both multi-robot deployment, and high degree-of-freedom robots, and was also tested in multi-robot hardware deployments. The work here can allow robots to carry out complex tasks, autonomously or with operator assistance. Altogether, these three components provide a comprehensive approach towards robotic deployment within the deactivation and decommissioning tasks faced by the Department of Energy.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

FIGURE PAGE

CHAPTER 1

Introduction

1.1 Motivation

Across the world, 31 countries use nuclear energy to handle their energy requirements and generate waste which has to be stored. Moreover, we are still coping with legacy waste left over from weapons development programs dating back to the Manhattan Project and Cold War. The facilities associated with generating and storing this waste have suffered accidents, whose already critical status is heightened by the nature of the material stored within. We might immediately think of the 2011 disaster at the Fukushima-Daiichi power plant. However, there are also structures such as the Hanford facility in Washington state, which houses high-level liquid radioactive waste inside large double-shell tanks, one of which has already leaked. The tanks are expected to be decommissioned by 2046 at a cost of $113.6 billion [han14]. Further, we have the Waste Isolation Pilot Plant (WIPP) shown in Figure 1.1, which in 2014 suffered a vehicle catching fire and the subsequent release of radioactive matter into the ventilation system. These structures are also mostly underground, making inspection difficult and dangerous [DT].

Much of the inspection and handling of radioactive material is still done by humans. Protective clothing and equipment can reduce the dosage received, but these measures are still ineffective when dealing with the acute, large doses expected when handling especially radioactive materials. The Department of Energy provides dose limits as in Table 1.1.

During the events following the Fukushima disaster, 167 workers received more than 100mSv, at which there is a slightly increased cancer risk. A further six workers received more than 250mSv, and two workers received upwards of 600mSv [Bru12]. A more extreme example can be found in the 1986 Chernobyl disaster. Here, people known as the

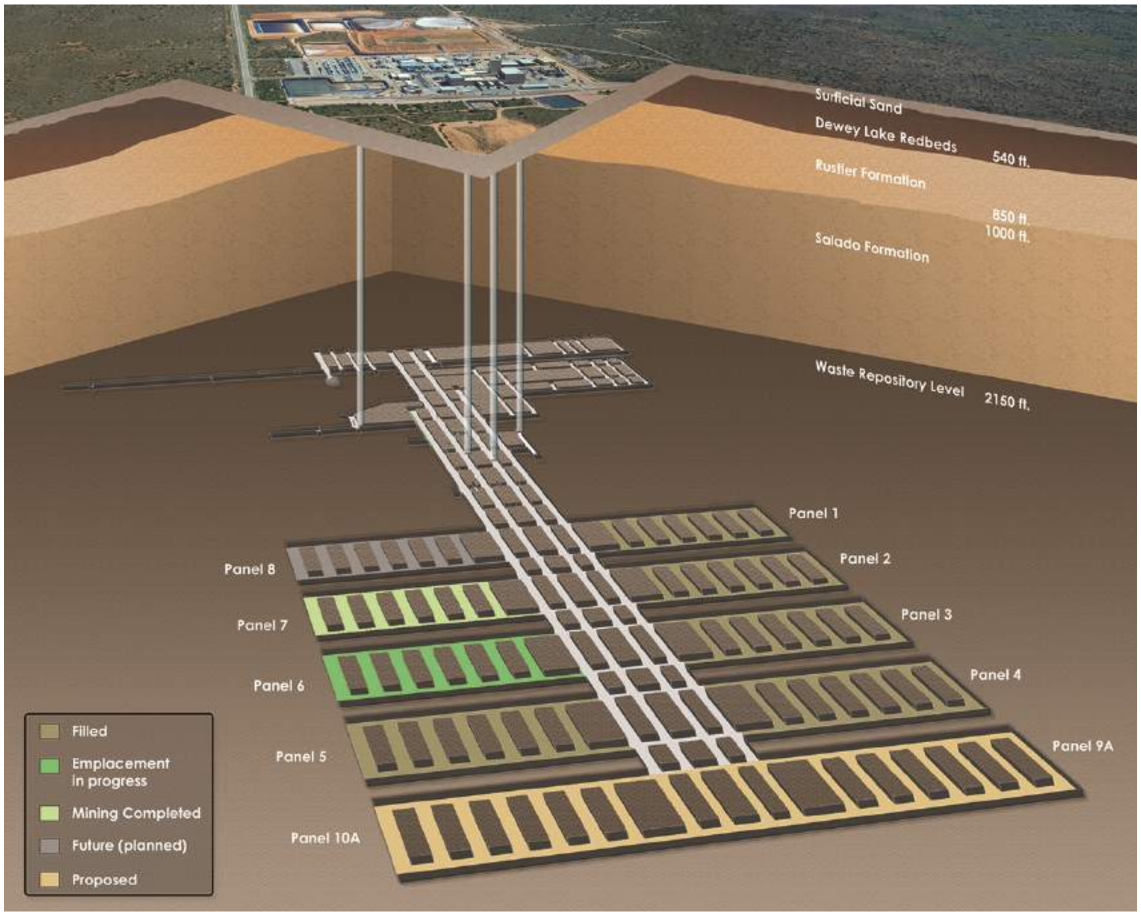


Figure 1.1: WIPP Facility Overview [nuk]

Dose Limit (Whole Body)	Activity Performed	Conditions
50 mSv	All	
100 mSv	Protecting major property	Where lower does limit not practicable.
250 mSv	Lifesaving or protection of large populations	Where lower does limit not practicable.
>250 mSv	Lifesaving or protection of large populations	Only on a voluntary basis to personnel fully aware of the dangers involved

Table 1.1: Department of Energy radiation dose limits. [par]

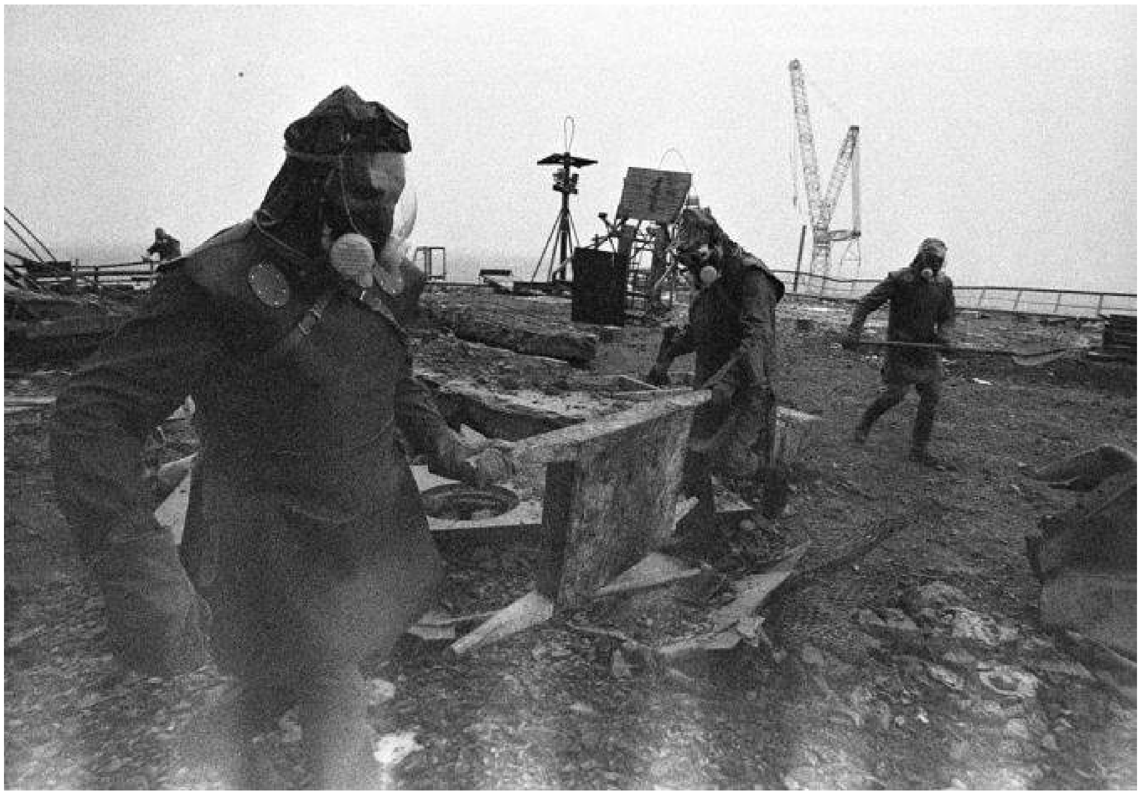


Figure 1.2: Workers known as “biorobots” working on the roof of reactor 3. They would run, remove a few shovels, and then leave. Workers recall feeling pain in their eyes and a metallic taste in their mouth due to the high radiation. [nat11]

Chernobyl liquidators were tasked with limiting the immediate and longer-term dangers from the disaster. Experiences were grim, with some men (later called “biorobots” shown in Figure 1.2) receiving instructions to throw a shovelful of radioactive dust and run, in the hopes of limiting their dosage. Several firefighters were exposed to over 180sV (note this is sieverts, not millisieverts), absorbing fatal doses in just 48 seconds. When they died two weeks later, they had to be buried in lead coffins welded shut.

Surprisingly, robotic solutions in the nuclear industry have been very limited. These solutions often take the form of Master-Slave Manipulators such as in Figure 1.3, which require human operators to use basic tele-operation tools.

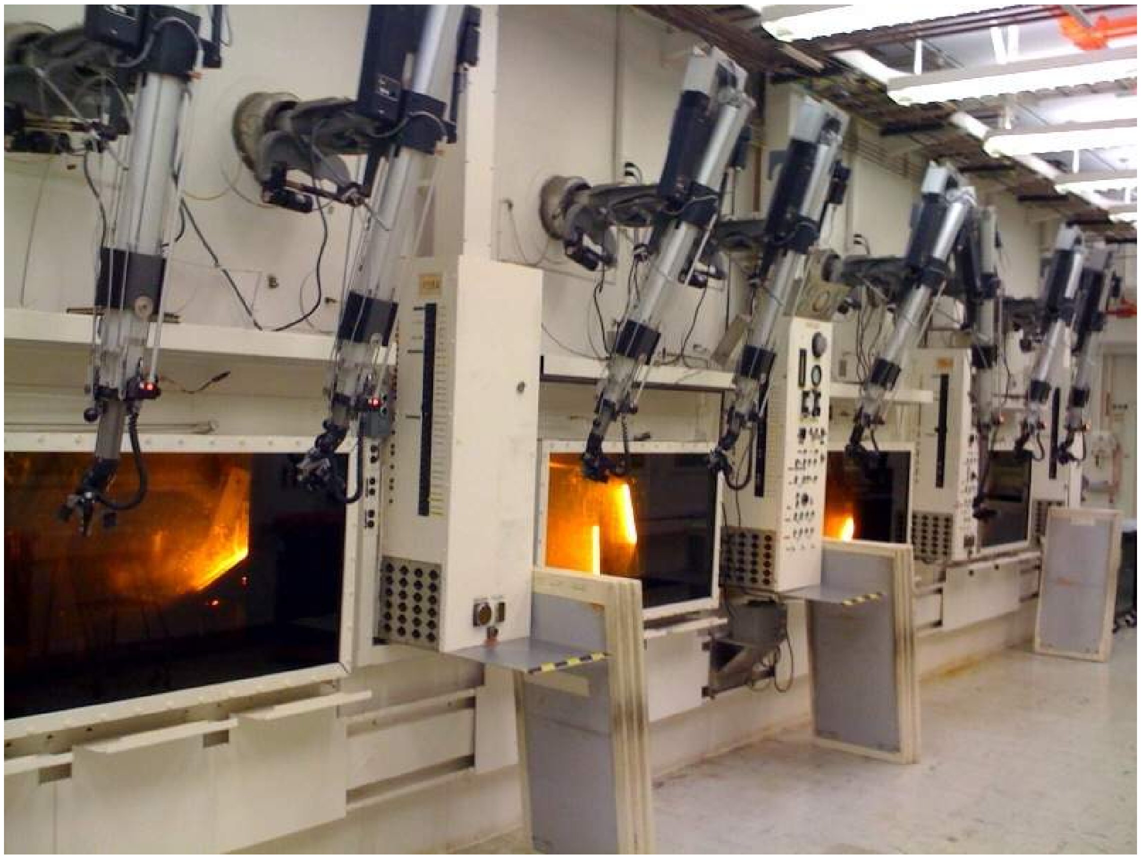


Figure 1.3: Alpha Gamma Hot Cell Facility at Argonne National Laboratory [alp]

(a)

(b)

Figure 1.4: Examples of remote-controlled robots. (a) Mighty Mouse [mig]. (b) Gemini-Scout [gem].

In the case of autonomous robots, we usually see a single robot with cameras such as the Mighty Mouse and Gemini-Scout in Figure 1.4, which are often used for disaster situations to probe an area ahead of human rescuers. Semi and fully autonomous robots can navigate in locations that are dangerous or inaccessible for humans - such as channels at the bottom of the tanks or pipes in waste treatment facilities, helping to maintain the safety of inspectors.

1.2 Mobile Robots

Long the domain of research laboratories and carefully controlled factory settings, robots have begun to move out into the world. These mobile robots are able to sense, localize, and navigate their environments. Recent efforts are being undertaken to perform these tasks in more complex environments, and with greater accuracy. Funding and R&D support is coming from both government and private institutions, such as:

- Government agencies: Department of Energy, Department of Defense, National Aeronautics and Space Administration, National Science Foundation
- Private industry: Alphabet/Google, iRobot, Rethink Robotics, Kuka, DJI
- Academic institutions: Florida International University, Carnegie Mellon, Texas A&M, École Polytechnique Fédérale de Lausanne, ETH Zurich

Examples of current robotic applications such as those in Figure 1.5 run the gamut from the Amazone BoniRob used to check plant phenotypes and perform precision spraying [Ama], to the MQ-9 Reaper UAV used for dynamic targets and intelligence collection [mq-15a]. Boston Dynamics is well-known for their Atlas humanoid robot's ability to not only walk, but also run and jump [atlb]. We also have NASA's R5/Valkyrie humanoid robot, which is capable of operating in degraded or damaged environments [Kis15]. NASA also has the Curiosity rover, equipped with an enormous suite of sensors, and operating semi-autonomously on Mars [mis]. More down to Earth, there is an early deployment of Starship Technologies' delivery robot [Bur18], which uses simpler sensors to accomplish tasks similar to Curiosity: navigating complex environments. Similarly, the Knightscope robot [knia] is being used in crowded areas to extend a human security guards awareness. Florida International University has developed and tested a tethered pneumatic crawler robot for inspecting pipes in nuclear facility settings, with similar applications as the work in this dissertation. Finally, there is the surge in competition for developing a self-driving car, such as Waymo's forays into this field [waya].

(a)

(b)

(c)

(d)

(e)

(f)

(g)

(h)

(i)

Figure 1.5: Examples of robots and their applications. (a) Amazone [bon]. (b) MQ-9 Reaper UAV [mq-15b]. (c) Atlas [atla]. (d) NASA R5/Valkyrie robot [Hal15]. (e) Curiosity Rover [mar]. (f) Delivery robot [sel]. (g) Knightscope patrolling robot [knib]. (h) FIU Applied Research Center Crawler [cra]. (i) Waymo self-driving car [wayb].

1.3 Fundamental Challenges for Robotic Tasks in Deactivation & Decommissioning

Many structures undergoing Deactivation and Decommissioning (D&D) are either too radioactive or inaccessible for humans to directly inspect. This can lead to deteriorating infrastructure, and potentially hazardous situations when no information is available. Semi and fully autonomous robots can navigate in locations that are inaccessible for humans, helping to maintain the safety of inspectors.

1.3.1 Modeling

Most everything that is done in robotics can be considered a modeling problem. In this dissertation, modeling often concerns itself with modeling both robots and their environment. The environments that robots will be working in are often not designed for remote inspection, and can be cluttered or very narrow. To simplify this problem, environments are often represented as a graph structure, where robots travel to and from vertices along edges. The vertices can represent locations of interest or safe locations for robots to stay in. When thinking about modeling, we often assume that the object of interest is a physical object, such as a wall or other obstacle; however, we may also seek to model other phenomena. As an example, in Chapter 2 we will cover how radiation might be mapped, and generate a regression model for predicting radiation intensity at unvisited locations.

1.3.2 Navigation

With our map in hand and a specified location to visit, navigation attempts to create a trajectory for the robot that takes it to the goal location - most likely while also minimizing some value such as total distance traveled, or maximizing another value such as information gain. Navigation entails both *path planning* to specify an overall movement strategy, and *obstacle avoidance* which concerns itself with more local modifications to the robot path. Path-planning is used throughout this dissertation. In Chapter 2, we include the addition of information gain when determining which path a robot should take. In Chapter 3, we introduce the issue of obstacle-avoidance when executing a path. In Chapter 4, paths can be generated for both robots as well as independently operated appendages for a robot.

1.3.3 Coordination

The use of multi-robot systems is being explored as a way for sampling environments and transporting material within long-term nuclear waste storage facilities. Using multiple robots has the benefit of distributing workloads and allowing for faster overall completion rates and more robust operations. However, this brings with it the new issue of how to best coordinate the robots so that they avoid collisions with each other, and effectively distribute work. This is at odds with current robot job allocation and scheduling, which is often performed in an ad-hoc manner and is complicated by unknown environments. Moreover, many of these robots are controlled by multiple operators, and clumsy operator controls have led to making mistakes and robots being damaged [ABB$^+$15]. This was evidence in the case of the Fukushima-Daiichi nuclear power plant, where robots requiring multiple operators suffered coordination issues and led to several robots being lost.

1.4 Scope and Overall Strategy

To address the shortcomings of existing approaches, the proposed research will address three questions impeding robotic inspection:

1. **Thrust 1: Informative path planning in constrained environments** - The environments that robots have to contend with are difficult to traverse; in the case of the Hanford tanks, robots must navigate small, 1.5-inch refractory slots at the bottom of the tanks, as well as carry a tether for safety purposes. These constraints hinder where samples can be taken and makes movement difficult. *How do we sample so that cost is minimized while still reducing model uncertainty?* We make use of Informative Path Planning (IPP) to determine where to sample, and to detect anomalies.

2. **Thrust 2: Robot task allocation in complex environments** - D&D structures may be partially unknown, as buildings can differ from their original blueprints. Therefore, robots must cope with unknown, possibly dynamic environments. Task allocation in these environments is further complicated as robots may have different capabilities, and tasks have various requirements and deadlines. The question then becomes: *how to best allocate tasks among many heterogeneous robots to improve the task completion rate?* We utilize a greedy heuristic approach to rapidly trim the large search space and arrive at an online solution.

3. **Thrust 3: Robot policy generation given operator constraints** - Robots often require operator oversight when executing complex maneuvers. However, assigning operator attention to multiple robots is challenging to scale to large numbers of robots or robots with the need for multiple operators. *How do we effectively allocate operator attention?* The core of our strategy is a geometric representation of the problem, which not only allows for a graphical representation of the problem but also for the use of motion-planning techniques to quickly arrive at a solution via sampling.

These main topics allow us to 1) reduce the search space when looking for areas of interest, 2) indicate how to dispatch robots in order to complete tasks, and 3) in the event that a robot cannot perform fully autonomously, coordinate operator attention to assist robots.

1.5 Organization of the Dissertation

This chapter concludes with an overview of the remainder of the dissertation. Chapters 2, 3, and 4 contain original contributions. Chapter 5 closes with a review of the significant contributions. The main topics of this dissertation are detailed in the subsequent chapters as follows:

- **Chapter 2** We review some of the difficulties we face when attempting to survey an area using robots in order to map out a field of interest. The chapter provides a methodology for mapping out a temperature field in a constrained environment and evaluates the methodology in a simulated environment. Early physical experiments are outlined, and potential avenues of future research are provided.
- **Chapter 3** Once the task of mapping has been accomplished, robots may need to perform specific tasks. This chapter described how tasks could be allocated amongst multiple heterogeneous robots in a manner that increases overall system performance. The work was tested in various simulated environments with good results.
- **Chapter 4** Finally, we consider situations in which a robot is not able to perform entirely autonomously, and must instead rely on the assistance of a human teleoperator. This chapter provides a solution which allows few operators to oversee a large number of individual robots or complex robots which would typically require

multiple operators. The solution was tested in simulation for both simple and high degree-of-free robots, as well as in hardware experiments.

CHAPTER 2

Adaptive Informative Path Planning

2.1 Introduction

During World War II and the Manhattan Project, large amounts of high-level radioactive waste were generated. Some of these wastes are in liquid form and stored in large double-shell tanks at the Hanford Facility in Washington state. These structures are now in a surveillance and maintenance phase which requires continuous monitoring to check for containment failures. Contamination of these and similar structures can result from leakage, and one tank has been confirmed to have leaked [EGHR].

Localizing the source of these leaks is difficult due to the structure of the tank: the tanks are buried approximately 15ft underground, and the bottom of the tank is another 45ft deeper, as illustrated in Figure 2.1 [BG13]. Inspection of the structural integrity of the tanks can only be accessed via narrow annuli at ground level, further complicating sensor deployment. Moreover, this only serves to reach the bottom of the tank along its perimeter. Access to the rest of the tank bottom must be done through a series of narrow 1.5-inch cooling refractory slots located at the bottom of the tank. We might consider deploying a sensor network throughout the refractory slots, however the sensors may interfere with the air being circulated through the slots, or the moving air could dislodge the sensors. Deployment in such a constrained environment faces the issues of how to transport and attach the sensors, and how to power and communicate with them over extended periods of time. Existing inspection approaches use a pole-mounted camera; however, this can only inspect the perimeter of the tank and the outermost segment of the refractory slots. Furthermore, pole-mounted visual inspection requires the operators to manually inspect each of the refractory slots [Gir15], leading to a labor-intensive, time-consuming process.

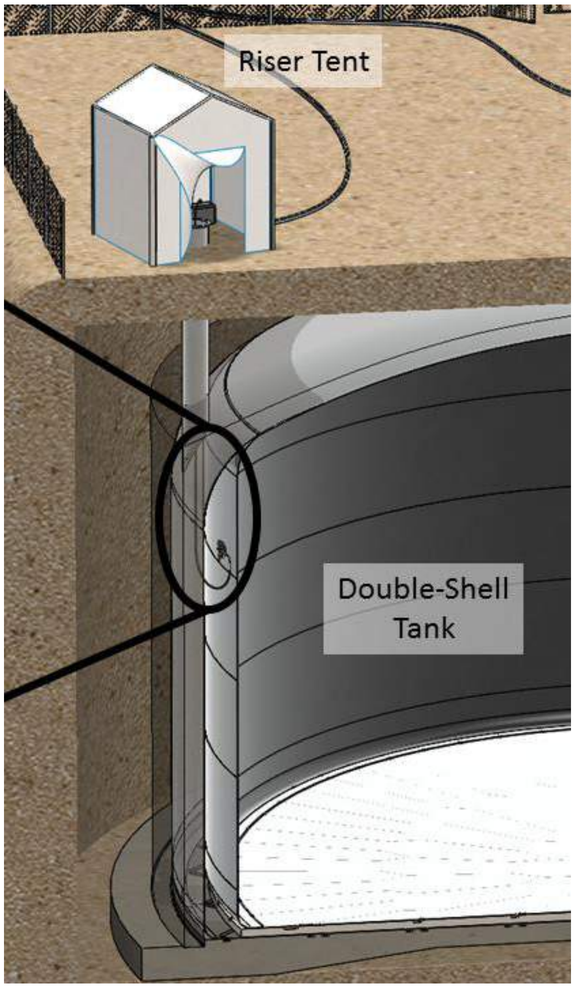


Figure 2.1: Tank cutaway showing buried tank and access annulus. [Gun15]

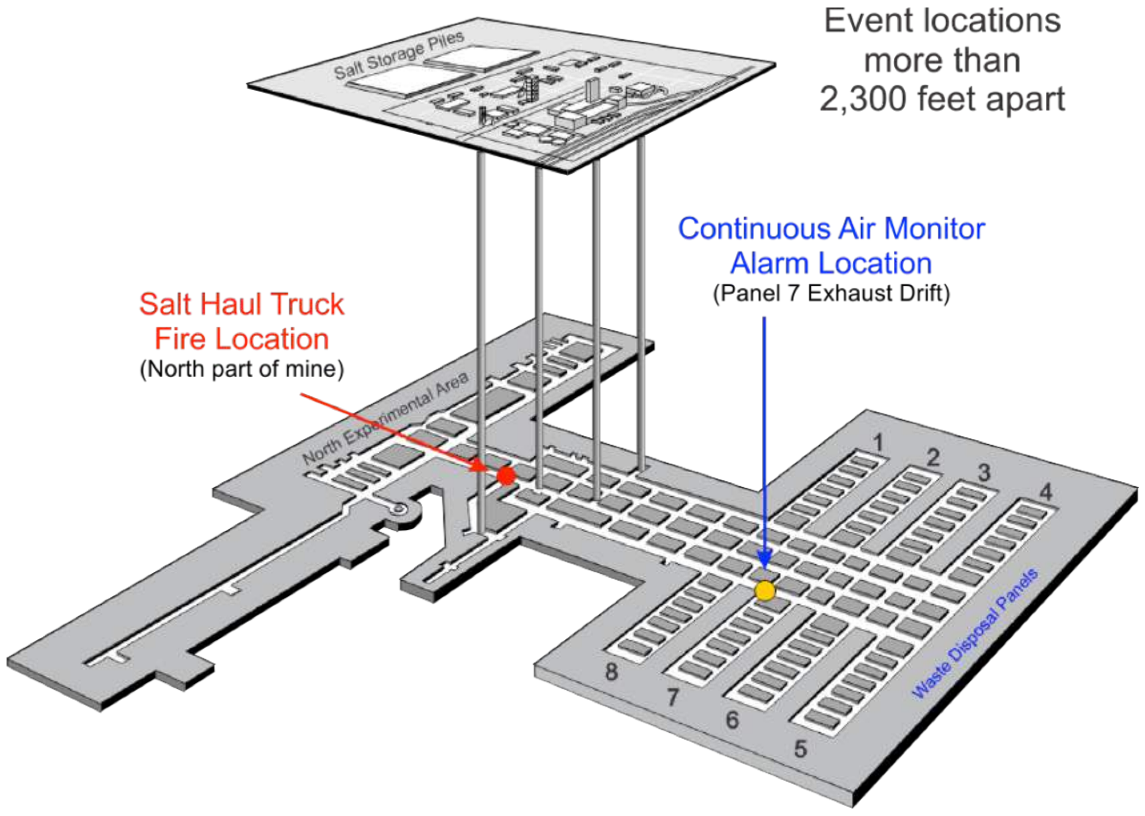


Figure 2.2: WIPP Facility 2014 accident details. [was]

As a further motivating example, we can look at the Waste Isolation Pilot Plant (WIPP) Facility. In 2014, an explosion (illustrated in Figure 2.2) from one of the barrels holding waste led to the entire facility being closed until teams could respond. Inspection of the area was greatly delayed due to safety concerns and a lack of rapid robotic response solutions.

