

Long-Term Monitoring for Robots with Probabilistic Models



Alex Stephens

Exeter College
University of Oxford

A thesis submitted for the degree of
Doctor of Philosophy
Michaelmas 2024

And so, does the destination matter? Or is it the path we take? I declare that no accomplishment has substance nearly as great as the road used to achieve it. We are not creatures of destinations. It is the journey that shapes us. Our calloused feet, our backs strong from carrying the weight of our travels, our eyes open with the fresh delight of experiences lived.

Brandon Sanderson, *The Way of Kings*

Acknowledgements

The journey of a PhD student is a long and demanding one. It can be invigorating, daunting, fulfilling, frustrating, empowering, demoralising—and that’s just over the course of a single supervision meeting. As I stand, finally, within reach of the finish line, I am immensely grateful to all of the wonderful people who have supported me along the way. There are too many to mention individually, so to anyone I may have missed, to all the friends, family, colleagues, and mentors who have helped me over the years: thank you.

First and foremost, my sincere thanks to my supervisors, Nick Hawes and Bruno Lacerda. They were exceedingly generous with their time, wisdom, and guidance—always encouraging about my next great idea, always patient when inevitably it took me longer than planned to deliver on it. The many hours spent discussing ideas in front of the whiteboard in Nick’s office were truly some of the most enjoyable and rewarding of my DPhil.

Funding for my DPhil came from the AIMS EPSRC Centre for Doctoral Training and the AWS Lighthouse Scholarship programme, and I am grateful for the incredible opportunities that their support has afforded me. I also acknowledge the use of the University of Oxford Advanced Research Computing (ARC) facility, which provided computational resources generous enough to spare me from the dreaded ordeal of optimising my code. Thank you to my friends in the AIMS CDT, and most of all to Wendy, our superstar of an administrator, for consistently going the extra mile to look after us all.

The work life of a PhD student can be isolating, but fortunately that was not my experience. For that, I thank my fellow GOALS students, past and present: Charlie, Marc, Anna, Mohamed, Michael, Matt, Clarissa, Lara, Branton, Alex (the other one), Carl, An, Michal, Alex (the other, other one), and Roland. Thank you all for fostering such a great work environment; it was a genuine privilege to work alongside such a lovely, brilliant group of people. I was fortunate to have the opportunity to collaborate with Matt on some of the work in this thesis, and I remain in awe of his ability to get stuff done blazingly fast under tight deadlines. I must also commend Charlie and Anna for setting an unreasonably high bar with their reading group baking efforts.

Moving halfway around the world amidst a global pandemic was a daunting prospect, and I feel fortunate to have had the community of the Exeter College MCR to welcome me to Oxford during those turbulent times. My thanks to Tom, my college dad, for

candidly sharing his experiences as MCR President, and having a valiant crack at talking me out of running for the role myself—I'm glad I knew what I was getting myself into. Thank you to my housemates, Emma, Matěj, and Chloe, for the many shared meals and living room chats, which were always welcome ways to unwind after long days of work. Thanks also to my friends from Sydney (Max, Janek, Rومان, Joel, Calida, Justin, and more!) for not forgetting about me—your support over the years is massively appreciated, and it is always a joy when our paths do cross.

Staying physically active and spending time outdoors throughout my DPhil has been invaluable to me, bringing much-needed balance to my life and preventing the walls from closing in around me when my stress levels crept up. I am indebted to all of the amazing people who have helped me to discover new sporting passions over the last four years. I'm grateful to the University Parks parkrun team, for giving me something to look forward to each and every week—particular thanks to Amy and Kieran for helping me to discover the joys of parkrun volunteering. Thank you to my friends from the Oxford Uni Triathlon Club (OUTriC) for their camaraderie and support; discovering triathlon has been a highlight of my time in Oxford, and I've now completed (and enjoyed!) events that would have been unthinkable to my younger self. My thanks also to the Headington RoadRunners; I feel very lucky to be part of such a friendly and welcoming community.

Behind everything I have achieved over the years has been the steadfast love and support of my family. I thank my cousins, Vinil and Ash, the role models who inspired many of my interests over the years, and my uncle Anil. I thank my Oma, Maria, for the love, encouragement, and food that she has provided me with over the years. I am grateful also to my late grandfather, my Opa, Bill: although he never saw me move to Oxford for my DPhil, the journey started long before that—he taught me from an early age to appreciate the privilege of education, and I have carried that lesson with me all the way.

My heartfelt thanks to my partner, Bronya, for her kindness and understanding through my busiest periods, and for her enthusiasm in celebrating my successes with me. My life is immeasurably better for having her in it, and I'm excited to see where the coming years will take us.

I reserve my final thanks for my parents, André and Sadhana. They have always encouraged and supported me in whatever I chose to do, even when it meant moving halfway around the world. The opportunities that I have been fortunate enough to have are the result of their hard work and sacrifices, and for that, I am forever grateful.

Abstract

As robots become increasingly capable and robust, there is a growing scope for *long-term* monitoring missions in which robots operate autonomously without human intervention. Long-term robot missions hold immense potential across a range of scientific and industrial applications, from environmental monitoring to precision agriculture. However, mission planning in this setting is challenging, requiring robots to reason about complex and uncertain environment models over long horizons. Many existing works on planning for long-term robot missions use simplistic models of robots and their environments, overlooking the complexity and uncertainty inherent to real-world robotics in unstructured environments.

This thesis addresses planning for long-term robot monitoring of unknown, continuous spatiotemporal environment processes. We use Gaussian processes to capture the dynamics of these environment processes using sparse observational data, adopting an online approach that assumes minimal prior knowledge. We develop planning algorithms that reason about these models in belief space over long horizons to produce effective reward-gathering policies. Our algorithms address practical challenges such as resource constraints and uncertainty over action durations, which are fundamental to long-term robot missions in the real world. We also present an algorithm for safe exploration and mapping in the presence of an unknown environmental hazard, enabling a robot to autonomously and safely map an unknown environment in which no prior map is available. Our empirical results demonstrate that our planning solutions are able to generate effective policies in realistic and challenging long-term robot monitoring scenarios.

Contents

List of Figures	x
List of Definitions	xi
List of Abbreviations	xii
List of Symbols	xiii
1 Introduction	1
1.1 Overview	1
1.2 Challenges	4
1.3 Contributions	5
1.4 Thesis structure	6
1.5 Publications	8
2 Background	9
2.1 Topological maps	9
2.2 Gaussian processes (GPs)	11
2.2.1 Model overview	11
2.2.2 Computational complexity	12
2.3 Markov decision processes (MDPs)	12
2.3.1 Finite horizon MDPs	13
2.3.2 Partially observable MDPs (POMDPs)	14
2.3.3 Partially known state MDPs (PKSMDPs)	14
2.4 Monte Carlo tree search (MCTS)	16
2.4.1 Fundamentals	17
2.4.2 Progressive widening	18
2.5 Modelling stochastic processes	19
2.5.1 Continuous-time Markov chains (CTMCs)	19
2.5.2 Phase-type distributions (PTDs)	19

3	Related Work	21
3.1	GP modelling of continuous environments	22
3.1.1	Modelling natural phenomena	22
3.1.2	Spatiotemporal modelling	22
3.1.3	Alternative techniques	23
3.1.4	Scaling GP methods	23
3.2	Planning in unknown environments	24
3.2.1	Adaptive informative path planning	25
3.2.2	Planning under partial observability	28
3.2.3	Planning with GP beliefs	29
3.2.4	Safe exploration	30
3.2.5	Mapping of unknown environments	32
3.3	Long-term planning	33
3.3.1	Long-term autonomy in robotics	34
3.3.2	Hierarchical planning	36
3.3.3	Persistent monitoring	37
3.4	Planning under resource constraints	39
3.4.1	Constrained path planning	40
3.4.2	Constrained sequential decision making	40
3.4.3	Long-term and persistent problems	41
3.4.4	Chance-constrained planning	43
4	Multi-Mission Monitoring with Gaussian Process Models	45
4.1	Introduction	45
4.2	Problem formulation	47
4.3	MCTS planning with GP beliefs	49
4.4	Algorithm	52
4.4.1	Spatiotemporal modelling with GPs	52
4.4.2	Evaluating trajectory rewards	54
4.4.3	Planning algorithm	55
4.5	Experiments	56
4.5.1	Experimental domains	56
4.5.2	Planning with a spatiotemporal GP kernel	58
4.5.3	Belief-based MCTS across multiple missions	60
4.6	Discussion	64
4.7	Summary	66

5	Hierarchical Planning for Resource-Constrained Long-Term Monitoring	67
5.1	Introduction	68
5.2	Problem formulation	70
5.3	Algorithm	71
5.3.1	Overview	71
5.3.2	High-level planning: Depot-MDP	73
5.3.3	Low-level planning: belief-based MCTS	76
5.4	Experiments	78
5.4.1	Planner configurations	78
5.4.2	Experimental domains	79
5.4.3	Scheduling of active hours	80
5.4.4	High-level planning horizon	82
5.4.5	High-level planning step	84
5.4.6	UAV wildfire monitoring	84
5.5	Discussion	87
5.6	Summary	88
6	Long-Term Monitoring with Stochastic Action Durations	89
6.1	Introduction	89
6.1.1	Problem formulation	91
6.1.2	Navigation dynamics	92
6.1.3	Resource model	93
6.2	Algorithm	94
6.2.1	Overview	94
6.2.2	High-level planning	94
6.2.3	Low-level planning	96
6.2.4	Checking reachability	97
6.3	Experiments	98
6.3.1	Battery charging model	100
6.3.2	Planner horizon	101
6.3.3	Environment stochasticity	103
6.4	Discussion	106
6.5	Summary	107
7	Safe Exploration of Unknown Environments	108
7.1	Introduction	109
7.2	Preliminaries	110
7.2.1	Constrained reachability	110
7.2.2	Estimated MDPs (EstMDPs)	112
7.2.3	SafeEstMDP-Process	114

7.2.4	Occupancy map models of unknown environments	116
7.3	Problem formulation	117
7.4	Algorithm	118
7.4.1	Algorithm description	118
7.4.2	Incremental navigation graph expansion	118
7.4.3	Choice of goal waypoint	120
7.4.4	Goal scoring function	123
7.5	Experiments	124
7.5.1	Safety hazard model	124
7.5.2	Simulated domains	125
7.5.3	Exploration of unknown environments	125
7.5.4	Hardware and system configuration	130
7.5.5	Deployment in a physical environment	131
7.6	Discussion	133
7.7	Summary	134
8	Conclusion	136
8.1	Summary of contributions	136
8.2	Future work	138
	References	140

List of Figures

1.1	Examples of autonomous robots in unstructured environments	2
2.1	Topological map example	10
2.2	Comparison of different GP kernels	12
2.3	MCTS algorithm illustration	17
4.1	Synthetic domain snapshots	57
4.2	Coral reef monitoring domain snapshots	58
4.3	Planner performance with different GP kernels	60
4.4	Coral reef acoustic monitoring results	61
4.5	Performance of different planner configurations	63
5.1	Wildfire monitoring dataset visualisation	69
5.2	Hierarchical planner architecture	73
5.3	UAV wildfire monitoring problem domain	80
5.4	Wildfire domain ground truth GP model snapshots	81
5.5	Flexible scheduling experiment results	83
5.6	Planner horizon experiment results	85
5.7	Planner discretisation step experiment results	86
5.8	UAV wildfire monitoring results	87
6.1	Stochastic action chain model	99
6.2	Piecewise linear battery charging model	101
6.3	Planner performance with different planning horizons	102
6.4	Planner performance with different environment stochasticity levels	105
7.1	SafeEstMDP-Process and SafeEstMDP-Map flow diagram	115
7.2	Navigation graph incremental construction	121
7.3	Progression candidates illustration	122
7.4	Safe exploration simulated domains	126
7.5	Simulated experiment results (reactor room domain)	128
7.6	Simulated experiment results (research mine domain)	129
7.7	SafeEstMDP-Map real-world deployment example run	132

List of Definitions

2.1	Topological map	10
2.2	Gaussian process	11
2.3	Finite horizon MDP	13
2.4	Partially observable MDP (POMDP)	14
2.5	Unknown processes	15
2.6	Partially known state MDP (PKSMDP)	16
2.7	Continuous-time Markov chain (CTMC)	19
2.8	Phase-type distribution (PTD)	20
4.1	Multi-Mission PKSMDP (M^2 -PKSMDP)	48
5.1	Long-Term Monitoring PKSMDP (LTM-PKSMDP)	71
5.2	Depot-MDP	74
6.1	Hybrid action model	93
6.2	Resource model	93
6.3	High-level PKSMDP	95
7.1	Reach-avoid paths	111
7.2	Cost accumulation over paths	111
7.3	Cost-optimal reach-avoid	111
7.4	GP posterior over an interval	112
7.5	Estimated MDP (EstMDP)	113
7.6	Safety function	114
7.7	Volumetric gain	116
7.8	Safe exploration objective	117

List of Abbreviations

AIPP		Adaptive informative path planning.
AUV		Autonomous underwater vehicle.
BAMDP		Bayes-adaptive Markov decision process.
BO		Bayesian optimisation.
CTMC		Continuous-time Markov chain.
EV		Electric vehicle.
GP		Gaussian process.
IPP		Informative path planning.
MCTS		Monte Carlo tree search.
MDP		Markov decision process.
PKSMDP		Partially known state Markov decision process.
POMDP		Partially observable Markov decision process.
PTD		Phase-type distribution.
SoC		State of charge.
UAV		Unmanned air vehicle.
UCB		Upper confidence bound.
UGV		Unmanned ground vehicle.

List of Symbols

Topological maps

D		A set of depot nodes, where $D \subseteq V$.
E		A set of topological map edges.
$\mathcal{G}$		A topological map.
V		A set of topological map nodes.

Markov decision processes

A		An action space.
C		A cost function.
F		A set of forbidden states, where $F \subseteq S$.
G		A set of goal states, where $G \subseteq S$.
H		A planning horizon.
$\mathcal{M}$		A Markov decision process (or variant thereof).
Π		A set of policies.
R		A reward function.
S		A state space.
S_e		A set of unknown states (of a PKSMDP).
S_k		A set of known states (of a PKSMDP).
T		A transition function.

Unknown environment processes

f		An unknown process.
O		The observation space of an unknown process.
X		An environment.
z		A noisy observation.

Gaussian processes

$\mathcal{D}$		A dataset of observations.
$\mathcal{GP}$		A Gaussian process.
k		A kernel function.
m		A mean function.
$p^{\mathcal{GP}}$		The posterior distribution of a Gaussian process.
θ		The hyperparameters of a kernel function.

Monte Carlo tree search

H		A planning horizon.
n		A node in a search tree.
n^B		A belief node in a search tree.
n^{BA}		A belief-action node in a search tree.
N_i		The visit count for node $n_i \in \mathcal{T}$.
Q_i		The value estimate for node $n_i \in \mathcal{T}$.
$\mathcal{T}$		A search tree.

1

Introduction

Contents

1.1 Overview	1
1.2 Challenges	4
1.3 Contributions	5
1.4 Thesis structure	6
1.5 Publications	8

1.1 Overview

The field of robotics has seen immense progress in recent years, producing robots that are more robust, capable, and widely accessible than ever before. This has enabled successful deployments in increasingly complex and challenging domains, from unknown subterranean environments, to forests, and the depths of oceans. Moreover, improved robustness across diverse domains has created new opportunities for *long-term* deployments, in which robots operate for prolonged periods—weeks, months, or years—without the need for human intervention or supervision. However, long-term robot missions present a unique combination of challenges which make it difficult to come up with effective plans. Key challenges include practical constraints on resources such as time and battery life, dynamic and partially observable environments, and



(a) A Clearpath Husky robot navigating the site of an ecological research project.



(b) A Boston Dynamics Spot robot exploring a disused military bunker.

Figure 1.1: Examples of mobile robots operating autonomously in unstructured environments

the high computational demand of long-horizon planning. In order to unlock the full potential of long-term robot deployments, we require planning algorithms that explicitly tackle these challenges.

Long-term monitoring for robots holds significant promise across a range of scientific and industrial applications. In precision agriculture, robots have been used to monitor pasturelands, providing long-term models that help farmers to optimise crop health and manage resources more effectively [1, 2]. Robots have been deployed to conduct long-term visual surveying [3] and mapping [4] of natural environments, where they can provide data of much higher spatial and temporal resolution than human surveyors or satellite imagery, at a significantly lower cost. Autonomous unmanned air vehicles (UAVs) have become increasingly widespread tools for detection, prevention and monitoring of wildfires [5, 6], providing unparalleled situational awareness which has proved particularly valuable in remote areas [7]. Scientists in many areas of earth and life sciences have also realised the potential of robots for large-scale and long-term data gathering. Autonomous robots have been used extensively by oceanographers to study

marine environments, as well as by geologists for monitoring of volcanic activity [8], and by meteorologists studying the atmosphere to learn about microclimates [9]. As these technologies mature, we can expect that the scope of viable applications will only continue to grow.

In this thesis, we address **planning under uncertainty for long-term robot monitoring of continuous spatiotemporal processes in unstructured environments**. Such processes might include bioactivity in coral reefs, or the distribution of wildfire hotspots—in each case, features that can be observed locally by robots using on-board sensors. Planning involves generating *policies* that specify actions for the robot to take in order to best serve its mission objectives, given the available information to it at each decision point. The planning problem for our long-term monitoring setting comes with several distinct yet interrelated challenges. One such challenge is that of modelling the processes of interest using the available data. The observational data produced by a robot in this setting will be sparse and often noisy. The environment processes of interest, on the other hand, are often complex and dynamic, making them difficult to capture accurately with limited data. Even the topology of the environment is often dynamic over the long term from a robot planning perspective. Obstacles might shift over time, or certain routes might be unsafe to traverse in adverse weather conditions. In some cases, the layout of the environment might not be known *a priori* at all, requiring the robot to build a map of its surroundings online—a task which has been the subject of considerable research.

Mobile robots are also inherently resource-constrained, primarily by battery life, which limits the duration for which they can remain continuously active. Long-term operation without any human intervention requires robots to manage their own resources in order to avoid resource depletion, whilst continuing to service their mission objectives. This presents technological challenges, in terms of developing reliable and cost-effective infrastructure for autonomous recharging, as well as algorithmic challenges in terms of incorporating these resource constraints into long-term planning. However, where independent resource management *is* possible, it can unlock the ability for a single

robot to monitor much larger environments—provided, of course, that the required infrastructure (e.g. battery charging stations) is present.

Planning over long horizons is challenging in general, and particularly so in the dynamic and highly uncertain environments on which we focus throughout this thesis. These environments tend to produce complex interactions with the robots that operate within them, resulting in further uncertainty over the outcomes of robot actions. Varied terrain might make it difficult to predict the time required to navigate between locations, for example, or the robot’s battery might deplete at a rate that varies significantly with the ambient temperature. These phenomena are often too complex to capture explicitly in deterministic models, so must instead be represented through uncertain, probabilistic models. Model uncertainties accumulate over the planning horizon, making it difficult to achieve reliable decision making. However, in missions set in large environments or over long timescales, long-horizon planning is *essential* to ensure good performance, since short-horizon planners may be unable to consider the full scale of the environment or the environmental process of interest.

In this thesis, we develop planning algorithms for long-term robot monitoring missions in natural environments. Existing works have addressed the challenges of this setting in different ways, but often in isolation, thereby avoiding the compounding effect of the interactions between them. We therefore aim to address the challenges of long-term robot monitoring in a more unified manner, developing algorithmic solutions that are modular and applicable to a wide variety of practical problems.

1.2 Challenges

This thesis aims to develop planning solutions that address the practical challenges of long-term robot monitoring missions. The difficulty of planning over long horizons is compounded by the contextual challenges faced by robots in the field. Each of these challenges has been addressed on some level by previous works, each with their own particular formulations and assumptions. Our goal is to demonstrate planning solutions that can tackle these challenges in combination, making them applicable to a wider range of practical problems. The challenges we focus on in this thesis are:

- C1 Environment Modelling:** The features of interest in natural environments come with rich and complex behaviours, varying both spatially and temporally. Effective monitoring requires flexible modelling solutions which can capture this complexity from the sparse, noisy, local observations produced by on-board sensors.
- C2 Resource Constraints:** Robots operating for extended periods are inevitably limited by their finite resources. Integrating resource management into planning adds complexity to the planning problem, while also unlocking a new layer of intelligent behaviour.
- C3 Stochastic Interactions:** Robots interact with natural environments in complex ways, and the outcomes of these interactions are often impossible to model deterministically. Instead, the uncertainty itself must be incorporated into planning, in such a way that produces robust and effective policies.
- C4 Limited Priors:** Prior knowledge can be an invaluable asset for planning, but in practice it is often expensive or impractical to acquire in unstructured environments. Maximising the value of the limited prior knowledge that is available leads to systems that can operate effectively across more diverse application domains.

1.3 Contributions

The core contributions of this thesis to addressing the challenges outlined in Section 1.2 are as follows:

- A planning algorithm for long-term monitoring of a continuous spatiotemporal process across *repeated* missions. This addresses **C1 Environment Modelling** by using a GP model with hyperparameters that are learned online, demonstrating that this model functions sufficiently well with sparse data to enable effective planning in this setting. To a lesser degree, it also addresses **C4 Limited Priors** by incorporating online learning of GP hyperparameters to reduce reliance on prior domain knowledge of the environment process.

- A novel batch evaluation method for GP-based rewards during MCTS rollouts, which improves computational performance by reducing the number of GP evaluations required. This method is used in Chapters 4–6 to make planning over a GP model (**C1 Environment Modelling**) computationally tractable.
- A hierarchical planning framework for long-term monitoring in resource-constrained settings (**C2 Resource Constraints**). With this new framework, we demonstrate the benefits of integrating resource management into long-term planning, showing that the flexibility of doing so can lead to better outcomes.
- An extension of the above hierarchical planner to consider uncertainty over action outcomes, thereby additionally addressing **C3 Stochastic Interactions**. We adopt a chance-constrained approach to resource management, a novel method in the context of long-term robot monitoring.
- A planning algorithm for safe exploration and mapping of unknown environments with no prior map (**C4 Limited Priors**). Although this contribution does not directly address long-term monitoring problems, its algorithmic framework has many similar elements, and offers a pathway to extend the above contributions to environments in which a prior map is not available.

1.4 Thesis structure

In this thesis, we develop planning algorithms that address the long-term robot monitoring challenges outlined in the previous section. Chapter 2 summarises the relevant background knowledge for this thesis, introducing models and techniques that are used throughout. Chapter 3 provides an overview of the relevant literature. Additional background material and literature are also presented within the later chapters as needed. In Chapter 4, we formulate the long-term robot monitoring problem as a series of repeated deployments, and present a planning approach to solve it. We use GPs to model continuous spatiotemporal environment processes, with a kernel function geared specifically towards the monitoring problems that motivate the work in this thesis. We

also incorporate online learning of GP hyperparameters to reduce reliance on domain knowledge, and demonstrate that our GP model can be used effectively for planning in this setting. Finally, we introduce a batch evaluation method for GP-based rewards in MCTS, enhancing the computational efficiency and overall performance of our planner. The work in this chapter is published in the proceedings of IROS 2024, the 37th IEEE/RSJ International Conference on Intelligent Robots and Systems [10].

In Chapter 5, we consider long-term monitoring in a resource-constrained setting, developing a hierarchical planning framework that integrates resource management directly into the planning process. We frame the problem using depots, pre-determined locations within the environment at which the robot can replenish its resources. We demonstrate the benefits of our planning approach, which allows the robot to decide when and where to replenish its resources in order to best serve its mission objective. The hierarchical approach also enables more effective long-term planning, by decoupling responsibilities for short-term action selection and longer-term, high-level mission planning between two levels of the hierarchy. The work in this chapter is published in the proceedings of ECAI 2024, the 27th European Conference on Artificial Intelligence [11].

Chapter 6 extends the work from the previous chapter to consider uncertainty over action outcomes. The durations of navigation actions, and the outcomes of resource replenishment actions, are modelled as random variables with known distributions. We then adopt a chance-constrained approach to resource management, enforcing constraints on the probability of the resource level dropping below a specified minimum level. This adds significant complexity to the planning problem, but we demonstrate that we are able to obtain comparable performance (measured by cumulative reward) to a deterministic planning baseline, while limiting the number of resource constraint violations.

Chapter 7 considers the problem of safe exploration and mapping of unknown environments. We extend existing work on safe exploration of an unknown hazard in a pre-mapped environment, developing an algorithm that instead incrementally constructs the navigation graph during exploration. This work bridges the gap between the long-term monitoring and environments in which no prior map is available, enabling

a robot to safely and independently construct a topological map which can subsequently form the basis of a monitoring mission. The problem formulated in this chapter shares many commonalities with the long-term monitoring work in the previous chapters, considering GP models of continuous features and planning over a graph-based environment representation. The work in this chapter is published in the journal *Autonomous Robots* [12].

Chapter 8 concludes the thesis and discusses potential avenues for future work.

1.5 Publications

The research presented in this thesis is published as follows:

- A. Stephens, B. Lacerda, and N. Hawes, “Planning for long-term monitoring missions in time-varying environments,” in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2024.
- A. Stephens, B. Lacerda, and N. Hawes, “Hierarchical planning for resource-constrained long-term monitoring missions in time-varying environments,” in *European Conference on Artificial Intelligence (ECAI)*, IOS Press, 2024.
- A. Stephens, M. Budd, M. Staniaszek, B. Casseau, P. Duckworth, M. Fallon, N. Hawes, and B. Lacerda, “Planning under uncertainty for safe robot exploration using Gaussian process prediction,” *Autonomous Robots*, vol. 48, no. 7, pp. 1–29, 2024.

2

Background

Contents

2.1	Topological maps	9
2.2	Gaussian processes (GPs)	11
2.2.1	Model overview	11
2.2.2	Computational complexity	12
2.3	Markov decision processes (MDPs)	12
2.3.1	Finite horizon MDPs	13
2.3.2	Partially observable MDPs (POMDPs)	14
2.3.3	Partially known state MDPs (PKSMDPs)	14
2.4	Monte Carlo tree search (MCTS)	16
2.4.1	Fundamentals	17
2.4.2	Progressive widening	18
2.5	Modelling stochastic processes	19
2.5.1	Continuous-time Markov chains (CTMCs)	19
2.5.2	Phase-type distributions (PTDs)	19

In this chapter, we introduce the key concepts in environment representation, modelling, and planning which underpin the work in this thesis.

2.1 Topological maps

This thesis addresses high-level decision making for applications in robotics. To support effective planning, we represent the environments occupied by robots as *topological maps* [13]. Topological maps represent environments compactly, as a discrete set of

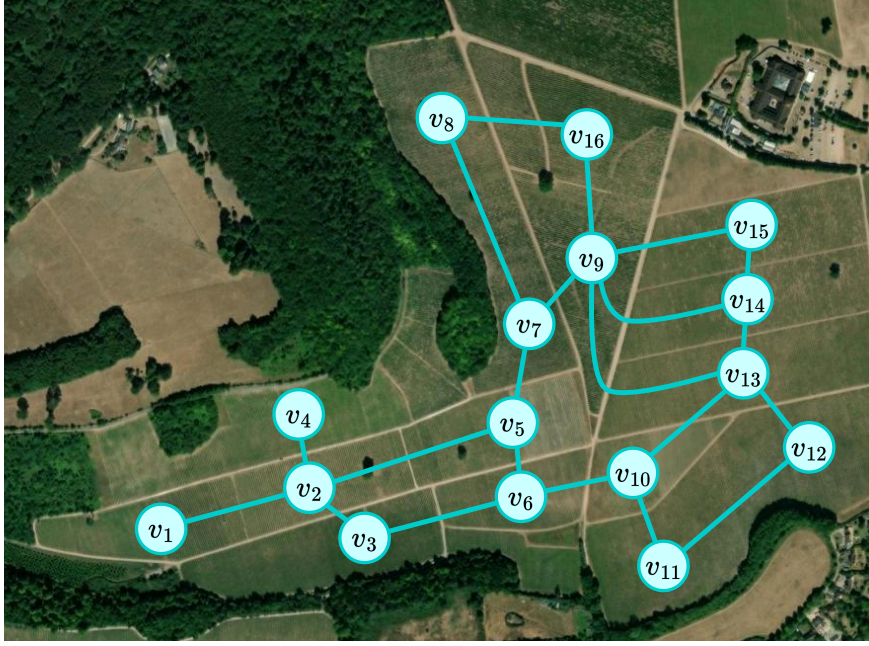


Figure 2.1: An example of a topological map of a vineyard in Surrey, UK. Satellite imagery from [14].

locations which are relevant to the planning problem at hand. We plan over topological maps, generating *actions* which can then be passed to lower-level subsystems to manage navigation and control during execution.

Definition 2.1 (Topological map) A topological map is a tuple $\mathcal{G} = \langle V, E \rangle$, where V is a finite set of nodes (also known as waypoints, or vertices) representing locations of interest in the environment, and $E \subseteq V \times V$ is a set of directed edges representing links between locations that can be traversed by a robot.

A key component of the work in this thesis is *spatiotemporal modelling*, which involves modelling a feature of the robot’s environment as a continuous spatiotemporal process. In particular, we employ Gaussian processes for this, which are discussed in Section 2.2. To support this, we explicitly associate each topological node $v \in V$ with a location in the robot’s physical environment, i.e. $v = (x, y) \in \mathbb{R}^2$. This allows our planners to consider spatial relationships between the locations represented by topological nodes. For example, if two nodes are close together, then we might expect that the observed values of environment features at those two nodes will be similar.

Figure 2.1 shows an example of a topological map of a vineyard, such as might be used for agricultural monitoring task. In this example, nodes are placed at key locations across the different plots of land that comprise the vineyard. These nodes are linked by edges, each of which indicates a link between the corresponding locations which can be traversed by a robot. In this case, the edges are bidirectional, meaning that they can be traversed in either direction.

2.2 Gaussian processes (GPs)

2.2.1 Model overview

A Gaussian process is a statistical model which defines a distribution over functions. GPs have received significant attention in the machine learning literature, as they provide a flexible and non-parametric framework for modelling complex, unknown distributions from data [15]. Importantly for the work in this thesis, GPs provide *probabilistic* predictions—that is, their outputs are probability distributions rather than point estimates. This allows for explicit reasoning about the uncertainty of the model, which is crucial in robotics applications such as ours where the environment is only partially observable.

Definition 2.2 (Gaussian process) A Gaussian process $\mathcal{GP}$ is a collection of random variables, of which any finite subset are jointly Gaussian distributed. The GP is fully specified by its mean function $m(s)$ and its kernel function $k(s, s')$, giving a distribution over functions $o(s) \sim \mathcal{GP}(m(s), k(s, s'))$. We can let $m(s) = 0$ without loss of generality. Given a dataset of $n_{\mathcal{D}}$ noisy observations $\mathcal{D} = \{(s_i, o(s_i) + \epsilon_i)\}_{i=1}^{n_{\mathcal{D}}}$ for states s_i , where $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$ is Gaussian observation noise, GP regression predicts unknown environment feature values at inputs s_* based on visited states $s_{\mathcal{D}}$. The kernel function k is parameterised by hyperparameters θ .