There are sampling methods such as [HGG^{+}14] which attempt to perform a spatial extrapolation given samples at selected discrete locations. Care must be taken when selecting and adopting a sampling approach, as an inappropriate regression model or utility function may cause problems ranging from non-representative samples to the absence of an essential but easily-overlooked location. Furthermore, inaccurate estimations of the spatial variability can lead to the incorrect modeling of the underlying field and contamination properties.

In this chapter, we present a methodology for automating and improving the inspection process of these tanks. We propose an Adaptive Informative Path Planning (IPP) approach that would allow a miniature robotic rover to inspect the tank for locations of interest efficiently. The IPP algorithm incorporates prior knowledge about the tank structure, balances exploration and exploitation to initially locate and then refine the location of locations of interest, and also accounts for the robot's movement constraints.

The rest of the chapter is organized as follows: In Section 2.2, we review existing approaches to IPP, including relevant similarities, and the major differences to our domain. In Section 2.3, we define the environment and problem being tackled, and Section 2.4 describes our approach. Sections 2.5 and 2.6 cover the simulated trials and an analysis of the results. In Section 2.7 we offer a discussion of the results obtained and conclusions.

2.2 Related Work

IPP has wide applicability, used to localize points of interest in forests, oceans, and disaster areas [CLD13]. As such, it has received much attention, and new solutions continue to be proposed due to the myriad domain-specific issues which can render existing approaches insufficient. Traditional localization such as [Mic17] often uses a rastering (zig-zag) pattern to cover an area to map it. This may take a long time to localize the source if it is opposite to the starting position. Another solution is to determine which locations might be most informative a-priori, and then execute a minimum-cost tour of those locations [HS13].

Adaptive sampling aims to provide better results with less time, by actively adapting its sampling locations. In this work, we utilize a modified Gaussian Process - Upper Confidence Bound approach (GP-UCB) [SKKS09] to select sampling locations efficiently. The goal of this work is to deploy a robotic system for localizing radioactive leaks, and thus has some similarities to [CCS$^+$16, QSBZ12] where robots are fitted with optical and radiation sensors to find radiation sources, and [CMS16] which has a strong showing of aerial vehicles and their associated mapping techniques. The approach in this chapter differs in its unique environment and the resulting constraints such as limited robot movement.

Unlike some existing IPP approaches which rely purely on the informativeness of possible sampling locations [GK11], we also incorporate robot dynamics such as movement and tether constraints that limit the robot's ability to visit certain locations easily. This bears some resemblance to work by [MR12, MR14] where a Gaussian Process is used to model both the phenomenon and the quality of possible paths. With regards to path-planning, we draw inspiration from existing approaches by Brass et al. [BVX15] which performs path planning for a tethered robot given polygonal obstacles, and Kim et al. [KL15] which use a Multi-Heuristic A^* algorithm to find paths for a tethered robot with a homotopy invariant augmented graph. In order for the robot to navigate between the vertices in the graph representation of the refractory slots, we use a modified A^* algorithm such as the one described in [ZBT17].

2.3 Problem Formulation

The goal of inspecting the tanks is to detect anomalies - in this case, possible leaks. As a proxy for finding the leak, we use the temperature distribution at the bottom of the tank, which would be impacted by the presence of a leak. We are looking to create a map of the temperature distribution at the bottom of the tank - which is represented by an unknown scalar field $f : \mathbb{R}^d \to \mathbb{R}$ - from samples Y selected from a set of potential sampling

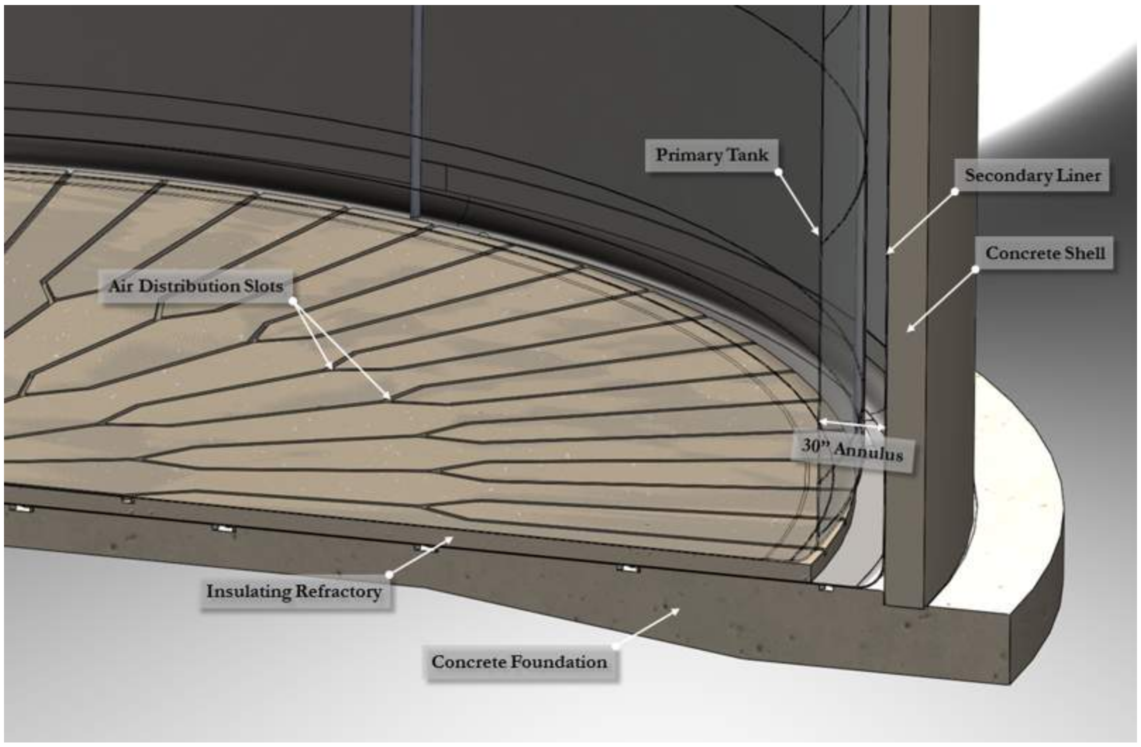


Figure 2.3: Tank cutaway showing the inner and outer shells, refractory slots, and annulus at the sides of the tanks. [BG13]

locations V. Given the samples, we seek to find the location with the highest temperature, corresponding to the most likely source of a leak. We desire to select the sampling locations which best update the model, but also keep the overall distance traveled as low as possible while respecting the kinematic constraints of the robot. Complete coverage would aim to map out the entire tank, and lends itself to an exhaustive approach. This formulation instead seeks to find the leak more quickly than with a traditional exhaustive approach.

The approach and simulation in this work were designed for deployment in the Hanford facility double-shell tanks. These tanks are composed of an inner tank that holds the high-level liquid radioactive waste and an outer shell serving as a fail-safe if the inner tank leaks. Figure 2.3 illustrates a cutaway view of the tank, indicating the inner storage vessel, refractory slots, and the gap between the inner and outer tank walls. Here, we will describe the structure in further detail.

2.3.1 Refractory Slot Structure

Sandwiched between the bottom of the two tanks is a series of air distribution slots seen in Figure 2.3, also known as refractory slots. These slots serve as an air distribution system to cool the primary tank and provide an avenue for inspecting the bottom of the inner tank without actually entering the tank itself. The tanks were built over multiple years and have slightly varying refractory slot designs. In this work, we focus on the design of the AY-series tanks, which consists of 1.5-inch-wide slots arrayed radially outwards as in Figure 2.4(a).

We model the refractory slots as a graph, with the slots represented as edges E, and the forks as vertices V. The robot cannot execute tight turns (cannot turn at a fork to go down an adjacent slot). Given this graph structure, we only considered vertices as valid sampling locations, rather than the continuous plane representing the tank floor. As such, we introduce additional evenly-spaced vertices along edges such that a minimum desired sampling resolution is achieved.

The bottom of the tank is formed by multiple steel plates welded together, meaning that there are weld seams between the various plates. These weld seams run in a North-South, East-West pattern as in Figure 2.4(b), with occasional overlaps along the refractory slots indicated by the purple points in Figure 2.4(c). Many of the weld seams had initial high rejection rates, and have been reworked several times [BG13]. Due to all the rework, there may be some uncertainty regarding the integrity of the weld seams, making them of higher interest.

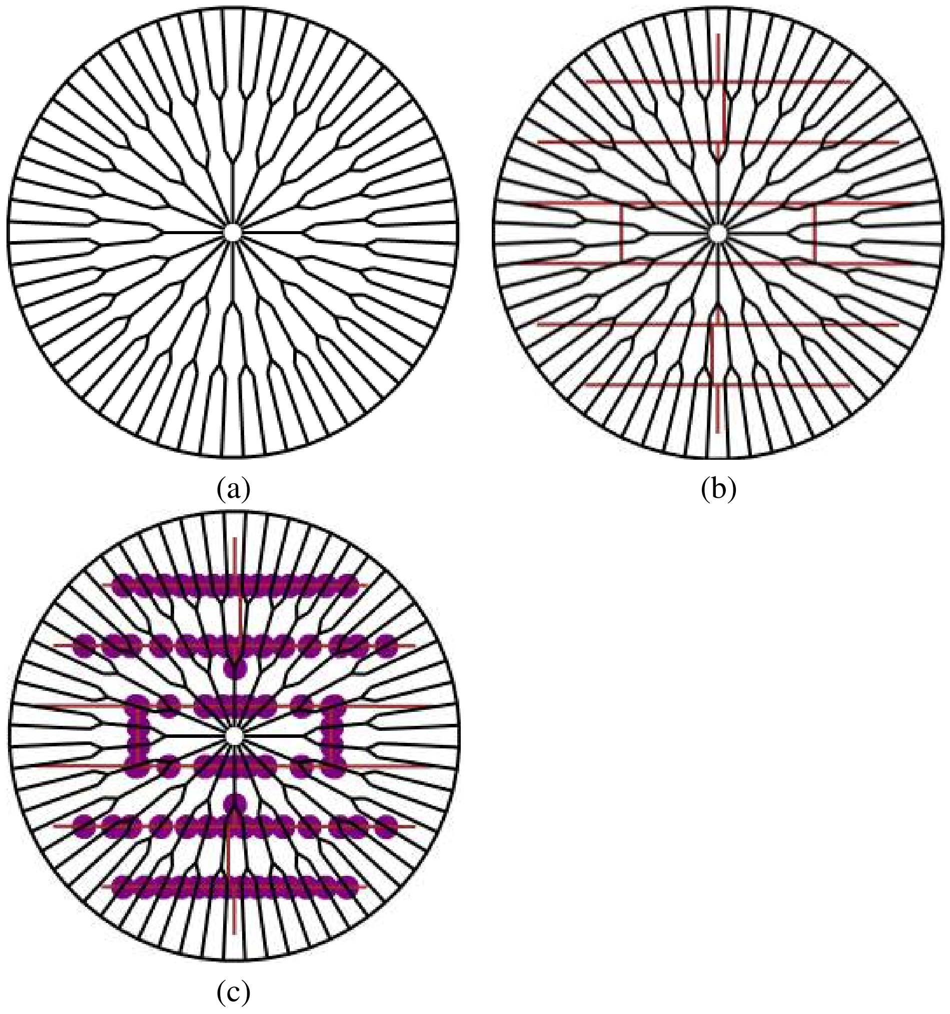


Figure 2.4: (a) Layout of the refractory slots at the bottom of the tanks. (b) View from (a) with the addition of weld seams. (c) View from (b), with locations where refractory slots intersect weld seams highlighted.

2.3.2 Temperature Distribution Modeling

Searching for the leak, we would initially consider searching for the liquid that has leaked out. However, there are confounding factors such as seepage from other sources, as well as the desire to avoid having the robot come into contact with the contamination. We might also mount a radiation sensor to the robot, however, a high-accuracy radiation sensor would not fit within a refractory slot. Moreover, radiation roughly follows an inverse-square law, and its measurements drop off quickly, making localization difficult. Instead, we look towards *temperature* as a proxy measurement, as these sensors are small and sensitive to temperature variations. The Gaussian nature of temperature assists in localizing the leak. The source of the leak corresponds to the peak, and moving further away from the leak leads to a decaying signal. The model was generated using a 2-dimensional multivariate normal distribution with a probability density function such as that in [RW06], with mean vector M, a randomly-generated positive definite covariance matrix Σ. Further details about the model can be found in Section 2.5.

2.3.3 Regression

Gaussian Processes provide a method for modeling unknown fields non-parametrically. Here, we aim to efficiently derive a Gaussian Process regression through a process such as that described in [RW06]. Given a set of N sampling locations, where each location $x_i \in \mathbb{R}^2$ has a noisy measurement $y_i \in \mathbb{R}$ given by $y = f(x) + \varepsilon$ where $f(x)$ is the ground-truth and $\varepsilon \sim \mathcal{N}(0, \sigma_n^2)$. The predicted mean μ^* and covariance σ^* at a specific target location $x^* \in X^*$ is given by:

$$\mu(x^*) = K(x^*, X)K_X^{-1}y$$

$$\sigma(x^*) = K(x^*, x^*) - K(x^*, X)K_X^{-1}(X, x^*)$$

where $K(X', X'')$ is the covariance matrix, X are the sampled observation locations, and $K_X = K(X, X) + \sigma_n^2 I$.

For the covariance function, our implementation uses a Matérn kernel. The finitely differentiable Matérn kernel can better model physical processes, and does not assume as much smoothness as other kernels - such as the infinitely differentiable Squared Exponential kernel - which can yield unrealistically smooth results when modeling a physical process [Ste12]. The Matérn kernel is described as follows:

$$k(x_i, x_j) = \sigma^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\sqrt{2\nu}\frac{d}{\rho}\right)^{\nu} K_\nu \left(\sqrt{2\nu}\frac{d}{\rho}\right) \tag{2.1}$$

Here, x_i, x_j are two locations and d is the distance between them, which is parameterized by $\rho > 0$. We control the smoothness of the function via $\nu > 0$. Γ is the gamma function, and K_ν is the modified Bessel function of the second kind [RW06]. We select $\nu = 1.5$ (a once-differentiable function) to avoid having to compute the Bessel function, allowing for a roughly 10 times faster computation. For optimizing the kernel's parameters, we use the Limited-memory BFGS (L-BFGS-B) [BLNZ95] optimization algorithm, which is designed to smooth functions, and has linear memory usage. The first run of L-BFGS-B is done with the kernel's initial parameters, and then an additional n times (restarts) using θ derived from a random log-uniform distribution within the allowed bounds. With experimentation, setting the number of restarts for the optimizer to 10 yielded good regression performance while reducing runtime.

2.3.4 Robot

In this work, we model the robot as a point robot, capable of moving along edges in the graph from one vertex to another. To simplify the problem, we assumed a movement time of 1ft/s, and a sampling time of 10s when measuring the temperature at a location. The robot has a tether, which was selected to be long enough to allow the robot to access any point in the refractory slots, but short enough that the furthest location would require using all the length of the tether with minimal slack. This tether is required to:

- Power the robot
- Send/receive commands and sensor data
- Allow for removal of the robot in the event of system failure

The tether limits the distance the robot can travel, as well as constrains its movement (loops and tight turns are not possible). The most noticeable effect of the tether is that the robot cannot wholly circle the tank, and must instead retract to the insertion point and then go the other direction when reaching locations on the far side of the tank. An illustration of this can be seen in Figure 2.5, where the robot must circle back before exploring the other half of the tank. Moreover, the tether must be dragged by the robot, which is difficult given the small size of the robot (1.4 x 1.2 x 2 in), and is complicated by turning around corners where the tether experiences additional friction.

2.3.5 Path Estimation

Given the structure of the refractory slots, a traditional Euclidean distance between points is not an adequate metric for considering the cost of traveling. Instead, the entire path-cost must be computed, taking into account the need to backtrack out of individual slots. We augment Dijkstra's algorithm [CLRS09] with the constraints that the robot has:

- a limited-length tether
- to travel backward to exit a slot
- to enter and exit via the same slot, before re-entering an adjacent slot (the robot cannot enter a slot and exit via another due to the space constraint at a fork)

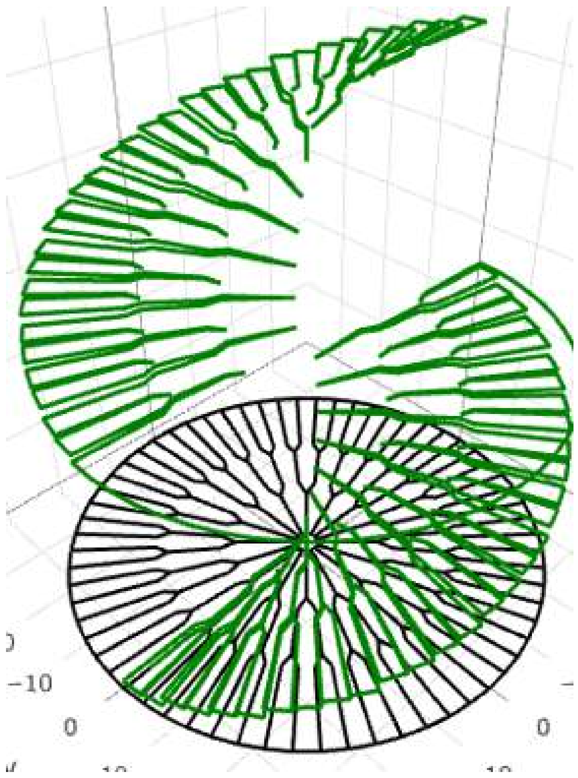


Figure 2.5: Time representation of a robot moving through refractory slots using an exhaustive approach. Vertical movement corresponds to the robot moving through time.

In Algorithm 1, we show the pseudocode for the modified Dijkstra. The inputs to the algorithm are the adjacency graph *G*, start and goal locations, cost of reaching the current location, the current tether occupancy, and the max length of the tether. In lines 1 – 7, we initialize a frontier priority queue that contains the vertices to explore in the order provided by a priority heuristic. The came_from dictionary contains the relationships showing how vertices are connected to each other; cost_so_far indicates the cost to reach each explored vertex, and the tethers dictionary shows the cells occupied by the tether to reach each explored vertex. Each of these is initialized with the starting location of the robot, the tether occupancy, and the cost to reach the current location. In line 8 and 9, we pop the frontier for unexplored vertices. Line 10 checks each neighbor of a vertex that is currently being expanded. In lines 11 – 15, we keep track of the stack of the tentative_tether, extending or shortening it as the robot moves along. Line 16 considers the cost of reaching the neighbor given the cost already expended, and the additional effort of moving to the next neighbor. In line 17, we check to see if the route under consideration exceeds the maximum tether length. If not, we also check if that neighbor has not already been explored, or if the route under consideration has a lower movement cost or tether length than the previously examined route. If these conditions are met, save the new route in lines 18 – 21. Finally, we check in line 22 to see if the goal has been reached. If so, we reconstruct the path from the saved neighbors in lines 23 – 24. Otherwise, no valid path exists, and we return None in lines 25 – 26. The solution here only takes the robot between two locations, start and goal. To plan a path through all the desired sample locations, the planning process is repeated in a sequential manner where the goal location of the previous search is assigned as the start location for the next search until all the sample locations have been visited. The corresponding cost and the tether occupancy of the robot are also updated along the searching iterations. Figure 2.6(c,d) shows the movement of a robot throughout the slot network, with the z-axis representing the order of sampling sequence.

2.4 Gaussian Process Modeling and Sampling Location Selection

Building on Section 2.3, we accept as input a graph $G = (V, E)$ representation of the refractory slots, and discretize the graph to the desired resolution by inserting additional vertices along the edges as needed. This process allows us to approximate the continuous sampling space using a more straightforward discrete representation. The vertices also encode the angles between each other, to prevent the robot from attempting tight turns which would cause the tether to become stuck.

Algorithm 1 Tethered A*

```
1: Input: $(G, x_{init}, c, x_{goal}, \mathrm{T}, \mathrm{T}_{max})$
2: frontier $\leftarrow$ PriorityQueue
3: came_from $\leftarrow$ Dictionary
4: tethers $\leftarrow$ Dictionary
5: frontier.put($x_{init}$, 0)
6: came_from[$x_{init}$] $\leftarrow$ c
7: tethers[$x_{init}$] $\leftarrow$ tether
8: while frontier$\neq \emptyset$ do
9:     current $\leftarrow$ frontier.pop
10:    for neighbor $\in$ g[current] do
11:        $\mathrm{T}_{tent}$ $\leftarrow$ tethers[current]
12:        if neighbor == previous_position then
13:            $\mathrm{T}_{tent}$.pop
14:        else
15:            $\mathrm{T}_{tent}$.append(current)
16:        $c_{new}$ $\leftarrow$ cost_so_far[current] + $c_{move}$
17:        if ($|\mathrm{T}_{tent}| \leq \mathrm{T}_{max}$) and ((neighbor $\notin$ cost_so_far) or ($c_{new} \leq$
       cost_so_far[neighbor]) or ($|\mathrm{T}_{tent}| \leq$ |tethers[neighbor]|))) then
18:            cost_so_far[neighbor] $\leftarrow$ $c_{new}$
19:            frontier.put(neighbor, $c_{new}$)
20:            came_from[neighbor] $\leftarrow$ current
21:            tethers[neighbor] $\leftarrow$ $\mathrm{T}_{tent}$
22: if $x_{goal}$ $\in$ came_from then
23:     path $\leftarrow$ reconstruct_path(came_from, $x_{init}$, $x_{goal}$)
24:     return path
25: else
26:     return None
```

These restrictive constraints permit the solution to be used in similar environments such as the tanks at the Savannah River National Laboratory [LLC17], or the Waste Isolation Processing Plant (WIPP), which also has a channel-like structure. Our ideas can also be applied to more traditional open environments. If a graph structure is not initially available, a Voronoi decomposition or cell decomposition [LaV06] may be used to generate a graph.

2.4.1 Sampling Site Selection

Our approach consists of a modified Upper Confidence Bound algorithm: Given the current state of a Gaussian Process Regression, we use the predictive output mean μ^* *(Exploitation)* and variance σ^* *(Exploration)* at the candidate sampling locations $\mathscr{S}$, which is initially equivalent to V. Weld seam bias w serves to increase the expected utility of prospective sampling locations that lie on top of a weld seam, given the expected higher failure rate of weld seams due to their high initial rejection rate.

The exploitation and exploration values are normalized at each step $t \in T$, with regards to the highest-valued predicted output in the set S. The weld seam value is set to a constant $w = 1$. These elements are then respectively weighted by λ, where $\lambda \in [0,1]^3$ to yield: $utility = \lambda \cdot [\mu^*, \sigma^*, w]$, where locations with a higher value are deemed more desirable.

We will now cover each of the parameters in detail, and how they affect the model's behavior:

Exploration vs. Exploitation

Here, we discuss the most critical component of the modified UCB algorithm: the trade-off between exploration and exploitation.

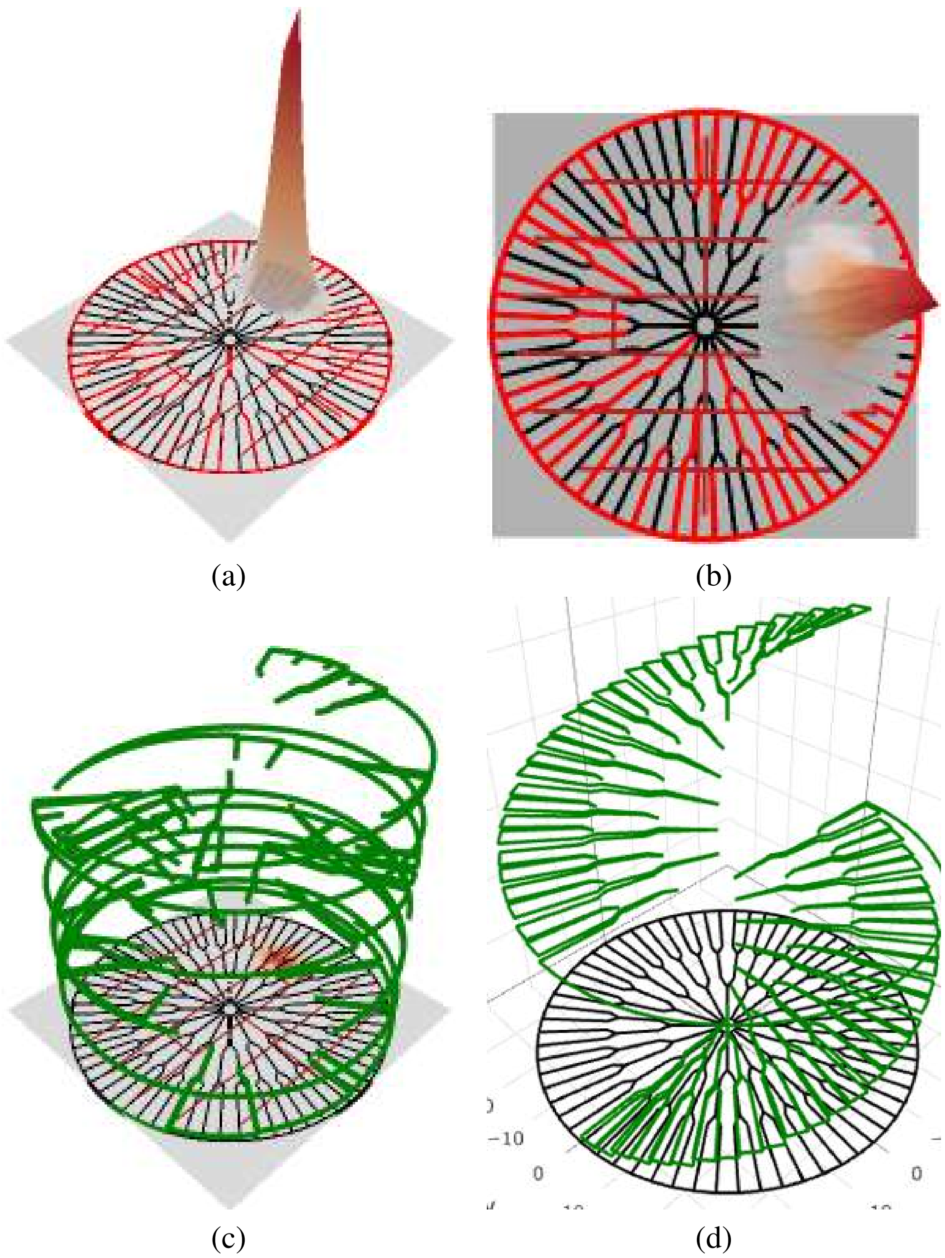

(a) (b) (c) (d)

Figure 2.6: (a) Isometric view of an example Gaussian-Process-like temperature distribution - with the center being the leak source - overlaid on refractory slots. (b) Overhead view of (a). (c) Example time representation of a robot moving through refractory slots, where the vertical z-axis is time. (d) Time representation of a robot moving through refractory slots using an exhaustive approach.

Start by constructing a distribution that describes the Gaussian Process we are looking to reconstruct. Adding more observations, the distribution improves, and the uncertainty

Algorithm 2 Bayesian Optimization

```
1: Input: Possible sampling locations $V$, Utility function $S$, Update Rate $r$, Number of
   samples to take $n$
2: $V' \leftarrow V$
3: for $t \in [1,n]$ do
4:     Evaluate $S(v)$ over $V'$
5:     $x \leftarrow argmax\ S(v)$
6:     $Sample(x)$
7:     $V' \leftarrow V' \backslash x$
8:     if $t\%r == 0$ then
9:         $UpdateGP(x)$
```

(variance) diminishes near sampled locations allowing us to determine which locations need to be further explored. The UCB algorithm shown in Algorithm 2 is a modification from [MR12, SKKS12], and selects a new sampling location based on the weighted mean and variance. A higher mean biases to rapid localization and a higher variance to total coverage. This process is done by finding the maximum of the UCB utility function, which serves as a computationally simpler proxy for the task of regression [SLA12, BCDF10]. We can also incorporate knowledge about the tank dimensions into the regression. The bounds of the kernel length scale are allowed to range between $[1e^{-5}, 80]$, corresponding to just over the maximum diameter of the tank, and we set the initial estimate for the length scale to be 40, the midpoint. We must also set α, which is the value added to the diagonal of the kernel matrix when fitting the model. Small values correspond to less noise, whereas high values indicate greater noise, equivalent to using an additional White Kernel. Here, we set $\alpha = 0.2$, which roughly correlates to the $\pm 2°C$ error margin of the temperature sensor model.

Weld-Seam Bias

The previous section assumes that the only way to gather information is via new samples. However, we would expect that the weld seams are more prone to failure than the steel plates themselves. With this in mind, we can bias our search to prioritize weld seams that intersect our available sampling locations (Figure 2.4(c)) by adding the weighted parameter w to the utility function. For vertices that lie on a weld seam $x \in V_{weld}$, we add the weighted w to the utility; otherwise, the value is 0.

Continuous Area

The above work assumes a discretized environment composed of a graph. For completeness, we may also adapt this work to function in a continuous environment where robots are not as tightly constrained. In this event, minor modifications are needed: rather than evaluating the utility of the vertices in a graph, we must find a way to accomplish this efficiently over an infinite number of points.

Algorithm 3 Continuous Area Bayesian Optimization

```
 1: Input: Number of warm-up locations $m$, number of optimization locations $l$, Utility
    function $S$, Update Rate, Number of samples to take $n$
 2: $V \leftarrow m$ random samples from the parameter space
 3: $V' \leftarrow l$ random samples from the parameter space
 4: for $t \in [1,n]$ do
 5:     Evaluate $S(v)$ over $V$
 6:     $v,x \leftarrow argmax\ S(v)$
 7:     for $v \in V'$ do
 8:         $v',x' \leftarrow$L-BFGS-B$(-S(V'))$
 9:         if $v' > v$ then
10:             $x \leftarrow x'$
11:     $Sample(x)$
12:     if $t\%r == 0$ then
13:         $UpdateGP(x)$
```

The algorithm is shown in 3 and described here. Begin by taking a random sampling of points V from the parameter space defined by the bounds of the environment, where $m = |V|$ is the number of random samples to take, and evaluate the utility function over those points. Next, we sample the parameter space more thoroughly over $l = |V'|$ points. For these points, we use the Bounded Limited-Memory Broyden–Fletcher–Goldfarb–Shanno algorithm (L-BFGS-B) to find the minimum of minus the utility function. The L-BFGS-B optimization algorithm was selected as it reduced the amount of memory needed, and extends traditional L-BFGS to allow for bound constraints on the variables, reducing the time needed to run. Still, the time needed to evaluate the optimization function is non-negligible, and often $l << m$. The minimum of all the points is then selected as the next location at which to sample.

2.5 Simulation

We performed a series of 200 independent trials, each consisting of a randomly generated hot-spot representing a leak. The hot-spots exhibit a distribution that can be described by a Gaussian Process, and the peak of each hot-spot lies within the bounds of the tank, as in 2.6(a, b). 100 of these trials had the hot-spot centered on a randomly-selected location along a weld seam, to reflect the higher failure rate associated with weld seams compared to the plates themselves. The other 100 trials had the hot-spot generated at a random location within the bounds of the tank. A visualization of this can be found in Figure 2.7.

Without loss of generality, the hot-spot peak intensity (the mean) was set to 100, while the covariance along the x, y axes was randomly selected from the range $[4.5, 18]$. This range was selected as 4.5 is approximately the maximum distance between two refractory slots - and therefore the minimum size the hot-spot must have so that at least one refractory slot intersects it. The upper value of 18 corresponds to 4 times 4.5 and was used to provide a varying range of spread. The resulting hot-spot was then used to evaluate the various weighting schemes.

While executing a trajectory such as in Figure 2.6(c), the robot would sample if it visited a previously un-sampled location, and remove that location from the candidate pool of future sampling locations S. The regression was fitted at every 3^{rd} new sample.

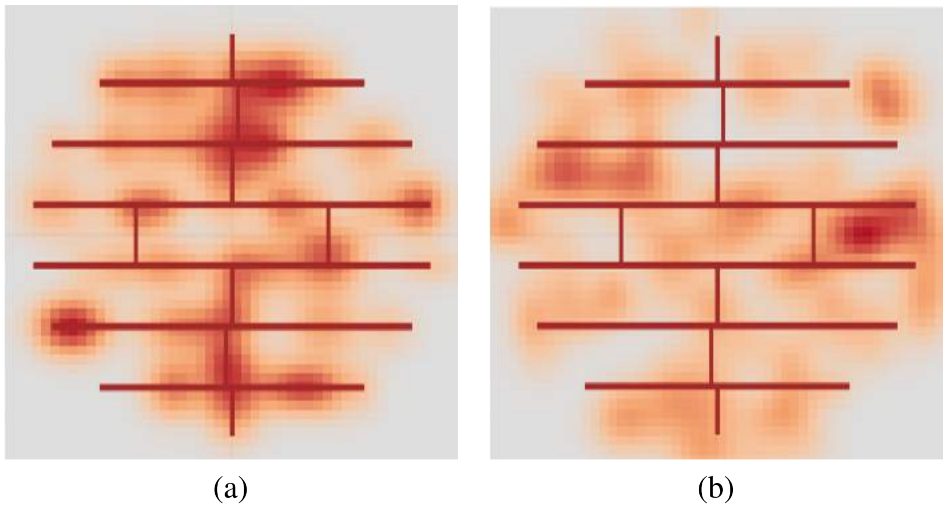


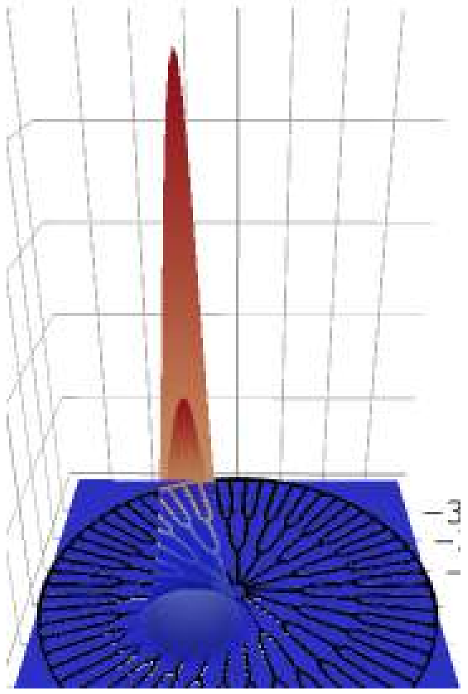
Figure 2.7: Example visualization of distributions. (a) Distributions centered on a random location coinciding with a weld-seam. (b) Distributions placed randomly throughout the tank. On average 3ft away from the nearest weld seam.

This process continued until all vertices in the graph had been visited and sampled. Apart from Informative Path-Planning, an exhaustive approach was executed against the same distributions to establish a baseline. The exhaustive approach used the trajectory shown in Figure 2.6(d) where x, y are the planar coordinates, and z is a representation of time.

2.6 Analysis

Testing of the efficacy of the different weighting schemes was done by comparing their Root Mean Square Error (RMSE) for the predicted value at locations throughout the tank, defined as: $RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}$ where Y is the vector of ground-truth values, and $\hat{Y}$ is the vector of n predictions.

The resulting non-negative loss-value is a measure of accuracy indicating the difference between the predicted values and the ground truth, where a value of 0 is the best score, and larger values correspond to a worse-performing model. A visual example can be found in 2.8(a), where the ground truth is shown overlaid on the prediction. The RMSE


(a)

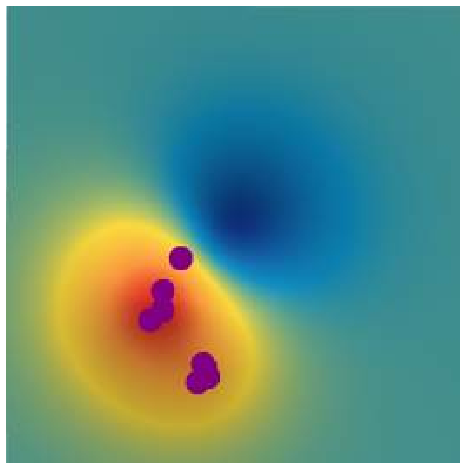

(b)

Figure 2.8: (a) Comparison of ground truth and predicted. (b) Radiation map from GPR given hardware data.

can be thought of like the difference between the two surfaces. We will also refer to "local RMSE", defined as the RMSE for a region surrounding the distributions' center, with a diameter of 1σ, the spread given by the covariance of the distribution. This is done to more clearly illustrate the performance of the regression for the point of interest, rather than the entire area.

2.6.1 Illustrative Examples

In this section, we will outline the weighting schemes that best illustrate how different strategies can affect the performance of the approach described above.

Randomly-Located Leaks

We begin by focusing on a model tank where the leak is generated at a random location. In Figure 2.9, we find a comparison of the average performance among the different weighting schemes across the 100 trials. The y-axis indicates the average local RMSE score every time the regression was updated, and the x-axis shows the average time-step at which the regression was updated across all trials. The models initially begin at an RMSE of approximately 31.

The traditional exhaustive approach (such as the one in [Mic17]) performs as expected, with a steady decrease in the local RMSE. A strictly Exploration-based approach of $\lambda = (0,1,0)$ will naturally perform poorly, as this will make the robot alternate between each side of the tank to visit the most extensive remaining unexplored region. Switching over to a purely Exploitation-based approach of $\lambda = (1,0,0)$, we find remarkably better performance than either the exploration or exhaustive methods as the robot quickly converges on the location with the highest temperature. Attempting to combine exploration

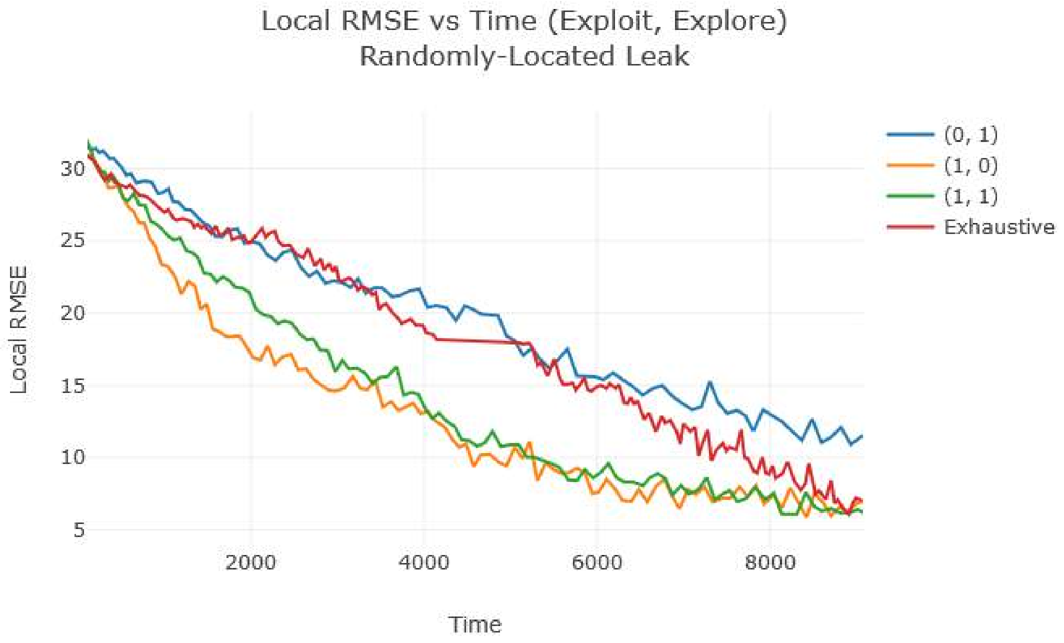


Figure 2.9: RMSE performance of various weighting schemes given leaks created in random locations throughout the tank. The y-axis corresponds to the RMSE, and the x-axis indicates the average time at which the corresponding score was achieved.

and exploitation as $\lambda = (1,1,0)$ results in the robot occasionally moving to distant parts of the tank which have high uncertainties.

Weld-Seam-Biased Leaks

We now turn our attention to the trials where the leaks were created over weld-seams. The accompanying results can be found in Figures 2.10, 2.11. The baseline using the exhaustive approach closely resembles that of the previous trials with a randomly-located distribution, beginning at an RMSE of ~ 31, and eventually decreasing to ~ 5. However, we are primarily concerned with how quickly we can localize the leak. If we consider the threshold to be a 50% reduction in the local RMSE (to be conservative, let's say an RMSE of 15), then an exhaustive approach reaches this threshold at time $6,147$. Using the previous-best weighting scheme of $[1,0,0]$, we reach a $> 50\%$ reduction of the local RMSE by time $3,257$, or almost half the time of the exhaustive method. Taking into account our prior knowledge about weld-seams having a higher failure rate, we compare this with $\lambda = (1,0,1)$. Incorporating prior knowledge provides a slight advantage, shaving off 21% of the exploitation approach to a time of $2,584$, or almost 42% of the exhaustive approach. Incorporating the exploration weight ($\lambda = (1,1,1)$) again shows the same issue as before, giving slightly worse results.

2.6.2 Path-Planning Considerations

A common theme throughout the results shown above is the harsh implicit penalty for moving. In a continuous plane or other obstacle-free environment such as in [CMS16], a robot can quickly move from one location to the next. In the case of the refractory slot structure, moving between locations - even nearby locations - requires the robot to exit the refractory slot, circle the tank, and then enter a different slot, leading to considerable

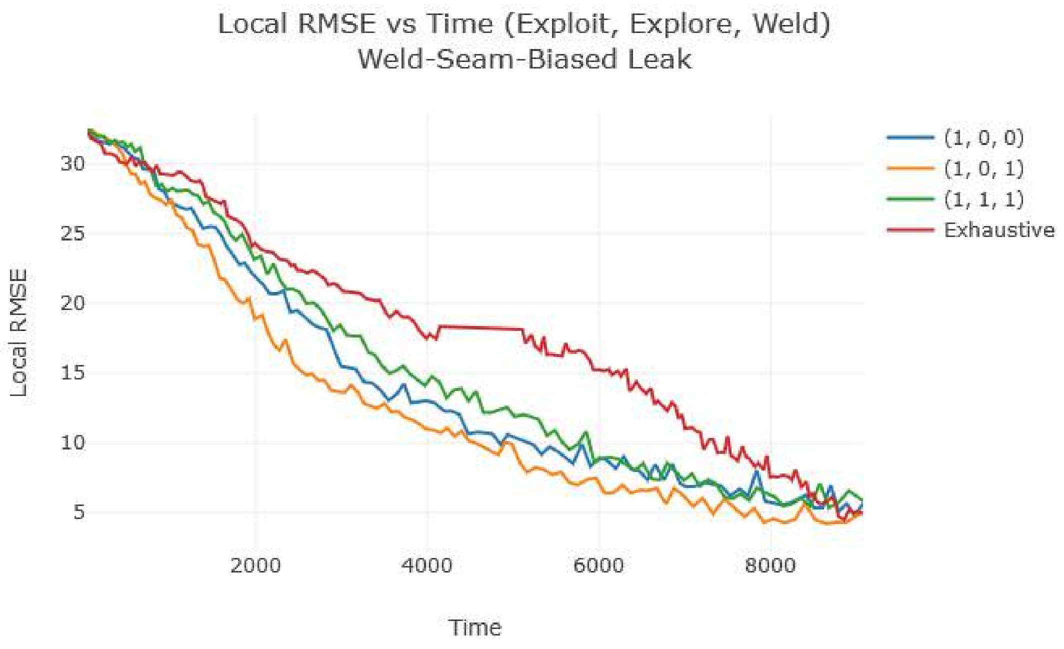


Figure 2.10: RMSE performance of various weighting schemes given leaks biased to weld-seams. The y-axis corresponds to the RMSE, and the x-axis indicates the average time at which the corresponding score was achieved.

movement costs. Motion capabilities are further constrained by the tether forbidding any turns within the forked refractory slots. Thus, any exploration is heavily penalized, as seen in strategies favoring σ.

Readers may notice that the exhaustive approaches have a slight plateau in the middle of their runs. This result is due to the structure of the exhaustive approach path, which must explore one half of the tank, and then circle back before exploring the other half due to the robot's tether constraint. During the circling back of the robot, the regression will be unaffected as those locations have already been sampled.

2.6.3 Leak Behavior and Effects on Weighting Performance

The trials shown here were all simulated with a single distribution (representing a single leak). In the event of two or more simultaneous leaks, we expect that slightly favoring Exploration (σ) might yield better performance as it would help in avoiding the robot becoming stuck on a local maximum. In the future, we plan to examine how the model performs given multiple unknown distributions.

2.6.4 Performance

As was previously shown in Section 2.6.1, we find better performance using an appropriate weighting scheme such as $\lambda = (1, 0, 1)$ than an exhaustive approach. Of interest is not just the overall reduction in the average time needed to localize the point of interest, but also the dispersion. In Figure 2.11 we show the whisker plot for various schemes when operating in a weld-seam-biased tank. Of particular importance is the high variability of the exhaustive approach, and the lower variability of the UCB techniques, showing more consistent times for localizing the leak.

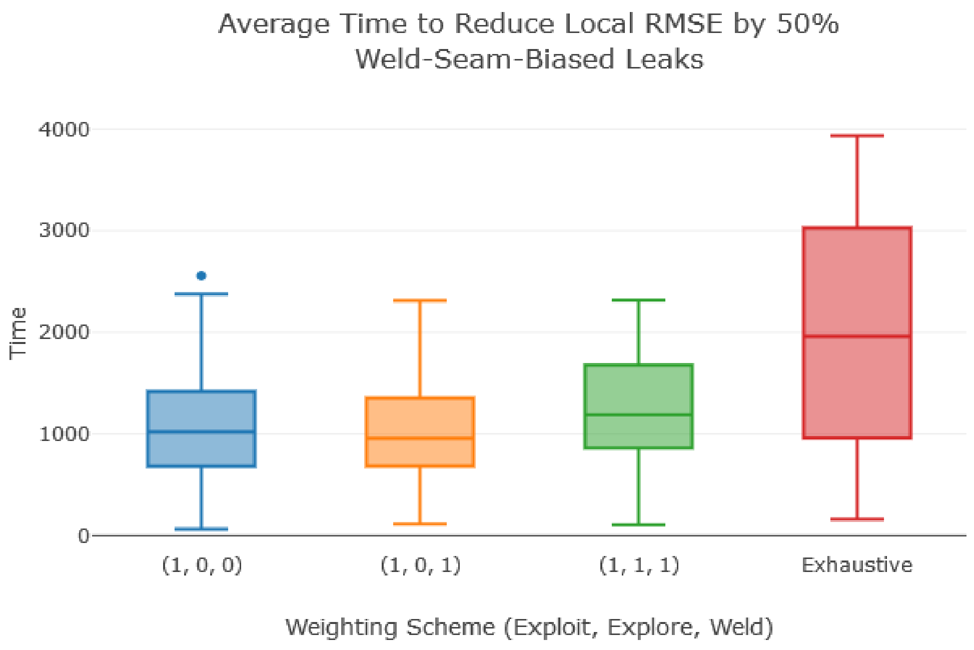


Figure 2.11: Times to reduce the local RMSE to 50% of the original value. Results shown are with regards to weld-seam biased leak locations.

2.6.5 Hardware Experiments

Preliminary experiments were run on a Turtlebot Waffle, equipped with an LDS LIDAR and CZT Spectrometer. We were able to use the robot to randomly visit locations and gather data based on a point-source placed within the environment. The data was then used to form a regression as in Figure 2.8(b). These early experiments performed well, with the point source being localized with just a few samples. In the future, we would like to continue these experiments using the online IPP approach. It is expected that given the performance using a random approach, the IPP approach will fare much better.

2.7 Conclusion

In this chapter, we illustrated a methodology for localizing potential leaks at the Hanford Facility high-level waste tank farm. Whereas current efforts only utilize a visual inspection with a pole-mounted camera for inspecting the periphery of the refractory slots [Gir15], our approach allows for robotic inspection of the bottom of the tank. We provide simulation results and an analysis of the results indicating that the solution is feasible. While much of the existing literature for IPP assumes a relatively open environment, this work describes a path-planning approach that can work in constrained environments where a robot can only sample a small portion of the total area. Moreover, the path-planning algorithm shown here [ZBT17] allows for the robot to be constrained in its movements by its turning radius and tether, unlike previous works such as [GK11] where those kinematic constraints are not considered. The IPP approach also incorporates prior knowledge of the environment, rather than having to start from no knowledge as several of the related works in Section 2.2 assume, and incorporating a travel penalty if desired. Compared to the existing approaches used at the Hanford facility with a manual probe, this solution permits inspection of the entire length of the refractory slots, and reduces the

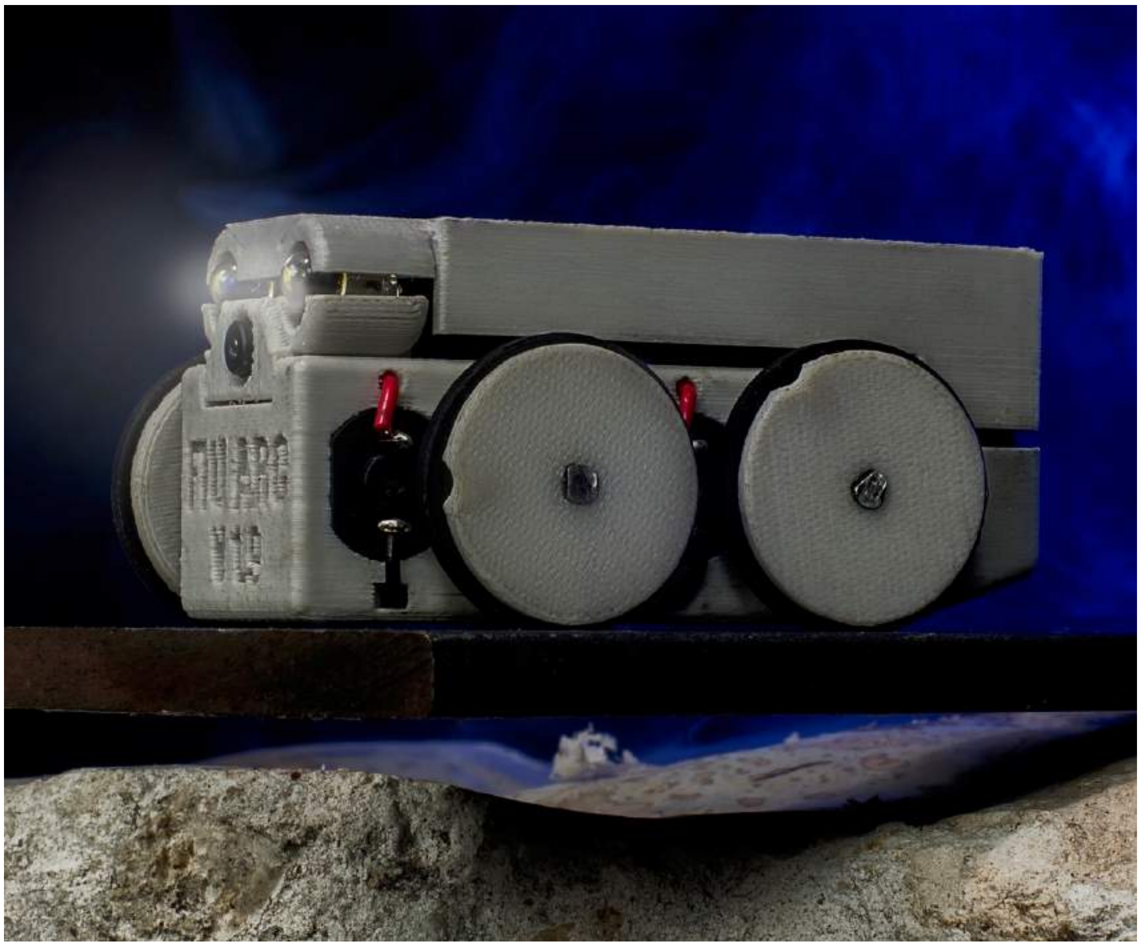


Figure 2.12: Prototype mini-rover for inspection of refractory slots.

time needed to localize a leak compared to the traditional rastering exhaustive approach such as in [Mic17]. The reduced time allows for better responses in both periodic inspections and emergency responses. This work can also be used when surveying other types of structures and buildings, allowing for effective remote characterization, assisting operators to make better decisions about what areas need decontamination.

We have also developed the robotic mini-rover with a video camera shown in Figure 2.12, and performed preliminary testing in mock-up refractory slots [DAMT17, DMA+18]. We are now working to integrate temperature and other sensors. Future directions for this work include deployment within the actual working tanks, and applying these techniques to a radiation signal rather than temperature.

CHAPTER 3

Task Allocation for Heterogeneous Robot Teams

3.1 Introduction

Allocating tasks to robots is a challenging problem. Robots may be spread out over an area, with tasks coming in from multiple locations. An example of this is sampling in hazardous environments. Robots must travel to the sample locations, and then either ferry the samples to a drop-off location or execute experiments using onboard equipment.

The components of the solution include:

1. Converting the environment into a Voronoi decomposition that yields a "roadmap" G with the best clearance from known obstacles as in Figure 3.1, followed by path-planning.
2. Collision checking, which permits robots to react to previously unknown obstacles and other robots X_{obs}, where the robot seeks a goal location shifted r steps after the obstacle.
3. Scheduling, consisting of an estimation of how including the task T^i would impact other tasks and a greedy heuristic which quickly prunes the large decision space into a manageable set by iteratively prioritizing and eliminating tasks according to a set of operator-defined criteria.