The GP kernel function captures information about the structure of the process f being modelled by the GP. Given hyperparameter priors $p_0(\theta)$, the values of θ can be learned by maximising the log marginal likelihood of the model given the dataset. Alternatively, the values of θ can simply be set manually. The choice of kernel function is

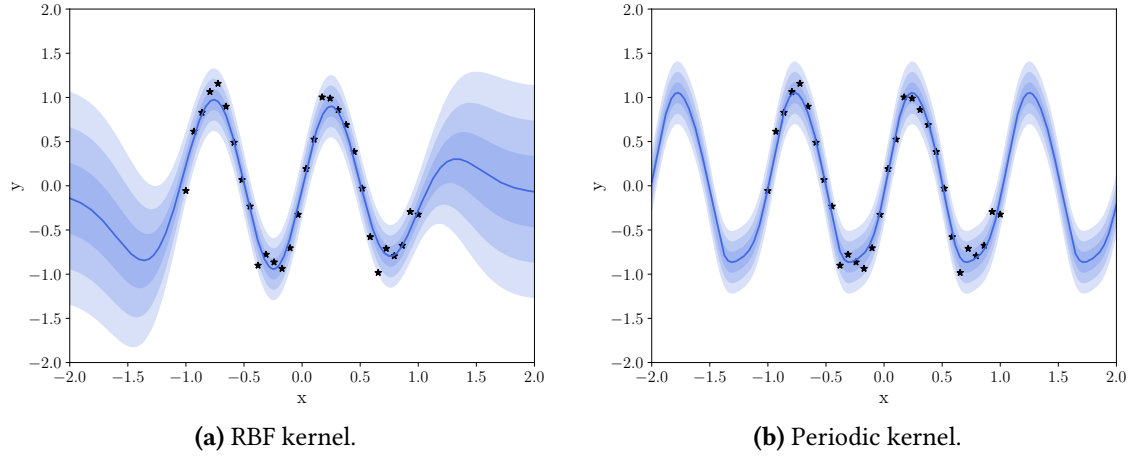


Figure 2.2: GPs with different kernels fit to the same data. The curves show the GP posterior mean, and the shaded regions show 1–3 standard deviations of divergence from the mean.

a key design decision in GP applications, as it can be used to encode domain knowledge about f , which can in turn dramatically improve model accuracy. To illustrate this, Figure 2.2 compares the predictions of GPs with two different kernels fit to the same dataset, which is sampled from an underlying periodic function. The first uses a radial basis function (RBF) kernel, which simply treats nearby states in the input space as similar. The second uses a periodic kernel, which uses a periodic distance function to reflect periodicity expected in the data. We see that if the kernel function is well-chosen, it can allow the GP predictions to generalise better across the input space, thereby producing more informative predictions.

2.2.2 Computational complexity

One of the primary limitations of GP models is their computational complexity. For a GP with n data points, the size of the kernel matrix is $n \times n$. Obtaining the predictive distribution at a new input point requires inverting this matrix, and as such exact GP inference has naive time complexity of $O(n^3)$ and storage demand $O(n^2)$ [16]. Techniques exist to speed up GP inference under certain conditions, which we discuss in Section 3.1.

2.3 Markov decision processes (MDPs)

The MDP [17] is a mathematical framework for modelling decision making under uncertainty. MDPs capture sequential interactions between a decision making agent

and its environment, where the agent selects actions which in turn affect its state within that environment, in order to maximise some notion of reward. MDPs are ubiquitous in robotics, and have been used in the literature to model a wide range of planning problems, including safe exploration [18, 19, 20], shared human-robot autonomy [21, 22, 23], informative path planning [24, 25, 26], and multi-robot coordination [27, 28, 29]. Throughout this thesis, MDPs will be used as the underlying model of robots as decision making agents, and the environments in which they operate. The following sections outline the key concepts and notation of MDPs, focusing on particular variants which are relevant to the work in this thesis.

2.3.1 Finite horizon MDPs

The finite horizon MDP is the basic model that forms the foundation of other, more specialised MDP models used throughout this thesis. Variations on this model will be introduced in later chapters as needed.

Definition 2.3 (Finite horizon MDP) A finite horizon MDP is a tuple $\mathcal{M} = \langle S, s_0, A, T, R, H \rangle$, where S is a finite set of states, $s_0 \in S$ is the initial state; A is a finite set of actions; $T : S \times A \times S \rightarrow [0, 1]$ is the transition function; $R : S \times A \rightarrow \mathbb{R}$ is the reward function; and $H \in \mathbb{N}^+$ is the time horizon.

In general, an agent's behaviour in an MDP is defined by a *policy*. A deterministic policy is a mapping $\pi : S \times \{0, \dots, H\} \rightarrow A$, where $\pi(s_j, j) = a$ specifies the action a to be executed at timestep j from state s_j . We denote the set of all such policies by Π . The standard MDP planning objective is to find a policy π^* which maximises the expected sum of rewards up to the planning horizon H :

$$\pi^* = \arg \max_{\pi} \mathbb{E}^{\pi} \left[\sum_{j=0}^{H-1} R(s_j, \pi(s_j, j)) \right]. \quad (2.1)$$

MDPs and their variants can equivalently be formulated with a *cost function* $C : S \times A \rightarrow \mathbb{R}$ in the place of the reward function. In this case, the planning objective is to find a cost-minimising policy.

2.3.2 Partially observable MDPs (POMDPs)

POMDPs [30] extend the standard MDP framework to the case where the agent does not have full observability over its environment. In robotics, partial observability often arises from robots' imperfect sensing capabilities, which prevent them from knowing with certainty their own location or pose within the environment. This leads to the problem of *state estimation*, which is a key area of robotics research [31, 32]. In this thesis, however, we assume that robots have perfect observability over their *internal* state (e.g. position, clock time, battery level), while the state of the environment can be observed only through noisy local measurements.

Definition 2.4 (Partially observable MDP (POMDP)) A POMDP is a tuple $\mathcal{M} = \langle S, s_0, A, T, R, O, Z, H \rangle$, where S is a finite set of states; s_0 is the initial state; A is a finite set of actions; $T : S \times A \times S \rightarrow [0, 1]$ is the transition function; $R : S \times A \rightarrow \mathbb{R}$ is the reward function; O is the observation space; $Z : S \times A \times S \rightarrow [0, 1]$ is the observation model; and $H \in \mathbb{R}^+$ is the time horizon.

Compared with the standard MDP, the POMDP additionally includes the observation space O and the observation model Z . After executing an action, the agent receives an observation $o \in O$ according to the observation model Z . Since the agent no longer has direct access to the true state of the environment, it must instead maintain a *belief* over this state based on the observations received. Thus, unlike in the MDP, where a policy can map directly from state to action, POMDP policies map from beliefs (or equivalently, histories) to actions.

While the work in this thesis does not explicitly formulate planning problems as POMDPs, we do consider models with partial observability. As such, the algorithms we develop are informed significantly by the literature on POMDP planning, which we discuss in Section 3.2.

2.3.3 Partially known state MDPs (PKSMDPs)

This thesis addresses robot planning problems which are concerned with *environment processes* that are unknown *a priori*. We consider robots operating within bounded

environments $X \subset \mathbb{R}^n$, within which they are able to navigate and take local observations of these processes using on-board sensors. These processes represent environment phenomena which vary spatially throughout the environment, and may also vary temporally, such as radiation levels in a nuclear inspection scenario, or acoustic bioactivity markers in coral reef monitoring. The data from local observations can be used to build a model of the environmental processes, which in turn enables inferences about unvisited states for the purpose of planning.

Definition 2.5 (Unknown processes) Unknown processes in an environment X are defined as a function $f : X \rightarrow O$, where $O = \mathbb{R}^m$ is the observation space of f , with m being the number of modelled unknown processes. A robot can observe noisy sensor measurement $z : X \rightarrow O$ of the form:

$$z(x) = f(x) + \epsilon, \quad (2.2)$$

where $\epsilon = [\epsilon_1, \dots, \epsilon_m]$ is an m -dimensional vector of Gaussian observation noise, i.e. $\epsilon_j \sim \mathcal{N}(0, \sigma_j^2)$ for all $j \in \{1, \dots, m\}$.

Throughout this thesis, we will address the case where $m = 1$, i.e. that of a single unknown process. However, the methods proposed can generalise fairly straightforwardly to more than one unknown process. Multiple correlated processes could be modelled using a multi-output GP [33], or independent processes could be modelled simply using multiple single-output GPs. The former assumes non-independent process dynamics, where learning about one process could improve predictions of another. The reward function could then be reformulated as a scalar function of multiple variables, in order to retain a single-objective problem—we do not address multi-objective planning in this thesis.

We also consider some components of the state, such as location and time, to be known. In order to clearly separate the known robot dynamics from the unknown behaviour of the environment, we represent the system as a partially known state MDP (PKSMDP), a model introduced by Budd *et al.* [20, 34] and formalised by Stephens *et al.* [12].

Definition 2.6 (Partially known state MDP (PKSMDP)) A PKSMDP is a tuple $\mathcal{M} = \langle S, s_0, A, T, R, H \rangle$, where $S = S_k \times S_e$ is a factored state space consisting of a known component S_k and an unknown component S_e ; $T : (S_k \times S_e) \times A \times S_k \rightarrow [0, 1]$ is the transition function; and s_0, A, R , and H are as in a standard MDP (Definition 2.3).

The definition of a PKSMDP is similar to that of an MDP, with the exception of the factored state space. In this thesis, the known component S_k is used to capture the robot’s internal state (e.g. its location or resource level), while the unknown component S_e represents an environment process, which is defined by an underlying (unknown) function $f : S_k \rightarrow S_e$. We assume f to be continuous. Because f is unknown but the value of $f(s_k)$ is uniquely defined, the transition function of the PKSMDP is not a complete transition function, in the sense that it only maps to known state components. $T((s_k, s_e), a, s'_k)$ is the probability of moving to s'_k given that a was executed in state s_k and the value of the environment process was s_e . This formulation allows for transition dynamics that depend on the unknown state component S_e , such as in the case of an underwater robot operating in the presence of unknown currents. Furthermore, the reward function R depends on the full state S , and thus considers the unknown state component.

We can also consider PKSMDPs with deterministic transitions, where we instead have $T : (S_k \times S_e) \times A \rightarrow S_k$. In Chapters 4 and 5 we make use of this deterministic formulation, before extending to a stochastic setting in Chapter 6.

2.4 Monte Carlo tree search (MCTS)

MDPs with finite state and action spaces can in principle be solved optimally using dynamic programming methods [35]. However, optimal solution techniques do not scale to large problems or continuous state spaces, so in practice heuristics or sampling-based methods are often used. In this thesis, we use MCTS [36] to plan in belief space over long horizons.

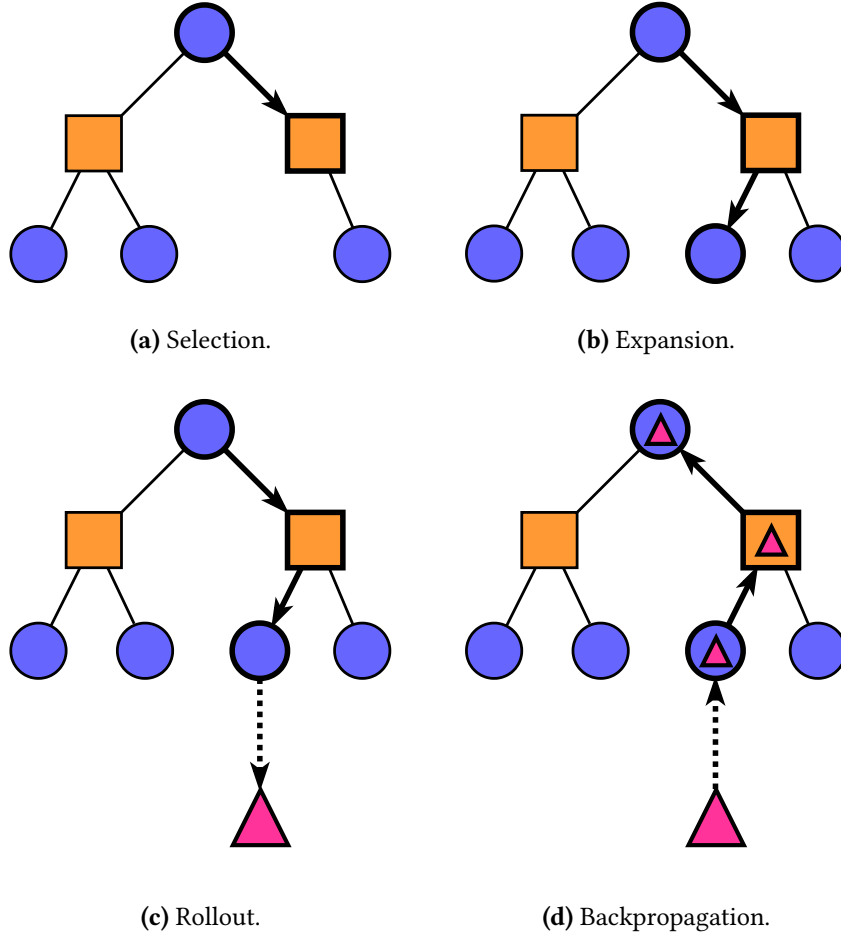


Figure 2.3: An illustration of the four phases of an MCTS trial. Blue circles are decision nodes, orange squares are chance nodes.

2.4.1 Fundamentals

MCTS methods incrementally build a search tree $\mathcal{T}$ over a series of Monte Carlo simulations, which we refer to as *trials*. In non-deterministic settings, the tree is composed alternately of two types of nodes: *decision nodes* $n(s)$, which represent states, and *chance nodes* $n(s, a)$, which represent action choices with stochastic outcomes. Each node stores a count $N(s)$ or $N(s, a)$ of the number of times it has been visited during simulation, as well as a state-value estimate $Q(s)$ for decision nodes and an action-value estimate $Q(s, a)$ for chance nodes. The tree is initialised with a single decision node, the root node, representing the current state of the system. Each trial then consists of four stages: *selection*, *expansion*, *simulation*, and *backpropagation*. The algorithm summarised visually in Figure 2.3.

In the selection phase, actions are chosen according to a *search policy*, and states are sampled from the transition model to traverse down the tree until a state is reached which is not in $\mathcal{T}$. We use the Polynomial Upper Confidence Tree (PUCT) policy [37] as our search policy, which balances the average value of an action across previous trials with an exploration term that favours less-simulated actions:

$$Q_{\text{PUCT}}(s, a) = \hat{Q}(s, a) + C \sqrt{\frac{N(s)^{e_d}}{N(s, a)}}, \quad (2.3)$$

where $\hat{Q}(s, a)$ is the average reward obtained by choosing action a from state s in previous trials; $N(s)$ is the number of times that node s has been visited; $N(s, a)$ is the number of times that action a has been selected from node s ; C is a constant that affects the exploration-exploitation balance; and e_d is a depth-dependent parameter as per Auger *et al.* [37]. This approach guides exploration during MCTS towards more promising actions, resulting in fewer wasted simulations and thus more efficient planning.

A new node is then added to $\mathcal{T}$ (expansion). If this node is not a terminal state, its value is estimated by simulating forwards up to a horizon H (or, until a terminal state is reached) using a *rollout policy* (simulation). In many MCTS implementations, a random rollout policy is used, which simply samples actions uniformly at random. The value of the new node estimated by the rollout is then backpropagated up the tree, updating the statistics of all nodes visited during the trial. Finally, once trials are completed—either after a fixed number of iterations, or when a time limit is reached—an action is selected based on the statistics of the root node.

2.4.2 Progressive widening

In problems involving continuous state, action, or observation spaces, the standard MCTS algorithm will produce a shallow tree of only one layer, since the outcome of any action will be unique and so the simulation step will never visit the same node twice [38]. The *progressive widening* approach from Couëtoux *et al.* [39] solves this problem by limiting the number of children of each node based on their visit counts, thereby ensuring that the tree remains balanced.

In this thesis, we consider discrete action spaces, but continuous state and observation spaces. As such, actions are guaranteed to have unique outcomes each time they are executed, so we use progressive widening during the MCTS trials to limit the width of the tree. We parameterise our progressive widening approach by a depth-dependent parameter α_d as per Auger *et al.* [37]. During the selection phase at node $n(s, a)$, if $\lfloor N(s, a)^{\alpha_d} \rfloor = \lfloor (N(s, a) - 1)^{\alpha_d} \rfloor$, then the successor state s' is randomly sampled from the existing children of $n(s, a)$; otherwise, a new node is generated by sampling from the relevant transition and observation models.

2.5 Modelling stochastic processes

Temporal uncertainty is an important consideration for planning in many robotics applications. In Chapter 6, we tackle the challenge of managing uncertainty over action durations in the context of planning for long-term robot monitoring missions. Here, we introduce some of the tools used to model stochastic action durations.

2.5.1 Continuous-time Markov chains (CTMCs)

CTMCs [40] are stochastic processes in which continuous-time transitions between states are governed by exponential distributions.

Definition 2.7 (Continuous-time Markov chain (CTMC)) A CTMC is a tuple $\mathcal{Q} = \langle S, \text{init}, \delta_E \rangle$, where S is a finite set of states, $\text{init} : S \rightarrow [0, 1]$ is the initial state distribution, $\delta_E : S \times S \rightarrow \mathbb{R}_{\geq 0}$ is the rate transition function.

For a given pair of states s_i and s_j , $\delta_E(s_i, s_j)$ is the rate parameter for the exponential distribution describing the transition between those states. The probability that the transition takes place within time t is then given by $1 - e^{-\delta_E(s_i, s_j)t}$.

2.5.2 Phase-type distributions (PTDs)

Phase-type distributions describe the time taken for a CTMC to reach an absorbing state.

Definition 2.8 (Phase-type distribution (PTD)) A phase-type distribution is a tuple $\mathcal{P} = \langle S, \text{init}, \delta_E, s^a \rangle$, where S , init , and δ_E are as in a CTMC, and $s^a \in S$ is the single absorbing state, with $\text{init}(s^a) = 0$.

We refer to the non-absorbing states in the PTD as *transient states*. The PTD can be represented in matrix form $\mathcal{P} = \langle \boldsymbol{\alpha}, T \rangle$, where $\boldsymbol{\alpha}$ is a vector containing the initial state probabilities, and T is a transition matrix between transient states, given by

$$T = \begin{cases} \delta(s, s') & \text{if } s \neq s' \text{ and } s, s' \neq s^a \\ \sum_{s'' \neq s} \delta(s, s'') & \text{if } s = s' \text{ and } s \neq s^a. \end{cases} \quad (2.4)$$

The probability density function and cumulative density function for a continuous PTD are given respectively by:

$$f(x) = \boldsymbol{\alpha} e^{T x} \mathbf{r}, \quad F(x) = 1 - \boldsymbol{\alpha} e^{T x} \mathbf{1}, \quad (2.5)$$

where $\mathbf{1}$ is a vector of ones, and $\mathbf{r} = -T\mathbf{1}$ captures the transition rates to the absorbing state s^a . In Chapter 6, we use PTDs to model the time taken for a robot to complete sequences of actions with stochastic durations.

3

Related Work

Contents

3.1	GP modelling of continuous environments	22
3.1.1	Modelling natural phenomena	22
3.1.2	Spatiotemporal modelling	22
3.1.3	Alternative techniques	23
3.1.4	Scaling GP methods	23
3.2	Planning in unknown environments	24
3.2.1	Adaptive informative path planning	25
3.2.2	Planning under partial observability	28
3.2.3	Planning with GP beliefs	29
3.2.4	Safe exploration	30
3.2.5	Mapping of unknown environments	32
3.3	Long-term planning	33
3.3.1	Long-term autonomy in robotics	34
3.3.2	Hierarchical planning	36
3.3.3	Persistent monitoring	37
3.4	Planning under resource constraints	39
3.4.1	Constrained path planning	40
3.4.2	Constrained sequential decision making	40
3.4.3	Long-term and persistent problems	41
3.4.4	Chance-constrained planning	43

In this chapter, we discuss existing models and techniques relating to long-term monitoring in unknown environments. We first summarise the relevant literature on Gaussian processes, the primary tool used to model continuous environments

throughout this thesis. We then consider techniques for decision making in unknown environments in general, before focusing specifically on long-horizon and resource-constrained problems.

3.1 GP modelling of continuous environments

3.1.1 Modelling natural phenomena

In recent years, GPs have become a popular tool for modelling in a breadth of applications, as they provide a flexible framework for capturing complex statistical distributions based on local observations [15]. GPs have been demonstrated as effective tools for modelling a wide variety of natural phenomena, including species distributions [41, 42, 43], biodiversity [44], dissolved oxygen levels in water [45], and ocean currents [46].

Golding and Purse [41] explore the power of GPs in the context of environment modelling, showing that they outperform a variety of other statistical methods for species distribution modelling, in both predictive accuracy and computational complexity. They incorporate ecological knowledge into their GP models through Bayesian priors over the mean function and the kernel hyperparameters, demonstrating that doing so can improve model accuracy in cases where limited data is available. More recent work in Ingram *et al.* [42] considers joint modelling of multiple species using *multi-output* GPs, exploiting correlations between species to improve models of rarer species for which less data is available. In this thesis, we use GPs to model acoustic bioactivity in coral reefs (Chapter 4), wildfire hotspots (Chapter 5), and hazardous gamma radiation (Chapter 7).

3.1.2 Spatiotemporal modelling

In the context of robot monitoring of dynamic environments, we are particularly interested in models that are able to capture *temporal* variations in features. This unlocks the capability to plan future actions based explicitly on estimates of *future* states, rather than simply the current state. Duan *et al.* [47] use a hierarchical GP model to represent medical data at multiple levels, introducing a shared GP that captures correlations between two GP models which would otherwise be independent. By doing

so, they are able to make more efficient use of clinical data, improving outcomes in predicting the progression of diseases. The ability to forecast accurately from limited data is extremely valuable in robotic monitoring, since it enables robots to plan ahead to be at particular locations when the most valuable measurements are expected.

GPs have also been used extensively in the planning literature, which we discuss in Section 3.2.3.

3.1.3 Alternative techniques

Several alternatives to GPs have also been used to model environment features based on noisy data. A common modelling approach in the ecological field is the generalised additive model (GAM) [48]. Although GAMs are relatively flexible, they tend to struggle with higher-dimensional data, requiring prohibitively large sample datasets in order to obtain an accurate fit [41]. This is a significant limitation in the context of robotics applications, where data-gathering is often costly, and it is therefore beneficial for models to perform well with less data. By contrast, GPs provide a favourable trade-off between capturing complex interactions between features, while also fitting smooth and plausible predictive surfaces using relatively small amounts of sample data.

Alternatively, graphical models such as Bayesian networks can model probabilistic relationships between multiple variables [49], and have been used successfully in a variety of similar environment monitoring applications [50, 51]. However, these are somewhat less suited to managing continuous data, and the discretisation process introduces additional challenges. These factors, combined with the ubiquity of GP models in the literature, motivate our choice of GPs as the primary modelling tool to address **C1 Environment Modelling**.

3.1.4 Scaling GP methods

The computational demand of GPs (see Section 2.2.2) is one of the key limitations to their use in large-scale problems. Our work considers GP models of unknown environment processes, which are updated online based on observations. In long-term or persistent monitoring missions, the GP dataset can grow without bound, ultimately

making inference operations intractable. The work in this thesis does not explicitly address bounding the size of the GP dataset, as we limit ourselves to experiments where GP inference with the full dataset remains tractable. However, there are several existing techniques that could be integrated with our work in order to scale it to larger problems—we provide a brief overview of some such techniques below.

The main class of techniques for scaling of GPs is known as *sparse approximations*. The idea of these methods is to approximate the GP posterior for a training set of n , using a small set of m *inducing variables*. This reduces the time complexity of inference from $O(n^3)$ to $O(nm^2)$ [52]. Many different sparse approximation methods have been developed, along with different methods for selecting the inducing points, each of which makes various assumptions about the form of the underlying data [53]. More recent works have also incorporated techniques from deep learning to further improve the capability of GPs [54, 55, 56, 57, 58, 59].

In this thesis, we use GPs as real-time models for data acquired through the on-board sensors of mobile robots. Osborne *et al.* [33] present several useful optimisations for this setting in their work with GPs for real-time information processing. First, they show that incoming data can be processed efficiently in an incremental manner, reducing the cost of model updates from $O(n^3)$ to $O(n^2)$. Second, they note that accurate inference typically does not require the full observation history, and introduce a scheme for identifying and discarding stale (i.e. less informative) data points. This provides a principled way of windowing the GP dataset, ensuring that its size remains bounded over long or indefinite horizons.

3.2 Planning in unknown environments

Objectives of exploration, information gathering, and learning in unknown environments have been the subject of many works in the planning literature. In this section, we review some of the problem formulations and solution techniques that have been used to enable effective planning in such environments.

3.2.1 Adaptive informative path planning

The adaptive informative path planning (AIPP) problem involves planning a trajectory that maximises the information acquired in an unknown environment, subject to resource constraints on the path [60]. AIPP is a core problem in robotics, arising across a breadth of applications including exploration, monitoring, inspection, and search and rescue [24, 60, 61, 62, 63]. The work on long-term monitoring in this thesis does not explicitly use an information-based goal, instead adopting an objective of cumulative reward maximisation in an unknown, dynamic environment. However, information gain is still an important consideration in the design of our planning approach, as we recognise that improving model accuracy can lead to better outcomes across the lifetime of the mission. As such, we make use of techniques from several areas of the AIPP literature, which we provide an overview of in this section.

The *adaptive* variant of the informative path planning (IPP) problem refers to planning *online*, reacting to incoming observations and adapting the plan accordingly. Conventional (i.e. offline) IPP algorithms typically make strong assumptions about the environment, relying on a stationary environment with a known information distribution [64]. These assumptions are extremely limiting in robotics, where domains are often characterised by high uncertainty and dynamic elements that cannot easily be modelled *a priori*. In Chapter 7, online replanning is an essential component of our safe exploration approach, allowing the robot to adapt to unexpected obstacles or failed traversal of difficult terrain. Furthermore, offline methods tend to scale poorly to large environments, and do not generalise well across diverse environments and problems [60, 65].

Environment representations

The information-gathering objective in AIPP has been formulated in many different ways, depending on the environment representation and specifics of the task at hand. In their recent review of learning-based methods for AIPP in robotics, Popović *et al.* [60] introduce a taxonomy of AIPP methods which divides them into four environment

representation categories: occupancy grids, GPs, graph-based, and neural representations. The choice of environment representation depends heavily on the specifics of the problem—for example, neural radiance fields [66] and variants thereof (e.g. [67]) are popular tools in mapping and 3D reconstruction. In the case of information-gathering problems over continuous features f , GPs are a natural choice, and have shown success in many applications including mapping temperature distributions [68], light [63, 69], water properties [70, 71], and vegetation [72, 73]. This, along with the wider success demonstrated by GPs as tools for modelling natural processes as discussed in Section 3.1, motivates our choice of GPs as the primary modelling tool in this thesis.

It is worth noting that while we do make use of a GP model in this thesis, we plan over a compact, discrete representation of the physical environment as a topological map (see Section 2.1). This is in contrast to the works mentioned above, many of whose domains allow for fine-grained trajectory planning—although some previous works, such as Ott *et al.* [74], have also used graph-based environment representations for AIPP problems over GPs. We employ the graph-based representation to make the long-horizon planning problem more tractable, in line with the literature in this area which we will discuss further in Section 3.3. Lower-level aspects such as motion planning are delegated to separate subsystems, and fall outside the scope of this thesis.

Evaluation metrics

Another key aspect in AIPP problems is the *evaluation metric*, which quantifies performance with respect to the mission objective, either as a direct representation or an approximation of the true objective [60]. GP-based AIPP methods have used evaluation metrics including entropy [75], mean uncertainty [76, 77], maximum value information [78], distribution divergence [76], and upper confidence bounds (UCBs) [79]. In Chapter 4, we incorporate a UCB reward into our planning system in order to explicitly encourage information-gathering actions, while in Chapter 7 we design a more application-specific volumetric gain measure to estimate information gain for environment mapping. In each case, the information-based metrics are not themselves the mission objectives, but rather act as proxies which encourage the desired exploratory behaviour.

Planning approaches

We now consider the planning approaches used in the AIPP literature. Many recent works have employed *supervised learning* techniques, which use labelled datasets to learn the desired output behaviour. Some works have created synthetic environments or simulators which approximate the real-world problem, using these to generate the training data [63, 68, 73, 80, 81]. Alternatively, training data can be acquired from previously executed missions [70, 77, 82] or open-source datasets [83, 84]. However, for the purpose of this thesis, we consider each of these approaches to be too restrictive. In the long-term missions we consider throughout this thesis, in areas such as coral reef monitoring (Chapter 4) and wildfire monitoring (Chapter 7), it cannot be assumed that data from previous missions exists, nor that there is any other freely available data of sufficient quality to create a realistic simulator. Indeed, in Chapter 5, we note that one of the key limitations of the wildfire hotspot dataset [85] is its low temporal resolution, which was limited by the frequency of satellite passes over the region. In general, long-term datasets with sufficient spatial and temporal resolution to build a high-quality simulator are rare, due to the high cost of data collection, so we choose not to use supervised learning methods which rely on the existence of such data.

We therefore turn to the *reinforcement learning* (RL) paradigm, which removes the reliance on pre-labelled data, and focuses instead on learning through feedback from interactions with the environment. The AIPP problem can be formulated as a POMDP [86], in which the unknown process f is a component of the state space, over which the robot maintains a belief. With this in mind, we can consider the AIPP problem within the broader context of the literature on POMDP planning, which we explore in the following section.

A further alternative that has been used with success in AIPP is *imitation learning* [87, 88], where the policy is learned from expert demonstrations. While such methods can produce highly effective policies, the work in this thesis assumes that we do not have access to expert demonstrations, and thus we rely entirely on online learning with minimal prior knowledge.

3.2.2 Planning under partial observability

Finding exact solutions to POMDP problems is known to be computationally intractable for problems of non-trivial size [89, 90]. This is often attributed in the literature to two key factors, known respectively as the *curse of dimensionality* and the *curse of history*. The curse of dimensionality refers to the fact that the size of the belief space used to solve POMDPs grows exponentially with $|S|$, the number of POMDP states [91]. The curse of history refers to the fact the belief-contingent planning space also grows exponentially with the planning horizon [92]. In this section, we discuss some of the key developments in the POMDP planning literature as they relate to the work in this thesis. We restrict this discussion to online solvers, which have demonstrated significantly greater ability to scale to large problems than offline methods [93].

The POMCP algorithm introduced by Silver and Veness [90] has been hugely influential in POMDP planning, and uses MCTS to tackle both of the core challenges mentioned above. POMCP extends the UCT algorithm from Kocsis and Szepesvári [36] to partially observable environments, using it to select promising actions during MCTS. Combining this with a particle filter approximation of the belief state then allows for efficient belief updates as part of the same Monte Carlo procedure as the tree search. The result is an algorithm that scales to much larger POMDP problems than were previously solvable. A contrasting approach which has also seen much success is the DESPOT algorithm [94], which achieves POMDP planning at scale by randomly sampling a set of determinised *scenarios*. While this approach has also demonstrated success in scaling to large POMDP problems, it has been shown to struggle in information-gathering problems in continuous spaces [38].

Subsequent works have extended POMDP techniques to continuous domains. The POMCPOW algorithm from Sunberg and Kochenderfer [38] adapts POMCP to continuous action and observation spaces using progressive widening (see Section 2.4.2), with the addition of a weighted particle filter to encourage convergence to better solutions. Continuous-time POMDPs have also been solved using discretisation [95], removing the continuous element altogether as a trade-off between model accuracy and computational tractability. Alternatively, partially observable semi-MDPs (POSMDPs) [96] explicitly

model continuous-time actions, and have since been extended to solve asynchronous planning problems for multi-robot systems [97, 98]. In Chapter 5 we make use of temporal discretisation in our hierarchical planner in order to enable long-term planning, before extending this to an explicitly continuous-time model in Chapter 6.

So far, this section has discussed some of the key developments that have paved the way for modern POMDP planning algorithms. Within the broad area of planning under partial observability, the work in this thesis is guided by two key considerations: the GP belief representation of environment processes, and the goal of long-term planning. We therefore dedicate the following section specifically to a discussion of planning with GP beliefs, before Section 3.3 delves into the literature on long-term autonomy.

3.2.3 Planning with GP beliefs

The versatility of GPs as modelling tools (see Section 3.1) has led them to increasing prominence in the planning literature. Our particular interest in this thesis is in GP representations of a robot’s *belief* over the dynamics of an unknown environment process, which can be updated online based on observations. GP beliefs have been used in planning problems to model natural phenomena including terrain traversability [99], radio signal strength [100], hazardous radiation [19, 20], magnetic fields [101], ocean currents [34, 46], and bacteria levels in a lake [70]. In this section, we discuss some of the techniques that have been developed for planning with GP beliefs, across a variety of different—albeit interrelated—fields of study.

Several recent examples of planning with GP beliefs have approached problems through the framework of *Bayesian optimisation* (BO) [102], which seeks to find the optimum of an unknown function f through an iterative process of noisy sampling. BO typically uses a GP model of f , which is updated after each iteration as new observations are added. Standard BO is a *myopic* decision making approach, in that it only considers the next step when deciding on the best action, thereby failing to account for future rewards and often exhibiting suboptimal behaviour in multi-step problems. The *sequential* Bayesian optimisation (SBO) framework presented by Morere *et al.* [103] bridges this gap, building on decision making techniques introduced

in previous works [79, 104]. They introduce the BO-POMDP model for sequential Bayesian optimisation over GP beliefs, and demonstrate its capability as a non-myopic approach to informative path planning. Note that the AIPP problem discussed in Section 3.2.1 can be viewed as a sequential BO problem. Flaspohler *et al.* [78] take a similar approach to the maximum seek-and-sample problem, a close relative of IPP. Their work leverages the maximum value information (MVI) heuristic from Bayesian optimisation to plan effectively for long-term information gathering with sparse data, using continuous-observation MCTS with a GP belief. One limitation of both of these works is that the POMDP transition function T is assumed to be deterministic and independent of f . In Chapter 6 we focus specifically on stochastic transition dynamics as a component of a more realistic model for our long-term robot planning problem. Moreover, while we do not explicitly consider cases where the unknown process can affect the transition dynamics, our formulation does leave this open to future work as a relatively straightforward extension.

These limitations are addressed in recent work by Budd *et al.* [34], which takes a Bayesian RL approach to planning in an unknown environment. They frame their problem as a Bayes-adaptive MDP (BAMDP) [105], and develop a variant of the BAMCP algorithm [106] which allows for efficient planning over a GP-modelled environment process. Importantly, the Bayesian framework optimally trades off exploration and exploitation, which ensures reliable performance in the single-mission setting they consider. Their work also makes use of the *root sampling* technique used in POMCP and BAMCP, demonstrating an equivalence relation which removes the need for computationally expensive GP belief updates during MCTS. Despite the demonstrated success of this approach, however, we shall see in Chapter 4 that this approach cannot be straightforwardly transferred to the long-term monitoring setting, where the mission lifetime might extend beyond the feasible planning horizon.

3.2.4 Safe exploration

Among the diverse environments in which autonomous robots are routinely deployed, many are marked by the presence of hazards or other safety-critical factors. *Safe*

exploration addresses the problem of exploring an environment whilst respecting safety constraints, in order to prevent harm to the robot or its surroundings. These constraints have been formulated in a variety of different ways, with the general goal of avoiding *unwanted* or *unrecoverable* states [107]. Sui *et al.* [108] consider the problem of learning by sampling from a GP-modelled unknown function, requiring the sampled points to exceed some pre-specified safety threshold. This work was later extended by Sui *et al.* [109] to separate the safety function from the objective function, allowing for efficient optimisation of the objective function while maintaining guarantees over the safety constraints. In Chapter 7, we consider the objective of mapping as much of an environment as can be safely accessed, only visiting states in which the value of a GP-modelled unknown process—in this case, hazardous radiation—is within a safe bound.

In the context of safe exploration of MDPs, it is necessary to reason about state *reachability* and *returnability*—in contrast with the Bayesian optimisation setting, which assumes that observations can be taken freely from throughout the full domain. Moldovan and Abbeel [18] posit that in most physical systems, it is indeed *not* the case that all states are accessible from all other states (a concept known as *ergodicity*), and introduce an algorithm for MDP exploration which ensures returnability to the initial state. The SafeMDP algorithm introduced by Turchetta *et al.* [19] extends this idea to safe exploration of MDPs using GPs, reasoning about both reachability and returnability. However, this approach only ever selects goal states that are a single step away from the visited set, resulting in inefficient exploration.

Budd *et al.* [20] addresses this limitation with a safe exploration algorithm that reasons about multi-step reachability and returnability. Their work models an unknown process as a GP, and incorporates a probabilistic transition function that can model influences of the unknown process on the robot’s dynamics. This is in contrast to the SafeMDP algorithm, which assumes deterministic transitions. Consider, for example, an AUV exploring an ocean environment, where the underwater currents are unknown *a priori*—in this scenario, the currents directly affect the robot’s navigation dynamics, and sufficiently strong currents could render certain states unsafe. Certain actions may even become entirely infeasible if the current velocity exceeds the maximum velocity of the

AUV. Effective safe exploration in this setting requires an environment model that can be adapted as the mission progresses, and incorporated into the robot’s reasoning about safety during exploration. Similar concerns can arise in other domains, such as a UAV exploring in a windy environment [110], or a ground robot navigating steep terrain [111]. However, the work by Budd *et al.* [20] assumes that the layout of the environment is given as a topological map, and considers only exploration of the unknown process itself. In Chapter 7, we address the problem of safe exploration in a *completely* unknown environment, where mapping of the environment is a component of exploration.

3.2.5 Mapping of unknown environments

So far, our discussion of unknown environments and partial observability has focused on cases where the unknown element has been some observable process or feature present throughout the physical environment, such as temperature or radiation levels. We have assumed that the *physical* layout of the environment as a topological map is known *a priori*. In this section, we consider the more general problem of exploration in completely unknown environments, in which a robot must explore without any prior map or knowledge of the environment layout. This forms the foundation of the work in Chapter 7.

The problem of exploration of unknown environments has been addressed in many works through frontier-based approaches, which aim to guide the robot towards unknown areas of the map [112, 113, 114, 115]. As defined by Yamauchi [112], a frontier is a location on the boundary between open space and unexplored space; thus, by moving to new frontiers, a robot can extend its map of the environment. The research on exploration of unknown environments received significant attention in recent years due to the DARPA Subterranean Challenge (SubT), in which the goal was to develop teams of robots to autonomously explore underground environments. In the context of SubT, teams of researchers developed and tested complex integrated systems which address problems such as localisation and mapping in GPS-denied environments, navigation in adverse terrains, or coordination of heterogeneous multi-robot teams under communication constraints [116, 117, 118, 119, 120]. Numerous exploration

algorithms were developed as part of these efforts, a significant portion of which rely on frontier-based methods [121, 122, 123, 124].

The SafeEstMDP-Map algorithm that we present in Chapter 7 also uses a notion of frontiers to drive the robot towards unknown areas of the map. However, because it must maintain safety with regards to the unknown process, we also consider possible points which are not at the edge of known space but for which the GP prediction of the unknown process still has high variance—in a sense, these are frontiers in our belief space rather than the map of the physical environment. Exploration approaches for large-scale domains are often hierarchical, with the aim of ensuring thorough exploration at a local level whilst considering when to move to other unexplored areas of the map at a global level [122, 124, 125, 126]. In the context of SubT, where unknown and challenging terrain was a key consideration, many works focus on the *traversability* of edges in the navigation graph. In contrast, we either assume that traversability is either given in the form of a navigation MDP, or that it can be checked by simple ray casting, where an edge is traversable to the robot unless there is a visible obstacle on the way. Moreover, we do not separate the exploration algorithm between local and global levels. Instead, our focus is on the use of GPs to model an external unknown process, and reasoning about safety with respect to that process. This aspect of our work goes beyond what is considered in the SubT works, where this type of safety was not a focus. We also note that our approach could in principle be integrated with more complex exploration approaches, in order to ensure an extra layer of safety against external processes.

3.3 Long-term planning

A fundamental challenge when planning in long-term settings is computational complexity, as the search space typically grows exponentially with the planning horizon [30, 92]. In this section, we provide a brief overview of existing application areas for long-term robot autonomy. We then discuss some of the key techniques that have been developed to address the challenges of planning in this setting, and planning over long horizons more generally. These techniques inform the design of our planners in Chapters 4 and 5.

3.3.1 Long-term autonomy in robotics

Long-term autonomy is a growing area of robotics research, with the potential to deliver immense value across a breadth of industrial and research applications. Different domains and mission objectives induce particular requirements on robot hardware, software, and algorithmic systems, which must be addressed in order to achieve successful long-term operation. Here, we provide a brief overview of some of the technical challenges that arise in long-term robot autonomy, and some of the existing works that have addressed these challenges.

The development of autonomous systems for some applications has been driven by constraints on communication systems, which in some cases makes a certain level of autonomy a necessity. In space robotics, communication delays due to signal propagation times can be on the order of minutes, making real-time teleoperation from Earth impossible. During its first year on Mars, the NASA Perseverance rover autonomously planned or mapped almost 90% of its 17.7 km total distance travelled [127]. The software pipeline required to achieve this included an autonomous navigation system (AutoNav); an automated system for selection of scientifically interesting features to examine based on visual data; and an on-board planner to schedule tasks and ensure efficient usage of the limited available energy [127]. Improvements to these planning systems through decades of research have enabled the Perseverance rover to pursue more ambitious scientific objectives than previous Mars rover missions were able to, and thus to make better use of the resources invested into the mission. Missions in marine environments face many similar challenges, since communication through water is limited and unreliable, and maps of these environments (particularly the deep ocean) are also often difficult to obtain. However, the technology in this area is significantly more mature, and there are countless examples in the literature of AUVs being deployed successfully on scientific missions. Ocean gliders, for example, are simple AUVs which have seen great success in ocean monitoring and sampling missions [128], with their long range and endurance making them particularly well-suited to long-term missions. Propellor-driven long-range AUVs can offer additional mobility and control compared with gliders, and have recently achieved up to 2000 km of continuous operation [129].

Terrestrial robots—also known as unmanned ground vehicles (UGVs)—are also seeing increasing use in long-term missions, in which they often contend with unstructured and dynamic environments. One of the most prominent application areas for such robots is in agriculture, which can require navigation over challenging terrain and execution of complex tasks. Robots are increasingly being used for tasks such as weeding, planting, harvesting, crop health monitoring, spraying of pesticides, and more [130]. Autonomous systems have been able to significantly improve upon the efficiency of traditional farming methods, boosting production and reducing the environmental impact of the agricultural sector. Aerial robots, by contrast, are most acutely constrained by energy, and long-term or perpetual flight remains a significant challenge. Solar power has been used to extend the flight time of UAVs, but such solutions tend to be contingent upon favourable weather conditions and carefully chosen flight paths [131]. One alternative approach to enabling long-term autonomy for UAVs is demonstrated by Peloquin *et al.* [132], who developed a UAV capable of autonomous landing and takeoff from lakes—between flights, this UAV could float passively on the surface of a lake while recharging its batteries through a solar panel. While this particular approach is rather specific and limited in scope, it is a testament to the research efforts being put towards the challenge of enabling long-term autonomy in aerial robots. UAVs offer incredibly wide-ranging application potential, from agricultural and industrial monitoring, to wildfire monitoring, border surveillance, and more [133].

Robots have also been deployed in more human-centred environments, where autonomous operation may involve navigation of complex systems and safe operation in the presence of humans. This is exemplified in the problem of autonomous driving, which requires robots to navigate complex road networks alongside other road users, and make real-time decisions based on their surroundings. Autonomous driving systems are becoming increasingly prolific, but as yet few have been able to achieve reliable operation for prolonged periods without human intervention [134]. Many similar challenges are faced by service robots, which work for or alongside humans in environments such as offices and care homes [135]. These robots additionally will often have to

manage complex planning tasks, adapting to ever-changing requirements in complex and dynamic settings [136, 137].

The focus of this thesis is on addressing the planning challenges of long-term robot autonomy in unstructured outdoor environments, for applications such as ecological monitoring and precision agriculture. The difficulty of planning effectively over long horizons in these settings is compounded by each of the challenges outlined in Section 1.2. Our aim is to develop models and planning approaches which address these challenges in meaningful and practical ways, and to demonstrate the effectiveness of these approaches in realistic planning scenarios.

3.3.2 Hierarchical planning

The challenges of long-horizon planning are often addressed through the use of hierarchical planning methods. These methods introduce abstractions or subgoals which reduce the search space of the problem—either by reducing the branching factor, or the length of plans that need to be computed—in order to make larger problems more tractable. In the context of sequential decision making under uncertainty, hierarchical planners are often designed with the rationale that constructing detailed plans far into the future is unhelpful, since the evolution of the state cannot be accurately predicted. In such cases, a coarse-grained model may be sufficient for long-horizon planning. We use this principle to guide our planner designs in Chapters 5 and 6.

The MAXQ model from Dietterich [138] is an early example of a hierarchical method in reinforcement learning. This approach works by decomposing an MDP into a hierarchy of smaller MDPs, whose value functions combine additively to form the value function of the original MDP. Whilst originally presented as a model-free RL approach, this work was later used as the basis for a model-based planning algorithm in Diuk *et al.* [139]. However, despite its successes, the MAXQ approach also has many limitations. Most notably, the value functions within the hierarchy are computed by backing up values computed from the lowest level upwards requiring computations over the full state-action space; this is known as a *bottom-up* approach. The MAXQ algorithm also relies on a manual process of task decomposition, which presents a significant

barrier to generalisation across different domains. Another canonical approach in the hierarchical planning literature is the hierarchical task network (HTN) introduced by Erol *et al.* [140, 141], which has demonstrated success in a variety of complex robot planning tasks [142, 143]. HTNs rely heavily on domain knowledge, working best in structured environments where a clear task decomposition exists. In their original form, they also assume deterministic action outcomes, although later works have extended the approach to non-deterministic domains [144].

In this thesis, hierarchical planning is of particular interest as a means of managing the computational challenge of planning over large domains and long horizons. The *options* framework from Sutton *et al.* [145] moves us in this direction, generalising the notion of actions to include closed-loop policies for taking actions over time. An option is an example of a *macro action*, a composite action consisting of a sequence of *primitive* actions. This idea provides a convenient mechanism for MDP planning across multiple timescales, since each macro action can be treated as a single decision-making unit during planning. However, the options framework is still a bottom-up approach, meaning it will tend to become quite slow in problems where precomputed low-level policies are not available—as is the case throughout this thesis. The abstract MDP (AMDP) introduced by Gopalan *et al.* [146] instead decomposes the original MDP problem into sub-tasks in a top-down manner, producing a more efficient planning approach which “decides what a good subgoal is before planning to achieve it”. Kaelbling and Lozano-Pérez [147] take a hierarchical approach to combined task and motion planning, by first planning over an abstract representation of the problem, then realising only the first step of that plan with a low-level motion planner before replanning. We incorporate elements from each of these works in developing our hierarchical planning framework in Chapter 5, adapting them to the setting of long-term monitoring of a continuous, GP-modelled environment.

3.3.3 Persistent monitoring

Throughout this thesis, we refer to the problem of *long-term* monitoring, in contrast with *persistent* monitoring, as we do not explicitly consider missions of indefinite length. However, as we shall discuss in Section 8.2, bridging this gap would be trivial, requiring

as a minimum only minor modifications to our handling of input data for the GP models. The literature on persistent monitoring therefore significantly informs the work in this thesis, so we discuss some of the relevant works here.

Many previous works have addressed persistent monitoring problems in single-robot [148, 149, 150, 151, 152] and multi-robot [153, 154, 155, 156, 157, 158, 159] settings. These works typically frame environment monitoring in terms of information gain maximisation or uncertainty reduction over GPs, similar to the AIPP problems discussed in Section 3.2.1. Persistent monitoring can be tackled hierarchically, as done by Nayak *et al.* [160], who use a multi-armed bandit framework to address the high-level problem of allocating regions to robots. Krause *et al.* [161] address the monitoring problem from the perspective of fixed sensor placement, developing a sequential greedy algorithm that maximises mutual information across the environment. Later work by Binney *et al.* [148] poses the problem as graph-based IPP over a continuous spatiotemporal environment process, a very similar setting to the one we consider in this thesis. In contrast to our work, however, the mission objective in their work is uncertainty reduction at waypoints in the environment, and they incorporate sampling along graph edges as well as at nodes, something we do not consider in this thesis. Ma *et al.* [152] incorporate online learning of GP hyperparameters into their spatiotemporal monitoring algorithm, enabling their environment model to improve over the course of the mission. We follow a similar process for GP hyperparameter learning in Chapter 4, specialised to our multi-mission setting to allow the learning to happen during the robot’s down-time between missions.

This thesis focuses on monitoring problems with the goal of explicit *reward maximisation*, rather than information gain. Whereas information gain objectives focus on learning about the environment, reward maximisation focuses on accumulating high-value samples or observations from it. Several problems in the path planning literature are closely related to this goal. In the *orienteering problem* [162], the goal is to find a reward-maximising path that respects a budget constraint (e.g. on energy consumption or time taken). Yu *et al.* [163] pose a correlated version of the orienteering problem for persistent surveillance tasks, where the utility of observations considers spatial correlations between locations based on proximity. They evaluate their approach

in a persistent surveillance task for a UAV, showing that it is effective in small domains for generating effective tours within the constraints of limited travel range. Yu *et al.* [164] instead consider rewards which are gathered by spending *time* at points of interest, introducing a more explicit trade-off between time spent travelling and time spent sensing.

The infinite horizon in persistent monitoring problems presents a distinct challenge from the general planning problems discussed in Section 3.2. Some works have explicitly planned over infinite horizons by finding periodic solutions that approximate the optimal infinite horizon trajectory [165, 166]. However, finding such solutions is intractable in general, particularly in the high-uncertainty environments that are the focus of this thesis. More commonly, problems with indefinite horizons are simply addressed using a receding planning horizon, which allow planners to look ahead as far as is necessary (and computationally tractable) to generate effective plans. Receding horizons can take several different forms—for example, planning with a fixed n -step lookahead [79], or enforcing a maximum search depth in MCTS [103]. Jones *et al.* [167] combine a receding horizon planner with linear temporal logic (LTL) constraints, finding a middle ground which allows for planning over finite horizons while providing guarantees over certain aspects of the infinite horizon behaviour. Once again, however, this approach makes highly restrictive assumptions about the domain representation, and is only tractable in small domains. The algorithms we present in Chapters 4 and 5 therefore adopt receding horizon approaches, which afford the greatest flexibility for planning in complex environments. We allow these planning horizons to be guided by practical constraints, such as the battery life of the robot.

3.4 Planning under resource constraints

Mobile robots operating out in the real world are invariably subjected to a variety of practical resource constraints, which may include battery life, on-board memory, or time. In this section, we consider different formulations of resource constraints that have been used in the planning literature, and the techniques used to address them and enable effective deployment in the real world.

3.4.1 Constrained path planning

Some works have addressed resource-constrained planning by extending classical graph-based path planning problems. Asghar and Tokekar [168] frame the problem of energy-aware routing for a battery-constrained robot as a variant of the travelling salesperson problem (TSP), in which a robot must efficiently visit every location in its environment while also recharging its battery at *depots* as needed. We adopt a similar framework of depot-based recharging in Chapter 5, which focuses on addressing **C2 Resource Constraints**. Constrained shortest path (CSP) problems are also a focal point in the literature on electric vehicle (EV) routing [169, 170, 171]. Baum *et al.* [169] extend the CSP problem to use realistic battery charging models, an element we introduce into our work in Chapter 6 to link **C2 Resource Constraints** and **C3 Stochastic Interactions**. Different versions of the EV routing problem have been formulated, for example with goals of finding the most energy-efficient [172] or fastest [173] routes while incorporating the necessary stops at charging stations. In EV routing, similar to our long-term planning setting, planning multi-step routes with charging stops greatly expands the range that the vehicles have access to.

The main limitation shared by these works is that they assume static and fully-known environments, and thus do not have straightforward extensions to the dynamic, partially observable environments we are interested in. However, we shall see in Chapter 4 that we are able to use precomputed shortest path lengths as a way to prune actions during MCTS.

3.4.2 Constrained sequential decision making

Resource constraints have also been incorporated into sequential decision making problems as extensions of various MDP models. *Constrained MDPs* [174] encompass a broad class of problems, which in general involve an agent seeking to maximise some utility function (e.g. information gain) whilst bounding a cost function (e.g. energy consumption). Subsequent work has generalised this notion to constrained POMDPs [175], for which a constrained variant of the POMCP algorithm can be used for planning [176]. The short-horizon planning sub-problems we solve in Chapters 4 and 5

can be viewed as instances of constrained POMDPs, which we solve using MCTS with action pruning.

The standard constrained MDP framework does not incorporate the notion of resource *replenishment*, which we introduce into our work in Chapter 5 to allow us more flexible longer-horizon planning. This is addressed by the *consumption MDP* introduced in Blahoudek *et al.* [177], which models resource depletion and replenishment—however, their work only addresses the objective of planning a path that will visit goal states infinitely often while avoiding resource exhaustion. Other works have incorporated resource replenishment into variations on the orienteering problem [170, 178] in order to achieve long-term autonomy. In Chapter 5, we consider a resource-constrained problem in which the robot state is fully observable, and resource consumption is deterministic. This allows us to incorporate resource constraints in a simpler way than the general POMDP setting in order to address **C2 Resource Constraints**. We also show in Chapter 6 that this approach can be extended to the stochastic setting to address **C3 Stochastic Interactions**, under certain assumptions about the form of the resource consumption model.

It is worth noting that resource constraints can factor into robot mission planning in many more ways, depending on both the scenario itself and the modelling assumptions used to represent it. For example, in the context of heterogeneous robot teams, Liu and Williams [1] consider that there might be variable costs associated with each *deployment*, and address the problem of deciding when to deploy. The placement of charging stations can also be optimised to improve mission outcomes [179, 180, 181], although this is outside the scope of this thesis.

3.4.3 Long-term and persistent problems

Resource constraints have been incorporated into long-term and persistent planning problems. In long-term robot missions, it is actually *necessary* to consider resource constraints to some extent, since mobile robots have finite battery life. Much of the work in this area to date has focused on persistent surveillance problems, often for multi-robot teams, with mission objectives and constraints formulated in a variety of

ways. Aksaray *et al.* [182] use temporal logic constraints to minimise the time between visits to locations of interest in the environment, while ensuring each agent periodically visits a base station to refuel. This approach allows them to provide formal guarantees over their mission specification, but does not extend easily to complex reward structures or partial observability. Other works address planning for persistent surveillance with energy-constrained UAVs [183, 184], but again these do not generalise beyond coverage objectives. A popular approach to persistent UAV missions involves the use of mobile ground robots as charging stations for the UAVs [185, 186, 187]. This approach creates a more complex planning problem (as well as adding to the technical difficulty of deployment), including decisions about where to rendezvous for recharging. However, it also allows for greater flexibility in mission planning, as charging stations can be relocated dynamically to improve monitoring efficiency.

As is the case with long-term planning in general, hierarchical methods can be used to make constrained long-term planning problems more tractable. Eiffert *et al.* [188] present a hierarchical planning framework for robust long-term operation of field robots in unstructured environments, geared towards agricultural applications. Their work focuses on balancing the challenges of long-term, mission-level planning under resource constraints with path planning in highly dynamic environments. This is handled in their hierarchy by separating these components into a low-level dynamic path planner, and a high-level, long-term mission planner. This allows the low-level planner to tackle the fine-grained problem of local path planning, while the high-level planner uses an abstract, graph-based representation of the domain to plan over longer horizons.

An alternative approach to resource management in long-term planning is in the work by Tomy *et al.* [189], which considers the problem of when to charge for a long-life autonomous robot performing a series of time-windowed tasks. Their work demonstrates the advantage of flexible scheduling in a stochastic setting compared with a more rigid, rule-based scheduling approach. Although we do not frame our missions as task scheduling problems, we reach a similar conclusion in Chapter 5 in the context of long-term monitoring—we see that in monitoring an uncertain environment,

performance improves as the robot learns about its environment and adapts its charging schedule accordingly.

3.4.4 Chance-constrained planning

In stochastic settings where future resource consumption cannot be predicted with certainty, it is often impossible to provide strict guarantees of resource constraint satisfaction—or, where it is possible, doing so would tend to result in overly conservative behaviour. Instead, chance-constrained approaches place a bound on the allowable *probability* of failure. This affords much greater flexibility in planning, allowing the robot to accept a certain level of risk in order to obtain higher rewards. In Chapter 6, our focus is on extending the hierarchical planner from Chapter 5 to this setting in order to address **C3 Stochastic Interactions**.

The chance-constrained MDP (CC-MDP) and chance-constrained POMDP (CC-POMDP) generalise the constrained MDP and POMDP respectively to the setting of stochastic resource consumption. Chance-constrained planning is a more difficult problem than regular constrained planning, since all actions and outcomes in a policy are now coupled by the chance constraint. Ayton and Williams [190] introduce an MCTS approach for CC-MDPs which overcomes this challenge by decomposing the chance constraint into constraints over individual state histories, allowing them to efficiently find approximately optimal solutions. The more recent ConstrainedZero algorithm [191] tackles CC-POMDPs using MCTS, applying neural networks to learn the optimal policy and value functions, as well as approximate the failure probability. This algorithm is able to maximise reward while ensuring a specified level of probabilistic constraint. However, as yet these approaches have not been demonstrated to scale to long-horizon planning. In contrast, Ono *et al.* [192] present a dynamic programming approach to chance-constrained planning, demonstrating it as a computationally efficient solution to real-world, safety-critical planning problems. However, this approach assumes a fully observable state, and would not extend to our highly uncertain, partially observable setting.

In the face of uncertain resource consumption, other techniques can be used to mitigate the risk of constraint violations. Mitchell *et al.* [193] incorporate the useful notion of a *recourse* strategy into their work on persistent surveillance. They frame persistent surveillance as a task planning problem with stochastic task costs, in which a chance constraint is applied to resource consumption. Their recourse behaviour is a recovery mechanism which triggers in the event of unexpected energy expenditure, re-routing the robot to reduce the number of tasks it will try to complete before its next opportunity to recharge at a depot. This allows it to avoid resource exhaustion in emergency situations, providing an additional layer of robustness in the face of uncertain resource consumption. Our stochastic planner in Chapter 6 follows a similar approach to this, adopting a chance-constrained objective and incorporating a recovery behaviour to avoid constraint violations in case of unexpected resource consumption. In addressing stochastic action outcomes in Chapter 6, we relax the requirement of satisfying the chance constraint over the full mission, instead applying it to individual outings of the robot away from depots. This both simplifies the planning problem, and provides a more sensible framework for risk management in a context where missions might go on indefinitely. We are then able to implement an efficient action pruning regime to ensure constraint satisfaction within belief-space MCTS.

4

Multi-Mission Monitoring with Gaussian Process Models

Contents

4.1	Introduction	45
4.2	Problem formulation	47
4.3	MCTS planning with GP beliefs	49
4.4	Algorithm	52
4.4.1	Spatiotemporal modelling with GPs	52
4.4.2	Evaluating trajectory rewards	54
4.4.3	Planning algorithm	55
4.5	Experiments	56
4.5.1	Experimental domains	56
4.5.2	Planning with a spatiotemporal GP kernel	58
4.5.3	Belief-based MCTS across multiple missions	60
4.6	Discussion	64
4.7	Summary	66

4.1 Introduction

As autonomous robots become more capable and reliable, we are seeing increasing potential for *long-term* monitoring and information gathering missions in unstructured environments. Robots deployed in such missions must be able to plan effectively in the face of unknown and ever-changing environment dynamics, often with limited and noisy

observational data. In many such applications, however, uninterrupted autonomous operation cannot yet be sustained indefinitely due to lack of autonomous docking or recharging capabilities. The problem of autonomous docking has been studied for aerial [194], ground [195], and underwater robots [196], but remains a significant challenge. Further, in many applications in unstructured environments, the required infrastructure may not be readily available, or may be prohibitively expensive. As such, many robots deployed on long-term missions still require periodic, scheduled returns to a base station for recharging, data offloading, or maintenance.

In this chapter, we consider a mobile robot deployed over a *sequence* of discrete, short missions over a period of weeks or months. For example, the robot might be deployed during daylight hours, returning to a base station each night to recharge. Our work takes inspiration from ecological domains and applications, from mapping and monitoring of coral reefs [78, 197], to crop monitoring [4] and wildfire prevention [6]. In these domains, environments typically exhibit daily cyclic patterns (due to the influence of the weather, for example) combined with slower (e.g. tidal or seasonal) variation. Our mission objective is to maximise the cumulative value of observations taken by the robot of some environment feature, about which it has limited prior knowledge. In the ecological setting, features of interest might be plant growth or biodiversity [198], which can be measured by a robot using an on-board sensor.

We formalise the problem of repeated monitoring missions in a partially observable, spatiotemporally-varying environment as a PKSMDP, and propose an online belief-based planning approach to solve it. At the core of our approach is the robot’s model of the unknown environment feature, which persists across missions, improving over time as the robot makes more observations. Here, we address **C1 Environment Modelling**, using a GP [15] that captures both spatial and temporal variation. GPs have been used to model continuous environment features in a variety of planning problems [20, 34, 46, 78, 79, 103], as they provide a flexible and principled predictive framework for settings with sparse observational data. We assume that our feature of interest varies smoothly through space and time, and focus specifically on features that exhibit periodic behaviour.

This reflects our interest in ecological and other outdoor settings which naturally have a daily periodicity, and informs the design of the GP kernel function.

The long-term setting gives rise to significant challenges in scalability, where it quickly becomes infeasible to plan over a full multi-mission horizon even in relatively simple environments. We therefore decompose the planning problem into individual missions, in which the robot plans its next action online using MCTS, forward-simulating the outcomes of different actions up to the end of the current mission using the GP model. Planning only within individual missions risks inducing myopic behaviour, so we explicitly reward exploratory actions that can benefit *future* missions. To do so, we plan in a belief-aware manner, balancing exploitation of current knowledge about the environment with exploration guided by GP uncertainty. This requires performing GP belief updates during MCTS trials, which is computationally costly. Thus, we propose an algorithm that performs GP updates only within the search tree, and estimates the reward for the rollout phase without requiring a belief update at every step. Finally, as a step towards addressing **C4 Limited Priors** and minimising our reliance on prior knowledge about the environment process, we do not assume that well-tuned GP hyperparameters are given at the start of the mission. Instead, the GP hyperparameters are optimised based on the current dataset between missions, allowing the robot to learn a better model of its environment as more observations are made.

The key contributions of this chapter are (a) the formulation of long-term, multi-mission robot monitoring of a spatiotemporal process as an MDP with a partially unknown state; (b) a planning approach for this setting using a GP belief over the unknown environment, which balances exploratory information-gathering with reward-seeking behaviour; and (c) a new method for evaluating GP-based rewards during MCTS rollouts for improved computational performance. The work in this chapter first appeared in [10], presented at IROS 2024.

4.2 Problem formulation

We consider a mobile robot deployed in a physical environment represented as a topological map $\mathcal{G} = \langle V, E \rangle$. Each vertex $v \in V$ represents a location of interest in

the environment, i.e. $v = (x, y) \in \mathbb{R}^2$, and edges $e \in E$ represent traversable paths between vertices. The robot can navigate through the environment according to the edges in E , and we assume here that the outcomes of navigation actions are deterministic. There is an *a priori* unknown spatiotemporal mapping $f : \mathbb{R}^3 \rightarrow \mathbb{R}$, where $f(x, y, t)$ is the value of an environment process of interest at location (x, y) and time t .

The objective is to maximise the cumulative reward from observations taken of f across a set of *missions* that span many days. We model this problem as a Multi-Mission PKSMDP (M^2 -PKSMDP).

Definition 4.1 (Multi-Mission PKSMDP (M^2 -PKSMDP)) A Multi-Mission PKSMDP is a pair $\langle \mathcal{M}, \mathcal{H} \rangle$, where $\mathcal{M} = \langle S, s_0, A, T, R \rangle$ is a PKSMDP (see Definition 2.6), and $\mathcal{H} = \{(H_s^i, H_f^i)\}_{i=1}^m$ is a multi-mission specification listing the start and finish times for m consecutive missions.