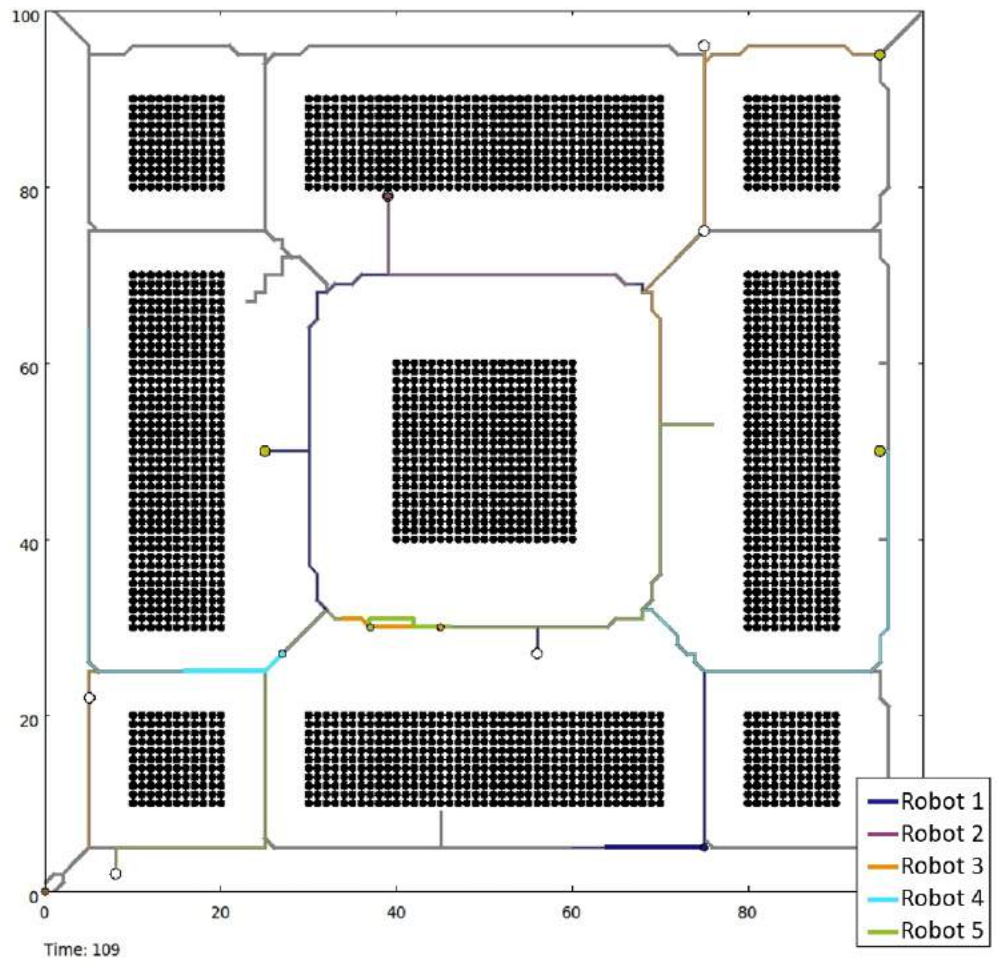


Figure 3.1: Example roadmap around buildings

3.2 Related Work

Traditional ferrying approaches aim to solve the problem of transporting messages or items from one location to another [ZAZ04]. The solution described here improves upon that of computational ferrying by [MAZ$^+$15], where traditional ferrying/muling [ST08, GK07] is augmented by allowing for computation. There are also connections with real-time scheduling where different jobs with deadlines must be scheduled as they come in [LL73, SAÅ+04, But11], where *wait time* must also take into account the physical distance between tasks, and multi-robot planning that assigns paths to robots [Par08, LaV06].

In Monfared et al. [MAZ+15], a set of robots are coordinated in order to satisfy the requests of multiple users scattered throughout an environment. This allows users to offload computation for mobile devices onto mobile cloudlets, providing greater processing and storage capabilities, as well as reduced power consumption. In order to allow for efficient usage of the ferries, the visit schedules must be carefully designed [ZAZ05]. The trajectories followed by robots in [MAZ+15] are direct lines between the pickup and delivery locations. In a realistic environment, there are obstacles and varying terrain in between locations. These obstacles affect vehicle trajectories and the travel time from one location to the next. All of these factors complicate scheduling estimates, and cannot be ignored if we are to have effective planning and scheduling strategies. With this in mind, we add obstacles and path planning to the original problem formulation. These additions enable for more robust performance in contested or dynamic environments. Operators gain the critical ability to view planned trajectories for vehicles, as well as more realistic travel time estimates [UGL+14]. Moreover, we also incorporate the ability to schedule given different priorities for tasks, and the requirement for specific hardware or software capabilities provided by different robots.

The rest of this chapter is organized as follows. In Section 3.3, we look at the system architecture and physical state space where path planning will take place. Section 3.4 will cover the methods used for allocating tasks to each robot and scheduling the robot visit orders. It will also examine the methods for generating and selecting trajectories between locations. Section 3.5 presents our results. Finally, Section 3.6 includes conclusions and suggestions for future work.

3.3 Preliminaries

3.3.1 Notation

We assume a 2-dimensional, partially known world $\mathscr{W} = \mathbb{R}^2$. Here, partially known means that we have both known $\mathscr{O}_{known}$ and unknown obstacles $\mathscr{O}_{unknown}$, which compose the set of polygonal obstacles $\mathscr{O} = \mathscr{O}_{known} \bigcup \mathscr{O}_{unknown}$, such that $\mathscr{O} \subset \mathscr{W}$.

Users in the workspace will generate or receive Tasks $\mathscr{T}$. There are n tasks, where each task is a tuple $\mathscr{T}^i$ with pickup and delivery locations $\mathscr{T}^i_{LP}, \mathscr{T}^i_{LD}$ (which can be the same location), job length $\mathscr{T}^i_L$, a start time $\mathscr{T}^i_S$, and a deadline by which the task must be delivered $\mathscr{T}^i_D \geq \mathscr{T}^i_S + \mathscr{T}^i_L$. Tasks may also have additional features which can be used to weigh them when designing a schedule. An example feature used here is priority $\mathscr{T}^i_P \in \mathbb{N}$, where completing a higher-priority task is preferred at the cost of missing the deadlines of lower-priority tasks. When scheduling, we augment the tasks with additional features: expected pickup and delivery times $\mathscr{T}^i_{TP}, T^i_{TD}$, and status indicators for whether the task has been picked up or delivered $\mathscr{T}^i_{PU}, \mathscr{T}^i_{DL}$. Multiple tasks can be announced in groups $\mathscr{T}^i_G$ while the system is running.

In the environment are also located robots. Robots are capable of carrying out the tasks offloaded to them. Execution of the tasks can take place regardless of whether the robot is adjacent to a pickup/delivery location or is moving. There are m robots $\mathscr{A}^j \in \mathscr{A}$, each a tuple

$$\mathscr{A}^j = (p, vo, \mathscr{A}, q).$$

Each robot has a number of processors $p \geq 1$ and visit order vo, representing the order in which tasks are scheduled to be picked up and delivered, such as $[1_p, 2_p, 2_d, 1_d]$, which would correspond to picking up task 1, followed by task 2, then delivering task 2 followed by task 1. $\mathscr{A}^j$ is the polygonal representation of the robot at location q. Robots are capable of long-range, but low-bandwidth communications to exchange simple location and planning data with the controller, as illustrated in the system architecture overview in Figure 3.2. They can also sense their environment in order to detect both known and unknown obstacles, and are capable of limited path-planning.

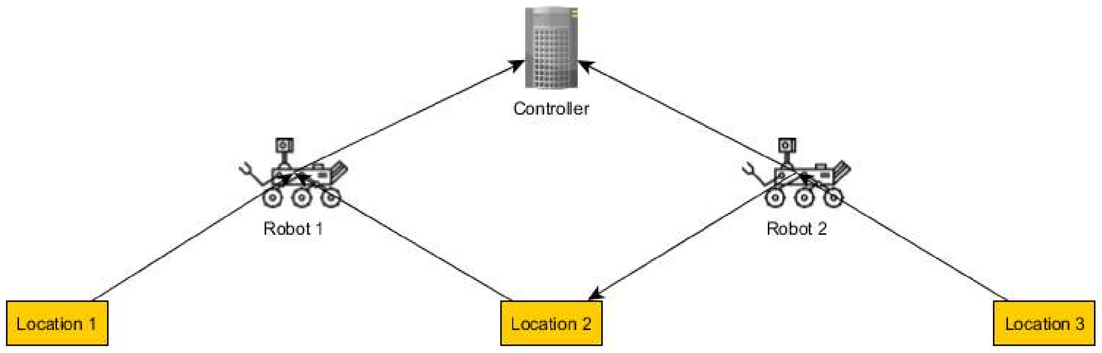


Figure 3.2: Example roadmap around buildings

$\mathscr{A}_q^j$ is the location occupied by robot $\mathscr{A}^j$ in configuration $q \in \mathscr{C}^j$. The obstacle state space X_{obs} consists of robot-robot collisions:

$$\mathscr{C}_{obs}^{jl} = \left\{ q \in \mathscr{C} | \mathscr{A}^j(q^j) \bigcap \mathscr{A}^l(q^l) \neq \emptyset \right\}, \tag{3.1}$$

and robot-obstacle collisions are:

$$q_{obs}^j = \left\{ q \in \mathscr{C} | \mathscr{A}^j(q^j) \bigcap O \neq \emptyset \right\}. \tag{3.2}$$

The union of these equations yields the complete obstacle region:

$$\mathscr{C}_{obs} = \left(\bigcup_{j=1}^{m} \mathscr{C}_{obs}^j \right) \bigcup \left(\bigcup_{jl, j \neq l} \mathscr{C}_{obs}^{jl} \right). \tag{3.3}$$

The obstacle-free region is defined as $\mathscr{C}_{free} = \mathscr{C} \setminus \mathscr{C}_{obs}$.

The controller is capable of receiving user requests and robot status updates (location and task progress). The controller must then determine a schedule and trajectory for each robot such that tasks are picked up, executed, and delivered, in a way that best satisfies the specified criteria (such as priorities, number of tasks completed, etc.). Given known obstacles and terrain, the trajectories will be precomputed to allow for more accurate schedule estimation. If a robot encounters an unknown obstacle, it may also report the location to the controller, which will incorporate it into its map.

3.3.2 Problem Definition

Problem 1 - Task Estimation: Given a set of tasks $\mathscr{T}$, Obstacles $\mathscr{O}$, and robots $\mathscr{A}$, determine when each task will be picked up and delivered.

Problem 2 - Task Allocation and Scheduling: Given a set of tasks $\mathscr{T}$, robots $\mathscr{A}$, and the ability to estimate when tasks will be picked up and delivered, allocate the tasks to robots *and* design a policy for each robot such that we attempt to meet as many task deadlines as possible.

Problem 3 - Path Planning and Unknown Obstacle Avoidance: Each robot begins in an initial state $q_I^j \in \mathscr{C}_{free}^j$, and ends in a goal $q_G^j \subset \mathscr{C}_{free}^j$. Given an unbounded time $\mathscr{T} = [0, \infty)$, we calculate a state trajectory h where the initial state is $h(0) = q_I$ and the final state is $h(t) \in q_G$ that takes the robot through $\mathscr{C}_{free}^j$ such that we avoid both known and unknown obstacles.

3.3.3 Complexity

In our first item of interest, we want to understand the computational complexity of the problem. In order to do that, we use the technique of *restriction* [GJ79]. We want to show that the *Computational Ferrying Problem* contains a known NP-hard problem as a special case (chosen to be the *Partition Problem*) [GJ79]. The *Partition Problem* decides whether a set of positive integers S can be split into two subsets $S^1 \subset S$ and $S^2 \subset S$, such that the sum of numbers in each set is equal $\sum_{s \in S^1} s = \sum_{s \in S^2} s$.

We restrict our problem of computational ferrying and set the pickup and delivery locations to be the same

$$T^i_{LP} = T^i_{LD} \forall T^i \in T,$$

where all tasks are made available at $t = 0$, and all deadlines are ignored $T^i_D = \infty$. Moreover, we assume that $\mathscr{A}$ consists of only two robots $\mathscr{A}^1, \mathscr{A}^2$, where their locations are also identical to the tasks $\mathscr{A}^1_x = \mathscr{A}^2_x = T^1_{LP}$. Each robot is also set to have a single processor $\mathscr{A}^1_p = \mathscr{A}^2_p = 1$.

Through this restriction, we remove all effects of traveling and arrive at the *Partition Problem*, where the positive integers are the task lengths T^i_L, which we desire to partition between both robots. Since the *Partition Problem* is NP-Hard, *Computational Ferrying* must also be NP-Hard.

To further motivate the issue of complexity, we point out that the problem of traveling between the various pickup and delivery locations can also be restricted to the Traveling Salesman Problem, though such a proof is trivial and not presented here for the sake of brevity.

3.4 Methods

Here, we describe the methods for solving the problem of allocating tasks to robots and designing trajectories. The accompanying flowchart in Figure 3.3 shows the relationships between the various components, and how we progress from start to finish.

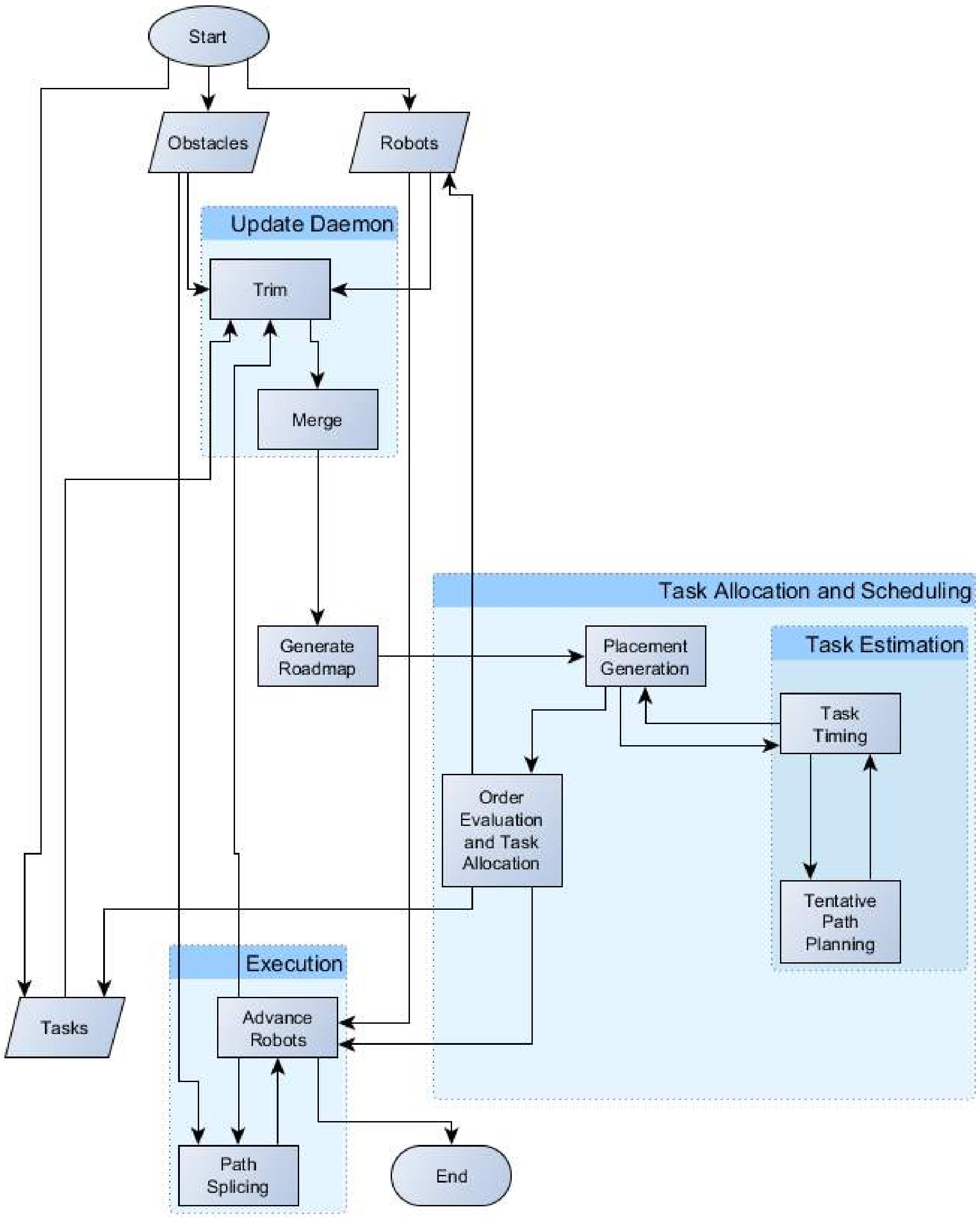


Figure 3.3: Workflow

3.4.1 Task Estimation

Initially, we are provided with a set of robots $\mathscr{A}$, Tasks $\mathscr{T}$, and Obstacles $\mathscr{O}$. While estimating when a task will be picked up or delivered, we are mainly concerned with the: task execution time, distance to travel, and task execution schedule. Execution time is provided by the user when a task is generated, and the schedule will be determined in part by this task length, as well as the distance a robot must travel. How then, can we estimate travel times?

Roadmap Construction

The controller accepts a set of known obstacles $\mathscr{O}_{known}$ and performs a Voronoi decomposition to obtain a graph G of edges. This graph will be used as a "roadmap" for path planning later on. A Voronoi decomposition was chosen as it 1) reduces the search space (as opposed to path-planning in the entire workspace), and 2) provides maximum clearance from known obstacles.

The controller begins by receiving a set of known obstacles $\mathscr{O}_{known}$ and creating a Voronoi decomposition. This graph, or *roadmap*, represents a set of safe paths which are equidistant from known obstacles. An example urban environment is shown in Figure 3.1. We augment the roadmap by adding task pickup and delivery locations, and the location of each robot, connecting them to the roadmap. Using a roadmap allows for faster path planning as we can now constrict our search to this graph, rather than an entire continuous environment. A roadmap could also allow operators to determine where to place assets that can ensure the safety of essential paths. If an area is deemed unsafe, operators can eliminate edges passing through that area.

With the roadmap now available to use for fast searching, we proceed to the next requirement: estimating trajectory times.

Tentative Paths

To generate a tentative path for a robot in Algorithm 4, we iterate over its visit order $\mathscr{A}_{vo}^{j}$. A graph search is performed in the roadmap between each pair of locations, concatenated to form a complete tentative path h through the known obstacle state space of q_{obs}^{j}. The length of the path (and subsequently, how long this takes to traverse) is also calculated as this is one of the factors used in the scheduling process.

Algorithm 4 Robot Tentative Path

Input: $\mathscr{A}^{j}$
Output: Tentative path representing robot trajectory

```
 1: $tentativePath \leftarrow []$
 2: $prevLoc \leftarrow \mathscr{A}_{x}^{j}$
 3: $nextLoc \leftarrow Null$
 4: $scheduled \leftarrow []$
 5: for $\mathscr{T}^{i} \in \mathscr{A}_{vo}^{j}$ do
 6:     if $\mathscr{T}_{PU}^{i} == true$ then
 7:         $nextLoc \leftarrow \mathscr{T}_{LD}^{i}$
 8:     else
 9:         $nextLoc \leftarrow \mathscr{T}_{LP}^{i}$
10:         $scheduled[i] \leftarrow true$
11:     $\tau \leftarrow GraphSearch(prevLoc, nextLoc)$
12:     $tentativePath.append(\tau)$
13:     $prevLoc \leftarrow nextLoc$
    return $tentativePath$
```

With the elements in place, we can now move forward with the components to begin allocating tasks to robots, and scheduling their execution.

Task Timing (Algorithm 5, Line 7)

In order to calculate the schedule, we will need to estimate the pickup, start, and delivery times of the tasks based on: robot location, task pickup/delivery locations, task job length, robot processor schedule, robot visit order, and estimated travel distance. For tasks of

Algorithm 5 Task Group Scheduling Algorithm

Input: $\mathscr{T}$, $\mathscr{A}$

Output: $\mathscr{A}'$

```
1: $\mathscr{T}' \leftarrow$ Sort $\mathscr{T}$ by ascending deadlines
2: for $\mathscr{T}^i \in T'$ do
3:     $orderings \leftarrow []$
4:     for $\mathscr{A}^j \in \mathscr{A}$ do
5:         $pList \leftarrow placementGeneration(\mathscr{A}^j, T^i)$
6:         for $p \in pList$ do
7:             $timedPermInfo \leftarrow taskTiming(\mathscr{A}^j, p)$
8:             $orderings.append(timedPermInfo)$
9:     Sort $orderings$ by user conditions
10:    Assign tasks to robots based on $orderings[0]$
11: return $\mathscr{A}'$
```

unknown length, we disregard the execution time, only focusing on the time needed to travel between locations, which is handled by the Task Timing algorithm.

$\mathscr{A}^j$'s tasks are distributed across $\mathscr{A}_p^j$, each one sequentially assigned to the processor with the shortest queue. Using the robot's visit order, a graph search on the roadmap finds the shortest path between following points, returning an estimated distance to be traversed. This is sufficient when picking up a task, as travel time is the only factor. When delivering tasks, we also incorporate the remaining task processing time: the max is taken between the estimated travel time and the remaining task computation time and used as the final delivery time. When completed, this returns the tuple:

$$pInfo = (FT, MD, TM),$$

where FT is the finish time of the last task in the visit order, MD is the number of deadlines expected to be missed, and TM are the tasks which will miss their deadlines.

3.4.2 Task Allocation and Scheduling

The goal of this section is to arrive at a set of task allocations to the available robots, and a schedule for each robot such that we can maximize the number of tasks that meet their deadlines, while simultaneously adhering to the constraints imposed by available resources, distances, and computing time.

In order to handle the task-length, we consider two cases:

- The task length is known or can be approximated given historical data
- The task length is unknown

Placement Generation (Algorithm 5, Line 5)

This algorithm is used by Task Scheduling to generate all valid placements of a visit order given an existing visit order $\mathscr{A}_{vo}^{j}$ and a new task $\mathscr{T}^{i}$. The task is inserted twice (representing pickup and delivery) in every valid location of the visit order. Each of these placements is stored in a list *pList*. In the case of a task with an unknown length, we ensure that the task is placed behind all tasks of higher priority, and ahead of tasks of lower priority. Within tasks of the same priority, tasks with an unknown computation length will be executed last, favoring the completion of a known set of tasks over the completion of a single unknown task that might overwhelm the others. Once all placements have been generated in time complexity $O(n^2)$, *pList* is returned.

As an example: we are given an existing visit order of $[T_{LP}^{1}, T_{LP}^{2}, T_{LD}^{2}, T_{LD}^{1}]$, indicating a pickup of task 1, followed by a pickup of task 2, and then the delivery of tasks 2 and 1. We now wish to generate valid orderings of pickup and delivery locations for new task 3, without perturbing the existing task orderings.

T^3_{LP}	T^3_{LD}	T^1_{LP}	T^2_{LP}	T^2_{LD}	T^1_{LD}
T^3_{LP}	T^1_{LP}	T^3_{LD}	T^2_{LP}	T^2_{LD}	T^1_{LD}
T^3_{LP}	T^1_{LP}	T^2_{LP}	T^3_{LD}	T^2_{LD}	T^1_{LD}
...	...	...	...	...	...
T^1_{LP}	T^2_{LP}	T^2_{LD}	T^3_{LP}	T^1_{LD}	T^3_{LD}
T^1_{LP}	T^2_{LP}	T^2_{LD}	T^1_{LD}	T^3_{LP}	T^3_{LD}

Table 3.1: Example Placements showing the visit order for when Tasks are picked up and then delivered.

Task Group Scheduling (Algorithm 5)

The controller first sorts the tasks in $\mathscr{T}'$ by their deadlines (line 1). Next, iterate over each task in the sorted $\mathscr{T}'$, and every robot in $\mathscr{A}'$ that is capable of executing the task (that has the specified, specialized hardware capabilities to handle the task), generating placements (line 5) of the visit order with the new task in every valid location of $\mathscr{A}^j$'s visit order. In line 9, the Task Timing Algorithm estimates each task's pickup, delivery, and start times.

This process is repeated for all $\mathscr{A}^j \in \mathscr{A}'$ giving a time complexity of $O(mn^3)$. Once all orderings and times have been calculated, the solution is then replaced by the placement order which fulfills the highest number of requirements in descending order(priorities, deadlines, etc.). The new greedy solution is then used in the next iteration over $\mathscr{T}'$.

During execution in a dynamic environment, many things can happen. Unknown obstacles can appear, or robots can be taken offline. In these cases, we must replan.

3.4.3 Path Planning and Obstacle Avoidance

Using the Task Allocation algorithm 5, we generate the different task placements and estimate the timings in order to evaluate the solutions. Finally, we select the best trajectories for the robots that allow them to visit the tasks.

3.4.4 Update

Here, we describe the process of checking for new incoming tasks and robots, removing completed tasks and robots and updating the status of tasks.

Merge

Upon receiving a group of robots, $\mathscr{A}$, and a list of tasks $\mathscr{T}'$, the merge algorithm combines existing tasks that have not been picked up by a robot with the incoming list of tasks. The tasks that have already been picked up by a robot are not altered.

Trimming

Given a generated base solution, the trimming algorithm tracks each robot's processor schedule, and each task's processing time, and determines if the task's deadline is missed or met. If the robot, $\mathscr{A}^j$, is at the current task's pickup location, picked up is set to true $\mathscr{T}^i_{PU} = true$. If a task has completed processing and $\mathscr{A}^j$ is at its delivery location, then $\mathscr{T}^i_{DL} = true$.

Third, in conjunction with Task Scheduling, we also check for new tasks and for robots being added or removed to ensure that all information is up-to-date. When a new group of tasks is available, we handle new or removed tasks and robots. The result is then sent to Task Group Scheduling which outputs the new $\mathscr{A}$ representing robots and their assigned tasks and paths.

Daemon

If a robot has completed a task with a previously-unknown execution length, we execute Update. If the remaining schedule for the robot's task queue has been shifted back by the task's execution length, we treat the remaining tasks as a set of new incoming tasks, allowing them to be “re-shuffled” among the robot’s as needed to maximize the completion rate.

3.4.5 Path Planning

In order for the robots to carry out their tasks, they must be able to navigate their environment to visit pickup and delivery locations. Navigation is broken into two components. In the first, the controller is responsible for designing a tentative path that can avoid known obstacles. This can also allow operators to designate known “safe lanes” which are expected to be obstacle-free. Complementing this is a dynamic path planning component which resides on each robot. By combining this with a sensor payload, robots can navigate around dynamic or unknown obstacles.

Adding Tasks and robots

As new tasks are made available, the controller adds the pickup and delivery locations of each task to the roadmap. Since the roadmap travels around every known obstacle, a line connecting the roadmap to the task will avoid known obstacles as well. The same process is repeated to add robots.

Hybrid Paths and Path Splicing

The paths designed above have the benefit of being relatively simple to design and search. However, they fail to take into account the changing environment that we are faced with. Moreover, it is often not possible for the controller to deal with the issue of dynamic path planning as it can suffer from intermittent connections and the large workload of designing and maintaining multiple dynamic trajectories. This can result in a single point of failure.