At the start of each mission $i \in [1, \dots, m]$, the robot is deployed at the initial location $v_0 = (x_0, y_0)$ at time H_s^i , and can explore the environment until the mission end time H_f^i . Note that, since the unknown process f is time-dependent, we include time as a component of the (known) PKSMDP state, i.e., $s_k = (x, y, t)$. Moreover, in contrast with the standard finite horizon MDP formulation (Definition 2.3), we do not use a fixed number of discrete timesteps for each mission's planning horizon. Instead, the horizons are based on the (real-valued) durations of actions, which are modelled explicitly. Within each mission, at step j , let $t(s_{k,j})$ denote the time at the current state $s_{k,j}$, and a_j the next action to be executed by the robot under the current policy. The robot transitions to the next state $s_{k,j+1}$ according to the transition function T , where the time component of the state is updated according to the duration of the action, so $t(s_{k,j+1}) = t(s_{k,j}) + \text{dur}(a_j)$. It then receives a noisy observation z_{j+1} of the unknown process f upon arrival at the new state $s_{k,j+1}$. These observations are subject to measurement noise $\epsilon \sim \mathcal{N}(0, \sigma_z^2)$, so $z_j = f(s_{k,j}) + \epsilon_j$. We refer to the sequence of known states visited and observations received during mission i as a *trajectory* $\varsigma^i = s_{k,1}^i z_1^i s_{k,2}^i z_2^i \dots s_{k,n_i}^i z_{n_i}^i$, where $s_{k,1}^i$ is the known component of the initial state of the i -th mission, i.e. $s_{k,1}^i = (x_0, y_0, H_s^i)$. The

length of each trajectory ς^i is bounded by the corresponding mission end time, so $t(s_{k,n_i}^i) \leq H_f^i$. The full multi-mission trajectory is denoted by $\varsigma = \{\varsigma^1, \dots, \varsigma^m\}$.

For the remainder of this chapter, based on our motivating setting of ecological monitoring, we consider the missions to be *daily*, such that the multi-mission problem described by $\langle \mathcal{M}, \mathcal{H} \rangle$ spans $|\mathcal{H}| = m$ consecutive days. We also assume, here and throughout this thesis, that f models a single environment feature only. However, as discussed in Section 2.3.3, this would be straightforward to generalise to multiple unknown state features.

Our monitoring objective is to find a policy π that maximises the expected cumulative reward across all missions:

$$V_\pi(s_0) = \mathbb{E}_\pi \left[\sum_{\varsigma^l \in \mathcal{S}} \sum_{s_j^l \in \varsigma^l} R(s_j^l) \right]. \quad (4.1)$$

In this work, for simplicity, we take the reward function to be the value of f itself, so $R(s_k, f(s_k)) = f(s_k)$. However, in principle a variety of other information-based objectives might be of interest in ecological or other scientific contexts. For example, Flaspohler *et al.* [78] use a maximum seek-and-sample objective, where the robot seeks to sample from the maximum of an unknown distribution. The work presented throughout this thesis permits any reward function to be used, provided it is sufficiently smooth and continuous to admit a GP representation.

Effective planning in this setting requires addressing three key challenges. First, the robot must be able to accurately model the spatiotemporal process f using sparse and noisy observational data (**C1 Environment Modelling**). Second, it must plan across multiple missions, balancing exploratory behaviour that may benefit future missions with exploitation of knowledge from past ones. Third, the planning approach must scale to large numbers of missions, in order to enable planning in truly long-term monitoring scenarios.

4.3 MCTS planning with GP beliefs

In this section, we outline an algorithm for belief-space planning with a GP observation model. This algorithm acts as the foundation of our approach to the monitoring problem

in this chapter, as well as those in Chapters 5 and 6. We consider planning for a PKSMDP in a setting where f can be observed and modelled as a GP, i.e. the GP is used to maintain a *belief* over the true value of f . This can be viewed as belief-space planning for a Bayes-adaptive MDP [105], with a GP encoding the belief over the unknown environment dynamics. Planning in belief space with GPs is computationally expensive, so we use MCTS to generate promising trajectories and select high-reward actions online. Algorithm 4.1 summarises our procedure for action selection, which we describe in detail below.

This algorithm is similar to the one presented by Flaspohler *et al.* [78], so we adopt their terminology, which refers to planning trees composed of two types of nodes: *belief* nodes, which represent planning states, and *belief-action* nodes, which represent action choices. In the standard MCTS nomenclature introduced in Section 2.4, belief nodes and belief-action nodes are decision nodes and chance nodes, respectively. The root of the search tree is a belief node representing the current state of the mission, which we denote with a timestep index j by $b_j = (s_{k,j}, g_j, s_{e,j})$, where $s_{k,j} \in S_k$ is the known robot state, g_j is a GP posterior over f based on the history of observations *prior* to timestep j , and $s_{e,j} \in S_e$ is an observation sampled from g_j at known state $s_{k,j}$.

Starting with just the root node, trials are performed, either for a fixed number of iterations or until some time budget is exhausted. In the selection step, starting from the root node, an action a is selected (Line 16) according to the PUCT value:

$$\hat{V}^*(b_j, a) = \hat{V}(b_j, a) + C \sqrt{\frac{N(b_j)^{e_d}}{N(b_j, a)}}, \quad (4.2)$$

where $\hat{V}(b_j, a)$ is the average reward obtained by choosing action a with belief b_j in previous trials; $N(b_j)$ is the number of times that node b_j has been simulated; $N(b_j, a)$ is the number of times that action a has been selected from node b_j ; C is a constant that affects the exploration-exploitation balance; and e_d is a depth-dependent parameter as per Auger *et al.* [37].

Since the observation space is the co-domain of f , which is continuous, any observation sampled from the GP belief is guaranteed to be unique. Thus, we use progressive widening to limit the size of the planning tree, parameterised by another

Algorithm 4.1 BELIEF-SPACE MCTS

Input: PKSM DP $\mathcal{M} = \langle S, s_0, A, T, R, H \rangle$, current GP model $\mathcal{GP}$ (dataset $\mathcal{D}$, hyperparameters θ , kernel $k(s, s')$).

Output: Action $a \in A$ to take from the current state s_0

```

1: procedure GETNEXTACTION( $\mathcal{M}, \mathcal{GP}$ )
2:    $n_{\text{root}}^B \leftarrow \text{INITIALISETREE}(\mathcal{M}, \mathcal{GP})$ 
3:   for  $i \leftarrow 1$  to  $N_{\text{trials}}$  do
4:      $n_{\text{leaf}}^B, r_{\text{tree}} \leftarrow \text{SELECTNODE}(\mathcal{M}, \mathcal{GP}, n_{\text{root}}^B)$ 
5:      $r_{\text{rollout}} \leftarrow \text{PERFORMROLLOUT}(n_{\text{leaf}}^B, H)$ 
6:      $r_{\text{total}} \leftarrow r_{\text{tree}} + r_{\text{rollout}}$ 
7:      $\text{BACKPROPAGATE}(n_{\text{leaf}}^B, r_{\text{total}})$ 
8:   end for
9:   return  $\text{BESTCHILD}(n_{\text{root}}^B)$ 
10: end procedure

11: procedure SELECTNODE( $\mathcal{M}, \mathcal{GP}, n_{\text{root}}^B$ )
12:    $n^B \leftarrow n_{\text{root}}^B$ 
13:    $r_{\text{tree}} \leftarrow 0$ 
14:   while  $\text{depth}(n^B) < d_{\text{max}}$  do
15:      $A_v \leftarrow \text{GETAVAILABLEACTIONS}(\mathcal{M}, n^B)$ 
16:      $n_{\text{child}}^{BA} \leftarrow \text{SELECTACTIONPUCT}(n^B, A_v)$ 
17:     if  $n_{\text{child}}^{BA} = \emptyset$  then
18:       return  $n^B, r_{\text{tree}}$ 
19:     end if
20:      $n_{\text{child}}^B, r_{\text{child}} \leftarrow \text{GENERATEBELIEFCHILD}(n_{\text{child}}^{BA})$ 
21:      $n^B \leftarrow n_{\text{child}}^B$ 
22:      $r_{\text{tree}} \leftarrow r_{\text{tree}} + r_{\text{child}}$ 
23:   end while
24:   return  $n^B, r_{\text{tree}}$ 
25: end procedure

26: procedure PERFORMROLLOUT( $n^B, H$ )
27:    $s \leftarrow s(n^B)$ 
28:   while  $t(s) \leq H$  do
29:      $A_v \leftarrow \emptyset$ 
30:     for  $a \leftarrow \text{GETENABLEDACTIONS}(\mathcal{M}, s)$  do
31:        $s' \leftarrow T(s, a)$ 
32:       if  $t(s') \leq H$  then
33:          $A_v \leftarrow A_v \cup \{a\}$ 
34:       end if
35:     end for
36:      $a \leftarrow \text{RANDOMCHOICE}(A_v)$ 
37:   end while
38:   return  $A$ 
39: end procedure

```

depth-dependent parameter α_d [37]. If $\lfloor N(b_j, a)^{\alpha_d} \rfloor = \lfloor (N(b_j, a) - 1)^{\alpha_d} \rfloor$, then the successor state b_{j+1} is randomly sampled from the existing children of node (b_j, a) ; otherwise, a new state $s_{k,j+1}$ is sampled from $T((s_{k,j}, s_{e,j}), a, \cdot)$ and a new observation $s_{e,j+1}$ is sampled from the GP posterior g_{j+1} at $s_{k,j+1}$. These are used to add a new leaf node to the tree, $b_{j+1} = (s_{k,j+1}, g_{j+1}, s_{e,j+1})$, where the time at the leaf node is given by $t(b_{j+1}) = t(b_j) + \text{dur}(a)$.

This process of alternating between action and belief-action nodes continues down the tree until a new (leaf) belief node is added (the expansion phase), as depicted in Figure 2.3. From this node, we enter the rollout phase (Line 5), in which we simulate a sequence of random actions up to the horizon H : that is, until we reach a node b'_j where $t(b'_j) + \min_a \text{dur}(a) > H$. This simulated trajectory yields a reward r , which is backpropagated (Line 7) to the tree root, updating the average reward and number of queries of each node visited in the selection step. Our method for computing this trajectory reward r is described in Section 4.4.2. Once trials are completed, the action corresponding to the most-visited belief-action child of the root node is selected (Line 9) and executed.

4.4 Algorithm

4.4.1 Spatiotemporal modelling with GPs

The presence of temporal variation amplifies the challenge of building accurate predictive models using limited observational data. Not only does it add to the dimensionality of the state space, but it also means—in the context of data gathered by a mobile robot—that no individual state can be observed multiple times. The robot may return to the same location multiple times, but the value of the environment process at that location may have changed in the interim. By accounting for the structure of the problem at hand in the design of our GP kernel function, we can reduce the amount of data required to accurately model the unknown process [199, 200].

In this section, we describe the design of a spatiotemporal GP kernel that is intended to be used in ecological monitoring and similar applications. The design encapsulates two key structural assumptions about the domains of interest: first, that the unknown

process varies smoothly in both space and time, and second, that it consists of an underlying cyclic component. While the latter assumption might seem rather specific, in fact many ecological processes exhibit periodic behaviour, whether due to daily cycles of temperature, humidity or sunlight, or the ebbs and flows of tides. Processes involving more complex (periodic or aperiodic) components could be captured through alternative GP kernel designs [15], provided the relevant domain knowledge is available, as our model is compatible with any GP kernel. Thus, for our purposes, these structural assumptions are relatively general, and allow us to demonstrate the capability of our model within a relevant application context.

In the following, let $\mathbf{x} = (x, y) \in \mathbb{R}^2$, $\mathbf{x}' = (x', y') \in \mathbb{R}^2$, $s = (\mathbf{x}, t) \in \mathbb{R}^3$ and $s' = (\mathbf{x}', t') \in \mathbb{R}^3$. Our kernel function can be expressed as a product of a spatial and temporal component:

$$k(s, s') = k_x(\mathbf{x}, \mathbf{x}')k_t(t, t'). \quad (4.3)$$

The spatial component is a radial basis function (RBF), which captures correlations between locations that are near one another:

$$k_x(\mathbf{x}, \mathbf{x}') = \sigma_x^2 \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\ell_x^2}\right), \quad (4.4)$$

and the temporal component is a product of a periodic and RBF kernel:

$$k_t(t, t') = \sigma_t^2 \exp\left(-\frac{2 \sin^2(\pi |t - t'| / p)}{\ell_p^2}\right) \exp\left(-\frac{(t - t')^2}{2\ell_q^2}\right). \quad (4.5)$$

Here, σ_x and σ_t are the spatial and temporal variances respectively; ℓ_x is the spatial lengthscale; p is the period of the temporal kernel; ℓ_p is the lengthscale of the periodic component; and ℓ_q is the lengthscale of the slowly-varying component. On its own, a periodic kernel assumes *purely* periodic behaviour; in combination with the RBF kernel, an element of periodicity is preserved, but points that are further apart in time are treated as less correlated, thereby allowing for slow variation between cycles with $\ell_q \gg p$.

Each of the hyperparameters mentioned above can either be assigned fixed values, or initialised with prior distributions and then learned during the mission. In our experiments in Section 4.5, for example, we assign a fixed value to the period of $p = 24$ h,

capturing our assumption that the environment will feature some diurnal rhythm. The remaining hyperparameters are optimised between episodes, by maximising the log marginal likelihood for the model given the current history of observations.

4.4.2 Evaluating trajectory rewards

The state-dependent reward in our mission objective (Equation 4.1) depends on the unknown function f , and thus cannot be evaluated directly. We therefore introduce an *approximate reward function* similar to Morere *et al.* [103], which is based on the GP belief in lieu of the true function f , and encodes the desired trade-off between exploration and exploitation. We use the GP belief at step j , denoted g_j , to compute an upper confidence bound (UCB) [201] reward for belief state $b_j = (s_{k,j}, g_j, s_{e,j})$, given by

$$\tilde{R}_{\text{UCB}}(s_{k,j}, g_j) = \mu_j(s_{k,j}) + \kappa \sigma_j(s_{k,j}), \quad (4.6)$$

where μ_j and σ_j are respectively the mean and standard deviation of g_j at state $s_{k,j}$, and κ is a constant that affects the exploration-exploitation balance. Note that the known state $s_{k,j}$ has both spatial and temporal components, so this reward is also spatiotemporal. The UCB reward adds an exploration bonus to states with greater uncertainty, which is essential in our multi-mission setting since we are only able to plan over a single-mission horizon. Exploratory actions that seem unfavourable within the current mission may still contribute to increased reward in future missions, and the UCB reward encourages taking such actions. Over time, the uncertainty in the GP belief will decrease, so the UCB reward will converge to the true reward.

Thus, the reward accumulated over a simulated trajectory $\varsigma = s_{k,1}z_1s_{k,2}z_2 \dots s_{k,n}z_n$ using the UCB reward is given by

$$R^+(\varsigma) = \sum_{j=1}^n \tilde{R}_{\text{UCB}}(s_{k,j}, g_j), \quad (4.7)$$

where g_j is the GP posterior given dataset $z_1z_2 \dots z_j$. However, evaluating each $\tilde{R}_{\text{UCB}}(s_{k,j}, g_j)$ requires sampling an observation from the previous GP belief g_{j-1} , and updating a copy of that belief. This is computationally costly, and reduces the capacity

to perform sufficient MCTS trials to accurately estimate the value of each action at the root node.

To mitigate this, we estimate the reward accumulated during the rollout phase using the GP belief g_ℓ from the *leaf* node b_ℓ ; i.e., we do not perform belief updates during this phase. In doing so, we ignore the changes in the GP belief that would occur during each step of a rollout, with the expectation that the GP uncertainty this far into the future would be large enough that such updates would be of limited value anyway. Formally, for a trajectory $\varsigma = s_{k,1}z_1s_{k,2}z_2 \dots s_{k,n}z_n$ with $1 \leq \ell \leq n$ being the index where the search moves to the rollout phase, we define the estimated cumulative reward as:

$$\tilde{R}^+(\varsigma) = \sum_{j=0}^{\ell} \tilde{R}_{\text{UCB}}(s_{k,j}, g_j) + \sum_{j=\ell+1}^n \tilde{R}_{\text{UCB}}(s_{k,j}, g_\ell). \quad (4.8)$$

This approach removes the need to create and sample from many copies of the GP during the rollout phase, instead allowing the reward accumulated during this rollout phase to be estimated through a single GP query. The in-tree component of each MCTS iteration is still quite computationally intensive. However, since the rewards depend explicitly on belief uncertainty, this cannot be easily circumvented, and we show in Section 4.5 that we are still able to obtain good performance within reasonable computation times.

4.4.3 Planning algorithm

Our planning approach is summarised in Algorithm 4.2. Each mission i begins with the robot at location $\mathbf{x}_0 = (x_0, y_0)$, at the mission start time $t = H_s^i$. At each step, actions are selected online (Line 10) using belief-based MCTS as described above, which plans over the *current* mission horizon using a GP model based on the history of observations up to that point. The selected action is then executed (Line 11), generating a new observation which is added to the dataset and used to update the GP model (Line 12). GP hyperparameters are optimised offline during the robot's down time between missions (Line 15), improving the model over the course of the mission.

Algorithm 4.2 MULTI-MISSION MONITORING

Input: M^2 -PKSMDP $\langle \mathcal{M}, \mathcal{H} \rangle$ where $\mathcal{M} = \langle S, s_0, A, T, R \rangle$ and $\mathcal{H} = \{(H_s^i, H_f^i)\}_{i=1}^m$, GP hyperparameter priors $p_0(\theta)$, kernel $k(s, s')$

Output: GP posterior over environment feature

```

1:  $i \leftarrow 1$  (current mission)
2:  $\mathcal{D} \leftarrow \emptyset$ 
3:  $\theta \leftarrow p_0(\theta)$ 
4: Initialise GP model  $\mathcal{GP}_{\mathcal{D}, \theta}$  with kernel  $k$ , hyperparameters  $\theta$ , dataset  $\mathcal{D}$ 
5: while  $i \leq |\mathcal{H}|$  do
6:    $\mathbf{x} \leftarrow \mathbf{x}_0$ 
7:    $t \leftarrow H_s^i$ 
8:    $s \leftarrow (\mathbf{x}, t)$ 
9:   while  $t \leq H_f^i$  do
10:     $a \leftarrow \text{GETNEXTACTION}(\mathcal{M}, \mathcal{GP}_{\mathcal{D}, \theta})$ 
11:     $s', o' \leftarrow \text{EXECUTEACTION}(a)$ 
12:     $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s', o')\}$ 
13:     $s \leftarrow \text{new state } s'$ 
14:   end while
15:    $\theta \leftarrow \text{OPTIMISEHYPERPARAMETERS}(\mathcal{GP}_{\mathcal{D}, \theta})$ 
16:    $i \leftarrow i + 1$ 
17: end while

```

4.5 Experiments

4.5.1 Experimental domains

We demonstrate the performance of our algorithm on domains based on acoustic monitoring of coral reef activity using an autonomous underwater vehicle (AUV). Studies of these environments have found that sound levels at low frequencies correlate with density and diversity of coral reef fish species, which vary periodically over both daily and seasonal timescales [197, 202]. The mission goal is to focus observations of the reef on areas where bioactivity is highest, using acoustic sensing data to build a spatiotemporal GP model of bioactivity in the reef.

Synthetic domains We conduct experiments on 10 STATIC domains and 10 MOVING domains, which capture different behaviours of the unknown environment process f . Each of these domains is an 8×8 grid map with a 4-connected structure. In the STATIC domains, f consists of a set of randomly-placed Gaussian point sources at *fixed* locations, whose amplitudes oscillate periodically over 24 hours; each source is spatially static, but still varies over time. In these domains, f is therefore purely periodic—it varies over the course of each day, but each day is identical to the one before. In the

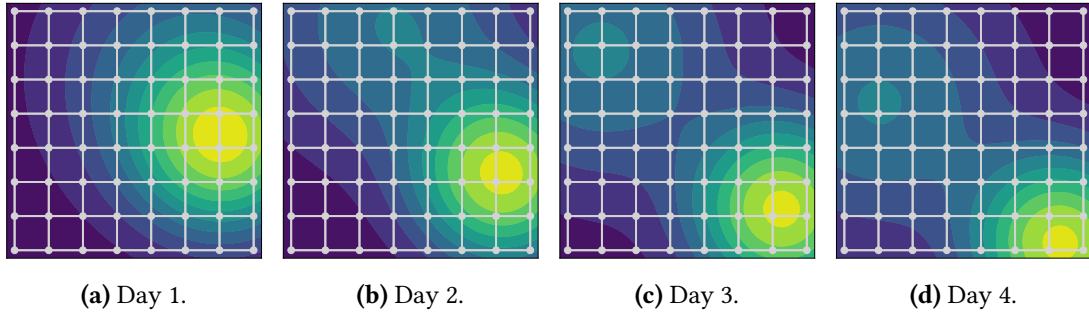


Figure 4.1: Snapshots of one of the 10 MOVING domains, shown at the same time across different days. The heatmap shows the value of the unknown process f with contours indicating regions of similar value per-day, where yellow represents the maxima.

MOVING domains, f consists of 1-2 *moving* Gaussian point sources, which oscillate in amplitude in the same manner as the static sources but also drift slowly across the map, emulating gradual seasonal variation. Figure 4.1 depicts an example of the gradual variation of the MOVING domains. The design of these domains reflects patterns seen in many ecological phenomena, which often display similar combinations of daily cycles and longer-term variation. In each case, the multi-mission duration is 20 days, and the robot is operational for 8 hours each day, performing 5 navigation actions per hour.

Coral reef acoustic domain Our SESOKO domain simulates a realistic coral reef monitoring scenario, using acoustic spectrogram data captured at three sites in a coral reef near the island of Sesoko, Japan [203]. Figure 4.4a shows a portion of the raw spectrogram data. We compute the signal power at each site as a normalised root-mean-square over frequencies from 400–600 Hz, a range within which acoustic signal strength is known to correlate with coral cover and fish density [204]. We also apply some smoothing to remove high-frequency noise. Each of the three signals is then used to modulate the amplitude of a Gaussian point source placed within a 10×10 grid environment, in order to generate a simulation of the acoustic signal across the full domain. Finally, we add a small drift velocity to one of the sources, to simulate possible long-term movement which could not be explicitly captured by the static acoustic sensor data. Figure 4.2 shows a series of snapshots of the domain over the course of a single day, in which subtle variations are visible in the intensity at each site. Over the course of a full mission, an effective planner should be able to identify which location within

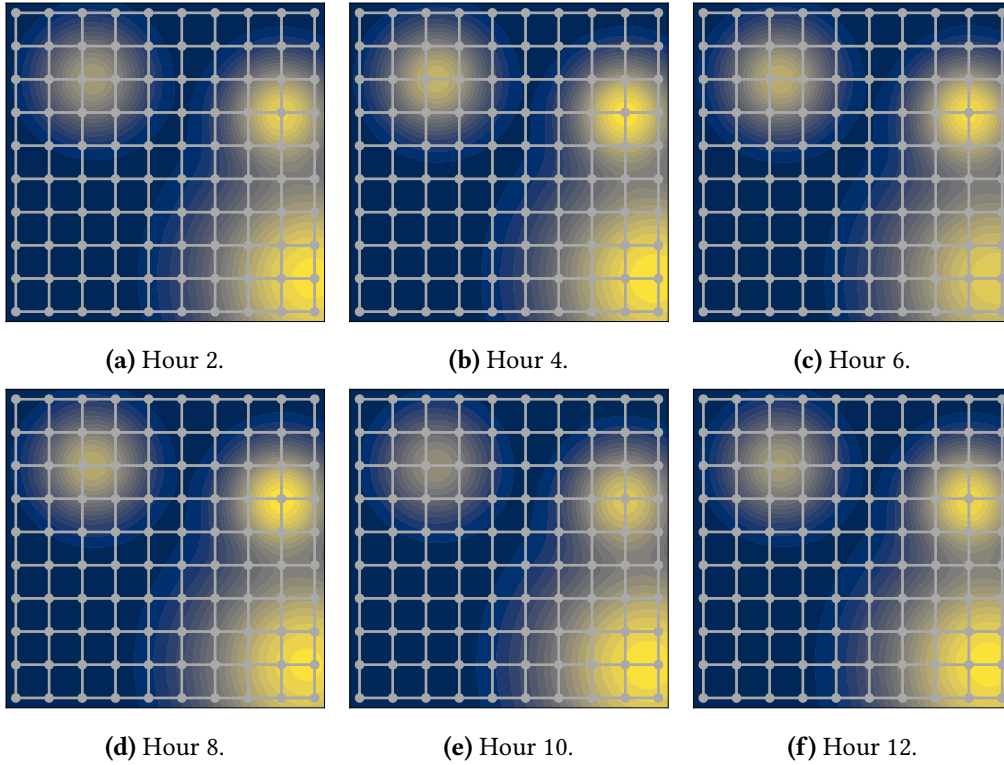


Figure 4.2: Snapshots of SESOKO reef monitoring domain, shown at intervals of two hours. The heatmap shows the value of the unknown process f with contours indicating regions of similar value per-day, where yellow represents the maxima. Subtle differences are visible in the intensity at each site over the course of the day, which can be exploited by the planner.

the domain is most advantageous to visit at a given time. These multi-missions span 30 days, with the robot active for 20 hours per day, performing 3 actions per hour.

4.5.2 Planning with a spatiotemporal GP kernel

Our first set of experiments investigates whether the spatiotemporal GP kernel described in Section 4.4.1 is able to produce improved planning behaviour in practice, compared with simpler baselines. In all experiments, we assign a fixed value of $p = 24$ h to the period of the temporal kernel. The remaining GP hyperparameters are initialised with relatively uninformative priors, and learned over the course of the missions. We run 10 multi-missions in each of the 10 STATIC and MOVING domains, testing three different GP kernel configurations:

1. **Spatiotemporal mixed (STM):** the full spatiotemporal kernel $k(s, s')$ from Equation 4.3, incorporating gradual spatial and temporal variation as well as

a periodic temporal component.

2. **Spatiotemporal periodic (STP)**: the spatiotemporal kernel with the gradual temporal variation component removed, so the kernel is purely periodic in time.
3. **Spatial static (SS)**: a purely spatial RBF kernel, corresponding to the component from Equation 4.4 only.

The results from these experiments are shown in Figure 4.3. We see that in the STATIC domains, the STP kernel achieves the best performance, since the unknown function in these domains is completely periodic and thus can be captured accurately by a kernel with a periodic temporal component. The STM kernel performs well, although its performance is reduced slightly by the presence of the non-periodic temporal component. This is as expected, since the additional term in k_t (Equation 4.5) reduces the weight of older observations in GP inference—in a purely periodic environment, this is a slight disadvantage, resulting in predictions being made using less data. Over a longer mission, we would expect to see the performance of the STM kernel converge to that of the STP kernel. The SS kernel performs worst, since it is unable to capture temporal variation; it can only identify the region of the map with the highest *average* reward across all missions.

In the MOVING domains, where the environment process evolves gradually between days, Figure 4.3 shows that the performance of both the STP and SS kernels are weaker than the STM kernel. In these domains, it is no longer possible to simply identify the best location to visit at particular times of day, since these locations change over time. However, the STM kernel is able to learn and exploit an accurate model after a few days, and continues to perform well across all subsequent missions.

Finally, Figure 4.4b shows the results of the SESOKO domain experiments. Here, we see that the planner with the STM kernel achieves the highest reward, with the SS and STP kernel performance degrading considerably in the latter half of the mission. The STM kernel captures the underlying periodicity in the acoustic data, despite significant noise and artifacts in the raw data. This demonstrates a strength of GP models in their ability to handle noisy data, effectively smoothing out this noise

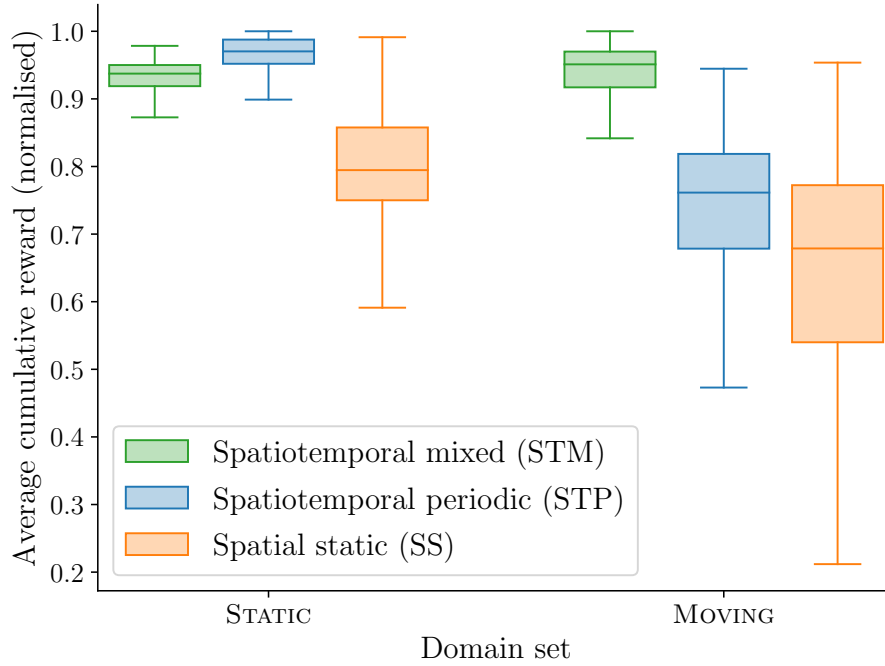


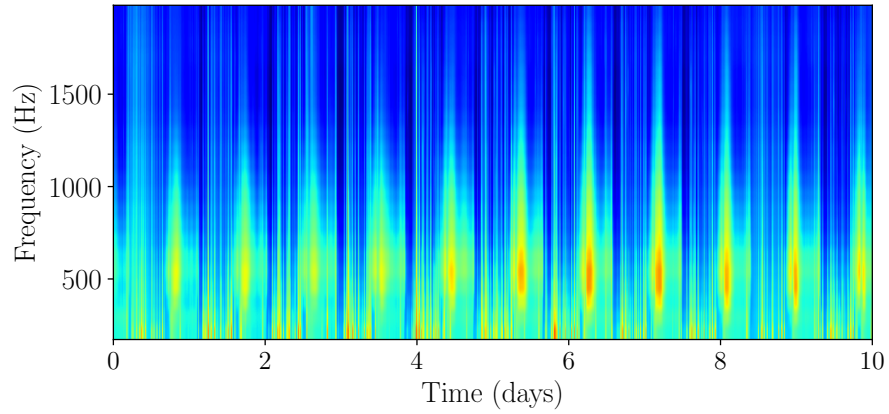
Figure 4.3: Average reward over 10 STATIC and 10 MOVING domains, using planners with different GP kernels. Rewards are normalised per-domain to the highest reward obtained on that domain across all complete runs (10 runs per domain). The STM kernel performs well in both domain groups, particularly outperforming the other two configurations in the MOVING domains.

during inference. Conversely, however, domains in which sharp transitions in the environment process occur would likely prove more challenging for the GP kernels explored here, and require more sophisticated kernel designs.

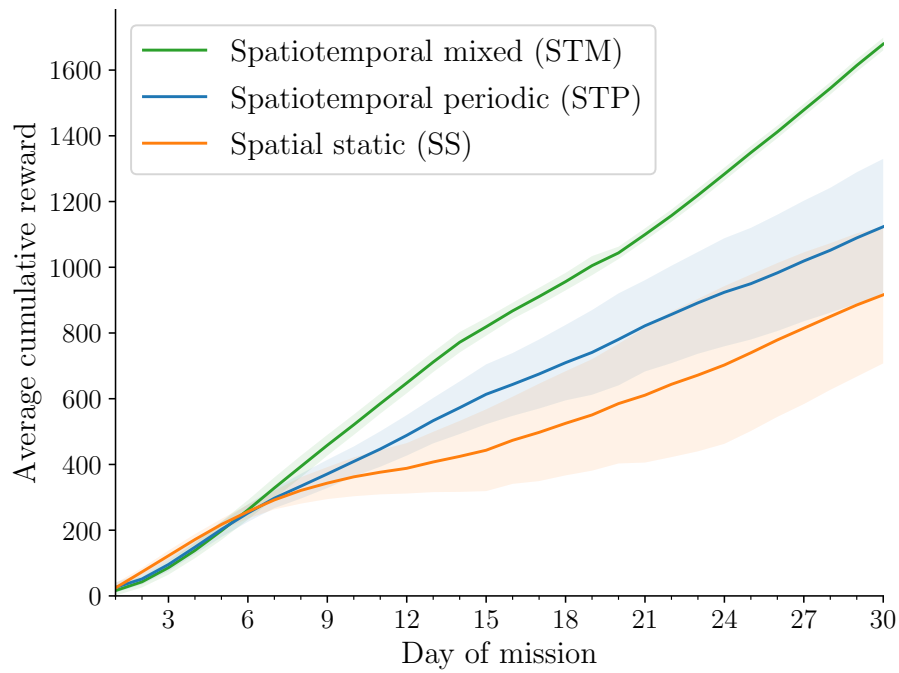
Overall, these experiments validate our GP kernel design, and more broadly demonstrates GPs as effective models for these monitoring tasks. We see that the combination of periodic and RBF kernels allows the robot to build informative models of different environments relatively quickly. These models then enable effective planning with respect to the long-term monitoring objective, despite the limited observational data and multiple layers of temporal variation present in the domains.

4.5.3 Belief-based MCTS across multiple missions

We now investigate how the core components of our M^2 -PKSMDP planner design contribute to performance. The planner uses a belief-based MCTS algorithm which rewards exploration of uncertain states using a belief-dependent UCB reward. This



(a) Frequency spectrogram data from Site C, north of Sesoko island.



(b) Average cumulative reward across 10 30-day missions in the Sesoko environment; shaded regions indicate standard deviation. This result demonstrates that the STM kernel is able to model real-world acoustic data well enough to support effective planning.

Figure 4.4: Input data and experimental results for the coral reef acoustic monitoring domain in Sesoko, Japan [203].

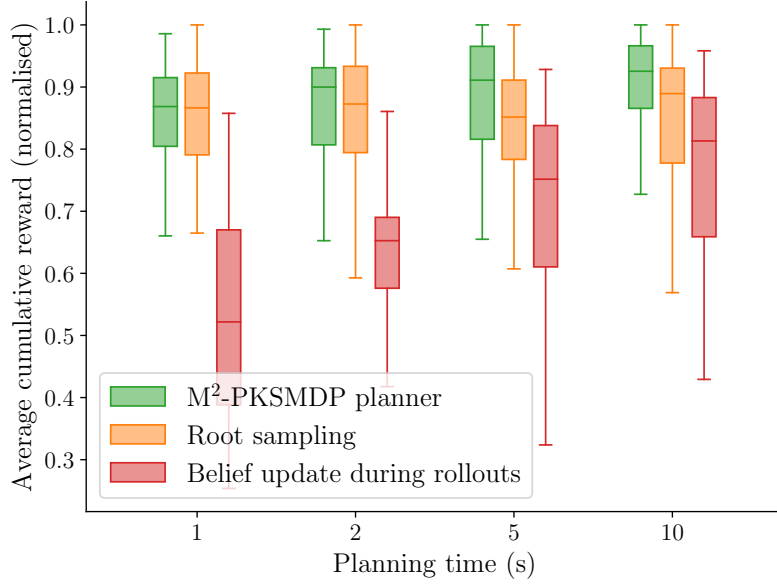
requires sampling from and updating a GP model each time a new node is added to the search tree. During the rollout phase, however, trajectory rewards are evaluated in a single batch *without* updating the GP, allowing for fast evaluation across many actions. We therefore compare our approach to two baselines which remove these elements.

The first baseline uses a root sampling approach similar to that of Budd *et al.* [34], where instead of updating the GP during MCTS, each observation is sampled directly from the belief at the root node, and no exploration bonus is used. In their work, this approach is shown to converge to the reward-maximising policy and significantly reduces the time taken for each trial. However, it does not allow for the explicit rewarding of exploration provided by the belief-dependent rewards in our approximate reward function (Equation 4.6). Thus, the policies generated by a root sampling approach will greedily maximise reward for each individual mission, without accounting for the benefits that exploration in one mission might have on performance in later missions.

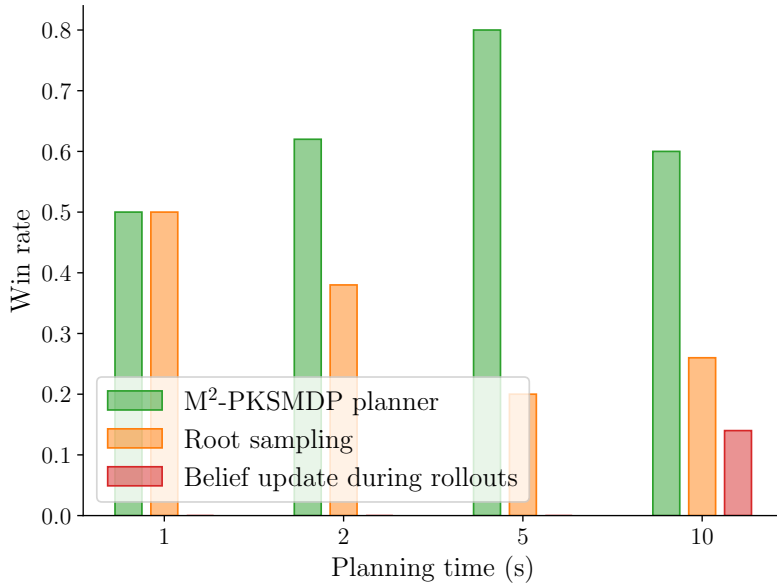
The second baseline modifies the rollout phase to update the GP after each action, rather than using the GP from the leaf node. In this configuration, each step in the rollout phase involves sampling from the GP model, updating the dataset, and then using the updated model to evaluate the reward term—similar to the process used in the in-tree part of MCTS. This should mean that the rollouts more accurately estimate the value of the leaf node, at the expense of a significantly higher computational cost.

Figure 4.5a compares the rewards obtained by the three algorithms in the MOVING domains when given different amounts of planning time per action. We see that performing GP belief updates during the rollout phase of MCTS does not improve overall performance, indicating that computation time is better spent simply performing *more* trials. The difference is most pronounced when computation time is limited, since in this case performing unnecessary belief updates results in the planner being unable to perform enough trials to select good actions. However, given sufficient computation time, the performance of this method slowly improves, as it more accurately estimates the value of each leaf node.

Figure 4.5b compares the win rates of each algorithm, indicating the proportion of runs in which each algorithm obtained a higher reward than the other two. We see



(a) Average cumulative reward (normalised per-domain).



(b) Win rate, indicating the proportion of runs on which each algorithm obtained the highest reward.

Figure 4.5: Planner performance under fixed per-action planning time. Each configuration was run 5 times on each of the 10 MOVING domains. We see that the M²-PKSMDP consistently performs well relative to the other approaches tested, although root sampling performs reasonably well when planning time is highly constrained.

that when given 10 s of planning time per decision step, the computationally expensive belief-updating method is able to achieve the highest reward on some runs. However, this is a significant amount of additional planning overhead for a relatively small domain, and as such it is likely that this approach would not be computationally feasible in larger domains. Furthermore, our planner with a planning budget of 1 s still outperforms the planner performing GP updates during rollouts when given a budget of 10 s. The root sampling baseline performs similarly to the M^2 -PKSMDP planner when planning time per action is limited to 1 s. However, as we increase the planning time, we see that the M^2 -PKSMDP planner outperforms the root sampling method, obtaining higher average cumulative reward and a win rate of at least 60% in all cases.

From these results, we can conclude that the belief-based exploration reward has the intended effect of encouraging exploration, to the benefit of overall performance on long-term missions. However, to achieve this improvement, updating of the GP belief should be done only during the in-tree component of MCTS. During the rollout phase, it is preferable to use a fixed belief in order to achieve faster computation and allow for *more trials* to be completed.

4.6 Discussion

The multi-mission framework adopted in this chapter provides a planning solution motivated by the practical constraints of long-term robot missions. Despite the advancements in robot technology to date, few robot systems are able to achieve long-term, uninterrupted autonomy in unstructured environments. This is due to many factors, including the need for a high level of robustness in challenging environments, and the costs associated with the required infrastructure. Thus, it is likely that in the near future, repeated deployments will remain a common way to enable long-term monitoring with mobile robots.

Our experiments demonstrate that GPs are a viable modelling tool for this type of long-term mission. In our experiments, we see that with a well-chosen GP kernel, we can achieve consistent monitoring performance across a range of simulated domains. Moreover, as previously discussed, clear avenues exist towards both scaling GPs to larger

problems (see Section 3.1.4), and to modelling more complex environment processes (see Section 4.4.1). In more complex time-varying environments, it is possible that GPs might struggle to provide useful predictions from the sparse data available from a robot’s local measurements. However, this could in turn be mitigated by incorporating additional sources of information as appropriate to the particular context, such as weather forecasts, or data from satellite imagery or static sensors. Although in this work we assumed that data from previous missions was not available, if such data were available, it could be a valuable source of information which could be used to improve the GP model.

GPs also offer additional flexibility which is not explicitly leveraged in our work. Our approach considers observations to be taken at waypoints only, but since GPs are continuous models, they can be queried at any point in their continuous input space. Allowing the robot to take more frequent observations during navigation would be a straightforward way to improve the rate of data collection, and thus the quality of the GP model. However, since these data points would be highly correlated due to their proximity in the input space, it is also possible that the model would not benefit significantly from the additional data.

Trial-based planning algorithms in general face a trade-off when allocating computational resources, between optimising the accuracy of the value estimates versus performing more trials. The baselines used in our experiments in Section 4.5.3 can be viewed as two opposite ends of this spectrum. The root sampling planner reduces the cost of each trial by using a fixed GP belief, allowing for more trials within the same computation budget by sacrificing some accuracy in its value estimates. On the other hand, performing belief updates during rollouts makes trials extremely computationally costly, but improves the accuracy with which they simulate the outcomes of future actions. Our planner strikes a balance between these two extremes by performing belief updates only during the in-tree component of MCTS, and using a fixed belief during rollouts. Our results show empirically that this approach produces better performance than either of the baselines, achieving higher rewards within the same computation budget. However, the optimal balance between these extremes will vary depending on the specific domain and objective, as well as the computational resources available.

It is likely that in some domains, the root sampling approach would be sufficient to obtain reasonable performance, particularly since our results show that it performs relatively well with limited planning time. More broadly, the planner would benefit from validation in larger-scale environments in order to assess its scalability in practice, in terms of both GP modelling and MCTS planning.

4.7 Summary

In this chapter, we introduced the M^2 -PKSMDP, a formalism for long-term, repeated robot monitoring missions in the presence of *a priori* unknown environment processes. We addressed **C1 Environment Modelling** using a GP kernel designed to capture spatiotemporally-varying processes, motivated by applications such as ecological monitoring. We presented a continuous-observation MCTS planning algorithm, which uses a GP belief to plan online, balancing exploitation of observations from past missions with exploration to benefit future ones. This algorithm used a novel technique for efficient estimation of the reward accumulated during rollouts, thereby mitigating the computational cost of GP belief updates and allowing more MCTS trials to be performed. Additionally, we incorporated online learning of GP hyperparameters to address **C4 Limited Priors**, reducing the need for manual tuning or domain-specific prior knowledge. We demonstrated the performance of our planner empirically in a series of simulated experiments, including in a coral reef domain using real acoustic monitoring data.

5

Hierarchical Planning for Resource-Constrained Long-Term Monitoring

Contents

5.1	Introduction	68
5.2	Problem formulation	70
5.3	Algorithm	71
5.3.1	Overview	71
5.3.2	High-level planning: Depot-MDP	73
5.3.3	Low-level planning: belief-based MCTS	76
5.4	Experiments	78
5.4.1	Planner configurations	78
5.4.2	Experimental domains	79
5.4.3	Scheduling of active hours	80
5.4.4	High-level planning horizon	82
5.4.5	High-level planning step	84
5.4.6	UAV wildfire monitoring	84
5.5	Discussion	87
5.6	Summary	88

5.1 Introduction

In Chapter 4, we considered the long-term monitoring problem as a sequence of shorter missions. This addressed the common scenario in which robots rely on human assistance for resource management tasks such as recharging batteries or offloading data. However, as robot technology improves, it is becoming increasingly feasible for robots to manage their resources independently. Given the appropriate infrastructure, robots can dock autonomously to recharge their batteries or offload data. While this does introduce additional requirements to the mission, in both the robot hardware and the infrastructure, it also comes with two key advantages. The first is that autonomous resource management can remove the need for regular human intervention, thereby reducing the running costs of the mission. The second, more interestingly from a planning perspective, is that the flexibility for a robot to manage when and where to recharge its resources can lead to better mission outcomes. Moreover, introducing multiple recharging locations can greatly expand the area that is accessible to a single robot, in much the same way that charging networks extend the moving range of electric vehicles [179].

In this chapter, we therefore turn our attention to **C2 Resource Constraints**, and consider *continuous* long-term deployments in which the robot must manage its own finite resources. The mission objective is as before: to maximise the cumulative value of observations of an environment process over a long horizon. The robot has a constrained resource, such as battery life or available on-board memory, which depletes steadily while it is active. The robot can spend time at a *depot* to replenish this resource. Figure 5.1 depicts an application domain in wildfire monitoring, where an unmanned aerial vehicle (UAV) can be used to provide real-time observations of areas where the fires are most severe.

There are three key challenges when it comes to planning in this setting. First, the robot must build and maintain an accurate probabilistic model of the environment from sparse, noisy, local observational data (**C1 Environment Modelling**). This was a focus of Chapter 4, where we demonstrated that our spatiotemporal GP model provided an effective modelling tool in this context. Second, the robot must then leverage this model

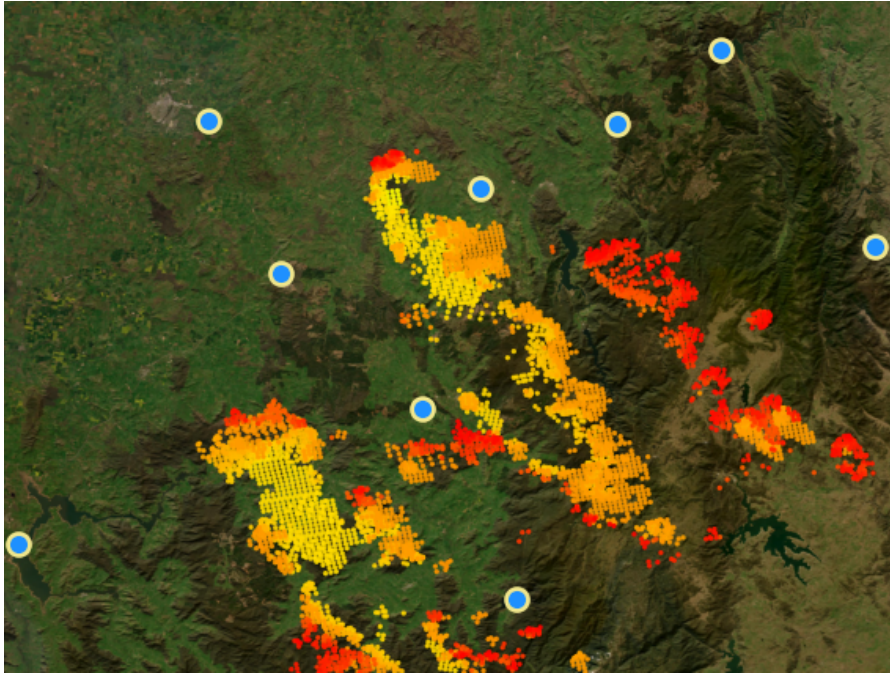


Figure 5.1: The progression of a wildfire in the Kosciuszko National Park in Australia. Yellow-red markers are wildfire hotspots detected via satellite over a period of 20 days, coloured by time of detection to show the evolution of the fires. Blue waypoints indicate weather stations where a monitoring UAV can recharge its batteries. Satellite imagery from [205].

to plan in a long-term manner, balancing short-term reward-seeking with exploration that may benefit later stages of the mission. In Chapter 4, we considered planning horizons as long as only a single resource cycle—here, we look to extend this to multiple cycles, enabling effective planning in much larger environments. Finally, the robot must manage its constrained resources, incorporating resource replenishment into its schedule in order to achieve the best overall performance (**C2 Resource Constraints**). At its most basic level, this final challenge can be handled simply by ensuring that the robot returns to a base periodically to replenish its resources. However, as we shall see in this chapter, there can be significant benefits to a more flexible approach to resource management.

Our contribution in this chapter is a hierarchical planning framework for long-term robot monitoring of an unknown environment process under resource constraints. This is a direct extension to the work presented in Chapter 4. We introduce a more flexible model for resource management, in which the robot must periodically plan to visit depots to replenish its resources. We then adopt a hierarchical planning approach to enable true long-term planning over multiple resource cycles. In this problem formulation,

resource management is a key component of the planning problem—the robot must decide *when* and *where* to replenish resources in order to maximise the reward obtained while active. To the best of our knowledge, this work is the first to consider long-term monitoring of an unknown environment process with resource management integrated into planning, and the first to tackle planning over a GP-modelled process with a hierarchical approach. This combination of elements yields a challenging problem, since the hierarchy introduces the need to extract information from the model at multiple levels of abstraction, and the resource constraints influence the exploration-exploitation trade-off available to the planning agent. We show that our approach can leverage the GP model to intelligently manage the resource constraints and plan over long horizons, and demonstrate the benefits of doing so through a series of experiments in synthetic domains and a wildfire monitoring application. The work in this chapter first appeared in [11], presented at ECAI 2024.

5.2 Problem formulation

Similar to Chapter 4, we consider a mobile robot deployed in a physical environment represented as a topological map $\mathcal{G} = \langle V, E \rangle$. Now, the robot has some finite resource, which it consumes as it navigates through the environment (for example, battery charge). This resource can be replenished at a set of *depots* $D \subseteq V$, which we assume to represent a small number of locations within the full topological graph. As in the previous chapter, we consider the navigation and resource dynamics to be *deterministic*, with navigation described by $\mathcal{G}$. That is, any action taken by the robot will result in a known state transition: the action duration, resulting state, and resource consumption are all known in advance. In Chapter 6, we will relax some of these assumptions and consider actions with stochastic outcomes.

While active, the robot’s resource depletes at a constant rate $C_{\downarrow}$, and while replenishing it increases at a constant rate $C_{\uparrow}$. The continuous resource level $b \in [0, 1]$ represents the battery level of the robot, with $b = 0$ corresponding to an empty battery, and $b = 1$ a full charge. Any non-depot state where $b = 0$ is a sink state. As previously, we consider an *a priori* unknown spatiotemporal mapping $f : \mathbb{R}^3 \rightarrow \mathbb{R}$, where $f(x, y, t)$

is the value of an environment feature of interest at location (x, y) and time t . The mission goal is to maximise the cumulative reward from observations taken of f across a long horizon H which corresponds to many days. We model this problem as a *Long-Term Monitoring PKSMDP*.

Definition 5.1 (Long-Term Monitoring PKSMDP (LTM-PKSMDP)) A Long-Term Monitoring PKSMDP is a tuple $\mathcal{M} = \langle S, s_0, A, T, R, H \rangle$ with $S = S_k \times S_e$, where $S_k = \{(x, y, t, b) \in \mathbb{R}^3 \times [0, 1] \mid (x, y) \in V \text{ and } 0 \leq t \leq H\}$ is the known state component; $S_e = \mathbb{R}$ is the unknown state component; $s_0 = (x_0, y_0, 0, 1, f(x_0, y_0, 0))$ is the initial state, with $(x_0, y_0) \in V$ being the initial location; $A = E \cup \{\text{CHARGE}\}$ is the action space; $T : (S_k \times S_e) \times A \times S_k \rightarrow [0, 1]$ is the transition function, which evolves the state according to the topological map and resource dynamics, depending on which action is selected; $R : S \times A \rightarrow \mathbb{R}$ is a reward function of the environment process f ; and H is the mission horizon.

As in the previous chapter, we take the reward function to be the value of the unknown process, i.e., $R(s_k, f(s_k)) = f(s_k)$.

5.3 Algorithm

5.3.1 Overview

The planning problem described above presents challenges through the computational demands of planning under uncertainty at scale, over large maps and long mission horizons, as well as management of the limited resources. In missions such as the UAV wildfire monitoring one depicted in Figure 5.1, effective exploration and continued monitoring of the environment requires some degree of *long-term* planning—where long-term, in this case, means spanning many resource depletion cycles. This is necessary because the UAV cannot traverse the whole environment in a single battery charge, so in order to plan at the scale of the full environment, it must plan over multiple resource cycles. Plans must incorporate periodic visits to depots in order to ensure that the UAV’s resources do not run out mid-flight, while still serving the overall mission goal. However, as discussed in Section 3.2.2, planning in belief space over long sequences of actions is

Table 5.1: Division of responsibilities between the high-level and low-level planner.

	Low-level planner	High-level planner
Planning horizon	Single resource cycle	Multiple resource cycles
State space	Full topological map	Depots only
Resource-aware	No	Yes
Belief-aware	Yes	No

extremely challenging, due to the large search space. We therefore take a hierarchical approach to planning, which introduces some abstraction between different components of the problem. Our planner uses a two-level hierarchy, which is summarised in Table 5.1.

The *high-level planner* is responsible for long-term planning and resource management. It plans over a horizon of multiple resource cycles, using a simplified model of the physical environment which considers *only* depot locations in its state space. Policies generated by the high-level planner specify *high-level actions*. These can either be resource replenishment actions, or navigation actions with depots as destinations, in each case with an associated duration. Resource replenishment actions are primitive, and can be executed directly. Navigation actions are macro actions, specifying a destination depot and an action duration—these are passed down to the *low-level planner*, which plans the specific sequence of actions to take in order to reach the destination in time.

The low-level planner does not need to directly consider resource management, as this is already captured by the high-level action choice. It plans over a shorter horizon, which is at most the time taken to deplete the robot’s resources once, allowing for more fine-grained reasoning about beliefs over the environment feature. Once a high-level action is executed (either directly, or by delegation to the low-level planner), a new high-level plan is generated to get the next action. This process repeats until the mission is completed (i.e., no more valid high-level actions can be executed within the mission horizon). Figure 5.2 depicts our hierarchical planning architecture, and we describe the components in detail below.

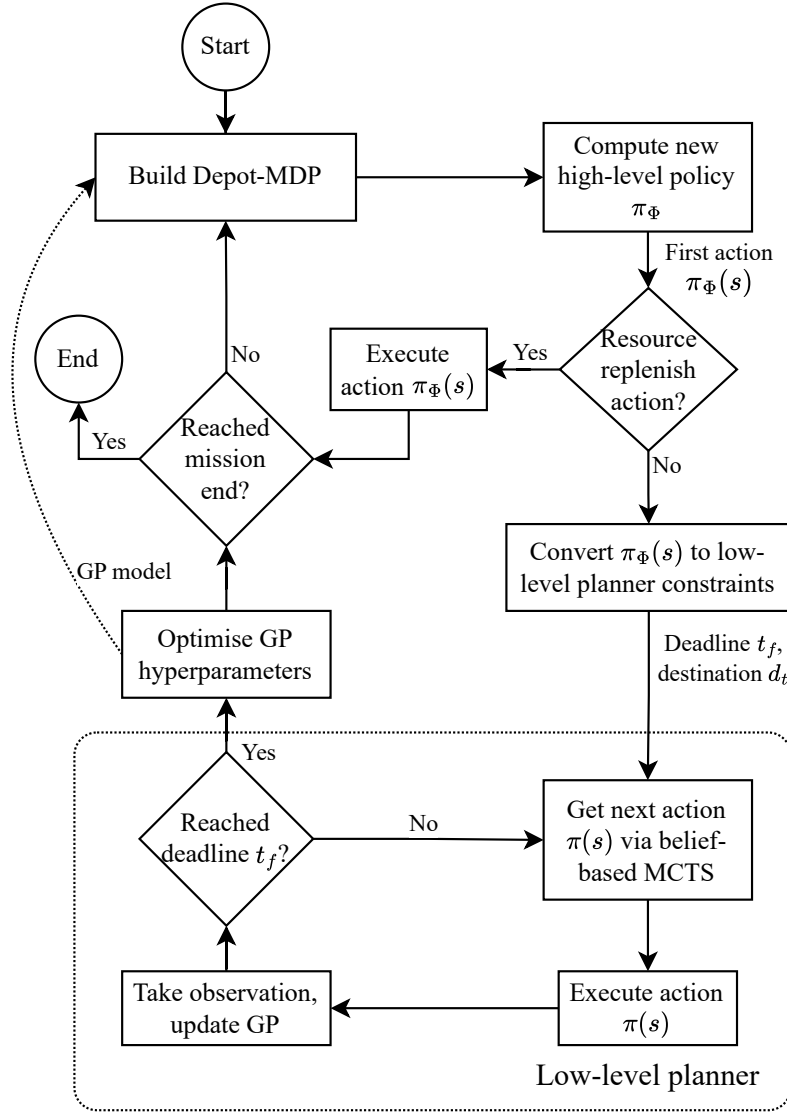


Figure 5.2: Hierarchical planner architecture.

5.3.2 High-level planning: Depot-MDP

The high-level planner builds a simplified representation of the full planning problem, which we refer to as a *Depot-MDP*. The Depot-MDP considers *only* depot locations, as it is used to plan the sequence of depots that the robot should visit, and not the specific navigation actions required to reach them. It uses a discrete representation of time and the resource level, with step values Δ_t and Δ_b respectively. We denote the maximum discrete resource level by $B_{\Phi, \max} = \lfloor 1/\Delta_b \rfloor$. The time step Δ_t can be much larger than the timescale of individual navigation actions, since the high-level planner is not responsible for decisions at this timescale. It must, however, be no larger

than the minimum amount of time required to fully drain or replenish the robot's resources, i.e. $\max(C_\uparrow, C_\downarrow)\Delta_t \leq B_{\Phi, \max}$.

We now present a formal definition of the Depot-MDP.

Definition 5.2 (Depot-MDP) For a given LTM-PKSMDP $\mathcal{M}$, planning horizon H_Φ , timestep Δ_t , and resource step Δ_b , the corresponding Depot-MDP is given by $\mathcal{M}_\Phi = \langle S_\Phi, \bar{s}_0, A_\Phi, T_\Phi, R_\Phi, H_\Phi \rangle$, with each component defined as below.

State space and planning horizon $S_\Phi = D \times T \times B$ is the discrete state space of the Depot-MDP, where D is the set of depots, $T = [0, \dots, H_\Phi]$ is the discrete time up to horizon H_Φ , and $B = [0, \dots, B_{\Phi, \max}]$ is the discrete resource level. The initial state $\bar{s}_0 \in S_\Phi$ corresponds to the initial state s_0 in the full PKSMDP.

Action space A_Φ is the discrete action space of the Depot-MDP, which consists of resource replenishment actions and movement actions. REPLENISH actions are available at any $d \in D$, with durations in the range $[1, \dots, \lfloor 1/(C_\uparrow \Delta_t) \rfloor]$. At the start of each mission, the shortest path durations between each pair of depots are precomputed, using the Floyd–Warshall algorithm [206] or similar. Then, for each $\{d_1, d_2 \in D \mid \text{SHORTESTPATH}(d_1, d_2) \leq 1/C_\downarrow\}$, two movement actions are defined: a TRAVERSE action $a_{\text{tr}}(d_1, d_2)$ has duration equal to the *minimum* number of Δ_t timesteps required for the journey, so $\text{dur}(a_{\text{tr}}(d_1, d_2)) = \lceil \text{SHORTESTPATH}(d_1, d_2) / \Delta_t \rceil$; and an EXPLORE action $a_{\text{ex}}(d_1, d_2)$ with duration equal to the *maximum* number of such steps within a single battery charge, so $\text{dur}(a_{\text{ex}}(d_1, d_2)) = \lfloor 1/(C_\downarrow \Delta_t) \rfloor$.

Transitions $T_\Phi : S_\Phi \times A_\Phi \rightarrow S_\Phi$ is a deterministic transition function which specifies the outcome of taking an action $\bar{a} \in A_\Phi$ from state $\bar{s} \in S_\Phi$.

Rewards $R_\Phi : S_\Phi \times A_\Phi \rightarrow \mathbb{R}$ is a reward function. To build the Depot-MDP, we need to estimate the reward that the robot will obtain for each navigation action, since the true reward values are not known in advance. We do this by using a constrained random walk algorithm to generate a set of candidate paths for each action in a computationally efficient way. The rewards over these paths are then estimated using the current GP

model of the environment. Our approach is summarised in Algorithm 5.1. This algorithm generates n paths from depot d_1 to d_2 by starting at d_1 and iteratively selecting a random valid successor—defined here as a state from which d_2 could still be reached before the action duration had elapsed, if the shortest path were followed (Lines 6-10). The reward at each step is evaluated as the UCB of the GP posterior (Line 12) at the time of the Depot-MDP construction, where this GP is *not* updated during the evaluation process. By using the UCB reward, as in Chapter 4, we encourage exploration in areas where the model has high uncertainty. We take the reward for the MDP to be the *maximum* over the n paths (Line 15), again in order to encourage exploratory behaviour. This method provides an approximation to the true value of different paths through the LTM-PKSMDP, which can be computed quickly for a large number of paths while remaining sufficiently informative to guide exploration at a high level.

Algorithm 5.1 COMPUTING DEPOT-MDP REWARDS

Input: LTM-PKSMDP $\mathcal{M}$, GP model $\mathcal{GP}$, depots $d_1, d_2 \in D$, initial time t_{init} , deadline t_f , number of trials n .

Output: Reward value $r \in \mathbb{R}$

```

1: procedure ESTIMATEREWARD( $\mathcal{M}, \mathcal{GP}, d_1, d_2, t_{\text{init}}, t_f, n$ )
2:   for  $j \in \{1, \dots, n\}$  do
3:      $r_j \leftarrow 0$ 
4:      $s \leftarrow d_1, t$  (initial state)
5:     while true do
6:        $A_v \leftarrow \text{GETAVAILABLEACTIONS}(\mathcal{M}, s, d_2, t_f)$ 
7:       if  $A_v = \emptyset$  then
8:         break
9:       end if
10:       $a \leftarrow \text{RANDOMCHOICE}(A_v)$ 
11:       $s \leftarrow \text{GETACTIONOUTCOME}(s, a)$ 
12:       $r_j \leftarrow r_j + \text{UCB}(\mathcal{GP}, s)$ 
13:    end while
14:  end for
15:  return  $\max_j(r_j)$ 
16: end procedure

17: procedure GETAVAILABLEACTIONS( $\mathcal{M}, n^B, d_t, t_f$ )
18:   $A_v \leftarrow \emptyset$ 
19:  for  $a \leftarrow \text{GETENABLEDACTIONS}(\mathcal{M}, s)$  do
20:     $s' \leftarrow T(s, a)$ 
21:    if  $t(s) + \text{dur}(a) + \text{SHORTESTPATH}(\mathcal{G}, s', d_t) \leq t_f$  then
22:       $A_v \leftarrow A_v \cup \{a\}$ 
23:    end if
24:  end for
25:  return  $A_v$ 
26: end procedure

```

The Depot-MDP can be solved using standard MDP solution techniques to obtain a high-level policy. The action specified by the policy for the current state is then passed as a constraint to the low-level planner, specifying a target depot $d_t \in D$ and the deadline t_f by which it must be reached. We assume that the low-level planner makes use of all of the available time for each high-level action. Once this action is executed, a new Depot-MDP is created and solved again for the next action, until the full mission is completed.