To cope with this problem, we must allow robots to respond to new obstacles. When a previously unknown obstacle $o \in \mathscr{O}_{unknown}$ is found, the robot re-plans its trajectory around it using an A* search [ZC09, Fan07] through q^j_{obs}. The robot selects a tentative goal q_G, which is located r steps after the start of the obstacle. r depends on the environment, and should be larger than most obstacles (in order to navigate around them), but small enough that it does not delay the responsiveness of the search. Next, a new trajectory is calculated through $\mathscr{C}^j_{free}$. If it is not possible for the robot to reach q_G because of additional obstacles in $\mathscr{O}_{unknown}$, then we re-plan the trajectory a further r steps ahead. This method is also applied to robot-robot collisions q^{ij}_{obs}, creating a path h through q^j_{obs}.

To summarize and illustrate the workflow, Figure 3.3 provides a flowchart showing how the various components interact.

3.5 Experimental Results

In this section, we discuss the changes and additions made to the existing work, as well as our preliminary results.

3.5.1 Software Simulation

We utilized a custom simulator to have better control over the robot's functioning, permitting fine-grained control over:

- Customized path-planning
- Unknown obstacle avoidance
- Processor/task scheduling and prioritization

The simulations were carried out using between 1 and 8 robots, 1 to 4 processors, and either 0 known and 0 unknown obstacles, or 8 known and 0 unknown obstacles, or

0 known and 8 unknown obstacles. There were 40 tasks present in the workspace, 10-each of 4 different priorities. An example of a generic environment is shown in Figure 3.5. Building off of the previous example of the WIPP facility, we also present a sample scenario in Figure 3.4

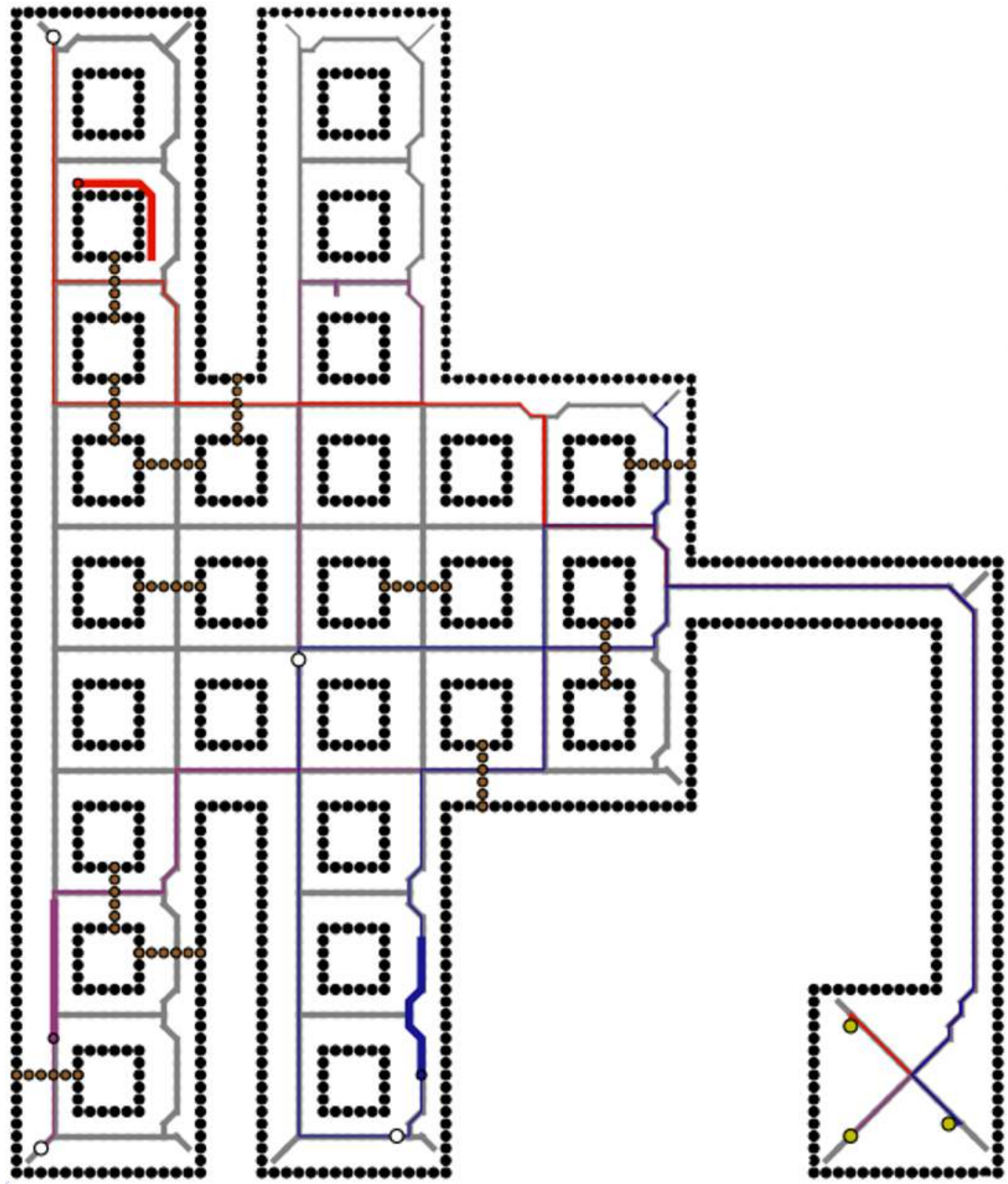


Figure 3.4: WIPP Facility simulation example

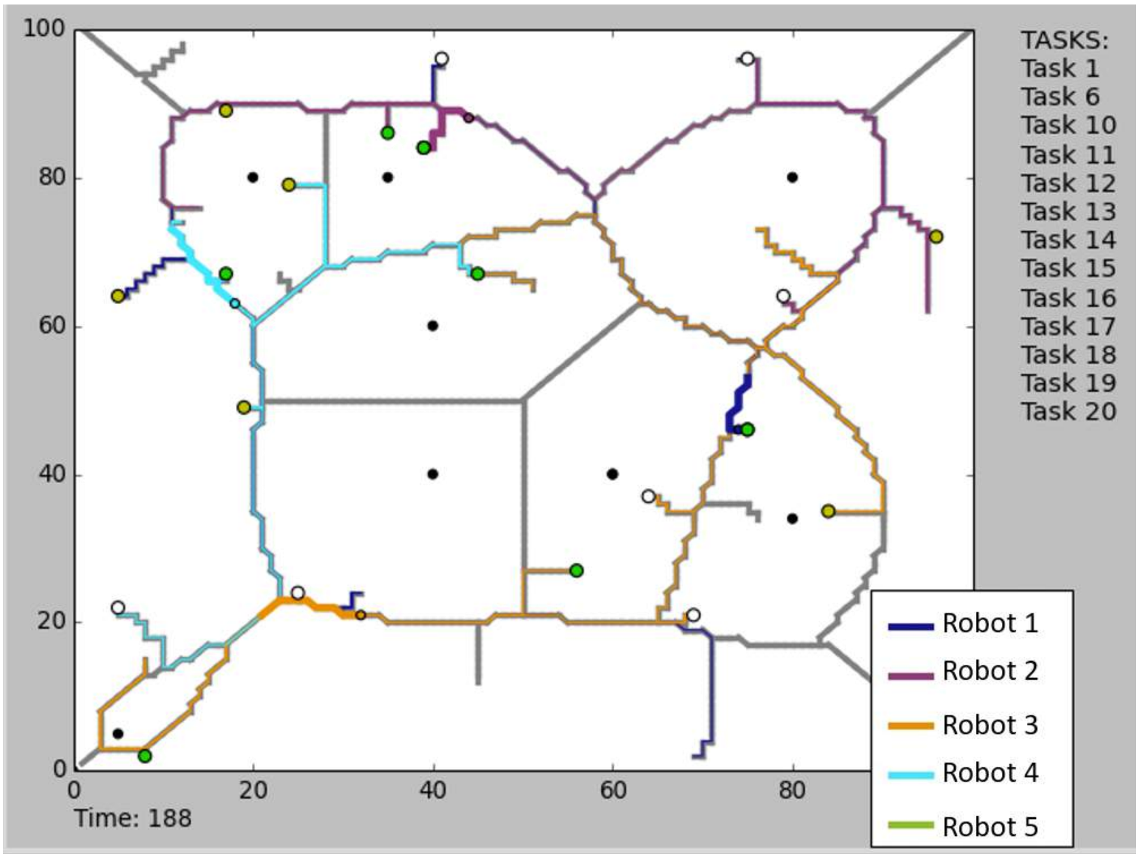


Figure 3.5: Simulation Example

3.5.2 Effects of Obstacles on robot Movement and Performance

As shown in Figure 3.6, the presence of unknown obstacles in the environment causes robots to travel further than with known obstacles, as path planning cannot be optimized. An interesting phenomenon is that the first few known obstacles decrease the steps taken. This is due to the resulting more complex roadmap providing more alternative routes. Future work will explore how the distribution and size of these known obstacles affect the efficacy of the roadmap.

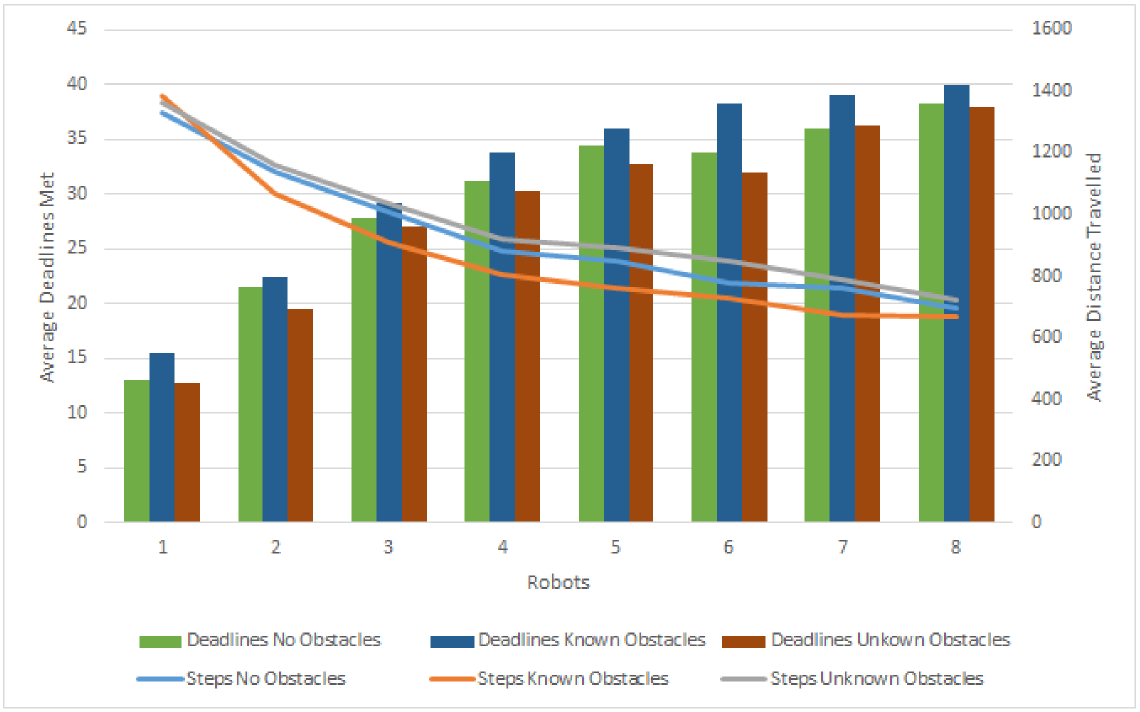


Figure 3.6: Visibility of Obstacles vs Average Distance Traveled and Deadlines Met

Turning our attention towards deadlines met, we see a mirroring of the effects on robot travel distances. With the roadmap generated for an environment containing no known or unknown obstacles, we see an average decline of 2.28 deadlines met compared to an environment with known obstacles. Likewise, when comparing the environment with 8 known obstacles vs 8 unknown obstacles, there is another decline with an average of 3.22 deadlines less being met.

For cases with more than one robot, we find that having an environment with known obstacles consistently performs better than all cases, again because of the more complex roadmap. The presence of unknown obstacles results in much worse performance, as expected.

3.5.3 Effects of Number of robots and Processors on Deadlines

Figure 3.7 shows that increasing the number of processors across the robots has a negligible effect. However, this is likely a result of our simulation's parameters, where tasks generally had a short run time. This results in the bottleneck being that of physical distance between locations, rather than the duration of tasks or the number of tasks being processed. It is expected that in the case of closely clustered tasks with longer run times, we could achieve better results with fewer robots so long as they were equipped with more processors. This could be utilized in order to allow mission planners to better allocate resources with regards to processing capabilities vs number of overall units. We plan to further analyze the effects of distance vs. runtime in future works.

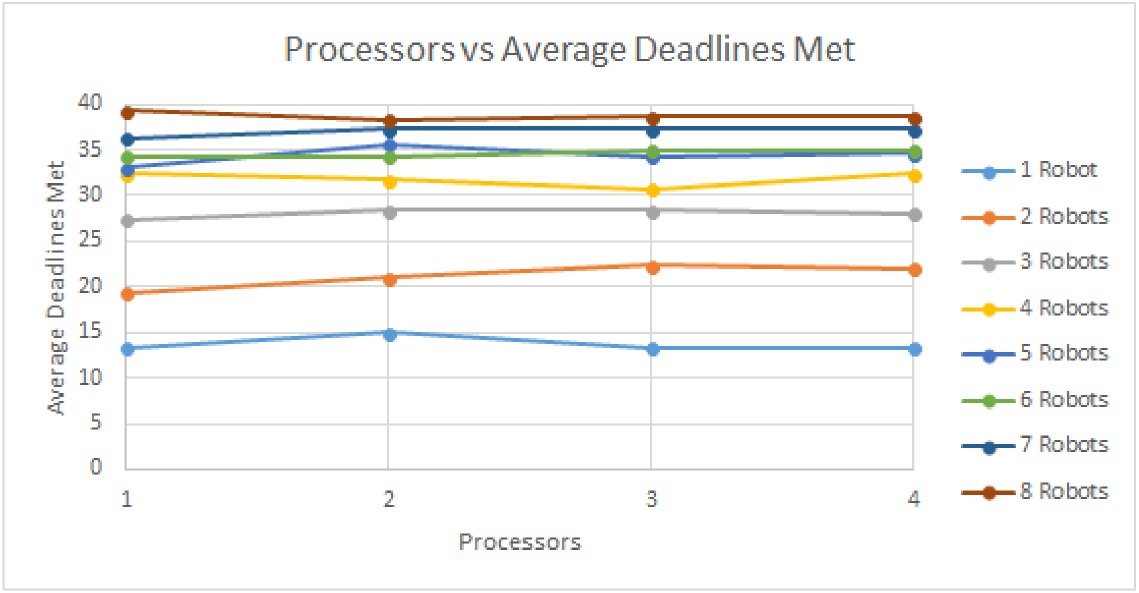


Figure 3.7: Average Deadlines Met vs Number of Processors

3.5.4 Effects of Removing Tasks on on Deadlines and Steps

We have done experiments to evaluate the effect of removing or not tasks. As illustrated in Figure 3.8 the more available robots, the higher the average deadlines met. Interestingly, there seems to be no effect of removing tasks, and the number of deadlines met. In contrast, as shown in Figure 3.9, removing tasks has an effect on the average steps taken. However, as the number of robots increase, this effect is reduced. Further investigations may be required to understand these effects fully.

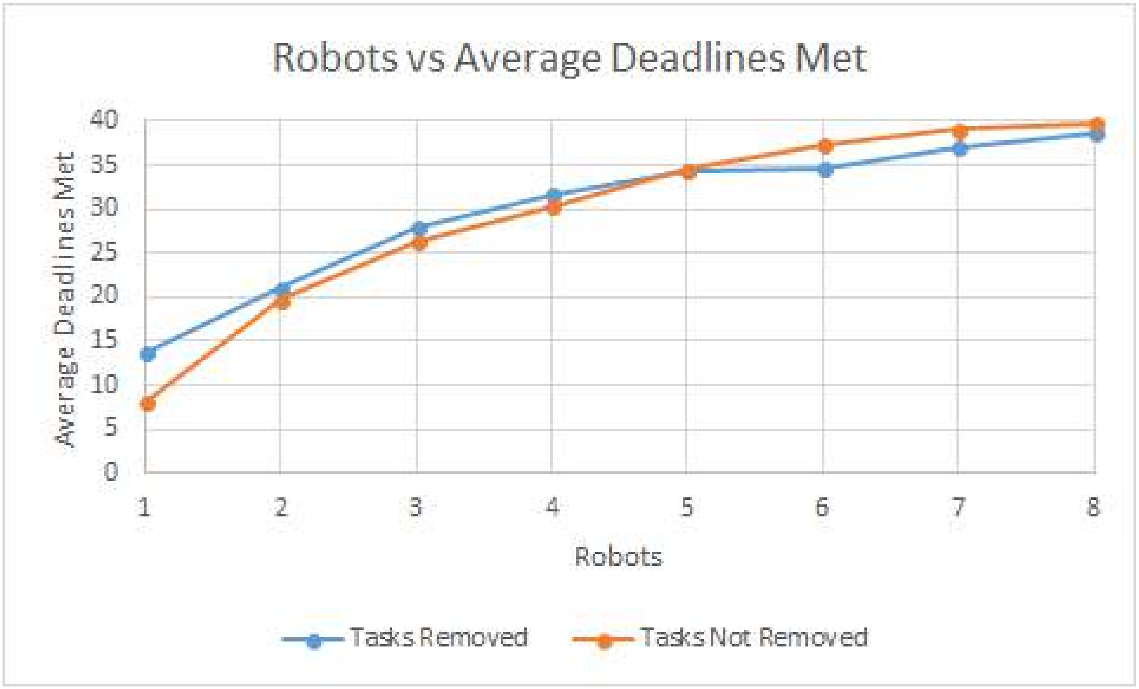


Figure 3.8: Effects of Removing Tasks on Deadlines

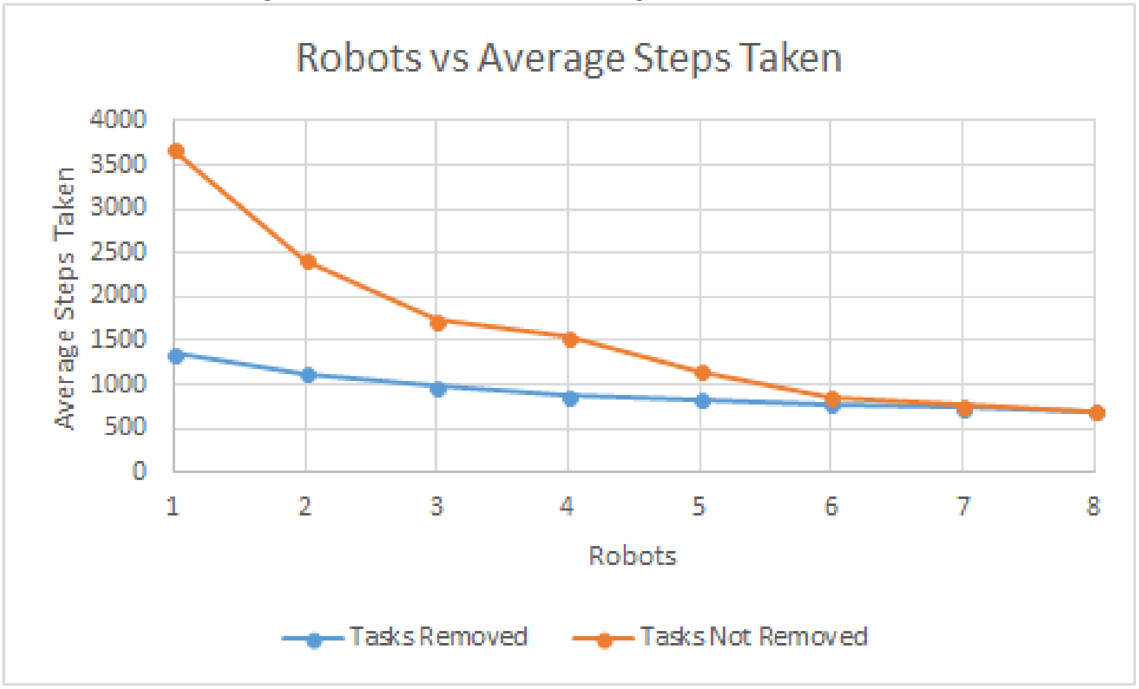


Figure 3.9: Effects of Removing Tasks on Steps

3.6 Conclusions and Future Work

In this chapter, we have extended and formulated a problem to schedule, plan, and deploy robots that can physically move to visit different locations in order to perform tasks and ferry samples or data. We have improved in several different aspects of the state-of-the-art [ZWBS16]. In prior related work [MAZ$^+$15], robots are only able to move in a straight line between task locations and did not explicitly incorporated obstacles. Given the complex and dynamic nature of the environment where this problem takes place, we cannot make this assumption and robots must be able to navigate around obstacles and other vehicles or robots. We model robots as autonomous vehicles operating within an environment populated with polygonal robots and obstacles. We have implemented path planning algorithms which can find reliable *a priori* estimation of distances between locations to obtain more accurate scheduling times. This is further augmented by the creation of safe "roadmaps" formed by the decomposition of the environment, which allows for safer operation and easier management by operators. The roadmap approach [BG08] and path planning algorithms proposed allow more flexibility in carrying out tasks.

We introduced several prioritization schemes that allow operators to assign prioritization weights to tasks, a feature which has been outlined as an avenue for development in [MAZ$^+$15]. This allows for greater flexibility when defining tasks. For instance, certain tasks may be given priority over others, such that high-priority tasks are executed at the cost of lower-priority tasks. In this chapter, our algorithms are implemented and tested in a computer simulation to understand the effect of completion time due to obstacles, the number of robots, and number of processors. Moreover, we allow for tasks and robots are heterogeneous, rather than the homogeneous assumption made in many of the works from Section 3.2. As an example, consider a robot that might have hardware capabilities not available in other robots or may have relevant data stored that is not replicated on other robots. This requires the scheduling algorithm to allocate the robot to whichever tasks need those specific capabilities. The suggested future works outlined in [MAZ+15] calls for the ability to remove and add robots at runtime. This capability has been added to our work. Any time a new robot is added or removed, its location is updated on the roadmap, and a rescheduling takes place. This allows for a more resilient system that can automatically tolerate robot failures.

CHAPTER 4

Robot Policy Generation Given Operator Constraints

4.1 Introduction

Thus far, we have covered how to use robots to map radiation, and how to allocate tasks to robots. However, what happens if a robot is incapable of executing a task on its own? There remain many problems that are not currently feasible with a completely autonomous robot. Tasks such as manipulation of irregular objects, or robots operating in hazardous environments are still difficult to automate and require some human-in-the-loop in order to oversee their operation. Here, we must design policies for not only the robots but for their operators as well.

Multi-robot systems are making a significant impact in key societal areas. From oceanic exploration to border surveillance, from robotic warehousing to precision agriculture, and from automated construction to environmental monitoring, collaborating groups of robots will play a central role in the coming years. In some of these scenarios, however, due to technical, ethical, or regulatory issues, one or more humans should monitor or help the robot during the execution of its tasks in certain critical parts. These critical segments of the robot trajectory can be for example kinematically or dynamically complex maneuvers, locations nearby obstacles, or regions where sensing is poor.

Most tele-operated systems assume at least one human operator per robot. In more complex scenarios, such as humanoids or mobile manipulator tele-operation, more than a human may be required for each subsystem (e.g. manipulation, locomotion, head positioning). Another dramatic example is control rooms in Unmanned Aerial Vehicles missions where several operators are needed for the operation of a single drone. Although we will not be able to completely remove this portion of a task that can not be automated, we can efficiently allocate human attention in this portions. As an application of our ideas, we envision scenarios where a single operator can coordinate the tasks of a group of automated construction machinery or several agricultural pieces of equipment. Effectively combining human and robot capabilities [SWWB11] has provided significant benefits in industrial applications [WNS12, HSH02] and more general methodologies are needed.

One of the motivating applications which will also serve as a study case to test our ideas is *robot-assisted search and rescue*. In traditional mobile robot search and rescue operations using unmanned vehicles, the ratio of operators to robots is commonly 2 to 1 [Mur04]. More recently, motivated by disasters such as the Fukushima nuclear plant there has been a need for robots with larger degrees of freedom that can operate in environments designed for humans. Concretely, lessons learned analyzing human-robot interfaces used by different teams in the Defense Advanced Research Projects Agency (DARPA) Robotics Challenge (DRC) [YNO$^+$15], gave two crucial reasons motivating our ideas to reduce the number of operators: 1) fewer operators reduces confusion and coordination overhead, and 2) the amount of human errors (one of the primary sources of problems in the DRC [ABB$^+$15]) is reduced.

In most situations, an operator is only required in specific parts of a robot's operation. Knowing this, we can schedule these operator interactions so that a single operator can perform multiple tasks. The contributions of this work are: extending our preliminary ideas from [ZRAB17, ZAL$^+$18] in the following directions: *First*, we analyze the complexity of this problem. *Second*, we present a sampling-based approach that allows us to design policies for a large number of tele-operators instead of a complete algorithm that only works for a small set of operators. *Third*, we allow re-planning of the robot's task alongside the operator. *Finally*, we present results of both simulated and physical experiments using mobile robots and a humanoid.

Our work deals with the problem of planning for robots using a small set of operators that can help the robot when needed. As a convention, we will use the term "robot" throughout this chapter; however, our work is done in the robots' configuration space and is agnostic to the type of robot. It can not only model multiple robots but also a single robot with multiple degrees of freedom such as a humanoid robot. To the best of our knowledge, our contribution is one of the few that attempts to formalize the problem of operator scheduling using a geometric approach.

The rest of the chapter is organized as follows: Section 4.2 discusses relevant related literature. Section 4.3 describes the preliminaries and formulates the problems of interest. Section 4.4 describes algorithms to solve the formulated problems in the previous section. In Section 4.5, we present an extension of the solution in Section 4.4 which can also re-plan robot trajectories. Section 4.6 presents both software and hardware experimental results, and a case study is provided in Section 4.7. Conclusions and future directions are presented in Section 4.8.

4.2 Related Work

Our work tries to address the scarcity of techniques for planning multi-robot missions that can assist in outlining mission requirements and robot policies. There are relevant approaches such as Crandall et al. [CCDPdJ11] which investigates the effects of allocating operator attention to robots, and [CM07, MS12, RFI$^+$15] which investigate additional methods of distributing operators across robots and the effects this has. We also find work on tele-operation using 1 : 1 operator: robot paradigms [YH12, GS07] for more critical operations. These works reflect the growing need for systems that facilitate operator oversight of multi-robot systems [SWWB11], and the insight that utilizing operators alongside partially autonomous robots yields systems that are less brittle and more effective than either one working alone [Hug08].