5.3.3 Low-level planning: belief-based MCTS

The low-level planner is a modified version of the planner from Chapter 4, which includes additional inputs representing the high-level action constraints. In effect, it solves a simplified version of the full planning problem represented by the LTM-PKSMDP, in which the planning horizon is much shorter and resource management is not explicitly considered. The low-level planning algorithm is summarised in Algorithm 5.2, with the changes from the general planner in Algorithm 4.1 highlighted.

The key change from the previous planner is that during the selection and rollout phases, an additional filtering step ensures that any selected actions will be feasible given the high-level action constraints. Each high-level navigation action specifies a target depot d_t and a deadline t_f . The modified `GETAVAILABLEACTIONS` method (called on Line 24) returns only the available actions which, once executed, would still leave the robot with enough time to reach depot d_t before time t_f . This ensures that the low-level plan will satisfy the resource constraint encoded in the high-level action. In the case of `TRAVERSE` actions, d_t is a tight deadline, so the path followed will be approximately a shortest path. `EXPLORE` actions, on the other hand, give the low-level planner more time to explore while still reaching d_t before the deadline.

During the rollout phase of MCTS, reward at each step is evaluated as the UCB of the GP model from the corresponding leaf node, as in Chapter 4. The UCB reward is used again to encourage exploration, since the full mission extends beyond the horizon of the low-level plan. This hierarchical approach gives the flexibility needed to intelligently tackle the long-term monitoring problem. In the early stages of a mission, uncertainty across the map is high, so the high-level planner will generate policies that guide the

Algorithm 5.2 LOW-LEVEL PLANNING

Input: LTM-PKSMDP $\mathcal{M} = \langle S, s_0, A, T, R, H \rangle$, current GP model $\mathcal{GP}$ (dataset $\mathcal{D}$, hyperparameters θ , kernel $k(s, s')$), **target depot** d_t , **deadline** t_f .

Output: Updated state and dataset.

```

1: procedure EXECUTEHIGHLEVELACTION( $\mathcal{M}, \mathcal{GP}, d_t, t_f$ )
2:   while  $t \leq t_f$  do
3:      $a \leftarrow \text{GETNEXTACTION}(\mathcal{M}, \mathcal{GP}, d_t, t_f)$ 
4:      $s', o' \leftarrow \text{EXECUTEACTION}(a)$ 
5:      $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s', o')\}$ 
6:      $s \leftarrow \text{new state } s'$ 
7:   end while
8:   return  $s, \mathcal{D}$ 
9: end procedure

```

```

10: procedure GETNEXTACTION( $\mathcal{M}, \mathcal{GP}, d_t, t_f$ )
11:    $n_{\text{root}}^B \leftarrow \text{INITIALISETREE}(\mathcal{M}, \mathcal{GP})$ 
12:   for  $i \leftarrow 1$  to  $N_{\text{trials}}$  do
13:      $n_{\text{leaf}}^B, r_{\text{tree}} \leftarrow \text{SELECTNODE}(\mathcal{M}, \mathcal{GP}, n_{\text{root}}^B, d_t, t_f)$ 
14:      $r_{\text{rollout}} \leftarrow \text{PERFORMROLLOUT}(n_{\text{leaf}}^B, d_t, t_f)$ 
15:      $r_{\text{total}} \leftarrow r_{\text{tree}} + r_{\text{rollout}}$ 
16:      $\text{BACKPROPAGATE}(n_{\text{leaf}}^B, r_{\text{total}})$ 
17:   end for
18:   return  $\text{BESTCHILD}(n_{\text{root}}^B)$ 
19: end procedure

```

```

20: procedure SELECTNODE( $\mathcal{M}, \mathcal{GP}, n_{\text{root}}^B, d_t, t_f$ )
21:    $n^B \leftarrow n_{\text{root}}^B$ 
22:    $r_{\text{tree}} \leftarrow 0$ 
23:   while  $\text{depth}(n^B) < d_{\text{max}}$  do
24:      $A_v \leftarrow \text{GETAVAILABLEACTIONS}(\mathcal{M}, n^B, d_t, t_f)$ 
25:      $n_{\text{child}}^{BA} \leftarrow \text{SELECTACTIONPUCT}(n^v, A_v)$ 
26:     if  $n_{\text{child}}^{BA} = \emptyset$  then
27:       return  $n^B, r_{\text{tree}}$ 
28:     end if
29:      $n_{\text{child}}^B, r_{\text{child}} \leftarrow \text{GENERATEBELIEFCHILD}(n_{\text{child}}^{BA})$ 
30:      $n^B \leftarrow n_{\text{child}}^B$ 
31:      $r_{\text{tree}} \leftarrow r_{\text{tree}} + r_{\text{child}}$ 
32:   end while
33:   return  $n^B, r_{\text{tree}}$ 
34: end procedure

```

► See Algorithm 5.1.

```

35: procedure PERFORMROLLOUT( $n^B, d_t, t_f$ )
36:    $s \leftarrow s(n^B)$ 
37:   while  $t(s) \leq d_t$  do
38:      $A_v \leftarrow \text{GETAVAILABLEACTIONS}(s, d_t, t_f)$ 
39:      $a \leftarrow \text{RANDOMCHOICE}(A_v)$ 
40:   end while
41:   return  $A$ 
42: end procedure

```

robot to explore it—the low-level planner is short-sighted, and so cannot do this on its own. Later, as the GP model improves, the high-level planner will generate actions more focused on reward-gathering in high-reward areas of the map, which are then executed by the low-level planner.

5.4 Experiments

We now present a series of experiments to explore the capabilities of our planner. We first test the planner’s scheduling and long-term planning abilities in synthetic gridworld domains. Then, in Section 5.4.6, we deploy the planner in a realistic simulation of a wildfire monitoring scenario.

5.4.1 Planner configurations

In order to assess the effectiveness of our planning approach, we present a series of experiments comparing it to two other planner configurations. We refer to the full hierarchical planning system described in Section 5.3 as the LTM-PKSMDP planner. The FIXED planner removes the flexible scheduling component of the full planner, using a restricted version of the LTM-PKSMDP with $\Delta_t = 12$ h; this puts the robot on a fixed schedule, where it is ACTIVE for 12 hours starting from the same time every day, and then REPLENISHING for 12 hours. The high-level planning horizon H_Φ for this configuration may still span multiple days. The final configuration, BASIC, implements the planner from Chapter 4. The BASIC planner operates on the same schedule as the FIXED planner, but uses only the low-level planner each day, thus removing the hierarchical component entirely. In this configuration, since there is no high-level planner, the low-level planner’s actions are instead constrained only to ensure that the robot concludes each ACTIVE period at a depot. This is done by pruning any action that would make it impossible for the robot to reach a depot before its resources ran out.

In all experiments, we use the robot’s battery charge as the constrained resource, letting $b = 1$ represent a charge capacity that can support 12 hours of activity, and $C_\uparrow = C_\downarrow = 1$ unit per hour—that is, the robot’s battery supports 12 hours of continuous

activity, and charges at the same (constant) rate at which it discharges. Each algorithm uses a GP with a product spatiotemporal kernel, as described in Section 4.4.1.

5.4.2 Experimental domains

Synthetic domains

Each synthetic domain is an $n \times n$ gridworld, where the unknown feature f is composed of a set of Gaussian point sources. These may oscillate in amplitude throughout the day and drift slowly around the map, as in the MOVING domains of the previous chapter. They are designed to reflect general behaviours that can be seen across a variety of ecological domains. The shorter-term movements represent changes that might be seen in tracking a herd of animals, for example, or observing local weather patterns. The longer-term drift is intended to capture slower changes, such as seasonal patterns.

Wildfire monitoring domain

Our wildfire monitoring problem instance uses fire hotspot data from the MODIS dataset [85] of the NASA Fire Information for Resource Management System (FIRMS) database. This dataset provides information about fire hotspots identified by satellite imagery in the mid-infrared range at a spatial resolution of 1 km. The temporal resolution of the data is somewhat limited, however, and we discuss in Section 5.4.6 how this might affect the observed results.

We simulate the wildfire by sampling the location and brightness of hotspots in Kosciuszko National Park in Australia from the MODIS dataset, between December 2019 and January 2020. These points are used to train a GP model with an RBF kernel, which simulates the “ground truth” of the environment process in our experiments. The robot is able to sample noisy observations from this simulated model during exploration. This process is required to convert the raw data into a form that can be sampled over continuous space and time. Figure 5.4 shows snapshots of our ground truth GP model on different days of the mission, revealing the general evolution of the wildfires over time.

We designate local weather stations as depots for battery charging, based on the assumption that a real-world deployment of a UAV for wildfire monitoring would aim

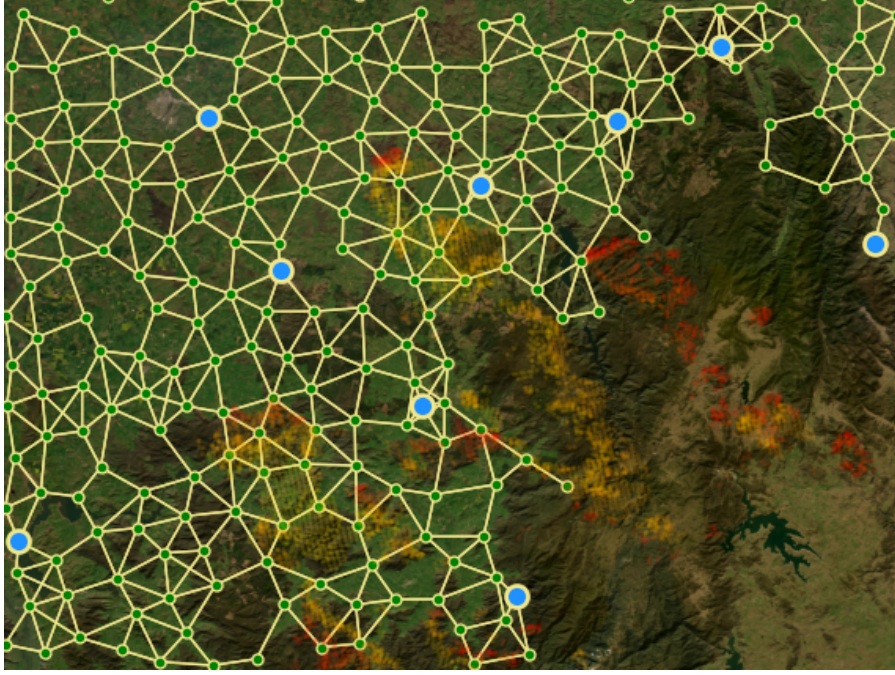


Figure 5.3: Problem domain for UAV wildfire monitoring in Kosciuszko National Park, Australia. The topological map is shown, with blue waypoints representing depots (weather stations). Wildfire hotspots from the NASA FIRMS database are shown faintly in yellow/red. Satellite imagery from [205].

to make use of as much existing infrastructure as possible in order to reduce setup and maintenance costs. The UAV is given a top speed of 4 km h^{-1} and battery life of 12 h, akin to that of a solar-powered surveillance drone such as the SolarXOne made by XSun [207]. Waypoints are then generated using Poisson disk sampling [208] within areas of the environment that are accessible based on its topography. This gives the map shown in Figure 5.3, which spans approximately $110 \text{ km} \times 90 \text{ km}$. The UAV is deployed on a 20-day mission, with rewards corresponding to observed hotspot brightness—that is, the mission objective is to provide real-time situational awareness of areas where the wildfires are most severe.

5.4.3 Scheduling of active hours

This experiment is designed to demonstrate the ability of the LTM-PKSMDP planner to schedule battery charging based on the reward-gathering objective. If more reward is available at particular times of day, the planner should learn to schedule the robot’s active hours to coincide with these times, and perform charging when the available

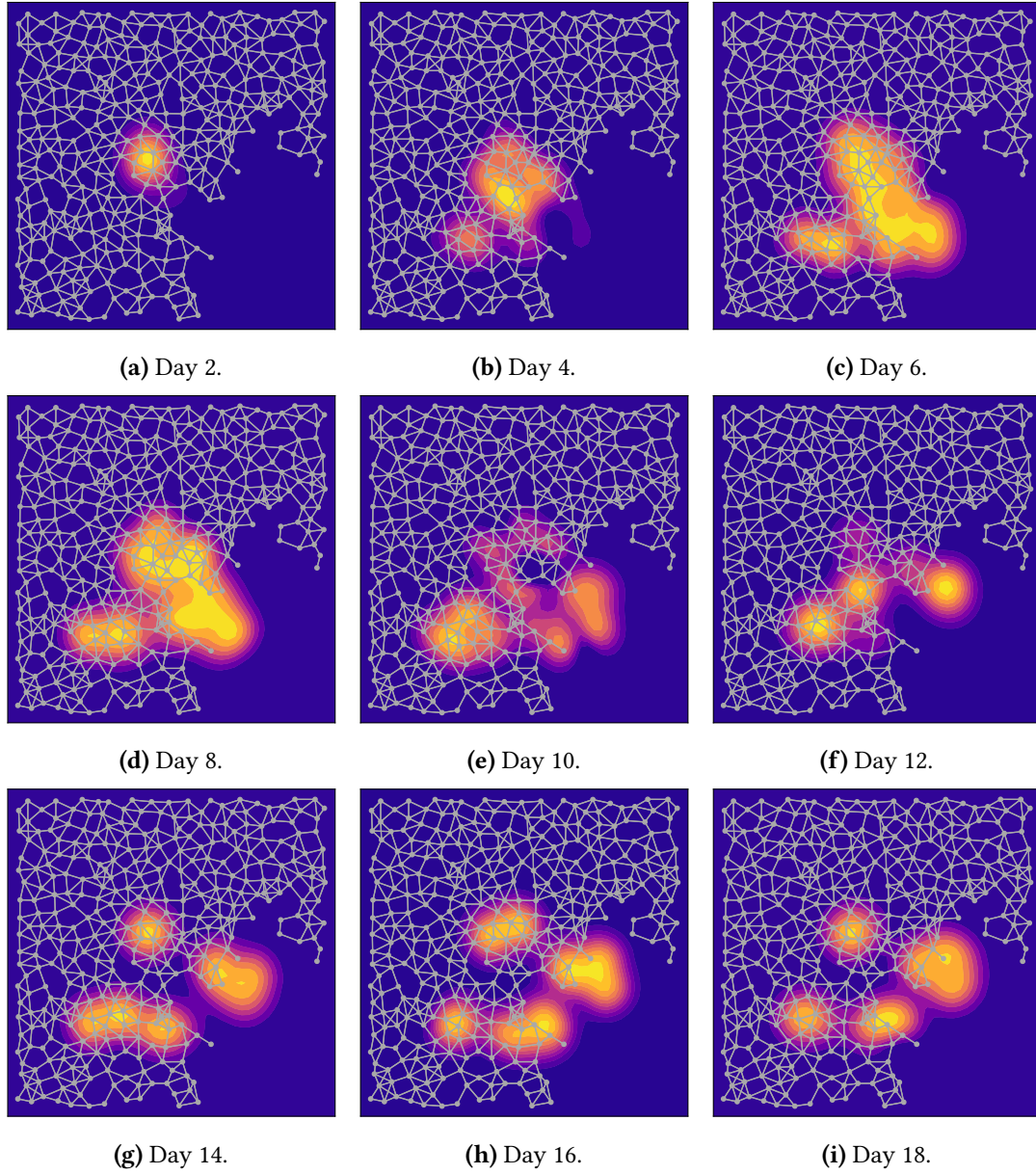


Figure 5.4: Snapshots of wildfire domain ground truth GP model, shown at the same time across different days. Yellow regions represent the highest wildfire intensity on a per-day basis.

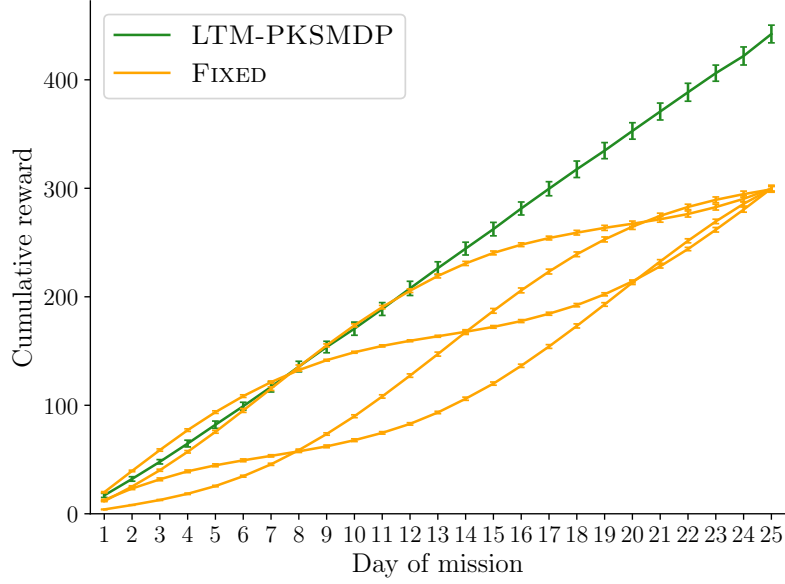
reward is lower. The domain is a small grid with a single depot, over which the value of f is spatially uniform but varies temporally. Therefore, in this context, it is only the timing of the robot’s active hours that affects the reward obtained, rather than the location of the robot. The value of f oscillates in amplitude throughout each day, with a peak that is initially in the morning, but shifts towards the evening over the 25-day mission. The LTM-PKSMDP planner uses a timestep $\Delta_t = 1$ h, and is compared to four instances of FIXED planners, each with different scheduled start times which are equally spaced throughout the day.

We see in Figure 5.5a that none of the FIXED planners are able to perform as well as the LTM-PKSMDP planner—flexible scheduling of resource replenishment is essential for good performance in this domain. Figure 5.5b shows the behaviour of the flexible planner, which learns to shift the robot’s active hours over the course of the mission to track the times with higher rewards. As expected, this tracking is not perfect, as the planner continues performing some exploratory actions throughout the mission.

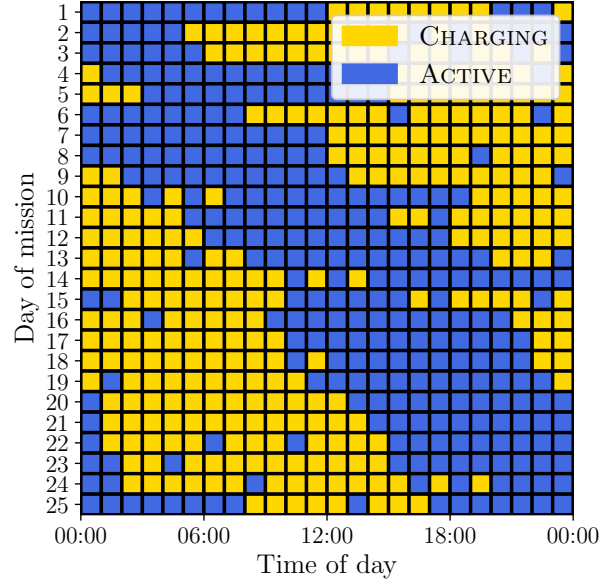
5.4.4 High-level planning horizon

In this experiment, we examine the effect that the high-level planning horizon H_Φ can have on performance. In principle, a longer horizon should enhance the planner’s capabilities, since it allows for decision-making based on a longer view of the future. However, in practice, the larger search space associated with a longer planning horizon can actually make it harder to find good solutions—particularly in problems with high uncertainty. Thus, in general, the choice of planning horizon will depend on the specific problem at hand, as well as the planning time and computational resources available.

The domain for this experiment, shown in Figure 5.6a, is a 20×20 grid map containing a chain of depots, spaced such that the robot takes multiple resource cycles to traverse it from end-to-end. We would expect sparse placement of depots to be common in practice, since installation and maintenance of many depots would be costly, and as such they would likely be placed in the minimal way required to grant the robot coverage of the environment. Here, the unknown function f has a high-value peak that drifts across the map from left to right over the course of a 50-day mission, as might be seen in



(a) Reward accumulated by the LTM-PKSMDP planner and four fixed-schedule planners with different starting times (10 trials each).



(b) The robot state at each hour of the mission during one run of the LTM-PKSMDP planner.

Figure 5.5: Results from the active hours scheduling test domain, demonstrating that the LTM-PKSMDP planner is able to adaptively schedule the robot's active hours to capitalise on available reward.

monitoring herds of wild animals [209] or tracking the drift of seaweed farms [210]. We ran MCTS with 250 trials.

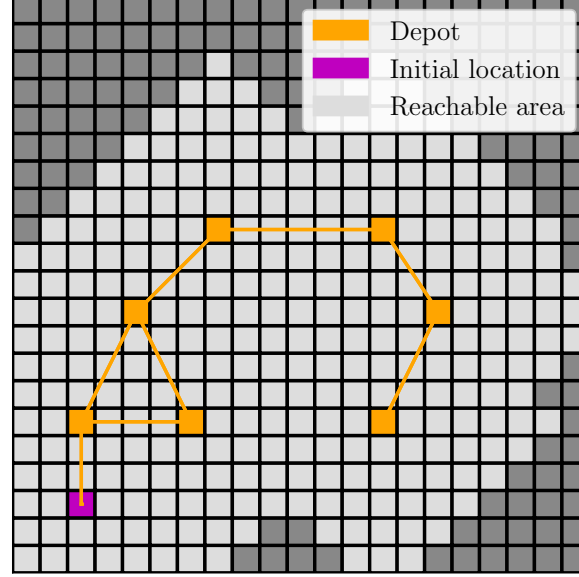
We see in Figure 5.6 that in order to perform well in this domain, the longer-horizon planning of the LTM-PKSMDP is required, and configurations with shorter planning horizons acquire lower total rewards. This also validates our approach to estimating rewards in the Depot-MDP, showing that even without planning in belief space at the high level, our hierarchical approach has significant benefits over the non-hierarchical BASIC baseline. The FIXED planner was tested with a horizon of 5 days, and the LTM-PKSMDP planner with the same horizon achieves on average 7% higher reward, reflecting the benefits of flexible scheduling in overall performance. The average computation time for the $H = 10$ d configuration was 14.0 s per step.

5.4.5 High-level planning step

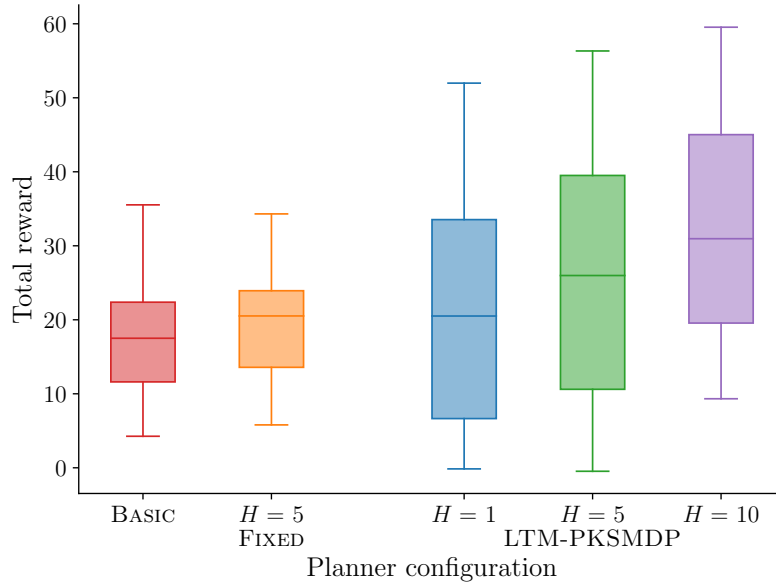
This experiment tests the effect of the high-level planning timestep Δ_t for the LTM-PKSMDP planner. Here we used a larger 40×40 grid world containing several randomly-generated high-interest areas which each shift slowly around the map, and ran MCTS with 250 trials. We see in Figure 5.7 that, as expected, planning with higher temporal resolution generally results in better performance. This does come with some additional computational demand, since the Depot-MDP grows with the number of high-level actions that can be executed within its horizon. However, for this configuration we found that the overall impact on computation time from decreasing the timestep was negligible, and all configurations took a similar computation time of approximately 30 s per step. This indicates that the computational load of planning was dominated by the low-level planner.

5.4.6 UAV wildfire monitoring

We now demonstrate a practical application of our approach to the large-scale UAV wildfire monitoring scenario described in Section 5.4.2. For each experiment, we ran MCTS with 500 trials, and used a high-level planner timestep of $\Delta_t = 4$ h. The results from this domain are shown in Figure 5.8.



(a) Gridmap of the domain. Orange lines link depots that can be reached within a single high-level planner timestep.



(b) Cumulative reward with different high-level planning horizons in days (30 trials each). Performance increases as the planning horizon increases, since this domain requires a long-horizon view in order to find an effective policy.

Figure 5.6: Experiment results with different planning horizons. The LTM-PKSMDP planner used a high-level timestep of $\Delta_t = 6$ h.

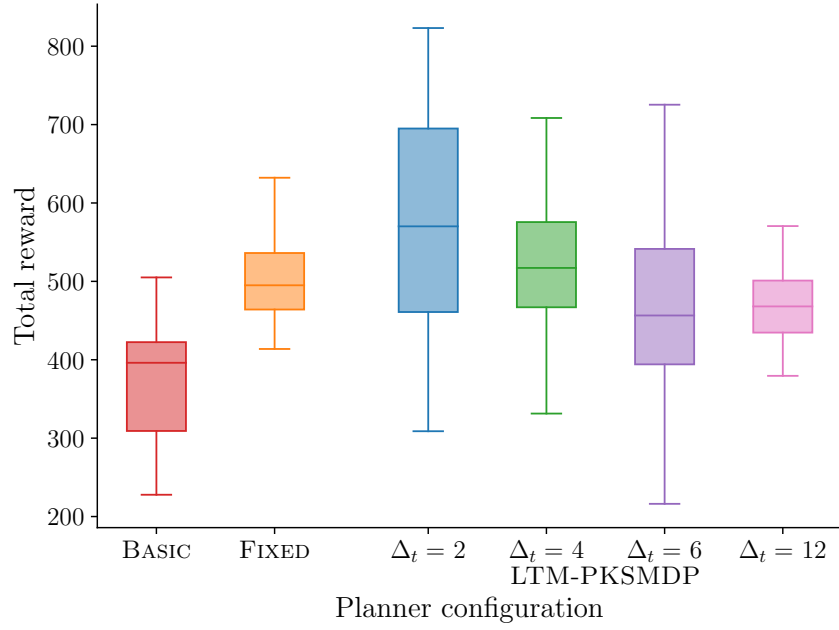


Figure 5.7: Cumulative reward with different high-level planning discretisation steps in hours (40 trials each). The LTM-PKSMDP and FIXED planners used a horizon $H = 5$ days. Performance generally improves as the discretisation becomes finer, at the cost of increased computation.

For the longest planning horizon of $H = 10$ d, our planner required an average computation time of 94.4 s per step, which is low enough to allow for online planning in a domain of this scale. We see that our hierarchical planning approach provides a clear benefit over the BASIC planner, which tends to get trapped in local minima and fails to usefully monitor the large environment. The FIXED planner is also outperformed by the LTM-PKSMDP planner with the same horizon, indicating that flexible scheduling can provide a significant advantage in this domain. This is as expected, since wildfires typically burn more intensely during the heat of the day—it should be the case that appropriate selection of active hours allows for more total reward to be accumulated.

It is also worth noting that the realism of our simulator is limited somewhat by the low temporal resolution of the MODIS satellite data. Hotspots can only be detected when the satellite passes overhead, and in this dataset each location typically only receives 1–5 satellite flyovers per day. This limits the amount of temporal variation from the true wildfires that could be captured by our ground truth GP model, and it is likely that with better ground truth data, the daily variations would be accentuated even more. Nonetheless, this experiment demonstrates that our planner can produce effective

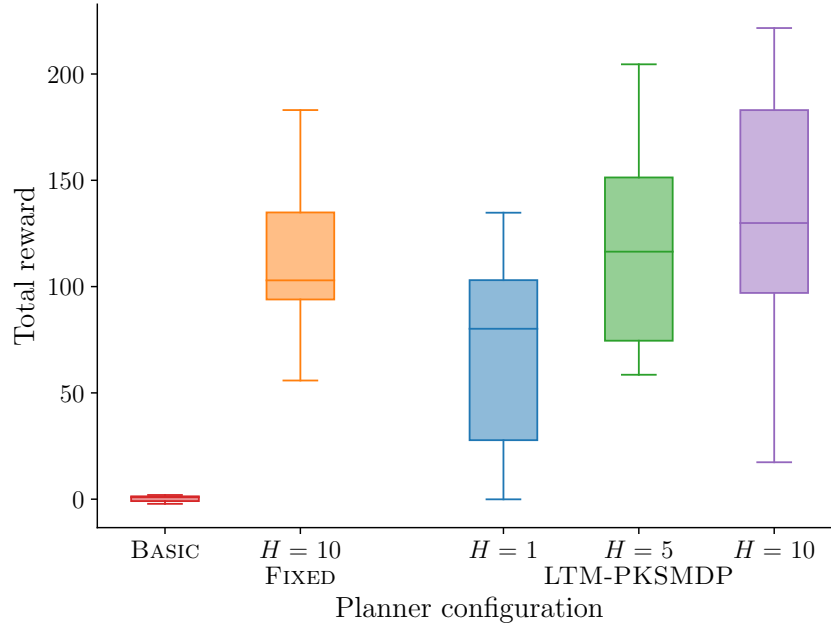


Figure 5.8: Cumulative reward obtained by different planner configurations on the wildfire domain (10 trials each). Planner performance improves as the planning horizon grows, and the LTM-PKSMDP planner outperforms the FIXED planner with the same horizon, indicating some benefit of flexible scheduling in this domain.

monitoring behaviour in a practical application. It enables a robot to autonomously provide situational awareness in a dynamic and time-critical setting, without relying on prior knowledge about the behaviour of the environment feature of interest.

5.5 Discussion

The multi-depot planning problem presented in this chapter opens up a richer planning space than the single-depot, fixed-schedule framework of the previous chapter. This is demonstrated in the synthetic domain experiments, where we see that our planner is able to adapt its active hours to improve its mission outcomes. Our planner is also able to benefit (albeit marginally) from a planning horizon of up to 10 days, achieving meaningful long-term planning even with a highly uncertain model. These behaviours are enabled by the high-level Depot-MDP model, which enables long-horizon planning over a simplified, abstract representation of the full state space.

However, the Depot-MDP model is somewhat rigid, and would likely struggle to scale to very large domains. Its primary limitation on this front is that the computationally

demanding reward estimation process (Algorithm 5.1) is always performed over the full state space of the Depot-MDP—that is, rewards are estimated through GP inference over the entire map, even areas far from the robot’s current location which are likely irrelevant to the immediate decision making. While this might be addressed by pruning the Depot-MDP to restrict its range, such an approach might inadvertently remove promising paths from consideration. Another key limitation of the Depot-MDP model is its use of discrete timesteps, which is suboptimal in the continuous-time problem settings we consider. In the deterministic-action setting considered in this chapter, this is not a major issue, but in a more realistic setting with stochastic action durations, the discrete-time assumption could lead to significantly greater loss to planning efficiency. In the next chapter, we will design a new, MCTS-based high-level planner which addresses each of these limitations.

5.6 Summary

We have presented a hierarchical planning approach for long-term robot monitoring in a GP-modelled unknown environment subject to resource constraints. Our work considers a setting in which the robot is able to autonomously recharge its resources at depots, enabling monitoring missions that span much larger areas than can be covered in a single battery cycle. We achieve effective long-horizon planning under uncertainty by separating high-level path planning and resource management from low-level action selection. Our planner is able to effectively fulfil monitoring mission objectives while replenishing resources as needed, balancing exploration and exploitation in a sensible manner. We demonstrated through a series of experiments that our planner is able to effectively fulfil its monitoring objective while replenishing resources as needed, and showed that it is able to perform well compared to relevant baselines in a wildfire monitoring application.