Particularly relevant to our research ideas is the work by Trautman [Tra12, TMMK15] where the expected behaviors of humans in an environment are incorporated into the planning phase of robots, allowing them to perform more elaborate plans than without this prediction. This argument also extends into more industrial settings, where it is often repeated, scheduled interaction between robots and operators [WNS12]. Our work also relates to motion planning approaches that generate joint plans for humans and robots[RBM$^+$18, RCB$^+$15, RCBM14].

This chapter builds upon [ZRAB17, ZAL$^+$18], to perform multi-robot planning [Par08, PJGH15]. We also find complementary goals in Hauser [Hau13] where a robot attempts to move from one location in its environment to another by calculating which obstacles can be minimally displaced to generate a possible trajectory. In our work, we will similarly generate a coordination space, where operator "collision obstacles" must be avoided, and seek to find the minimal displacement needed to avoid them. In work by LaValle and Hutchinson [LH96, LH98], as well as by Wang et al. [WZG$^+$14], the complexity of coordinating both many robots and operators is handled by separating the planning and scheduling aspects into two separate steps. This division dramatically assists in devising a feasible solution and is echoed here as well.

4.3 Preliminaries

We start with a set of m of bodies, which can be kinematic chains or mobile robots, $\mathscr{A} = \{\mathscr{A}^1, \cdots, \mathscr{A}^m\}$. Each robot $\mathscr{A}^i \in \mathscr{A}$ has a configuration space $\mathscr{C}^i$ representing the set of all possible transformations, where the set of valid configurations is called the free space $\mathscr{C}^i_{free}$. Robots also have initial $q^i_I \in \mathscr{C}^i_{free}$ and goal $q^i_G \in \mathscr{C}^i_{free}$ configurations, where the trajectory $\lambda^i : [0, t^i_f] \to \mathscr{C}^i_{free}$ takes the robot from $\lambda^i(0)$ - corresponding to q^i_I - through

$\mathscr{C}^i_{free}$ to the final configuration $\lambda^i(t^i_f)$ - corresponding to q^i_G, where t^i_f is the total runtime for $\mathscr{A}^i$ to execute λ^i given a dedicated operator.

When executing λ^i, $\mathscr{A}^i$ may enter critical configurations $\mathscr{C}^i_{att} \subset \mathscr{C}^i_{free}$ during which it will require one of the p operator's supervision. A conflict occurs when more than p robots require supervision at the same time. Given a range of time $T = [0, t_f]$ where the mission is executing, we will attempt to minimize $t_f = max(t^1_f, \ldots, t^m_f)$ when all robots have finished, while also providing operator attention when required.

Problem 1: Scheduling for Multiple Operators: *Given the number of operators p, a set of robots $\mathscr{A}$ - each with their trajectories λ^i, and a set of critical configurations $\mathscr{C}^i_{att}$ - determine a policy $\pi^i : T \rightarrow \mathscr{C}^i_{free}$ for each robot such that 1) all robots are only in critical configurations when an operator can supervise them, 2) the number of operators requested at any time does not exceed p, and 3) attempt to reduce the total runtime of the mission t_f.*

Building on this problem, we can add the following condition: Is it possible to yield a shorter mission runtime by generating alternative trajectories for bodies such that they do not require supervision at the same time as other robots in the first place, thus avoiding operator attention "collisions" altogether? This question leads us to a concrete extension of Problem 1:

Problem 2: Scheduling with Re-Planning: Instead of a pre-determined trajectory, we use a sequence of *waypoints* $\tau^i = [\tau^i_1, \ldots, \tau^i_o]$ - where each waypoint is a specific configuration the robot must achieve, and the application-specific function $plan(A^i, \tau^i, t_{den})$ yields a trajectory that visits τ^i while avoiding $\mathscr{C}^i_{att}$ during operator-denied times t_{den} - an example of which can be found in Section 4.6.

Given p operators, a set of robots $\mathscr{A}$ each with a sequence of sub-goals τ^i, and a set of critical configurations $\mathscr{C}^i_{att}$. Determine a trajectory λ^i and policy $\pi^i : T \rightarrow \mathscr{C}^i_{free}$ for each robot satisfying the waypoints such that 1) robots are in critical configurations only when *an operator can supervise them, 2) the number of operators requested at any time is less than or equal to p, and 3) an effort is made to minimize the ending time of the mission t_f.*

4.4 Scheduling Operator Attention

In this section, we will propose solutions to the problems defined in Section 4.3.

4.4.1 Scheduling for Multiple Operators

In [ZAL$^+$18], we describe the operator scheduling problem and present a novel geometric approach for the solution. There were several issues with the provided approach, mainly the computational complexity of creating the entire set of obstacles with the coordination space. To motivate this, we provide a short sketch proving the complexity of this problem.

We prove this by reduction [GJ79]. We start with the *Multiprocessor Scheduling* problem, which consists of a set of J jobs, each job j^i has a corresponding length l^i. Provided p processors, we must schedule this set of jobs so that they 1) do not overlap, and 2) execute in the minimum amount of time. Given our operator scheduling problem, assume that all possible configurations for the robot will require operator attention, meaning that the entire execution of λ^i will need an operator. The runtime of this plan is t_f^i, and is analogous to the length of a job in the original Multiprocessor Scheduling problem. These jobs are scheduled and allocated to p operators, which would be the processors in the original formulation. This problem then reduces to the Multiprocessor Scheduling problem where we schedule j jobs across p processors and indicates that the problem we are trying to solve is at minimum NP-hard.

A sampling-based solution

Knowing that the problem is NP-hard we ask: how can we effectively arrive at a solution using a heuristic approach?

We start by creating a Coordination Space $x = [0, \tilde{t_f^1}] \times \cdots \times [0, \tilde{t_f^m}]$ (following a procedure similar to [LaV06]) representing all possible configurations of the robots along their trajectories. Each of the m axes corresponds to the normalized execution time $\tilde{t_f^i}$ of robot $\mathscr{A}^i$, given by $\tilde{t_f^i} = \frac{t_f^i}{max(t_f^1, \ldots, t_f^m)}$, with the position along the axis corresponding to progress along the trajectory. Let x_{obs} be the set of invalid configurations where the number of robots requesting supervision exceeds p, and $x_{free} = x \backslash x_{obs}$ be the set of all valid configurations where the number of requests does not exceed p. At $x_{init} = (0, ..., 0) \in x_{free}$ all robots are in their initial configurations, and at $x_{goal} = (\tilde{t_f^1}, ..., \tilde{t_f^m}) \in x_{free}$ all robots are in their final configuration.

We define auxiliary functions, borrowing the notation from [KF11]: $d(x_1, x_2)$ is the Euclidean distance between two points, and $c(\cdot)$ is the cost of a path corresponding to the sum of the pairwise Euclidean lengths of the pairwise linear points within it.

The above formulation serves to create a coordination space where the position along axes represents robot configurations and invalid configurations where multiple robots request obstacles represent an operator. This process allows us to convert the coordination problem into a path-planning problem. We must find a path $h : [0,1] \rightarrow x_{free}$ from $h(0) = x_{init}$ to $h(1) = x_{goal}$. Following h will then give us an implicit representation of time where the positions of each robot along their trajectory, such that each robot will move from its initial state to goal state, with at most p robots requiring operator attention. We performed this calculation by mapping h to the trajectory λ^i corresponding to a particular robot. Define $\sigma : h \rightarrow [0, t_f^i]$, which indicates the position of the robot along its trajectory λ^i at the corresponding point of path h through x_{free}. We then perform the composition $\phi : \lambda \circ \sigma$, which yields $\phi : h \rightarrow \mathscr{C}_{free}$, mapping from the path h to $\mathscr{C}_{free}$. This allows us to determine the configuration of a robot at any point q in h via $\phi(q) = \lambda(\sigma(q))$. We can now obtain the series of configurations $\tilde{x}$ for each robot that will guarantee that at most p robots require operator attention at any given time, and reduces the total run-time of the mission.

Our original solution required generating the entire set of obstacles within the coordination space. Here, we instead use a lazy approach which only checks sampled locations. This is combined with a modified version of the Bidirectional RRT^* originally described in [KF10, KF11, JP], and shown in Algorithm 7 for reference. Define graphs $\mathscr{G}_a = (V_a = \{x^a_{init}\}, E = \emptyset) \in x_{free}$, $\mathscr{G}_b = (V_b = \{x^b_{init}\}, E = \emptyset) \in x_{free}$, where $x^a_{init} = x_{init}$ and $x^b_{init} = x_{goal}$. The objective will be to derive an obstacle-free path $h : [0,1] \rightarrow x_{free}$ such that $h(0) = x_{init}, h(1) = x_{goal}$. Given a user-defined function that can estimate when robots will enter a critical section $\mathscr{S} \leftarrow CriticalSegments(A)$ we can check if a point $x \in x$ is obstacle-free as in Algorithm 6, where for the point being evaluated, we iterate over each robot's critical segments (lines 3, 4) and check if the corresponding axis of x lies within the segment (line 5). If the number of collisions is greater than the number of operators (line 7), then the location is not obstacle-free. With some abuse of notation, we also use this to refer to checking if an edge is obstacle-free by sampling along the edge and checking if the samples are all within x_{free}.

The modified $BidirectionalRRT^*$ is presented in Algorithm 7 In lines 1, 2, we initialize the final path as currently being none, and the corresponding cost to be infinite. Subsequently, we perform the following procedure over N samples: Beginning with G_a - the graph starting at the origin - in lines 4, 5 we draw a randomly-selected point from x_{free}. Checking if the point lies within x_{free} is done using Algorithm 6, and select the nearest point in the graph (we use an r-tree to accomplish this efficiently). In line 6, create a point x_{new} that is closer to x_{rand} than $x_{nearest}$. Then in lines 7-9, select the r points in

Algorithm 6 CollisionCheck

Input: Point x; Number of operators p; robots $\mathscr{A}$
Output: True if obstacle-free, False otherwise

```
1: $n_{colls} \leftarrow 0$
2: for $i \in [1,m]$ do
3:     $q \leftarrow \lambda^i(x_i)$
4:     if $q \in \mathscr{C}^i_{att}$ then
5:         $n_{colls} \leftarrow n_{colls} + 1$
6:         if $n_{colls} \geq p$ then
7:             return False
8: return True
```

$\mathscr{G}_a$ that are nearest to x_{new} and sort them in order of increasing distance from x_{new}, where the sorted list L_s consists of tuples of the form (x', c', σ'), where $x' \in X_{near}$, σ' is an edge from x' to x_{new}, and c' is the cost of that path, and select the closest one with an obstacle-free path to x_{new} as in [QA15]. If there is a valid “best parent” - defined as the vertex with the lowest combined cost-to-come and cost-to-go - we insert it into the graph and rewire as in [QA15] (lines 10-13). We then attempt to connect both trees. In lines 14-17, we select the nearest vertex in the opposite graph $\mathscr{G}_b$ and attempt to draw a straight path from the newly-added vertex $x_{new} \in \mathscr{G}_a$ to $\mathscr{G}_b$, if possible. We then check if the resulting path is better than our current best-path σ_{best} and update σ_{best} if necessary.

At this point in the algorithm, we may have a valid path σ_{best} through x_{free}. We then perform *RandomContraction* as in [QA15] to attempt reducing the length of σ_{best}. The user may assign a probability p_{early}, corresponding to the likelihood of checking for an early-exit solution; this is to balance between the run-time of *B-RRT** and to yield a better path. We evaluate this in lines 20-23, returning a valid solution if one exists. Otherwise, we swap $\mathscr{G}_a$ and $\mathscr{G}_b$ and continue until all N samples have been drawn and return σ_{best}.

We then proceed by mapping h to the sequence of configurations $\tilde{x}^i$ that correspond to robot $\mathscr{A}^i$. Movement parallel to an axis corresponds to that robot moving at full speed,

Algorithm 7 *B-RRT**

Input: Coordination Space x, Operators p; Critical Segments $\mathscr{S}$; Samples N, Probability of early exit $p_{early} \in [0,1]$

Output: Obstacle-free path σ_{best} through x

```
$\sigma_{best} \leftarrow \emptyset$
$c_{best} \leftarrow \infty$
**for** $i \in [0,N]$ **do**
    $x_{rand} \leftarrow SampleFree$
    $x_{nearest} \leftarrow Nearest(x_{rand}, \mathscr{G}_a)$
    $x_{new} \leftarrow Extend(x_{nearest}, x_{rand})$
    $X_{near} \leftarrow Near(x_{new}, \mathscr{G}_a, r)$
    $L_s \leftarrow Sort(x_{new}, X_{near})$
    $x_{min} \leftarrow BestParent(L_s)$
    **if** $x_{min} \neq \emptyset$ **then**
        $\mathscr{G}_a \leftarrow Insert(x_{new}, x_{min}, \mathscr{G}_a)$
        $\mathscr{G}_a \leftarrow Rewire(x_{new}, L_s, E)$
    $x_{conn} \leftarrow Nearest(x_{new}, \mathscr{G}_b)$
    $\sigma_{new} \leftarrow Connect(x_{new}, x_{conn}, \mathscr{G}_b)$
    **if** $\sigma_{new} \neq \emptyset$ **and** $c(\sigma_{new}) < c(\sigma_{best})$ **then**
        $\sigma_{best} \leftarrow \sigma_{new}$
    $RandomContraction(\sigma_{best})$
    $u \sim U([0,1])$
    **if** $\sigma_{best} \neq \emptyset$ and $u \leq p_{early}$ **then**
        return $\sigma_{best}$
    $SwapTrees(\mathscr{G}_a, \mathscr{G}_b)$
return $\sigma_{best}$
```

perpendicular segments indicate the robot is paused, and diagonal segments to velocity-tuning depending on the slope.

4.5 Scheduling with Re-Planning

The previous solution provides us with a coordination space and corresponding path that yields a velocity-tuning approach preventing operator collisions. We now look for a solution that yields a shorter mission runtime by also altering the robot trajectories. This solution is found by comparing the current path through the coordination space h and the desired shortest-path path h_{des} which would be a straight line. Given the example in Figure 4.1(a, b), where we see the robots and environment, and the resulting coordination space, we indicate an "ideal" path as in Figure 4.1(c). When searching for a path through the coordination space, we may find a point $x \in X$ such that $h_{des}(x) \bigcap x_{obs} \neq \emptyset$, representing an obstacle. In the example shown in Figure 4.1(c), this is indicated by the blue region, meaning that the ideal path is not valid as it intersects the obstacle. In these situations, the solution is to either plan around the obstacle, corresponding to tuning the velocity of the robots involved - as in the solution for Problem 1 - or creating alternative plans for the robots. In the latter case, the number of operators requested during the original set of times corresponding to the obstacle can now be fulfilled, potentially reducing the overall mission runtime if the resulting plans are shorter than the wait times.

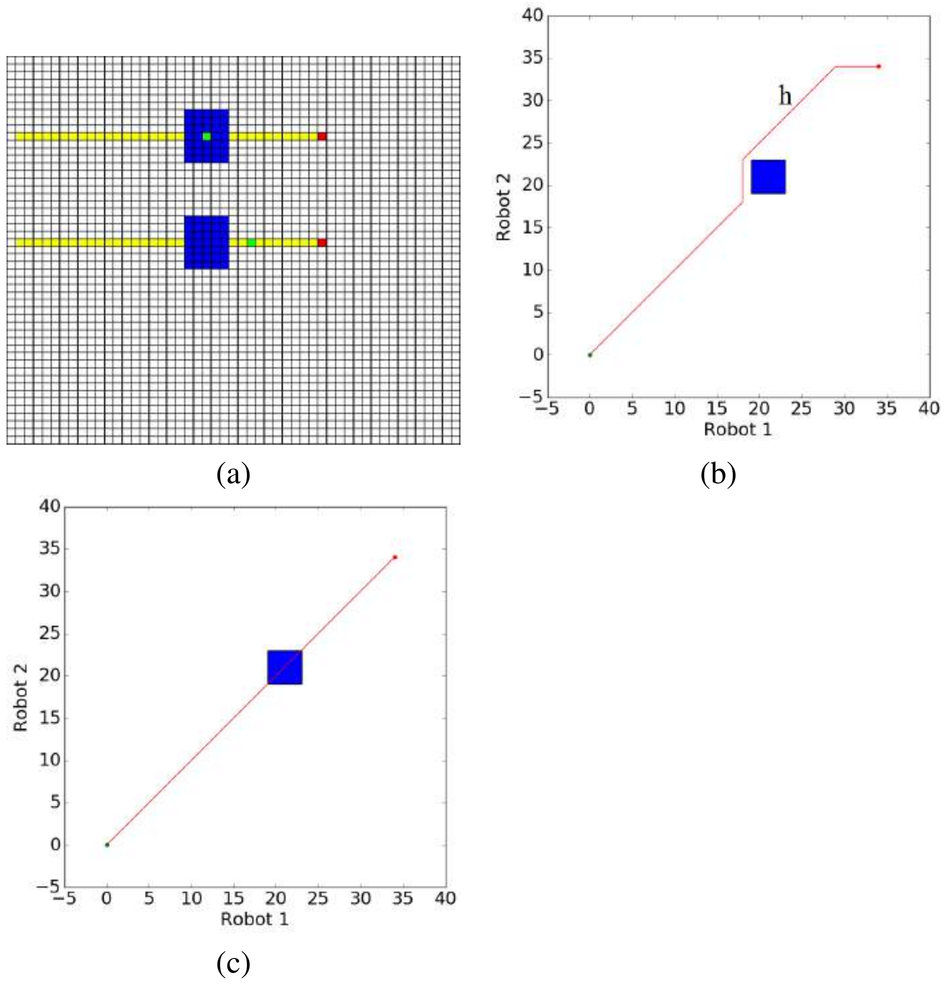
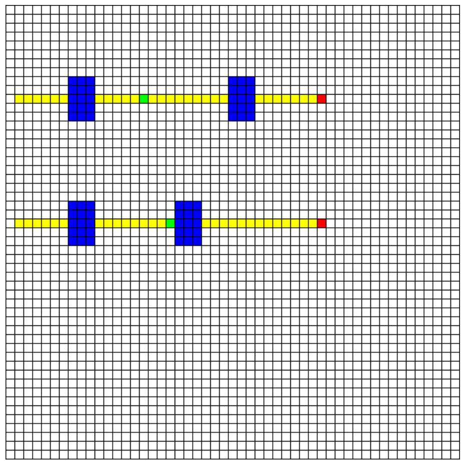
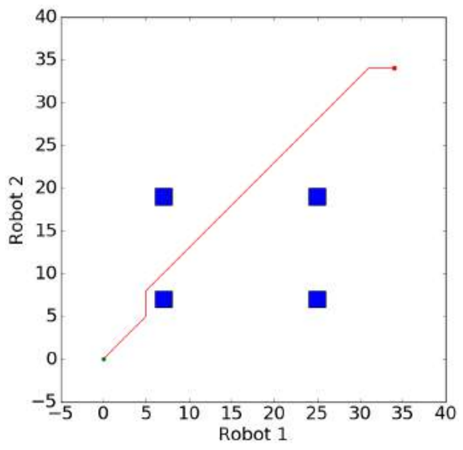


Figure 4.1: Example Environment and resulting Coordination Space. (a) A planar environment with dangerous regions requiring operator supervision to traverse shown in blue, and robot trajectories in yellow. (b) The 2-dimensional Coordination Space resulting from (a). Each axis corresponds to the positions of robots along with their trajectories. The red line indicates an attention-conflict-free path through the coordination space. (c) Coordination space from (b), with the desired (optimal) policy shown as the red line.

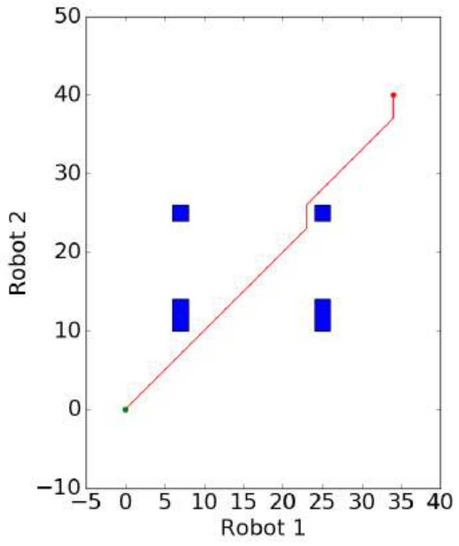
A critical side-effect to keep in mind is that by modifying the trajectories of robots when avoiding any collisions caused by conflicting operator attention requests, we are also potentially changing later parts of their trajectory. This change will lead to a different coordination space, and the possibility of shifting, creating or removing subsequent obstacles. As an illustrative example, Figure 4.2(a) shows two robots, which enter regions requiring supervision at the same time, and produce the coordination space in Figure 4.2(b). The vertical segment of the path h shown in red corresponds to the collision being resolved by pausing robot 1 until robot 2 has finished its operator request before continuing. This scenario could also be solved by re-planning robot 2 so that it avoids operator requests during the original times. However, the robot 1 will then require more time to travel around the dangerous region, causing it to encounter its second critical section at a later time - precisely when robot 1 is entering its second request as well (Figure 4.2(c)) - creating another conflict that must be solved.



(a)



(b)



(c)

Figure 4.2: Example Environment, resulting Coordination Space, and Shifting Conflict Regions. (a) Robots in their environment and their expected trajectories; (b) Original Coordination Space resulting from (a); (c) Final Coordination Space after re-planning around the first attention obstacle.

To handle the complexity of the problem, we use a heuristic approach shown in Algorithm 8. Start by constructing a coordination space $x_{curr} = [0, \tilde{t_f^1}] \times \cdots \times [0, \tilde{t_f^m}]$ and path h_{curr} through x_{free} from $x_{init} = (0, \ldots, 0)$ to $x_{curr\_goal} = (\tilde{t_f^1}, \ldots, \tilde{t_f^m})$ using B-RRT* as in Solution 1. This setup yields our initial solution via velocity-tuning. Then create an ideal path h_{opt}, given by a straight line that assumes no robots require supervision (line

Algorithm 8 Scheduler

Input: $\mathscr{A}$, robots to plan

Output: h, path through x used to derive policy

```
$x_{init} \leftarrow (0,\dots,0); x_{goal} \leftarrow (\tilde{t_f^1}, \tilde{t_f^m})$
$x_{curr} \leftarrow [\tilde{t_f^1},\dots,\tilde{t_f^m}]; h_{curr} \leftarrow$ B-RRT$^*(x_{curr}, x_{init}, x_{goal}, p, \mathscr{C}_{att})$
$x_{des} \leftarrow [0, \tilde{t_f^1},\dots,\tilde{t_f^m}]; h_{des} \leftarrow line(x_{init}, x_{goal}); \mathscr{C}_{desatt} \leftarrow \emptyset$
$o \leftarrow FirstObstacle(h_{des}, C_{att})$
**while** $o \neq \emptyset$ **do**
    $\mathscr{A}_{inv} \leftarrow Sort(o_{\mathscr{A}_{inv}})$
    $\mathscr{A}_{min} \leftarrow \mathscr{A}_{inv}[0 : |o_{\mathscr{A}_{inv}}| - p]$
    $\mathscr{A}_{alt} \leftarrow (\mathscr{A} \setminus \mathscr{A}_{min})$
    $plan(\mathscr{A}^i, t^i_{den}) \forall \mathscr{A}^i \in \mathscr{A}_{min}$
    $\mathscr{A}_{alt} \leftarrow \mathscr{A}_{alt} \bigcup \mathscr{A}_{min}$
    $x_{altgoal} \leftarrow (\tilde{t_f^1},\dots,\tilde{t_f^m}) \forall \mathscr{A}^i \in \mathscr{A}_{alt}$
    **if** $d(x_{init}, x_{altgoal}) \leq c(h_{curr})$ **then**
        $x_{alt} \leftarrow [0, \tilde{t_f^1}] \times \cdots \times [0, \tilde{t_f^m}]; h_{alt} \leftarrow$ B-RRT$^*(x_{alt}, x_{init}, x_{goal}, p, \mathscr{C}_{att})$
        **if** $c(h_{alt} \leq c(h_{curr})$ **then**
            $x_{goal} \leftarrow x_{altgoal}$
            $h_{curr} \leftarrow h_{alt}$
            $x_{curr} \leftarrow x_{alt}$
            $x_{des} \leftarrow x_{alt}$
            $\mathscr{A} \leftarrow (\mathscr{A} \setminus \mathscr{A}_{min}) \bigcup \mathscr{A}_{at}$
        **else**
            $\mathscr{C}_{desatt} \leftarrow \mathscr{C}_{desatt} \bigcup o_{\mathscr{C}_{att}}$
    **else**
        $\mathscr{C}_{desatt} \leftarrow \mathscr{C}_{desatt} \bigcup o_{\mathscr{C}_{att}}$
    $h_{des} \leftarrow$ B-RRT$^*(x_{des}, x_{init}, x_{goal}, p, C_{att})$
    $o \leftarrow FirstObstacle(h_{des}, \mathscr{C}_{att})$
return $h_{des}$
```

3). Next, we verify if the optimal solution is valid by checking for collisions between h_{des} and obstacles in the coordination space and return the first obstacle encountered - if any in line 4. $FirstObstacle$ returns the robots involved in the "collision" $o_{\mathscr{A}_{inv}}$, along with the corresponding configurations $o_{\mathscr{C}_{att}}$ and times that each robot has in conflict $o_{t_{den}}$. If the ideal path is invalid (line 5), we can resolve this in two ways:

1. Alter the involved robots policies (as in the previous solution).
2. Re-plan the involved robots trajectories to eliminate the obstacle.

We now describe how to re-plan the robot's trajectories. Given the robots involved in the collision, $o_{\mathscr{A}_{inv}}$, we sort them in order of ascending length of execution time and select the shortest $|o_{\mathscr{A}_{inv}}| - p$ - the minimum number of robots to re-plan to remove the attention collision (lines 6, 7). This procedure is performed on the robots with the shortest current plans so that possible extensions to their plans due to re-planning should have a minimal effect on the overall length of the mission. Then generate alternative trajectories for the robots, provided operator-denied times $o_{t_{den}}$, and create an alternative goal location $x_{altgoal}$ to account for any shifts in the ending times of the robot plans (lines 8 - 11).

If the distance between x_{init} and the alternative $x_{altgoal}$ is longer than the current solution, then velocity-tuning will yield a better solution, and we incorporate the obstacle into the "desired" set of obstacles (lines 12, 22). Otherwise, we test if the alternative, re-planned solution is better (lines 13, 14). If it is, then update the robots with their re-planned trajectories, and replace the current coordination space and goal to account for any changes in execution times (lines 15 - 17); else we incorporate the obstacle into the "desired" set of obstacles as before (line 19).

We repeat this process of generating desired solutions (line 24) and testing them until the desired path h_{des} no longer intersects any obstacles, at which point we return the final h_{des} that will have no operator conflicts.

4.6 Simulation

In this section, we cover the design and of both simulated and physical experiments, and the results obtained.

4.6.1 Software Simulation for Scheduling with Re-Planning

Here we describe our simulation and provide an example *plan* algorithm that re-plans a robot's trajectory around unsafe areas in the environment - which would require operator supervision - given operator-denied times.

The simulated environment consisted of a discretized 2-dimensional grid-world where robots can only move either horizontally or vertically. The environment also contains hazardous regions (shown in blue) which require operator supervision to traverse, corresponding to configurations in $\mathscr{C}_{att}$.

Example Re-plan Algorithm: The *plan* algorithm used in this example attempts to find the shortest path between x^i_{init} and x^i_{final} within the robot's environment, which can be easily attained via the A^* algorithm [BHH59, Mat02]. However, this path may intersect with regions requiring supervision. First, denote the starting time of the mission as $T_i = 0$. Given times when an operator will not be available for the robot, t_{den}, we modify A^* as follows: Augment A^*'s nodes with an additional *time* parameter. When visiting a node, update its neighbor's *time* attributes to $time + travel\_time$ where *time* is the current time, and $travel\_time$ is the time required to move from the current node to the neighbor. If the neighbor physically resides within $\mathscr{C}_{att}$ and the neighbors *time* is inside t^i_{den}, then we treat it as an obstacle. This modification of A^* provides paths that circumvent obstacles during operator-denied times, with an example shown in Figure 4.3.

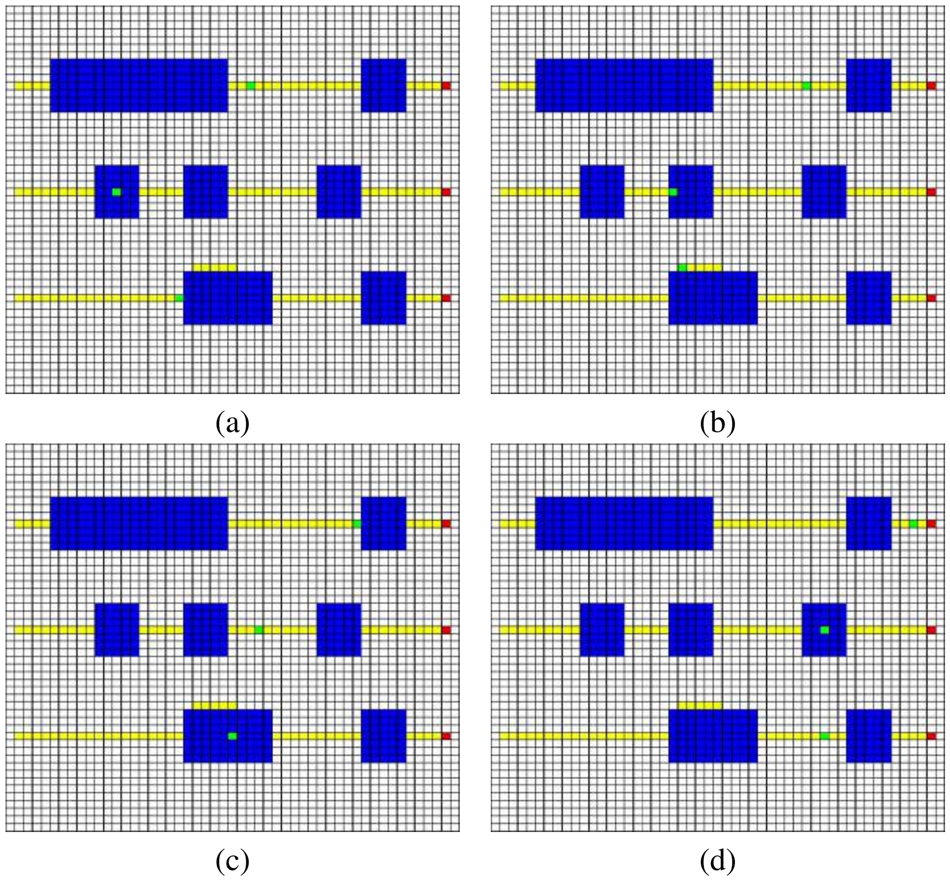


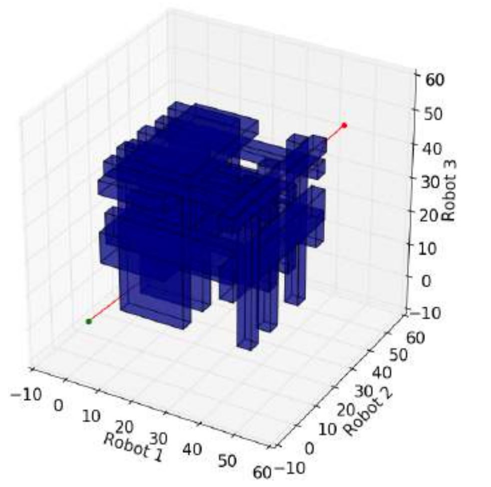
Figure 4.3: Example Simulation Environment. Example simulation. The robots are numbered 1, 2, 3 from top to bottom. (a) Robot 3 stops while Robot 2 passes through its dangerous region. (b) Robot 3 has re-planned its trajectory and is going around the dangerous area, allowing Robot 2 to be supervised. (c) Robot 1 stops to allow Robot 3 enter its dangerous area with supervision. (d) All robots continue to their final goal locations.

In Figure 4.3, we show a simulated example given an environment with three robots. The blue areas in the environment are dangerous, and require operator supervision to prevent an accident. The example was designed to show several operator attention "collision" scenarios. As the robots move from left to right, the following operator requests might arise:

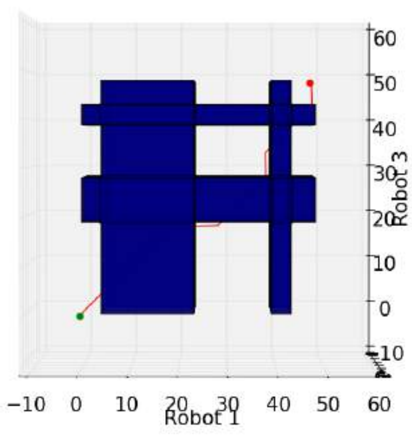
- $\mathscr{A}^1$ requiring an operator
- $\mathscr{A}^1$ and $\mathscr{A}^2$ require an operator at the same time

- $\mathscr{A}^1, \mathscr{A}^2, \mathscr{A}^3$ require an operator at the same time
- $\mathscr{A}^3$ requiring an operator while $\mathscr{A}^1$ and $\mathscr{A}^2$ leave their critical regions
- $\mathscr{A}^2$ requiring an operator
- $\mathscr{A}^1$ and $\mathscr{A}^2$ require an operator at the same time

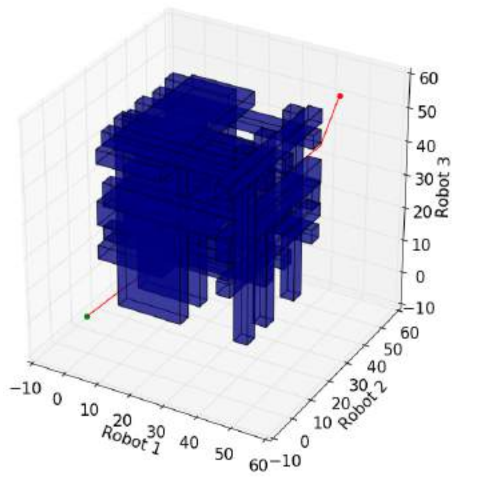
The resulting coordination space is shown in Figure 4.4, where (a, b) is only velocity-tuning, and (c, d) is with re-planning the robot trajectories, which yields a slightly shorter mission ending time than strictly velocity-tuning.



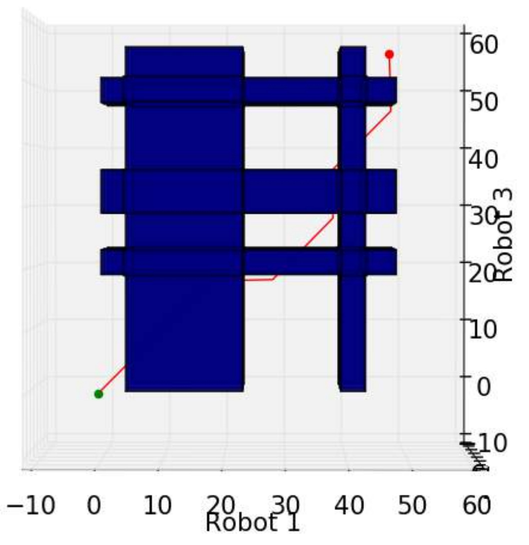
(a)



(b)



(c)



(d)

Figure 4.4: Example Simulation Coordination Space resulting from the example shown in Figure 4.3. (a) Original Coordination Space resulting from the environment and robots in Figure 4.3; (b) Side view of (a); (c) Final Coordination Space after replanning; (d) Side view of (c).

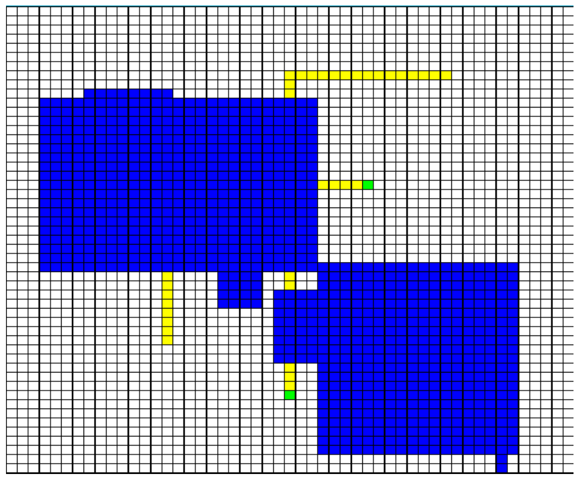


Figure 4.5: Example Random Environment. Example of a randomly-generated environment and trajectories intersecting critical regions.

For further validation, simulations were run using 2-dimensional environment populated with a set of randomly-sized, randomly-placed dangerous regions, and robots placed in randomized obstacle-free starting and goal locations along with a corresponding path between them as shown in Figure 4.5. Across each iteration of the simulations, environments and the starting and goal positions for the robots were randomly generated. In each generated environment, trials were run using 2, 4, or 8 robots, moving at 1 cell/second. These trials were then solved using the solutions for Problem 1 (Scheduling) and Problem 2 (Scheduling with Re-Planning), with 1, 2, 4, or 8 operators. The results can be found in Table 4.1.

Robots	**Operators**	**Average Savings**
2	1	1.126
2	2	0
2	4	0
2	8	0
4	1	1.937
4	2	3.402
4	4	0
4	8	0
8	1	NA
8	2	0.218
8	4	5.284
8	8	0

Table 4.1: Average time savings via re-planning vs velocity-tuning

There is an increase in average time saved when dealing with larger numbers of robots, as re-scheduling can simultaneously resolve multiple robots at once. We purposefully ran the simulations with equal numbers of robots and operators to ensure that there would be no time saved - as there would be no obstacles generated in the first place - and this performed as expected. All tests with 2 and 4 robots completed successfully. In trials with 8 robots and single operator, a solution was not found with the RRT^* parameters that were used. Given 2 operators, $\sim 30\%$ completed, and $\sim 60\%$ for 4 operators. This result was due to the low sample count used when running Attention RRT^*, and the large *steer* length, which prevented it from exploring paths in narrow gaps between obstacles. The tuning of the sample count, steer length and rewire count lie outside the scope of this work, but is nonetheless an stimulating problem we expect to incorporate in future work.

4.6.2 Hardware Experiment for Scheduling with Re-Planning

Here, we further illustrate the problem and solution via a hardware example. This example consisted of a single operator that had to be allocated across three line-following robots in a discrete grid environment.

The robots use a deterministic finite state machine to keep track of the position and orientation, and a transition function given by a second transition-state machine that ensures the robots inter-state path does not deviate from a grid line.

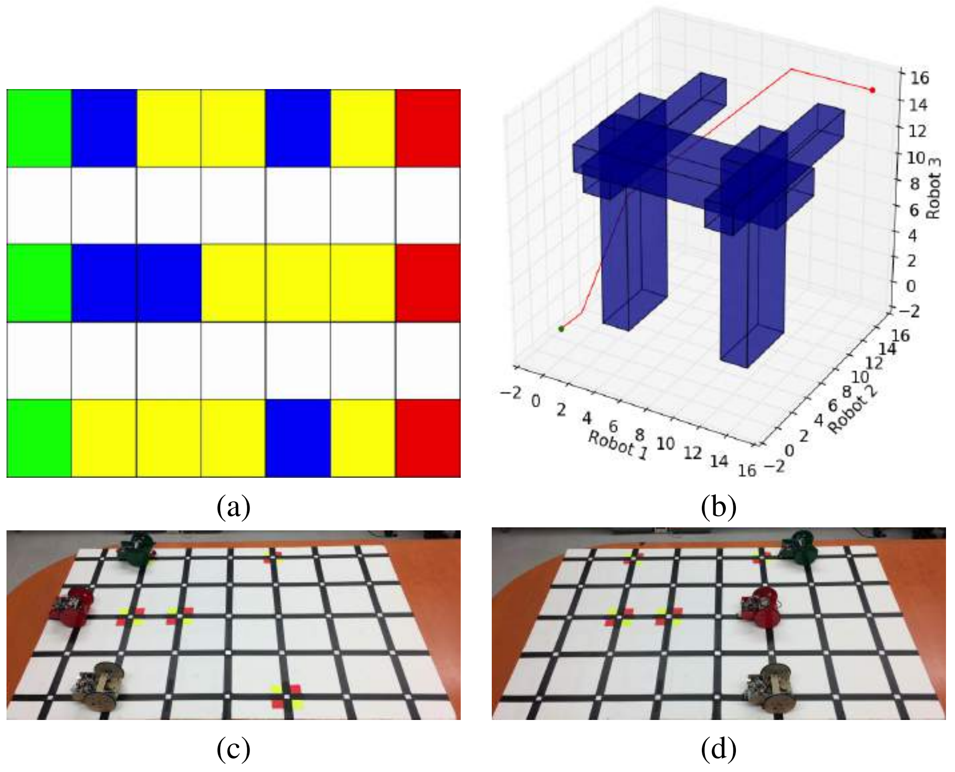


Figure 4.6: Hardware Experiment Example. (a) Simulated Environment; (b) Coordination Space resulting from (a); (c) Analogous hardware simulation at $t = 1$; (d) Hardware simulation at $t = 5$.

The hardware experiment in Figure 4.6 has an equivalent simulated environment shown in Figure 4.6(a). The robots have initial trajectories shown in yellow, which pass through dangerous areas of the environment (blue) requiring operator supervision. The physical implementation represents the dangerous areas using red/yellow squares, in the

same locations as in the virtual simulation. The resulting coordination space in Figure 4.6(b), provides a set of policies enabling the robots to execute their trajectories while ensuring that the operator is not split among multiple robots at the same time. The robots then executed their corresponding policies, moving and pausing when appropriate, with at most one robot entering a dangerous region at a time. Additional experiments and videos can be found at:
http://users.cis.fiu.edu/~jabobadi/oa/

The hardware experiments that were run and shown in the above link show successful runs using the above procedures to design trajectories and policies for three different robots under the supervision of a single operator. The mission ended in the shortest time possible, and the operator did not receive multiple concurrent requests.

4.7 Study Case: Humanoid Robots

In this section, an application of the proposed method to NASA's humanoid robot Valkyrie [RSH$^+$15] as shown Figure 4.7, is presented. Humanoid robots are high degree of freedom complex systems that have been proposed for diverse applications including nuclear-decommissioning tasks [LLP17], disaster response assistance [DDD$^+$15], and vehicles of space exploration [RSH$^+$15]. For many of these tasks, it is desirable to have a *human-in-the-loop* controller to ensure critical and hazardous sub-tasks are completed. The supervised autonomy frameworks to make humanoid robots applicable in performing complex tasks require a practical design for a shared operator control interface which remains an open question. As seen during the DRC, completion of complex tasks in simulated environments with humanoids requires large teams of operators and shared control is indispensable [DDD$^+$15]. Indeed even a simple manipulation task requires coherent operator collaboration or inter-operator communication problems can have detrimental effects [ABB$^+$15]. Thus it is preferable to enforce a 1:1 ratio between humanoids and operator [YNO$^+$15].

4.7.1 Methodology

We propose partitioning the humanoid robot into two serial kinematic chains, the left and right arm, which are denoted as $\mathscr{A}^l$ and $\mathscr{A}^r$ respectively. The desired task is modeled as a typical pick and place operation where the robots must visit designated picking and placing zones defined by the bounding boxes $\mathscr{X}_{i=1...n}$. For example, $\mathscr{A}^r$ picks an object from $\mathscr{X}_1$ and places it in $\mathscr{X}_2$. Next, $\mathscr{A}^l$ collects the object from $\mathscr{X}_2$ and places it in a final location $\mathscr{X}_3$. The picking and placing actions are executed by the end effectors of the right and left arms whose positions are respectively given by $\mathbf{p}^r$ and $\mathbf{p}^l$. When an end effector (robot's hand) is within a bounding box $\mathscr{X}_{i=1...n}$, it requires operator attention, i.e., the action is considered sensitive and require operator supervision. Thus $\mathscr{X}_{i=1...n}$ constitute configuration space constraints that must be transformed into critical regions in the coordination space. Thus, the constraints are represented in the configuration space such that for all times, $\lambda^l(t) \bigcap \lambda^r(t) = \emptyset$, where $\lambda^l(t), \lambda^r(t)$ are inside a bounding box.

Additionally, the re-planning algorithm is modified as follows: Given a set of waypoints τ and operator-denied times t_{den}, *plan* will re-plan sections of λ that reside within $\mathscr{X}$ during times t_{den} if possible. If re-planning is not possible, or if there are critical waypoints that should not be altered (such as waypoints denoting pick and place actions) the waypoints and relevant sections of λ will be untouched and returned to the scheduler as-is.

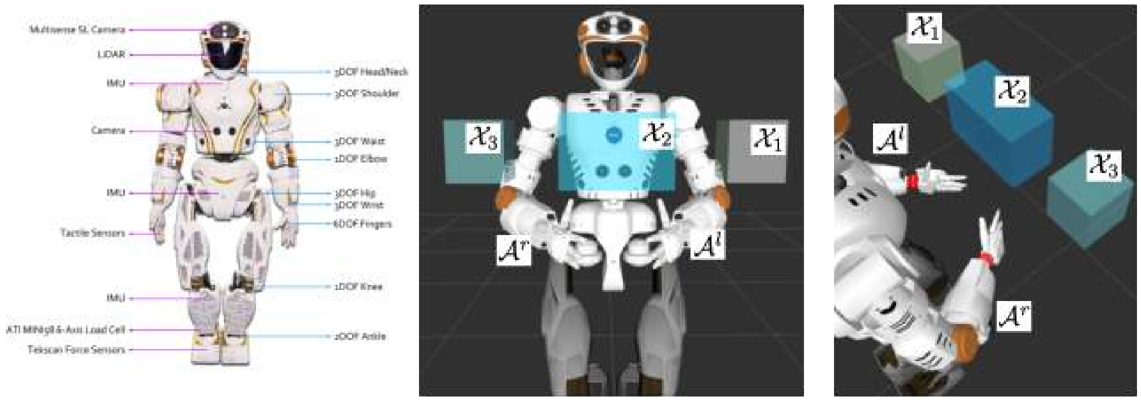


Figure 4.7: (Left) NASA's humanoid robot Valkyrie. (Middle, Right) Experimental setup showing coordination space obstacles and kinematic chains that are treated as independent robots.

4.7.2 Results

The simulation experiments are executed using the dynamic simulator *Gazebo*. An initial set of waypoints are defined for $\mathscr{A}^l$ and $\mathscr{A}^r$. These waypoints consist of a set of Cartesian positions and velocities for the kinematic chains such that λ^r and λ^l satisfy the pick and place task constraints. The initial waypoints are passed to the scheduling algorithm which generates a new set of waypoints that - when separated by a monotonic time step - satisfy both the configuration and coordination space constraints. A cubic interpolation of the waypoints is used to generate a continuous trajectory for execution on the robot. A comparison between the executions before and after the scheduling algorithm is shown in Figure 4.10 and Figure 4.13. The coordination space of these trajectories is shown in Figure 4.14.

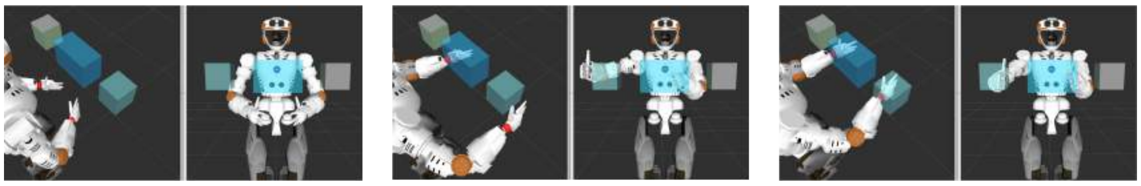


Figure 4.8: Initial trajectory with three attention zones

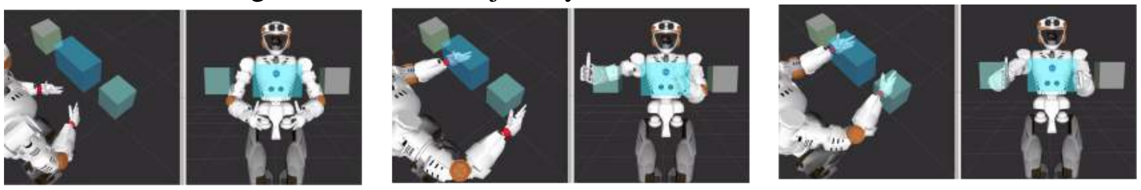


Figure 4.9: Rescheduled and re-planned trajectory with three attention zones

Figure 4.10: Pick and place task with three attention obstacles. The planning reference frame is located at the wrist of the respective arms and is highlight by a red square. Left: Both plans start in a valid position. Middle: Both plans approach the bounding in the same manner, but in the rescheduled case, the right arm execution is slowed down to ensure that before entering the bounding box the left hand has already left the attention zone (Right).

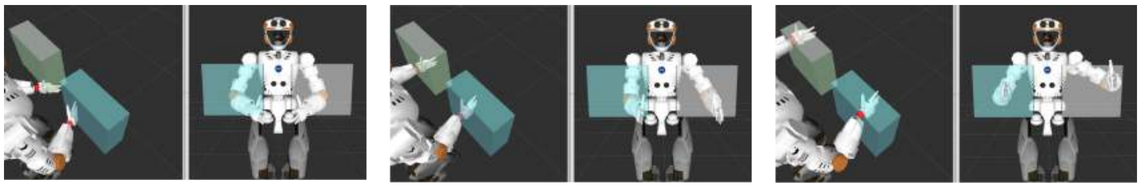


Figure 4.11: Initial trajectory with two attention zones

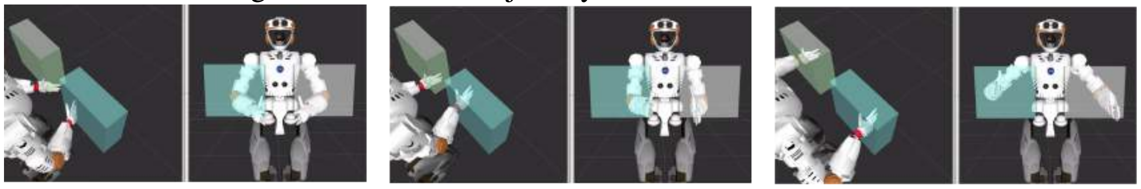


Figure 4.12: Rescheduled and re-planned trajectory with two attention zones

Figure 4.13: Pick and place task with two attention obstacles. The planning reference frame is located at the wrist of the respective arms and is highlight by a red square. Left: Both plans start in a valid position. Middle: The initial trajectory immediately violates attention constraints while the rescheduled trajectory slows the left arm to prevent entry into the area. Right: The right arm is slightly withdrawn (re-planning) to ensure target frame is outside the bounding box before the left has to enter.

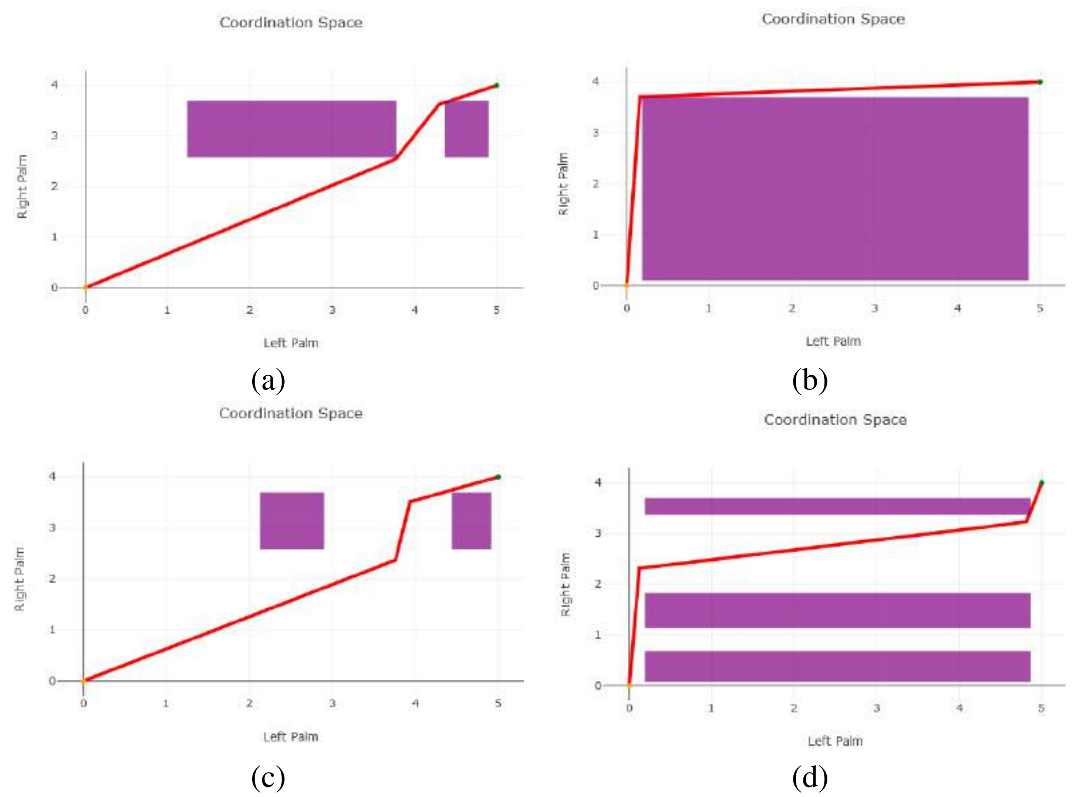


Figure 4.14: Purple areas represent times when both palms will be in a critical zone while the red line is the scheduled times to reach a point for each palm. (a) Trajectory of Figure 4.8. (b) Trajectory of Figure 4.11. (c) Trajectory of Figure 4.9. (d) Trajectory of Figure 4.12.

The two original trajectories are shown in Figures 4.14(a) and 4.14(b) have conflicts in critical areas as illustrated by the line passing through purple areas. The reduced purple areas in Figures 4.14(c) and 4.14(d) demonstrate the re-planning of waypoints, and the altered slope of the line through space indicates a change in time through the waypoints. Both trajectories use a combination of re-planning and rescheduling to generate a collision-free path through the coordination space.