6

Long-Term Monitoring with Stochastic Action Durations

Contents

6.1	Introduction	89
6.1.1	Problem formulation	91
6.1.2	Navigation dynamics	92
6.1.3	Resource model	93
6.2	Algorithm	94
6.2.1	Overview	94
6.2.2	High-level planning	94
6.2.3	Low-level planning	96
6.2.4	Checking reachability	97
6.3	Experiments	98
6.3.1	Battery charging model	100
6.3.2	Planner horizon	101
6.3.3	Environment stochasticity	103
6.4	Discussion	106
6.5	Summary	107

6.1 Introduction

Mobile robots operating in natural and unstructured environments interact with those environments in complex ways. Navigation through the environment can be affected by factors including terrain, weather conditions, unexpected obstacles, and even the

presence of other agents [27]. Likewise, the charging and discharging curves of a robot's battery are influenced by the ambient temperature in the environment [211], as well as general battery health and the actions being performed by the robot. These complex interactions mean that it is often difficult to accurately predict the outcomes of actions. From a planning perspective, this poses a considerable challenge, as a planner must be able to reason about the consequences of actions in order to make effective decisions. So far, we have focused on deterministic models of action outcomes, which make the assumption that the outcomes of actions are known with certainty. Stochastic models provide a richer framework for capturing the variability and uncertainty of planning in the real world, allowing for more robust and effective plans to be generated. In this chapter, therefore, we extend the work presented so far to tackle **C3 Stochastic Interactions**, by incorporating stochastic models of the outcomes of robot actions.

Specifically, we introduce stochasticity into our models of navigation and resource consumption, capturing the variability and incomplete information that is inevitably present in real, unstructured environments. For navigation actions, we describe the duration of each action as a probability distribution, rather than a deterministic value. This allows us to explicitly model the fact that the time taken to traverse a path through the environment may vary in ways that cannot be predicted in advance. For resource replenishment actions, we specify the time spent charging, and model the resulting resource level as a distribution, capturing variability in the charging process. This adds significant complexity to the planning problem, since actions now have uncertain outcomes, in addition to the uncertainty that already in the environment process. The effects of these uncertainties compound over time, making long-horizon planning particularly challenging. The addition of stochastic action durations also compounds the challenge of resource management (**C2 Resource Constraints**), since it is no longer possible to ensure satisfaction of the resource constraints with certainty. Instead, we move to a chance-constrained approach, where we provide a probabilistic guarantee that the robot battery charge will remain above some minimum threshold with a certain probability.

Our contribution in this chapter is a planning framework for long-horizon mission planning that handles resource constraints in stochastic environments, tackling **C2 Resource Constraints** and **C3 Stochastic Interactions** simultaneously. We extend the work from Chapter 5, which assumed deterministic action outcomes, and relied on a discrete representation of time and resource levels for planning at the high level. We introduce stochastic models for navigation and battery charging actions, which better represent the challenges of robotics in unstructured outdoor environments. By modelling action durations as exponential distributions with deterministic delays, we are able to enforce a chance constraint over the robot’s battery level on each outing away from a base station. We plan at the high level using MCTS in belief space, which handles continuous spaces using progressive widening rather than discretisation, making efficient use of computation time by focusing on promising areas within the search space. We demonstrate that our approach is able to plan effectively in challenging settings, obtaining higher performance than the previous work while producing fewer constraint violations.

6.1.1 Problem formulation

We extend the model from the previous chapter to consider stochasticity in action outcomes. For navigation actions, we model each action as consisting of a fixed minimum duration, as well as an exponentially-distributed stochastic component—we outline this model in full in Section 6.1.2. Stochastic durations in turn imply stochastic resource consumption for navigation actions, since an action which takes longer to execute will also consume more resources. We also consider a stochastic model of resource replenishment, where charging for a particular length of time now results in a *distribution* over resource levels, rather than a precisely known outcome. We describe the resource model in Section 6.1.3.

Our mission objective, as before, is to maximise the cumulative reward from observations taken of f up to a mission horizon H , which corresponds to many days. This is now subject to the chance constraint that on each *outing* where the robot ventures away from a depot, its resource level must remain above a threshold value $B_{\min} \in [0, 1)$

with probability $p_{\min}$. This threshold is introduced as an additional parameter that can be used to set an allowable level of risk, depending on the context of the mission. We formulate the chance constraint in terms of outings rather than across the entire mission to better accommodate the long-term nature of the planning problem. In a context where missions might span very long periods of time, or even continue indefinitely, it is more intuitive to consider the chance constraint on a per-outing basis, decoupled the constraint from the length of the mission. We assume that in most practical applications, there would be a high cost associated with a robot running out of battery in the field, since this would typically require human intervention to return the robot to a depot. Therefore, the battery threshold acts as a reserve level that can be used when constraint violations do occur, ensuring that complete depletion of the battery will be rare.

As in the previous chapter, we model this monitoring problem as a LTM-PKSM DP (Definition 5.1). This time, we consider stochastic action outcomes in the transition dynamics.

6.1.2 Navigation dynamics

Action durations have been modelled as continuous stochastic processes in previous works using several different distributions, including exponential [212, 213, 214], χ^2 [215], phase-type [27], and Gaussian [216]. PTDs, which we introduced in Section 2.5.2, can be fit to arbitrary non-negative distributions [217]. Street *et al.* [27] used them in the context of action duration modelling in robotics, by fitting PTDs to real data in order to construct realistic models for robots navigating under congestion.

Here, we adopt the simpler exponential distribution with a deterministic delay to model action durations. On its own, the exponential distribution is a somewhat unrealistic model, as it offers a non-zero probability that any action will complete arbitrarily quickly. The deterministic delay mitigates this, by enforcing minimum durations for each action. Since PTDs are essentially mixtures of exponential distributions, formulating action durations in this way allows us to evaluate the planner in a scenario where it can accurately model stochastic action durations. Further, the fact that PTDs can be fit to

arbitrary non-negative distributions means that our approach remains compatible in principle with other action duration models—albeit at a greater computational cost.

Definition 6.1 (Hybrid action model) For a given environment represented as a topological map $\mathcal{G} = \langle V, E \rangle$, each edge $e_i \in E$ has a corresponding hybrid action model $\rho(e_i) = \langle \eta_i, \lambda_i \rangle$, where η_i is a deterministic delay (i.e. minimum duration), and λ_i is the rate parameter of an exponential distribution representing the remainder of the action duration.

This provides a somewhat more realistic model for robot navigation than a pure exponential distribution, while retaining some of the computational benefits of that model. Importantly, the techniques we shall use later in this chapter for enforcing the chance constraint are applicable to general PTD models with deterministic delays, of which the exponential distribution is the simplest case. Thus, our methods could still apply to scenarios where higher-fidelity models are required, at the cost of some extra computation depending on the complexity of the chosen models. It is also worth noting that while we now consider stochasticity in the durations of navigation actions, the outcome of these actions is still known—that is, if the robot executes an action to move from location v_i to v_j , it is guaranteed to end up at the correct location, and the only uncertainty is over the time of arrival.

6.1.3 Resource model

In the previous chapter, we modelled the robot’s resource dynamics using fixed charging and discharging rates, effectively modelling the resource dynamics as a simple linear system. While this simple model might be sufficient in some applications, its scope is somewhat limited. We now introduce a more expressive *resource model*, is able to accurately capture a broader range of resource dynamics.

Definition 6.2 (Resource model) A resource model $\mathcal{B}$ is a tuple $\langle c_\downarrow, c_\uparrow \rangle$, where $c_\downarrow : [0, 1] \rightarrow \mathbb{R}$ is a discharging rate function and $c_\uparrow : [0, 1] \rightarrow \mathbb{R}$ is a charging rate function. Each rate function depends on the current resource level $b \in [0, 1]$.

The rates $c_{\uparrow}$ and $c_{\downarrow}$ describe the dynamics of the robot’s finite resources while it is ACTIVE and REPLENISHING, respectively. As previously, any non-depot state where $b = 0$ is a sink state. In Section 6.3, we shall use this resource model to express a realistic model of battery life, the most ubiquitous constrained resource in robotics problems. Importantly, however, the model is not specific to batteries, and could be used to model any resource that is consumed and replenished during a robot mission, such as data storage capacity, thermal coolants, or other consumables. In the case of battery, the charging rate $c_{\uparrow}$ specifies a battery charging curve, while the discharging rate $c_{\downarrow}$ specifies the rate at which battery life is consumed while the robot is active. As we shall see in Section 6.2.2, we can also incorporate uncertainty or noise over these rates, to account for stochasticity or unmodelled variability in the charging or discharging processes.

6.2 Algorithm

6.2.1 Overview

We now extend the planning approach from the previous chapter to the stochastic setting. In doing so, we introduce a new high-level planner which addresses the limitations discussed in Section 5.5. We also modify the action pruning mechanism from the low-level planner to consider the chance constraint. We describe each of these components in detail in the following sections.

6.2.2 High-level planning

In Section 5.3.2, we introduced the Depot-MDP as the high-level model for our hierarchical planning framework. One of the key features of this model is that it represents the high-level planning problem in a *discrete* state space, even though the full problem is continuous. Discretisation of the time and battery life state factors introduces a degree of slack into the planning process, since the continuous values are conservatively rounded to the nearest discrete level. This slack results in a less precise representation of the true state space, and therefore worse planner performance. However, we showed that this approach could be effective in practice nonetheless, and indeed became more effective as the discretisation became finer—at the cost, of course,

of increased computation time (Section 5.4.5). This approach also assumes a fixed belief representation while planning, as each Depot-MDP is constructed by sampling from the current GP model. This is not a major limitation in the deterministic setting, since the UCB heuristic was still able to encourage optimistic exploration of the state space, and each high-level plan was only ever executed for one step before replanning. However, as we move to the stochastic setting, the discretisation-based approach becomes more limiting, since actions can no longer necessarily be guaranteed to complete within any fixed number of timesteps—particularly in cases of action duration distributions with long tails. We therefore introduce a new high-level planning model for this setting.

In particular, we now formulate the high-level planning problem as a *high-level PKSMDP*, which captures navigation between depots as well as management of the constrained resource. Unlike the Depot-MDP, this model is continuous in both time and resource level, and provides a framework for the improved, belief-space planning approach which we describe below.

Definition 6.3 (High-level PKSMDP) For a given LTM-PKSMDP $\mathcal{M}$, navigation model ρ , and resource model $\mathcal{B}$, threshold $B_{\min}$ and probabilistic bound $p_{\min}$, we define a corresponding high-level PKSMDP $\mathcal{M}_{\Psi} = \langle S_{\Psi}, s_{0,\Psi}, A_{\Psi}, T_{\Psi}, R_{\Psi}, H_{\Psi} \rangle$. The known state space is $S_{\Psi,k} = D \times T \times B$, where D is the set of depots, $T \in [0, H_{\Psi}]$ is the time, and $B \in [0, 1]$ is the resource level; and $s_{0,\Psi}$ is the initial state; the unknown component $S_{\Psi,e}$ represents the value of the unknown process, as in the full LTM-PKSMDP. $T_{\Psi} : S_{\Psi} \times A_{\Psi} \times S_{\Psi,e} \rightarrow [0, 1]$ is the stochastic transition function, which specifies the outcomes of REPLENISH actions based on the resource model ρ , and NAVIGATE actions based on the navigation dynamics $\mathcal{B}$. $R_{\Psi} : S_{\Psi} \times R_{\Psi} \rightarrow \mathbb{R}$ is the reward function, which is described below, and H_{Ψ} is the planning horizon.

The action space A_{Ψ} consists of REPLENISH actions, which specify a duration, and NAVIGATE actions which specify a destination depot as well as a duration in which to reach it while satisfying the probabilistic constraints $(B_{\min}, p_{\min})$. REPLENISH actions can be specified for any duration from any (depot) state. NAVIGATE actions must only specify destinations that can be reached from the current location with probability $p_{\min}$

within the specified duration, without resources dropping below $B_{\min}$. Our method for generating actions that satisfy these properties is described in Section 6.2.4. The reward function R_Ψ estimates the reward that would be obtained for executing a NAVIGATE action $a \in A_\Psi$; the reward for all REPLENISH actions is zero.

We plan over the high-level PKSMDP using MCTS in belief space, as described by Algorithm 4.1, with a few modifications to adapt this approach to the high-level planning problem. In the in-tree selection phase (the SELECTNODE method), rewards are estimated using the method from Algorithm 5.1, which simulates a sequence of random walks and takes the highest reward obtained from these as its approximation of the true reward. The same method is used in the random rollout phase (the PERFORMROLLOUT method). This makes each trial of the high-level MCTS routine somewhat computationally demanding, but since the high-level PKSMDP is much smaller than the full PKSMDP, fewer trials are required to obtain reasonable planning performance, as we shall see in Section 6.3. Since charging actions now have stochastic outcomes, we now also use progressive widening over the outcomes of these actions when generating new child nodes (Line 20), to ensure that the tree remains balanced during planning. Rewards for navigation actions are estimated by sampling from the GP, whereas outcomes of resource replenishment actions are sampled from the resource model.

6.2.3 Low-level planning

As in the previous chapter, the job of the low-level planner is to execute actions generated by the high-level planner, which it receives in the form of constraints—a target depot d_t , and a deadline t_f by which to reach it. With the introduction of stochastic action durations, however, the low-level planner can no longer *guarantee* reaching the depot before the deadline, as it could in the deterministic case. Instead, it must select actions which probabilistically guarantee that depot d_t will be reached before time t_f with probability $p_{\min}$, without the resource level dropping below $B_{\min}$. In the following section, we describe our method for filtering out actions that do not satisfy this constraint.

6.2.4 Checking reachability

For an action $a \in A$ to be considered feasible by the low-level planner, we need to ensure that if action a were executed, then it would be possible for the robot to return to a depot in time with resource level $b \geq B_{\min}$ with probability $p_{\min}$. This is the stochastic analogue to the deterministic reachability check used previously in Algorithm 4.1 (Line 6). Exact computation of stochastic shortest paths over continuous distributions is extremely challenging [218]. We therefore use an approximation for the shortest path, over which we check the probabilistic reachability condition.

For the topological graph $\mathcal{G}$ with stochastic edge models, we construct a *determinised* graph $\mathcal{G}_{\text{det}}$, in which the cost $C(e)$ of each edge is simply equal to its expected value in the probabilistic model:

$$C(e_i) = \mathbb{E} [\rho(e_i)] = \eta_i + 1/\lambda_i. \quad (6.1)$$

We then introduce a stochastic version of the GETAVAILABLEACTIONS method used previously (Algorithm 5.1, Line 24), which filters available actions based on whether it will be possible to reach the target depot after executing them. The new method is outlined in Algorithm 6.1. For each action $a \in A$ available in state s , we generate the *chain* of actions corresponding to the shortest path onwards to the target depot (Line 5), computed in the determinised graph. Each action in this chain consists of a deterministic component η_i and an exponentially-distributed stochastic component λ_i . The minimum duration for the chain of actions to execute is simply the sum of the deterministic components. The stochastic components $(\lambda_1, \dots, \lambda_n)$ then form a special case of the phase-type distribution known as the *hypoexponential* distribution. We can compute its CDF (Equation 2.5), which we invert to obtain the duration required to complete the action chain with probability $p_{\min}$ (Line 17).

Figure 6.1 provides a visual representation of these action chains, and example distribution functions for the time taken to complete action chains of different lengths. We use the same approach of computing the duration of shortest path action chains when generating NAVIGATE actions for the high-level planner, in order to ensure that they are always feasible within the probabilistic constraint. This ensures that the inputs

Algorithm 6.1 CHECKING AVAILABLE ACTIONS WITH STOCHASTIC DURATIONS

Input: LTM-PKSMDP $\mathcal{M}$, determinised graph $\mathcal{G}_{\text{det}}$, current state s , probabilistic bound $p_{\min}$, battery threshold $B_{\min}$, target depot d_t , deadline t_f .

Output: Reward value $r \in \mathbb{R}$

```

1: procedure GETAVAILABLEACTIONSSTOCHASTIC( $\mathcal{M}, \mathcal{G}_{\text{det}}, s, p_{\min}, B_{\min}, d_t, t_f$ )
2:    $A_v \leftarrow \emptyset$ 
3:   for  $a \leftarrow \text{GETENABLEDACTIONS}(\mathcal{M}, s)$  do
4:      $s' \leftarrow T(s, a)$ 
5:      $A_{\text{chain}} \leftarrow a \cup \text{SHORTESTPATH}(\mathcal{G}_{\text{det}}, s', d_t)$ 
6:     if  $t(s) + \text{PROBDURATION}(A_{\text{chain}}, p_{\min}) \leq t_f$  then
7:        $A_v \leftarrow A_v \cup \{a\}$ 
8:     end if
9:   end for
10:  if  $A_v = \emptyset$  then
11:     $A_v \leftarrow \text{GETFALLBACKACTION}(\mathcal{M}, s, d_t, t_f)$ 
12:  end if
13:  return  $A_v$ 
14: end procedure

15: procedure PROBDURATION( $A_{\text{chain}}, p_{\min}$ )
16:   $\mathcal{P} \leftarrow \text{GETACTIONCHAINPTD}(A_{\text{chain}})$ 
17:  return  $\text{MINDURATION}(A_{\text{chain}}) + F_{\mathcal{P}}^{-1}(p_{\min})$ 
18: end procedure

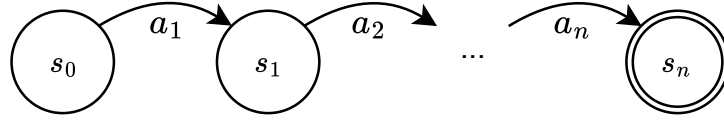
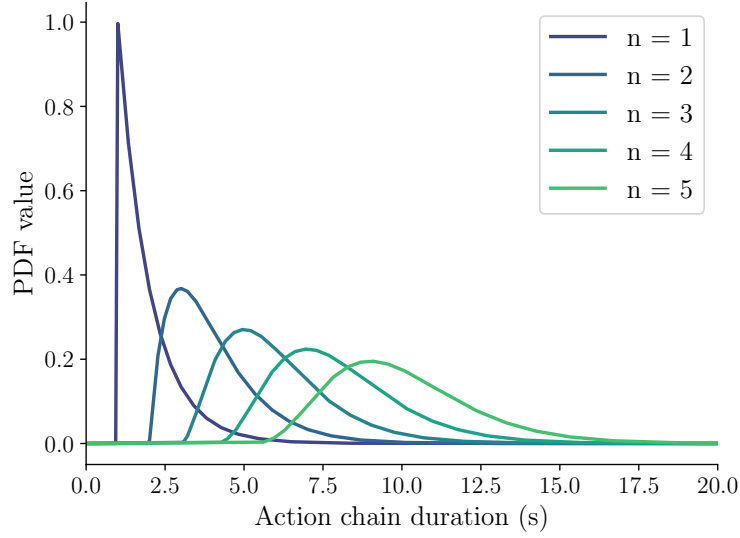
```

received by the low-level planner are always valid. In each case, the shortest path durations can be precomputed, since $\mathcal{G}$ (and hence $\mathcal{G}_{\text{det}}$) is known in advance.

As an additional safety layer for the stochastic setting, we also incorporate a *recourse strategy*, taking inspiration from the work by Mitchell *et al.* [193] on multi-robot persistent coverage. In our case, the recourse strategy is a fallback policy which triggers in the event that no actions are available which satisfy the probabilistic constraint (Line 11). This policy simply navigates the robot along the shortest path to the nearest depot, where this path is calculated over $\mathcal{G}_{\text{det}}$. The effect of this is that when an action takes longer than expected and the robot is at risk of violating the resource constraint, it will abandon its current high-level action and prioritise reaching a depot to recharge. This helps to reduce the number of constraint violations, ensuring that the robot can safely continue its mission even in the event of unexpected delays.

6.3 Experiments

We now evaluate the performance of our planner over a series of simulated experiments. As a baseline, we compare to the Depot-MDP planner from Chapter 5. This planner

(a) A chain of n actions to reach the absorbing state s_n .(b) PDFs for the durations to complete chains of n actions. Each action has a minimum duration $\eta_i = 1$ s, and an exponentially-distributed component with expected duration $1/\lambda_i = 1$ s.**Figure 6.1:** Model for a chain of actions with stochastic durations.

does not consider stochasticity in the action durations or resource dynamics. Thus, we adapt it to plan in this setting by using expected values for action durations and resource consumption.

For the experiments presented in this chapter, we adopt a piecewise linear battery model, as described below in Section 6.3.1, wherein a full charge supports 12 hours of activity. Similar to the previous chapter, we define a small high-level action set A_Ψ in order to narrow the scope of the planning problem. In principle, however, the planner described in this chapter could use full the continuous space of action durations using progressive widening. This action space consists of two types of NAVIGATE actions: the TRAVERSE action, which uses the shortest duration valid under the chance constraint to reach its destination, and the EXPLORE action which allocates the full available duration based on the current battery level. For REPLENISH actions, we define three actions which charge the battery up to 50, 80, and 100 percent respectively, representing the boundary

points of the piecewise linear charging model. In all experiments, we set the chance constraint of the stochastic planner to $p_{\min} = 0.99$.

6.3.1 Battery charging model

Following Fu and Dong [219], we adopt a piecewise linear model for battery charging, as shown in Figure 6.2. This model provides a reasonably good approximation to the typical charging curve for an electric vehicle (EV) or other lithium-ion battery device [220]. The key feature of these charging curves is that they increase quickly and (approximately) linearly up to a state of charge (SoC) of around 80%, after which the charging rate slows down significantly. Thus, while more complex non-linear battery models might be able to capture some additional nuances of real-world charging behaviour, the piecewise linear model provides a good balance between accuracy and computational efficiency for our purposes. In the case shown in Figure 6.2, the battery takes 6 hours on average to charge from empty up to 80%, and a further 6 hours to reach 100%. A consequence of this charging behaviour is that EVs are typically operated mostly within the 20–80% SoC range, as this is where their charging characteristics are most efficient [221]. To account for any variability in the charging rate, we also include Gaussian noise over the final SoC value, which accumulates proportionally to the duration of the charging action, as shown in Figure 6.2.

We model the robot’s battery discharging as a constant rate over time, so the amount of battery life consumed by a navigation action is directly proportional to the duration of that action. Note that, since action durations are stochastic, this implies that battery consumption per-action is also stochastic. Whilst we have selected a simple model of battery discharging here for illustrative purposes, our framework is more general and can capture more complex discharging models. For example, the discharge rate c_d could be dependent on the type of action being executed, or be specified individually per action (e.g. if navigating on more challenging terrain draws more power), but for simplicity we use a fixed value here.

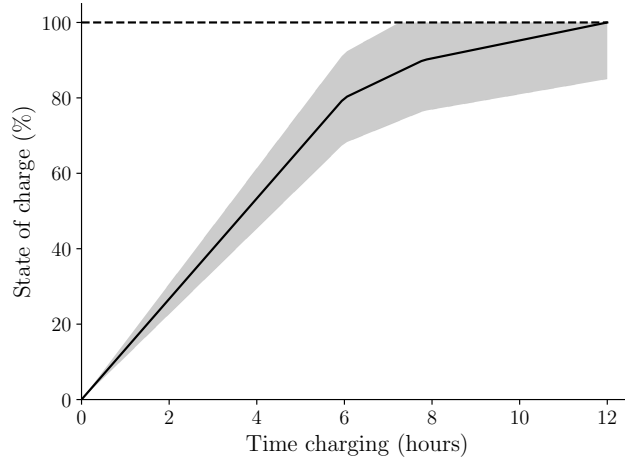


Figure 6.2: A piecewise linear model for battery charging. The curve shows the expected state of charge over time when starting from an empty battery, and the shaded area represents one standard deviation to either side.

6.3.2 Planner horizon

We first examine the performance of the planner across different planning horizons, in order to gauge its overall capability in the stochastic setting. In this experiment, we adapt the 20×20 grid map domain from Section 5.4.4. We transform the domain replacing the fixed cost $C(e)$ of each edge with a stochastic model $\rho(e)$ with the same *expected* cost. We randomly set the stochastic component to represent 20–40 % of the total cost. This variation in the amount of stochasticity is an important element, since it creates a richer and more realistic problem, where planners can consider trade-offs between paths with different levels of uncertainty over outcomes. We use 250 MCTS trials for both the high- and low-level planners, and run each planner configuration 20 times—results are presented in Figure 6.3 and Table 6.1.

A first observation is that, compared with the deterministic domain results from the previous chapter (Figure 5.6), we no longer observe a clear trend of increasing performance as the planner horizon increases. This is likely because with the additional uncertainty introduced by the stochastic dynamics, there is no longer such a clear benefit to planning further ahead. While it is possible that a larger number of MCTS trials (and experiment trials) might reveal some advantage of longer planning horizons, this would also come with a significant computation cost, and it is likely that any

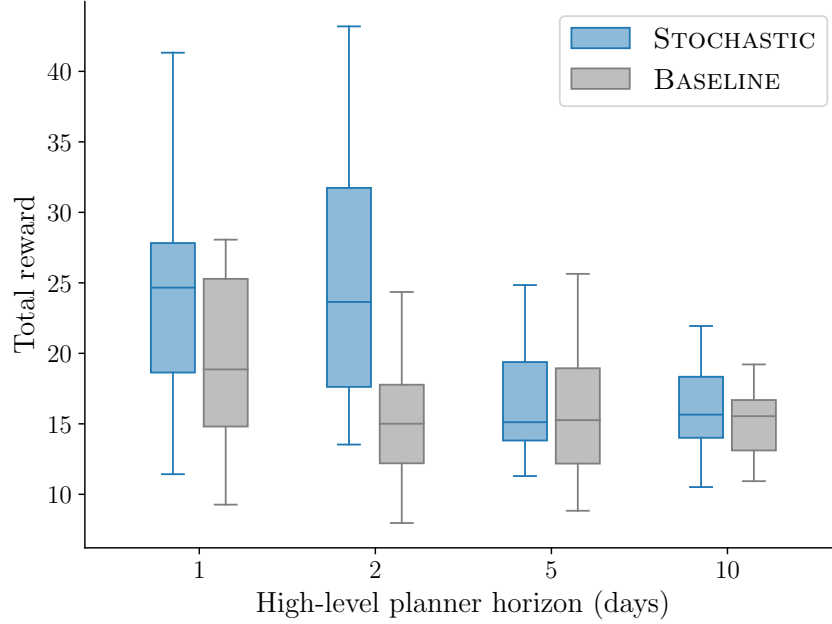


Figure 6.3: Cumulative reward obtained by each planner across different high-level planning horizons. Each configuration was run for 20 trials. We see that the stochastic planner outperforms the baseline at shorter planner horizons, and that the performance of *both* planners deteriorates once the planning horizon becomes too long. This reflects the fact that in stochastic environments, there may not always be any benefit to longer planning horizons.

Table 6.1: Planner performance data in a stochastic environment with different planning horizons.

Planner	Planner horizon (days)	Average reward	Computation time per step (s)	Constraint violations
STOCHASTIC	1	24.56	15.8	1
	2	25.67	25.9	4
	5	17.59	47.6	6
	10	16.22	67.9	3
BASELINE	1	19.39	6.7	18
	2	15.50	10.3	27
	5	16.29	15.3	14
	10	16.09	25.8	25

such advantage would be marginal. We shall revisit this idea briefly in Section 6.4. Comparing the two algorithms, we see that the stochastic planner outperforms the baseline in terms of cumulative reward for planner horizons of 1–2 days. For longer horizons, however, the two algorithms achieve similar rewards on average, with similar variability in performance. However, the crucial difference is that the stochastic planner is able to achieve this performance with a much lower rate of constraint violations, as shown in Table 6.1. On average, the baseline violates the constraints 6 times more frequently, indicating that the stochastic planner does provide a significant advantage even in the longer horizon cases, by obtaining a comparable reward to the baseline while planning much more cautiously and avoiding constraint violations. The stochastic planner also requires 2–3 times more computation time per step, due to the increased computational overhead of computing action duration distributions. However, as shown in Table 6.1, planning times still remain within a feasible range for the experimental timescales we consider for long-term robot missions, where actions would typically be no shorter than 10 minutes.

6.3.3 Environment stochasticity

We now examine the performance of our stochastic planner across environments with different levels of uncertainty over action durations. We refer to this uncertainty as the *level of stochasticity* in the environment. Similar to Section 6.3.2, we use a 20×20 grid map with a randomly generated environment process consisting of moving point sources. Here, we use a planning horizon of 2 days. We vary the level of stochasticity in the environment, quantified by the average proportion of the expected cost of each edge that is stochastic. For example, an environment stochasticity value of 0% corresponds to the deterministic case, whereas 20% means that on average across the environment, 20% of the cost of each edge comes from the exponentially-distributed stochastic component. In each (non-deterministic) case, we randomly sample the stochastic component for each edge uniformly from a range of $\pm 10\%$ around the average value. This ensures that there is some variation in the level of stochasticity in different regions of the environment, allowing the planner to strategically trade off between paths with different levels of risk.

For example, a more risk-averse planner might prefer lower-uncertainty paths, whereas a risk-seeking one might attempt traversing high-uncertainty paths in order to obtain higher rewards. The results of this experiment are shown in Figure 6.4 and Table 6.2.

We see in Figure 6.4 that the performance of the baseline planner degrades as the level of stochasticity in the environment increases. This is as expected, since it only considers *expected* action durations when planning, resulting in policies that fail to effectively adapt online to the stochastic dynamics. Even in the deterministic environment, however, we see that the stochastic planner performs significantly better than the baseline. In this scenario, the chance constraint reduces to a hard constraint, so the difference in performance is due to the high-level planner architectures—the baseline plans at the high level using the Depot-MDP introduced in Chapter 5, whereas the stochastic planner uses MCTS over the high-level PKSMDP as described in Section 6.2.2. We see in Table 6.2 that the MCTS approach achieves this performance advantage with no additional computational cost, since the advantage comes instead from focusing computation on the more relevant and promising areas of the search space. Further, as the level of stochasticity increases, the average reward obtained by the stochastic planner actually *increases*. This is perhaps a little counterintuitive, since these environments are in a sense more challenging, and should require more cautious planning in order to satisfy the chance constraint. However, it is possible that the higher reward is obtained because the planner is able to leverage the uncertainty over action durations to execute riskier behaviours, which can be exploited to obtain higher rewards in instances when actions execute faster than expected.

Table 6.2 also validates the ability of our planning approach to satisfy a chance constraint while maintaining high performance. We see that even in the most challenging environment, the stochastic planner incurs very few constraint violations. By contrast, the number of violations incurred by the baseline planner escalates rapidly as the level of stochasticity increases, since it does not explicitly account for uncertainty over action durations while planning. The results show that the computation time per step of the stochastic planner remains within feasible bounds. The computation time also remains similar for all three levels of stochasticity considered, since the

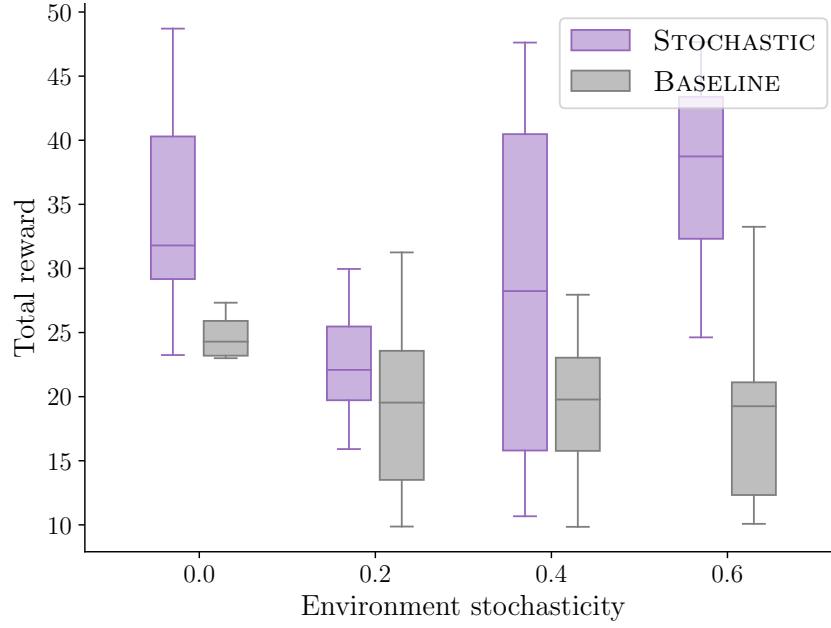


Figure 6.4: Cumulative reward obtained by each planner for different levels of environment stochasticity. Each configuration was run for 10 trials. The stochastic planner consistently outperforms the baseline, most strongly in environments with higher stochasticity. As expected, planner performance in stochastic domains also shows higher variance than the deterministic domain results from previous chapters.