4.8 Conclusion

In this chapter, we provide a novel geometric approach for converting robot trajectories and supervision requests into a set of policies for the robots that permits operators to oversee critical sections of robot plans without being over-allocated [ZRAB17]. The provided solution is also capable of determining when re-planning robots would yield a better solution than velocity-tuning [ZAL$^+$18].

The geometric representation and sampling-based approach has been received with much interest by the community. The visual representation is intuitive easily understood by operators. The solution is also unlike much of the related works in Section 4.2 as it can be applied to a generic set of "bodies", permitting this solution to be used for multiple robots, robots with a high degree of complexity, or a combination of both. Moreover, operators can easily specify safety bounds within the scheduling process by artificially "inflating" obstacles within the coordination space. As an example, operators require time to switch their attention from one robot to another. This time can be represented by extending obstacles in the coordination space towards the origin, and was tested with the Valkyrie robot described previously. Similarly, a robot's path may have some element of uncertainty, especially when outside of a factory setting. In this case, we can "inflate" the obstacles within the coordination space, which would provide a more cautious solution. The ability to schedule and tune the various robots, combined with an intuitive method for setting safety "buffers" allows for more robust operator control, avoiding issues such as those in [YNO$^+$15].

In the future, searching through the coordination space might be modified to use a receding horizon approach to allow for more rapidly changing robot plans if presented with a dynamic environment. We would like to include the stability constraints and interdependence between kinematic chains when working with humanoid robots.

CHAPTER 5

Discussion and Conclusions

In this chapter, we briefly summarize the contents of this dissertation. We used robots in several resource-constrained scenarios. The various approaches described in this dissertation allow for the robots to carry out Deactivation & Decommissioning tasks such as surveying and mapping, ferrying and computing, and interacting effectively with operators. These solutions were verified using both simulated and physical robots.

5.1 Summary of the Dissertation

Effective mapping of fields of interest in complex environments requires careful selection of sampling locations to avoid extended operating times. Traditional approaches often utilize a rastering approach, which provides complete coverage of an area, at the expense of a significant operating time. Moreover, unlike several other types of informative path planning, we are also able to incorporate prior knowledge when selecting locations to visit. Here, we provided an adaptive informative path planning methodology which can quickly localize locations of interest. The path-planning component also allows for a robot with restrictive kinematic constraints given by a tether, and obeys the effects on turning and navigation that this imposes. The approach was tested extensively in simulations, and preliminary hardware experiments also support its efficacy.

Following the mapping of an area, we then focused on how to efficiently use multiple heterogeneous robots to accomplish different tasks within an environment. This work builds upon the existing literature by allowing for the prioritization of certain tasks, permitting for heterogeneous robots and tasks as opposed to homogeneous sets, and the inclusion of environments with both known and unknown obstacles. The provided set of algorithms was shown to improve upon current solutions, allowing for a higher completion rate of tasks.

Finally, we consider situations where tasks cannot be fully automated. In these cases, a human operator must assist the robot in completing its tasks. While traditional $1:1$ ratios of operators to robots have been used with good results, this is no longer capable of scaling given the rising number of robots being used simultaneously, and with high degree-of-freedom robots which can require multiple operators. Existing work on these problems is often difficult for operators to understand, and can have issues scaling when there are large numbers of robots. To alleviate these issues, we provide a sampling-based methodology which can scale to many robots, even when there are relatively few operators. This work also has the benefit of functioning with both discrete robots, and robots with multiple operator requirements, or a combination of both. The solution was tested in both of these scenarios, using simulated and hardware experiments.

In this work, we have developed strategies for tackling several of the problems faced in the process of robotic deactivation and decommissioning. The strategies were based on current state-of-the-art and extended this work. The significant contributions have all been tested and met with approval from peer-reviewed venues. The advances in this work will enable more effective monitoring of legacy structures, and protect operators from having to enter dangerous environments.

BIBLIOGRAPHY

[ABB+15] Christopher G Atkeson, Benzun P Wisely Babu, Nandan Banerjee, Dmitry Berenson, Christoper P Bove, Xiongyi Cui, Mathew DeDonato, Ruixiang Du, Siyuan Feng, Perry Franklin, et al. No falls, no resets: Reliable humanoid behavior in the darpa robotics challenge. In *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 623–630. IEEE, 2015.

[alp] Alpha-gamma hot cell facility image. `https://www1.anl.gov/images/ARRA_AGHCF-200.JPG`.

[Ama] Amazone. Bonirob agriculture robot. `http://info.amazone.de/DisplayInfo.aspx?id=29417`.

[atla] Atlas robot. `https://www.bostondynamics.com/sites/default/files/styles/max_1300x1300/public/2017-05/MPM_4381-2.jpg`.

[atlb] Atlas robot image. `https://www.bostondynamics.com/atlas`.

[BCDF10] Eric Brochu, Vlad M Cora, and Nando De Freitas. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv:1012.2599*, 2010.

[BG08] Priyadarshi Bhattacharya and Marina L Gavrilova. Roadmap-based path planning-using the voronoi diagram for a clearance-based shortest path. *Robotics & Automation Magazine, IEEE*, 15(2):58–66, 2008.

[BG13] Travis J Barnes and Jason R Gunter. 241-ay-101 tank construction extent of condition review for tank integrity. `https://digital.library.unt.edu/ark:/67531/metadc834350/`, 2013.

[BHH59] J. Beardwood, J.H. Halton, and J.M. Hammersley. The shortest path through many points. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 55, pages 299–327. Cambridge Univ Press, 1959.

[BLNZ95] Richard H Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16(5):1190–1208, 1995.

[bon] Bonirob robot image. `http://info.amazone.de/DisplayInfo.aspx?id=29417`.

[Bru12] Geoff Brumfiel. Fukushima's doses tallied. `https://www.nature.com/news/fukushima-s-doses-tallied-1.10686`, May 2012.

[Bur18] Matt Burgess. These tiny autonomous robots are ready to deliver lunch to your office. `https://www.wired.co.uk/article/starship-technologies-robots-deliveries-intuit-compass-test`, Apr 2018.

[But11] Giorgio C Buttazzo. *Hard real-time computing systems: predictable scheduling algorithms and applications*, volume 24. Springer Science & Business Media, 2011.

[BVX15] Peter Brass, Ivo Vigan, and Ning Xu. Shortest path planning for a tethered robot. *Computational Geometry*, 48(9):732–742, 2015.

[CCDPdJ11] J.W. Crandall, M.L. Cummings, M. Della Penna, and Paul M.A. de Jong. Computing the effects of operator attention allocation in human control of multiple robots. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 41(3):385–397, 2011.

[CCS$^+$16] Carina Cai, Bryan Carter, Mihir Srivastava, Joseph Tsung, John Vahedi-Faridi, and Caroline Wiley. Designing a radiation sensing uav system. In *Systems and Information Engineering Design Symposium (SIEDS), 2016 IEEE*, pages 165–169. IEEE, 2016.

[CLD13] Nannan Cao, Kian Hsiang Low, and John M Dolan. Multi-robot informative path planning for active sensing of environmental phenomena: A tale of two algorithms. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pages 7–14. International Foundation for Autonomous Agents and Multiagent Systems, 2013.

[CLRS09] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2009.

[CM07] M.L. Cummings and P.J. Mitchell. Operator scheduling strategies in supervisory control of multiple uavs. *Aerospace Science and Technology*, 11(4):339–348, 2007.

[CMS16] D Connor, PG Martin, and TB Scott. Airborne radiation mapping: overview and application of current and future aerial systems. *International Journal of Remote Sensing*, 37(24):5953–5987, 2016.

[cra] Pneumatic crawler image. `https://arc.fiu.edu/robotics-technology/pneumatic-pipe-crawler/`.

[DAMT17] M. DiBono, A. Abrahao, D. McDaniel, and Y. T. Tan. Development and testing of robotic inspection tools for the hanford high-level waste double shell tanks. In *WM Symposia*, 2017.

[DDD$^+$15] Mathew DeDonato, Velin Dimitrov, Ruixiang Du, Ryan Giovacchini, Kevin Knoedler, Xianchao Long, Felipe Polido, Michael A Gennert, Taşkın Padır, Siyuan Feng, et al. Human-in-the-loop control of a humanoid robot for disaster response: A report from the darpa robotics challenge trials. *Journal of Field Robotics*, 32(2):275–292, 2015.

[DMA$^+$18] M. DiBono, D. McDaniel, A. Abrahao, L. Lagos, and Y. T. Tan. Engineering scale testing of robotic inspection tools for double shell tanks at hanford. In *WM Symposia*, 2018.

[DT] M. Braza J. Delicath S. Fletcher R. Johnson J. Larson A. Liles K. Stetler D.C. Trimble, D. Feehan. Condition of tanks may further limit doe's ability to respond to leaks and intrusions. Technical Report GAO-15-40, United States Government Accountability Office.

[EGHR] J K Engeman, C L Girardot, D J Harlow, and C L Rossenkrance. *Tank 241-AY-102 Leak Assessment Report*.

[Fan07] Andrew Fanton. Point seeking: a family of dynamic path finding algorithms. `https://scholarworks.rit.edu/theses/82/`, 2007.

[gem] Gemini-scout mine rescue vehicle image. `https://www.sandia.gov/research/robotics/unique_mobility/gemini-scout.html`.

[Gir15] Crystal Girardot. Hanford double-shell tank visual inspections. In *ASNT Annual Conference 2015*, pages 52–56, 2015.

[GJ79] Michael R Garey and David S Johnson. *Computers and intractability: a guide to NP-completeness*. WH Freeman and Company, San Francisco, 1979.

[GK07] Shimin Guo and Srinivasan Keshav. Fair and efficient scheduling in data ferrying networks. In *Proceedings of the 2007 ACM CoNEXT conference*, page 13. ACM, 2007.

[GK11] Daniel Golovin and Andreas Krause. Adaptive submodularity: Theory and applications in active learning and stochastic optimization. *Journal of Artificial Intelligence Research*, 42:427–486, 2011.

[GS07] Michael A Goodrich and Alan C Schultz. Human-robot interaction: a survey. *Foundations and trends in human-computer interaction*, 1(3):203–275, 2007.

[Gun15] Jason R Gunter. Hanford review of double-shell tank construction. In *ASNT Annual Conference 2015*, pages 67–71, 2015.

[Hal15] Laura Hall. Valkyrie robot image. `https://www.nasa.gov/sites/default/files/thumbnails/image/valkyrie-robot-4.jpg`, Jun 2015.

[han14] 2014 hanford lifecycle scope, schedule and cost report. Technical Report MSU-CSE-06-2, U.S. Department of Energy, February 2014.

[Hau13] K.K. Hauser. Minimum constraint displacement motion planning. In *Robotics: Science and Systems*, 2013.

[HGG$^+$14] Gregory Hitz, Alkis Gotovos, Marie-Éve Garneau, Cédric Pradalier, Andreas Krause, Roland Y Siegwart, et al. Fully autonomous focused exploration for robotic environmental monitoring. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 2658–2664. IEEE, 2014.

[HS13] Geoffrey A Hollinger and Gaurav S Sukhatme. Sampling-based motion planning for robotic information gathering. In *Robotics: Science and Systems*, volume 3. Citeseer, 2013.

[HSH02] Evert Helms, Rolf Dieter Schraft, and M Hagele. rob@ work: Robot assistant in industrial environments. In *IEEE International Workshop on Robot and Human Interactive Communication*, pages 399–404. IEEE, 2002.

[Hug08] T. Hughes. Human-automation coordination in multi-uav control. In *AIAA Guidance, Navigation and Control Conference and Exhibit*, page 6315, 2008.

[JP] Matthew Jordan and Alejandro Perez. Optimal bidirectional rapidly-exploring random trees. `https://dspace.mit.edu/bitstream/handle/1721.1/79884/MIT-CSAIL-TR-2013-021.pdf`.

[KF10] Sertac Karaman and Emilio Frazzoli. Incremental sampling-based algorithms for optimal motion planning. *Robotics Science and Systems VI*, 104, 2010.

[KF11] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *The international journal of robotics research*, 30(7):846–894, 2011.

[Kis15] Erin Kisliuk. R5 valkyrie robot. `https://www.nasa.gov/feature/r5`, Sep 2015.

[KL15] Soonkyum Kim and Maxim Likhachev. Path planning for a tethered robot using multi-heuristic a* with topology-based heuristics. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 4656–4663. IEEE, 2015.

[knia] Knightscope faq. `https://www.knightscope.com/faq/`.

[knib] Knightscope image. `https://static1.squarespace.com/static/599249e2bf629ad8f9ce4d84/5a1f33df8165f518b0a7636f/5a1f340ce2c483cbdb6576ec/1511997763289/IMG_1326.JPG`.

[LaV06] S.M. LaValle. *Planning Algorithms*. Cambridge University Press, Cambridge, U.K., 2006. Also available at http://planning.cs.uiuc.edu/.

[LH96] S.M. LaValle and S.A. Hutchinson. Optimal motion planning for multiple robots having independent goals. In *IEEE International Conference on Robotics and Automation*, pages 2847–2852, April 1996.

[LH98] S.M. LaValle and S.A. Hutchinson. Optimal motion planning for multiple robots having independent goals. *IEEE Transactions on Robotics and Automation*, 14(6):912–925, December 1998.

[LL73] Chung Laung Liu and James W Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the ACM (JACM)*, 20(1):46–61, 1973.

[LLC17] Savannah River Remediation LLC. Radioactive liquid waste facilities. https://www.srs.gov/general/news/factsheets/srr_rlwf.pdf, 2017.

[LLP17] Xianchao Long, Philip Long, and Taşkın Padir. Compositional autonomy for humanoid robots with risk-aware decision-making. In *Proceedings of the 2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, pages 553–560. IEEE, 2017.

[mar] Curiosity rover. https://www.jpl.nasa.gov/spaceimages/images/wallpaper/PIA14156-1920x1200.jpg.

[Mat02] James Matthews. Basic a* pathfinding made simple. *AI Game Programming Wisdom*, pages 105–113, 2002.

[MAZ$^+$15] Alireza Monfared, Mostafa Ammar, Ellen Zegura, David Doria, and David Bruno. Computational ferrying: Challenges in deploying a mobile high performance computer. In *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2015 IEEE 16th International Symposium on a*, pages 1–6. IEEE, 2015.

[Mic17] Ed Waller Michael E. Hosmar, Scott B. Nokleby. Experimental Testing of an Autonomous Radiation Mapping Robot. *CCToMM Mechanisms, Machines, and Mechatronics (M3) Symposium*, 2017.

[mig] Mighty mouse (m2). https://www.sandia.gov/research/robotics/advanced_manipulation/mighty_mouse.html.

[mis] Mars science laboratory curiosity rover. https://www.jpl.nasa.gov/missions/mars-science-laboratory-curiosity-rover-msl/.

[mq-15a] Mq-9 reaper. https://www.af.mil/About-Us/Fact-Sheets/Display/Article/104470/mq-9-reaper/, Sep 2015.

[mq-15b] Mq-9 reaper image. https://media.defense.gov/2009/Mar/17/2000608254/-1/-1/0/090127-F-7383P-001.JPG, Sep 2015.

[MR12] Roman Marchant and Fabio Ramos. Bayesian optimisation for intelligent environmental monitoring. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 2242–2249. IEEE, 2012.

[MR14] Roman Marchant and Fabio Ramos. Bayesian optimisation for informative continuous path planning. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 6136–6143. IEEE, 2014.

[MS12] R. Murphy and J. Shields. The role of autonomy in dod systems. Technical report, Technical report, Department of Defense, Defense Science Board Task Force Report, 2012.

[Mur04] Robin R Murphy. Human-robot interaction in rescue robotics. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 34(2):138–153, 2004.

[nat11] Pictures: ”liquidators” endured chernobyl 25 years ago. `https://news.nationalgeographic.com/news/energy/2011/04/110426/chernobyl-25-years-liquidators-pictures/`, Apr 2011.

[nuk] Wipp cross section. `https://www.nukewatch.org/graphics/NEW-WIPP-Xsection.jpg`.

[par] *10 CFR 835, Occupational Radiation Protection.*

[Par08] L.E. Parker. Multiple mobile robot systems. In *Springer Handbook of Robotics*, pages 921–941. Springer, 2008.

[PJGH15] S.S. Ponda, L.B. Johnson, A. Geramifard, and J.P. How. Cooperative mission planning for multi-uav teams. In *Handbook of Unmanned Aerial Vehicles*, pages 1447–1490. Springer, 2015.

[QA15] Ahmed Hussain Qureshi and Yasar Ayaz. Intelligent bidirectional rapidly-exploring random trees for optimal motion planning in complex cluttered environments. *Robotics and Autonomous Systems*, 68:1–11, 2015.

[QSBZ12] Kui Qian, Aiguo Song, Jiatong Bao, and Huatao Zhang. Small teleoperated robot for nuclear radiation and chemical leak detection. *International Journal of Advanced Robotic Systems*, 9(3):70, 2012.

[RBM$^+$18] M. M. Rahman, L. Bobadilla, A. Mostafavi, T. Carmenate, and S. A. Zanlongo. An automated methodology for worker path generation and safety assessment in construction projects. *IEEE Transactions on Automation Science and Engineering*, 15(2):479–491, April 2018.

[RCB$^+$15] Md Mahbubur Rahman, Triana Carmenate, Leonardo Bobadilla, Sebastian Zanlongo, and Ali Mostafavi. A coupled discrete-event and motion planning methodology for automated safety assessment in construction projects. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 3849–3855. IEEE, 2015.

[RCBM14] Md Mahbubur Rahman, Triana Carmenate, Leonardo Bobadilla, and Ali Mostafavi. Ex-ante assessment of struck-by safety hazards in construction projects: A motion-planning approach. In *Automation Science and Engineering (CASE), 2014 IEEE International Conference on*, pages 277–282. IEEE, 2014.

[RFI$^+$15] Sarvapali D Ramchurn, Joel E Fischer, Yuki Ikuno, Feng Wu, Jack Flann, and Antony Waldock. A study of human-agent collaboration for multi-uav task allocation in dynamic environments. In *IJCAI*, pages 1184–1192, 2015.

[RSH$^+$15] Nicolaus A Radford, Philip Strawser, Kimberly Hambuchen, Joshua S Mehling, William K Verdeyen, A Stuart Donnan, James Holley, Jairo Sanchez, Vienny Nguyen, Lyndon Bridgwater, et al. Valkyrie: Nasa's first bipedal humanoid robot. *Journal of Field Robotics*, 32(3):397–419, 2015.

[RW06] Carl Edward Rasmussen and Christopher KI Williams. Gaussian processes for machine learning. 2006. *The MIT Press, Cambridge, MA, USA*, 38:715–719, 2006.

[SAÅ$^+$04] Lui Sha, Tarek Abdelzaher, Karl-Erik Årzén, Anton Cervin, Theodore Baker, Alan Burns, Giorgio Buttazzo, Marco Caccamo, John Lehoczky, and Aloysius K Mok. Real time scheduling theory: A historical perspective. *Real-time systems*, 28(2-3):101–155, 2004.

[sel] Starship technologies robot image. `https://wi-images.condecdn.net/image/9qLWVkZmZmv/crop/810/f/128_starshipintuit-1.jpg`.

[SKKS09] Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. *arXiv preprint arXiv:0912.3995*, 2009.

[SKKS12] Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias W Seeger. Information-theoretic regret bounds for gaussian process optimization in the bandit setting. *IEEE Transactions on Information Theory*, 58(5):3250–3265, 2012.

[SLA12] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pages 2951–2959, 2012.

[ST08] Antonios Skordylis and Niki Trigoni. Delay-bounded routing in vehicular ad-hoc networks. In *Proceedings of the 9th ACM international symposium on Mobile ad hoc networking and computing*, pages 341–350. ACM, 2008.

[Ste12] Michael L Stein. *Interpolation of spatial data: some theory for kriging*. Springer Science & Business Media, 2012.

[SWWB11] Julie Shah, James Wiken, Brian Williams, and Cynthia Breazeal. Improved human-robot team performance using chaski, a human-inspired plan execution system. In *6th international conference on Human-robot interaction*, pages 29–36. ACM, 2011.

[TMMK15] Pete Trautman, Jeremy Ma, Richard M Murray, and Andreas Krause. Robot navigation in dense human crowds: Statistical models and experimental studies of human–robot cooperation. *The International Journal of Robotics Research*, 34(3):335–356, 2015.

[Tra12] Pete Trautman. Probabilistic tools for human-robot cooperation. In *Human Agent Robot Teamwork Workshop HRI*, 2012.

[UGL$^+$14] Kyle Usbeck, Matthew Gillen, Joseph Loyall, Andrew Gronosky, Joshua Sterling, Ralph Kohler, Richard Newkirk, and David Canestrare. Data ferrying to the tactical edge: A field experiment in exchanging mission plans and intelligence in austere environments. In *Military Communications Conference (MILCOM), 2014 IEEE*, pages 1311–1317. IEEE, 2014.

[was] Wipp 2014 accident overview. `http://wipp.energy.gov/wipprecovery-accident-desc.asp`.

[waya] Waymo ipace. `https://storage.googleapis.com/sdc-prod/v1/press/WaymoIPACE-city4.jpg`.

[wayb] Waymo journey. `https://waymo.com/journey/`.

[WNS12] Ronald Wilcox, Stefanos Nikolaidis, and Julie Shah. Optimization of temporal dynamics for adaptive human-robot interaction in assembly manufacturing. *Robotics Science and Systems VIII*, pages 441–448, 2012.

[WZG+14] J.J. Wang, Y.F. Zhang, L. Geng, J.Y.H. Fuh, and S.H. Teo. Mission planning for heterogeneous tasks with heterogeneous uavs. In *International Conference on Control Automation Robotics & Vision (ICARCV)*, pages 1484–1489. IEEE, 2014.

[YH12] Erkang You and Kris Hauser. Assisted teleoperation strategies for aggressively controlling a robot arm with 2d input. In *Robotics: science and systems*, volume 7, page 354, 2012.

[YNO+15] Holly A Yanco, Adam Norton, Willard Ober, David Shane, Anna Skinner, and Jack Vice. Analysis of human-robot interaction at the darpa robotics challenge trials. *Journal of Field Robotics*, 32(3):420–444, 2015.

[ZAL+18] Sebastian Zanlongo, Franklin Abodo, Philip Long, Taskin Padir, and Leonardo Bobadilla. Multi-robot scheduling and path-planning for non-overlapping operator attention. In *2018 Second IEEE International Conference on Robotic Computing (IRC)*, pages 87–94. IEEE, 2018.

[ZAZ04] Wenrui Zhao, Mostafa Ammar, and Ellen Zegura. A message ferrying approach for data delivery in sparse mobile ad hoc networks. In *Proceedings of the 5th ACM international symposium on Mobile ad hoc networking and computing*, pages 187–198. ACM, 2004.

[ZAZ05] Wemi Zhao, Mostafa Ammar, and Ellen Zegura. Controlling the mobility of multiple data transport ferries in a delay-tolerant network. In *INFOCOM 2005. 24th annual joint conference of the IEEE computer and communications societies. Proceedings IEEE*, volume 2, pages 1407–1418. IEEE, 2005.

[ZBT17] Sebastián A. Zanlongo, Leonardo Bobadilla, and Yew Teck Tan. Path-planning of miniature rovers for inspection of the hanford high-level waste double shell tanks. In *Florida Conference on Recent Advances in Robotics, 2017*, 2017.

[ZC09] W Zeng and RL Church. Finding shortest paths on real road networks: the case for A*. *International Journal of Geographical Information Science*, 23(4):531–543, 2009.

[ZRAB17] Sebastián A Zanlongo, Mahbubur Rahman, Franklin Abodo, and Leonardo Bobadilla. Multi-robot planning for non-overlapping operator attention allocation. In *Robotic Computing (IRC), IEEE International Conference on*, pages 109–112. IEEE, 2017.

[ZWBS16] Sebastián A Zanlongo, Alexander C Wilson, Leonardo Bobadilla, and Tamim Sookoor. Scheduling and path planning for computational ferrying. In *Military Communications Conference, MILCOM 2016-2016 IEEE*, pages 636–641. IEEE, 2016.

VITA

SEBASTIÁN A. ZANLONGO

May 30, 1991	Born, Miami, Florida, U.S.A.
2013–2018	Ph.D., Computer Science Florida International University Miami, FL
2015–2018	DOE Fellow Florida International University Miami, FL
2018	Student Guest Savannah River National Laboratory Aiken, South Carolina
2017	Student Guest Sandia National Laboratories Albuquerque, New Mexico
2016	Student Guest Los Alamos National Laboratory Los Alamos, New Mexico
2015	Research Intern G2, Inc. Annapolis Junction, Maryland
2013–2015	Graduate Teaching Assistant Florida International University Miami, FL

CONFERENCES

Zanlongo, Sebastian, et al. *Informative Path Planning for Mapping Radiation.* American Nuclear Society Winter Meeting and Expo; Orlando, FL, USA; November 11 - 15, 2018.

Zanlongo, Sebastian, et al. *Multi-Robot Scheduling and Path-Planning for Non-Overlapping Operator Attention.* International Conference on Robotic Computing; Laguna Hills, CA, USA; January 31 - February 2, 2018.

Zanlongo, Sebastian, et al. *Path-Planning of Miniature Rovers for Inspection of the Hanford High-Level Waste Double Shell Tanks.* Florida Conference on Recent Advances in Robotics; Boca Raton, FL, USA; May 11-12, 2017.

Zanlongo, Sebastian, et al. *Multi-Robot Planning for Non-Overlapping Operator Attention Allocation*. International Conference on Robotic Computing; Taichung, Taiwan; April 10-12, 2017.

Zanlongo, Sebastian, et al. *Scheduling and Path Planning for Computational Ferrying*. Military Communications Conference; Baltimore, MD, USA; November 1-3, 2016.

Abrahao, Anthony, et al. *Remotely Operated Multi-Tracked Robot for Visual Inspection in D&D Activities*. Florida Conference on Recent Advances in Robotics, Miami, FL, May 12-13, 2016.

Rahman, Md Mahbubur, et al. *A coupled discrete-event and motion planning methodology for automated safety assessment in construction projects*. International Conference on Robotics and Automation; Seattle, WA, May 26-30, 2015.

Carmenate, Triana, et al. *Non-Invasive Sensing System for Decoding Occupancy Behaviors Affecting Building Energy Performance*. ASCE Computing in Civil Engineering Workshop; Austin, TX, USA; June 21-23, 2015.

JOURNALS

Zanlongo, Sebastian, et al. *Planning, Scheduling, and Deploying for Computational Ferrying*. International Journal of Next-Generation Computing; 2018.

Rahman, Md Mahbubur, et al. *An Automated Methodology for Worker Path Generation and Safety Assessment in Construction Projects*. IEEE Transactions on Automation Science and Engineering; 2016.

Guo, Mingming, et al. *In-Network Trajectory Privacy Preservation*. ACM Computing Surveys (CSUR); 2015.