Table 6.2: Planner performance data at different levels of environment stochasticity.

Planner	Environment stochasticity	Average reward	Computation time per step (s)	Constraint violations
STOCHASTIC	0.0	347.07	7.78	0
	0.2	241.82	25.85	0
	0.4	288.12	25.91	1
	0.6	373.98	24.89	1
BASELINE	0.0	230.52	7.50	0
	0.2	191.41	8.00	6
	0.4	201.89	8.16	24
	0.6	185.42	8.66	43

reachability calculations have the same computational demand for any non-zero level of stochasticity. However, it is possible that in some environments, more MCTS trials would be required in order to obtain reasonable planning performance. In such cases, then, the computation time required to achieve good planning performance would indeed increase with the level of stochasticity.

6.4 Discussion

In this chapter, we have extended the planning approach from Chapter 5 to address **C3 Stochastic Interactions** by considering uncertainty over action durations and resource dynamics. The experiments in Section 6.3.2 demonstrate that our planner is able to achieve performance comparable to the baseline from the previous chapter while enforcing a chance constraint over the resource level. The previous approach relied on action pruning based on shortest path costs, which can be computed efficiently using algorithms such as Floyd–Warshall. This approach does not extend directly to the stochastic setting, so we use a deterministic approximation (Section 6.2.4) to the shortest path to enable action pruning. By doing so, we are able to ensure satisfaction of the chance constraint without the need for computationally prohibitive stochastic shortest path algorithms. The results in Section 6.3.2 show that both the baseline planner and the stochastic planner struggle to plan effectively over long horizons, with performance degrading as the planning horizon increases. This could likely be mitigated to a degree by increasing the number of MCTS trials, at the cost of additional computation. However, planning over long horizons is fundamentally challenging when the outcomes of actions are uncertain, so it might also be the case that longer planning horizons would not provide a significant benefit even in the limit of infinite computation.

We have also seen that the importance of explicitly considering stochasticity in environment dynamics is highly dependent on the level of stochasticity present. The results in Section 6.3.3 show the stochastic planner outperforming the baseline in all environment configurations. In the case of deterministic dynamics (i.e., when the environment stochasticity is 0), the performance gap is explained by the new high-level planner architecture. Better performance is achieved by discretising the state space on-the-fly using MCTS with progressive widening, compared with the fixed discretisation used in the previous chapter. In the stochastic environments, we see that the gap in performance between the planners is notably smaller when the level of stochasticity is low, and grows significantly as the variance in action durations increases. This is consistent with intuition, indicating that in certain, relatively predictable, environments, good planning performance might be achieved with a deterministic model. In such a case,

a small level of uncertainty could be managed by simply using a deterministic model with slightly conservative action durations. However, as the level of uncertainty increases, such approaches would either introduce significant inefficiencies into planning, or result in a high rate of constraint violations—we observe the latter in the results in Table 6.2.

6.5 Summary

In this chapter, we have extended the planner architecture from Chapter 5 to address **C3 Stochastic Interactions**, by incorporating stochastic models of action durations and battery charging. We model action durations using exponential distributions with deterministic delays, and integrate a realistic piecewise linear model of battery charging into our planner. We further improve upon the high-level planning approach from the previous chapter, using MCTS to better handle planning over continuous spaces with uncertain action outcomes. Our experiments demonstrate that our planner is able to achieve good performance while enforcing a chance constraint over the robot’s resource level, even in environments with high degrees of stochasticity.

7

Safe Exploration of Unknown Environments

Contents

7.1	Introduction	109
7.2	Preliminaries	110
7.2.1	Constrained reachability	110
7.2.2	Estimated MDPs (EstMDPs)	112
7.2.3	SafeEstMDP-Process	114
7.2.4	Occupancy map models of unknown environments	116
7.3	Problem formulation	117
7.4	Algorithm	118
7.4.1	Algorithm description	118
7.4.2	Incremental navigation graph expansion	118
7.4.3	Choice of goal waypoint	120
7.4.4	Goal scoring function	123
7.5	Experiments	124
7.5.1	Safety hazard model	124
7.5.2	Simulated domains	125
7.5.3	Exploration of unknown environments	125
7.5.4	Hardware and system configuration	130
7.5.5	Deployment in a physical environment	131
7.6	Discussion	133
7.7	Summary	134

7.1 Introduction

In this thesis, we consider planning problems for mobile robots deployed on long-term monitoring missions in unstructured environments. So far, we have assumed access to a known topological map of the environment, providing a structured representation of the area of interest that is safely navigable by the robot. However, in many of the real-world scenarios that motivate this work, such maps may not be available, and may be difficult or costly to obtain. In this chapter, we present a novel algorithm which enables a robot to autonomously explore and map an unknown environment, incrementally constructing a topological map whilst ensuring safety with respect to an unknown hazard. In doing so, we address **C4 Limited Priors**, reducing the reliance on prior knowledge of the environment topology, and thereby broadening the applicability of our planning to a wider range of real-world problems.

The ability to explore and map unknown environments is a fundamental capability for autonomous robots, enabling operations in a wide range of unstructured environments. However, many of the environments that are of interest in this thesis have the potential to present safety hazards for robots. Challenging terrain or strong underwater currents, for example, can present dangers to a robot attempting to navigate them, or create states from which the robot may not be able to return once entered. To explore safely, the robot must avoid states which are explicitly unsafe, as well as those from which it would not be able to return safely.

In this chapter, we address the problem of safe exploration in the presence of a hazard represented as an unknown process f , which is observable locally by the robot via an on-board sensor. We propose a planning algorithm named SafeEstMDP-Map, which enables the robot to incrementally build a topological map of an unknown environment, whilst remaining safe with respect to the unknown hazard. The hazard is modelled based on local observations as a GP, which is used to reason about the safe reachability and returnability of unexplored states. This algorithm is an extension to the SafeEstMDP-Process algorithm from [12], which safely explores a *known* topological map with the objective of estimating the unknown process. In contrast, our objective is to map the maximal safely reachable area of the environment. SafeEstMDP-Map

outputs a topological graph spanning this maximal safely reachable area, which can then be used as the basis for subsequent robot missions, such as the monitoring missions considered in previous chapters.

The work in this chapter is published in the journal *Autonomous Robots* as part of Stephens *et al.* [12], which is joint work with M. Budd. The contributions presented here are the work of the author of this thesis.

7.2 Preliminaries

This section provides background on some of the key concepts and methods that underpin the work in this chapter, in addition to the general background on safe exploration and exploration of unknown environments provided in Section 3.2.

7.2.1 Constrained reachability

To formulate our safe exploration objective and find safe policies, we are interested in *cost-optimal reach-avoid* problems [222], which are a special case of the constrained reachability problem from control theory and RL. The aim in such problems is to minimise the cost to reach a set of goal states, whilst avoiding a set of unsafe (or, *forbidden*) states.

Consider an MDP $\mathcal{M} = \langle S, s_0, A, T, C \rangle$, noting that here we use the cost function formulation from Definition 2.3. We do not specify a planning horizon for $\mathcal{M}$, since we are interested in the cost associated with reaching a set of goal states rather than the cost accumulated within a given number of timesteps, as we shall explain further below. The MDP costs could represent, for example, time taken to execute actions, or the energy consumed in doing so. We assume this MDP to be *time-invariant*—that is, the transition probabilities and costs have no dependence on time. A *path* through an MDP is a sequence $w = s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} \dots$ where $T(s_i, a_i, s_{i+1}) > 0$ for all $i \in \mathbb{N}$. We denote the set of all paths through $\mathcal{M}$ starting from state s by $Path_{\mathcal{M},s}$. Then, given an MDP $\mathcal{M}$ and a deterministic policy $\pi \in \Pi$, we can define a probability measure $Pr_{\mathcal{M},s}^\pi$ over the set of paths $Path_{\mathcal{M},s}$ [223]. For a measurable function over paths, $\zeta : Path_{\mathcal{M},s} \rightarrow \mathbb{R}$, we let $\mathbb{E}_{\mathcal{M},s}^\pi(\zeta)$ denote the expected value of ζ with respect to $Pr_{\mathcal{M},s}^\pi$.

Definition 7.1 (Reach-avoid paths) Let $G \subset S$ be a set of goal states and $F \subset S$ be a set of forbidden states in an MDP $\mathcal{M}$. We define the set of paths that reach G whilst avoiding F as:

$$\begin{aligned} reach_{\neg F, G} = \{ (s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} \dots) \in Path_{\mathcal{M}, s_0} \mid \exists i \in \mathbb{N} \text{ s.t.} \\ s_i \in G \text{ and } s_j \notin F \text{ for all } j \leq i \}. \end{aligned} \quad (7.1)$$

In addition to minimising the expected cost to reach the goal set, we consider that there may be cases where reaching the goal set is not possible. In such cases, a cost-optimal solution minimises the expected cost to reach a state from which reaching the goal set is *not* possible.

Definition 7.2 (Cost accumulation over paths) Let $w = s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} \dots \in Path_{\mathcal{M}, s_0}$. We define ℓ_w as the timestep until which cost will be accumulated for path w :

$$\ell_w = \begin{cases} \min_{\ell} \text{ s.t. } s_{\ell} \in G & \text{if } w \in reach_{\neg F, G} \\ \min_{\ell} \text{ s.t. } Pr_{\mathcal{M}, s_{\ell}}^{\max}(reach_{\neg F, G}) = 0 & \text{otherwise,} \end{cases} \quad (7.2)$$

where $Pr_{\mathcal{M}, s_{\ell}}^{\max}(reach_{\neg F, G})$ denotes the maximum over all policies $\pi \in \Pi$ of the reach-avoid probability $Pr_{\mathcal{M}, s_{\ell}}^{\pi}(reach_{\neg F, G})$.

Note that the second condition in Definition 7.2 encompasses the case where a forbidden state is visited before a goal state, as well as the case where no goal state can be reached. We can now define the cost-optimal reach-avoid problem.

Definition 7.3 (Cost-optimal reach-avoid) Let $cumul_{\neg F, G} : Path_{\mathcal{M}, s} \rightarrow \mathbb{R}_{\geq 0}$ be a function that maps a path $w = s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} \dots$ to the cost accumulated up to timestep ℓ_w :

$$cumul_{\neg F, G}(w) = \sum_{i=0}^{\ell_w-1} C(s_i, a_i), \quad (7.3)$$

and $\Pi^* = \{\pi \in \Pi \mid \pi = \arg \max_{\pi'} Pr_{\mathcal{M}, s}^{\pi'}(reach_{\neg F, G})\}$ be the set of policies that maximise the probability of reaching G whilst avoiding F . The cost-optimal reach-avoid problem is defined as finding the policy $\pi_{\neg F, G} \in \Pi^*$ that has minimal expected cumulative cost:

$$\pi_{\neg F, G} = \arg \min_{\pi \in \Pi^*} \mathbb{E}_{\mathcal{M}, s}^{\pi}(cumul_{\neg F, G}). \quad (7.4)$$

This is a variant of the stochastic safest and shortest path problem from Teichteil-Königsbuch [224], for which the minimal expected cost is known to converge to a finite value. We will write $\mathbb{E}_{\mathcal{M},\bar{s}}^*(\text{cumul}_{\neg F,G})$ to denote the (minimal) expected value corresponding to $\pi_{\neg F,G}$. To find $\pi_{\neg F,G}$, we encode the constrained reachability problem in *co-safe linear temporal logic* and use the approach presented in Lacerda *et al.* [225]. This allows us to formulate a scoring function to evaluate candidate goal states in terms of their expected cost to reach safely, which is the subject of Section 7.4.4.

7.2.2 Estimated MDPs (EstMDPs)

The SafeEstMDP-Process algorithm addresses the problem of planning for safe exploration of an unknown, GP-modelled continuous environment feature. Planning in this setting is achieved using a novel MDP formalism known as an *Estimated MDP* (EstMDP). The key idea of the EstMDP is to partition the codomain of the unknown process into a finite set of intervals, abstracting away the continuous aspect of the planning problem and allowing it to be solved using standard MDP techniques.

First, consider the unknown process $f : X \rightarrow O$, where O is partitioned according to the set of intervals $\mathcal{I}$. Given an element $o \in O$ of the observation space, $I(o)$ denotes the interval $I \in \mathcal{I}$ that contains o . The GP posterior $p^{\mathcal{GP}}$ over an interval I is defined as below.

Definition 7.4 (GP posterior over an interval) Let $\mathcal{D}$ be a dataset of observations of the unknown process f , $v \in V$ be a waypoint, and $I \in \mathcal{I}$ be an interval. The probability that the value of f at v is within interval I is defined as:

$$p^{\mathcal{GP}}(f(v) \in I \mid \mathcal{D}) = \int_{z \in I} p^{\mathcal{GP}}(f(v) = z \mid \mathcal{D}) \, dz. \quad (7.5)$$

It is worth noting here that in this chapter, we consider environment processes (and thus, planning problems) which are static in time, and in which navigation has deterministic outcomes. As such, for a PKSMDP $\mathcal{M}$ used to describe planning over an environment represented by a topological map $G = \langle V, E \rangle$, the known state space S_k is precisely the set of waypoints V . To aid with intuition for the remainder of this chapter, we shall describe the planning models in terms of waypoints of the topological map. We

can also describe actions in this manner, since each navigation action $a \in A$ has a direct correspondence with an edge $e \in E$ —we can therefore write $(v, tmnode')$ to denote the action a that moves the robot from waypoint v to waypoint v' .

This leads us to the *Estimated MDP* (EstMDP), which encodes both the probabilistic transition function of the PKSMDP and the GP model, given the observed dataset $\mathcal{D}$. The EstMDP acts as the basis for planning in the SafeEstMDP-Process algorithm, which we describe in the following section.

Definition 7.5 (Estimated MDP (EstMDP)) Let $\mathcal{M} = \langle S, s_0, A, T, C \rangle$ be a PKSMDP, $\mathcal{D}$ be a dataset of observations of the unknown process f , and $\mathcal{I}$ be a set of intervals partitioning the codomain of f . The estimated MDP is a tuple $\mathcal{M}^{\mathcal{D}} = \langle S^{\mathcal{D}}, s_0^{\mathcal{D}}, A, T^{\mathcal{D}}, C \rangle$, where:

- $S^{\mathcal{D}} = V \times \mathcal{I}$, i.e. the state space consists of pairs of known PKSMDP states and intervals over the unknown process value at those waypoints;
- $s_0^{\mathcal{D}} = (v_0, I(z(v_0)))$ is the initial state, where $v_0 \in V$ is the initial location of the robot;
- The transition function $T^{\mathcal{D}} : (V \times \mathcal{I}) \times A \times (V \times \mathcal{I}) \rightarrow [0, 1]$ is defined as

$$T^{\mathcal{D}}((v, I), a, (v', I')) = T((v, I), a, v') p^{\mathcal{GP}}(f(v') \in I' \mid \mathcal{D}), \quad (7.6)$$

where $T((v, I), a, v')$ is the average of $T((v, o), a, v')$ along interval $I = [l, u]$:

$$T((v, I), a, v') = \frac{1}{u-l} \int_l^u T((v, o), a, v') do. \quad (7.7)$$

The EstMDP formalism manages the continuous codomain of f in a different way to the MCTS-based planning methods used in the previous chapters, by discretising it *a priori* into intervals. As we shall see in the following section, this approach is particularly well-suited to reasoning about the safety of policies to reach goal states, compared with the progressive widening approach used in MCTS.

7.2.3 SafeEstMDP-Process

The primary contribution of this chapter is the extension of the SafeEstMDP-Process algorithm from Stephens *et al.* [12] to completely unknown environments. We summarise the key features of the SafeEstMDP-Process here—see the original paper for the complete algorithm description.

SafeEstMDP-Process considers the problem of a robot exploring an environment which is represented as a PKSMDP $\mathcal{M} = \langle S, s_0, A, T, C \rangle$ (Definition 2.6), with the objective of estimating an unknown process f . At each step of its mission, the robot takes a noisy observation of the unknown process, over which it maintains a GP model. During exploration, the robot must avoid states that are considered *unsafe*, a notion which is formalised by way of a safety function over the PKSMDP state space.

Definition 7.6 (Safety function) A *safety function* is a mapping $\phi : S \rightarrow \{0, 1\}$ where $\phi(s) = 1$ when s is considered safe and 0 otherwise. The sets of safe and unsafe states for a state space S can then be defined as $safe_\phi = \{s \in S \mid \phi(s) = 1\}$ and $unsafe_\phi = S \setminus safe_\phi$.

Figure 7.1 presents a flow diagram of the SafeEstMDP-Process algorithm. The main control loop of the algorithm involves repeatedly selecting goal states, generating policies to reach them, and executing those policies. Over the course of the mission, the robot improves its knowledge of the PKSMDP through the observations it makes of the unknown process, and uses the EstMDP formalism to represent this knowledge for planning. Goal states are selected on the basis of a scoring function, which considers how beneficial the state would be to visit from the current state, as well as the cost to safely reach the state, and its safe reachability/returnability properties. When a goal state is selected by the CHOOSEGOAL method, a policy to safely reach it is also returned—this policy having been generated during the reachability check for that goal state. After each action in the policy is executed, the robot performs a safety check of the current policy, re-evaluating the reachability criterion from the current state given the latest data. If the safety probability of the current policy falls below some threshold, the policy is aborted and a new goal state is selected. Otherwise, the robot

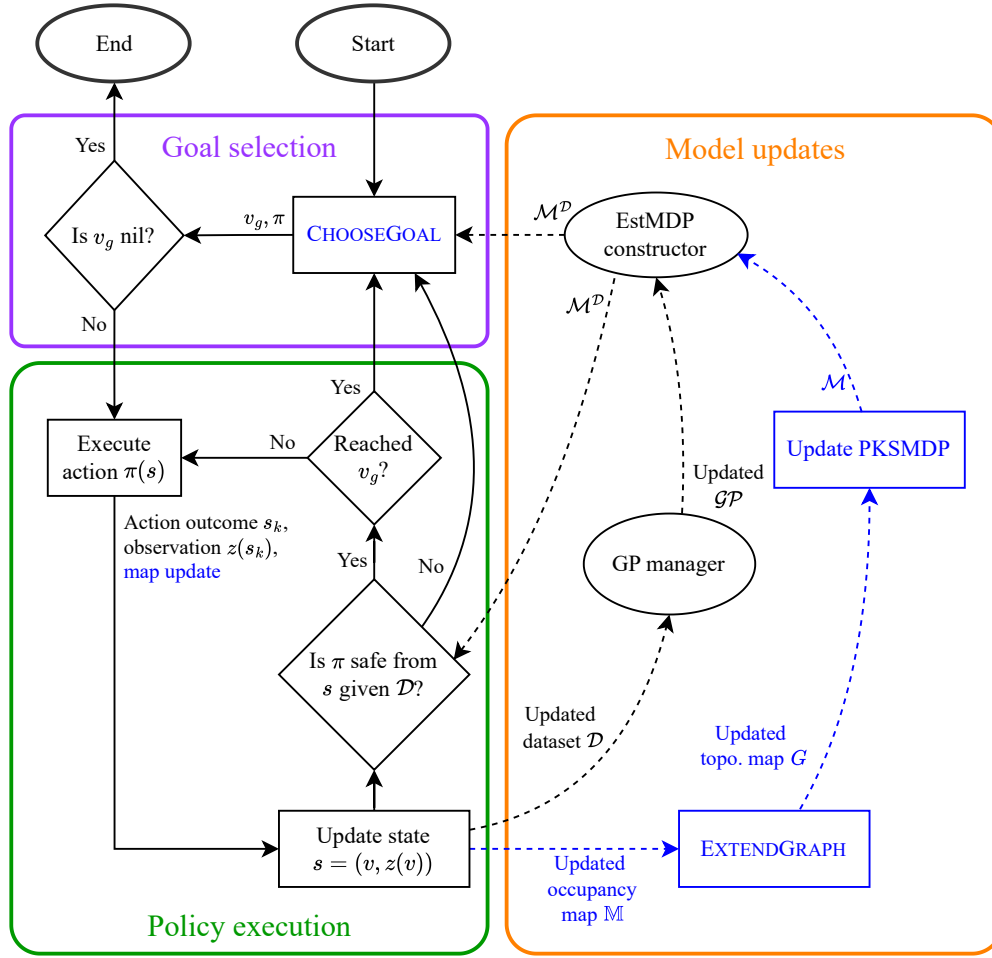


Figure 7.1: Flow diagram of the SafeEstMDP-Process and SafeEstMDP-Map algorithms. Components in blue are specific to SafeEstMDP-Map, or—in the case of the CHOOSEGOAL method in particular—modified from the SafeEstMDP-Process equivalent. Solid lines indicate the main control flow of the algorithm, dashed lines indicate structures that manage data and perform model updates online.

continues executing the policy until the goal state is reached. Once no more valid goal states can be found, the algorithm terminates.

In previous chapters, we used MCTS to plan online over PKSMDPs, discretising the continuous GP codomain online using progressive widening. In contrast, the EstMDP formalism incorporates this discretisation directly into the planning model, and distinguishes explicitly between safe and unsafe states according to the current GP model of f . This enables efficient offline reasoning about the safety of entire policies to reach particular goal states, making it a more suitable method for goal selection in the safety-critical setting.

7.2.4 Occupancy map models of unknown environments

In order to extend SafeEstMDP-Process to unknown environments, we require an environment representation that can be used as the basis for planning. Our chosen representation for this purpose is the *occupancy map* [226], a widely-used and versatile framework for representing unstructured environments observed through noisy sensor measurements in robotics. Consider a bounded environment $X \subset \mathbb{R}^n$, partitioned into free space X_{free} and occupied space X_{occ} . In this thesis, we consider the case where $n = 2$, i.e. we assume the robot is deployed in an environment which can be navigated in a horizontal 2D plane. However, the methods we propose can generalise to 3D environments. Thus, we represent the environment as a grid map $\mathbb{M} = \{m_1, \dots, m_\ell\}$ of ℓ cells, which each represent a region in 2D space, i.e. $m_i \subset \mathbb{R}^2$. We keep a classification of cells $m \in \mathbb{M}$ as representing either free space, occupied space, or unknown space. There are several ways to maintain such a classification. In our implementation, we use the OctoMap framework [227], an octree-based library that provides a probabilistic occupancy estimate of each 3D voxel in the environment, projecting it to 2D by taking a slice of the OctoMap at the height of the robot's lidar sensor.

As a component of our planning problem in this chapter, we require a method to evaluate prospective goal states in terms of how much they are likely to contribute to the goal of mapping the environment. To quantify this objective, we use the notion of *volumetric gain* from Dang *et al.* [122], which estimates the expected unmapped volume that would be perceived by an on-board sensor from a given location $x \in X$.

Definition 7.7 (Volumetric gain) Let N^x be the number of cells observed by the robot from location $x \in X$. We define volumetric gain at x as a weighted sum over the cells in each class observable from x , by casting uniformly-spaced rays outwards in all directions:

$$\text{gain}(x) = \alpha_{unk} N_{unk}^x + \alpha_{free} N_{free}^x + \alpha_{occ} N_{occ}^x, \quad (7.9)$$

where N_c^x is the number of cells observable from x of class c , so $N_{unk}^x + N_{free}^x + N_{occ}^x = N^x$ and $\alpha_{unk} > \alpha_{free} > \alpha_{occ} \geq 0$.

The α coefficients are normalised such that a gain value of 1 represents every ray cast seeing purely unknown space, and a value of 0 represents the robot being completely enclosed by occupied space. The specific values of the α parameters that produce the best performance will depend on the structure and connectivity of the environment: for example, whether it is made up of constrained corridors or wide-open spaces.

It is worth noting that in this chapter, we make use of an OctoMap-based environment representation obtained by projecting a 3D environment representation to 2D. This assumes that robot navigation occurs in a 2D plane, which is the case for all environments considered here, and many real-world scenarios. However, the exploration algorithms developed here also generalise naturally to 3D, so it would be straightforward to extend our proposed framework to more complex 3D exploration problems.

7.3 Problem formulation

We now pose the problem of safely exploring and mapping a completely unknown environment, with no assumed prior knowledge of the environment layout. We assume the robot is equipped with a sensor (e.g. lidar) and software which together enable it to build an occupancy map representation of its environment. This occupancy map is then used to incrementally extend a topological map, which forms the basis of the PKSMDP and EstMDP that are used for planning.

The formal definition of our safe exploration and mapping problem is presented below.

Definition 7.8 (Safe exploration objective) Let X be a bounded environment, $\mathbb{M}$ a grid map representation of X , and $\phi : S \rightarrow \{0, 1\}$ a safety function. Classify as much of $\mathbb{M}$ as possible as free or occupied space, whilst avoiding states $s = (v, f(v))$ where $s \in \text{unsafe}_\phi$. It is assumed that the robot is always initialised in a safe state, i.e., $\phi(s_0) = 1$.

To solve this problem, we adapt the SafeEstMDP-Process algorithm from Stephens *et al.* [12] (see Section 7.2.3) to incrementally grow a PKSMDP through exploration, rather than assuming that a fixed PKSMDP is provided as input. We denote our algorithm as SafeEstMDP-Map, reflecting that the goal is now to safely map the unknown

physical environment X , rather than to explore unknown process f as in the case of SafeEstMDP-Process.

7.4 Algorithm

7.4.1 Algorithm description

In this section, we present the SafeEstMDP-Map algorithm, which extends the SafeEstMDP-Process algorithm from Stephens *et al.* [12] to address the problem described in Definition 7.8. The objective is to safely explore a bounded environment X represented by a grid map $\mathbb{M}$, and classify—as free or occupied—as many cells in $\mathbb{M}$ as possible. For this problem, we will incrementally build a PKSMDP $\mathcal{M}$ over a topological map $\mathcal{G} = \langle V, E \rangle$ for navigation, halting exploration when the volumetric gain associated to all safely reachable waypoints in V is less than a specified threshold.

Our algorithm introduces two key changes with respect to the SafeEstMDP-Process algorithm. First, it adds a new process running in parallel with the main algorithm control loop, which incrementally grows the topological map and PKSMDP as the robot explores the environment. Each time the robot reaches a new state, it runs the EXTENDGRAPH operation described in Section 7.4.2, which uses the latest occupancy map data to extend the navigation graph into any new free space. This process produces updates to the environment PKSMDP, which are in turn used (in addition to the latest GP model) to construct the EstMDP for planning. Second, we modify the candidate waypoint selection, scoring function, and stopping condition used by the CHOOSEGOAL process, to reflect the new exploration objective of safely mapping the environment. These differences are highlighted in Figure 7.1, and described in detail in the following sections.

7.4.2 Incremental navigation graph expansion

Our exploration algorithm requires a mechanism for incrementally growing $\mathcal{M}$ based on local sensor measurements. For this, we assume that the robot navigation is *deterministic*, i.e., the transition function is such that $T((v, o), (v, v'), v'') = 1$ if $v'' = v'$, and 0 otherwise. Furthermore, we consider the PKSMDP cost to be based on the Euclidean distance, i.e., $C((v, o), (v, v')) = \|v, v'\|_2$. The deterministic navigation assumption is used so we do

not need to estimate the outcome distribution of traversing a specific edge from sensor data, which is outside the scope of this work. However, if we have a mechanism that can do so, then our approach can deal with stochastic navigation, as there is nothing in the exploration algorithm that assumes deterministic dynamics. A PKSMDP with a deterministic transition function and Euclidean distance cost can be fully defined by its underlying graph $\mathcal{G} = \langle V, E \rangle$. In the remainder of this section, we describe how to incrementally build $\mathcal{G}$. Note that any other cost function that maps pairs of waypoints to a positive number can be considered; we use Euclidean distance for ease of presentation. Furthermore, whilst uses volumetric gain derived from lidar scans to drive exploration, any other reward metric could be used as suited to the particular application—for example, information gain [228] or reduction of model uncertainty [229]. The graph-extending process is described by the EXTENDGRAPH process in Algorithm 7.1, which can be seen in the context of the full SafeEstMDP-Map algorithm in Figure 7.1.

Algorithm 7.1 EXTENDGRAPH

Input: Existing graph $\mathcal{G} = \langle V, E \rangle$, current waypoint v , angle increment δ , range r , lengthscale ℓ , proximity factor f .

Output: Updated graph $\mathcal{G}$.

```

1: for  $\theta \in [0, 2\pi]$  in increments of  $\delta$  do
2:    $v_{prev} \leftarrow v$ 
3:   for  $d \in [\ell, r]$  in increments of  $\ell$  do
4:      $v_q \leftarrow v_{prev} + d u_\theta$  ▷  $u_\theta$  is the unit vector in direction  $\theta$ .
5:      $v_c \leftarrow \text{CLOSESTVERTEX}(V, v_q)$ 
6:     if  $\|v_q, v_c\|_2 \leq f\ell$  then
7:        $v_q \leftarrow v_c$  ▷ Use existing vertex.
8:     end if
9:     if  $\text{PATHBETWEEN}(v_{prev}, v_q)$  then
10:       $V \leftarrow V \cup \{v_q\}$ 
11:       $E \leftarrow E \cup \{(v_{prev}, v_q), (v_q, v_{prev})\}$ 
12:     else
13:       break
14:     end if
15:      $v_{prev} \leftarrow v_q$ 
16:   end for
17: end for
18: return  $\mathcal{G} = \langle V, E \rangle$ 

```

The algorithm casts rays outwards from the robot's current location v in all directions within a horizontal plane, in angular increments of δ . Along each ray, stepping outwards based on a lengthscale ℓ , a new query waypoint v_q is created (Line 4). If v_q is too close to the nearest pre-existing waypoint v_c , then the nearby waypoint is used as the query

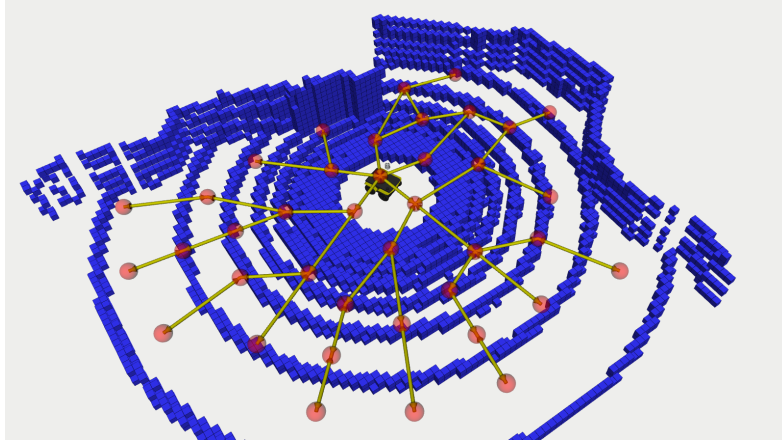
waypoint instead (Line 7). Then, the occupancy map is queried to check if there is a path through free space between the previous query waypoint v_{prev} and v_q (Line 9). If such a path exists, v_q is added to the vertices in the navigation graph, and a bidirectional edge is added linking it to the previous vertex. Otherwise, the current chain of waypoints ends, as it is presumed blocked by an obstacle. In this way, the algorithm constructs chains of waypoints outwards from the current waypoint v through free space, connecting them to existing nearby waypoints whenever possible.

Figure 7.2 shows the navigation graph construction algorithm in action, depicting the graph itself as well as the 3D occupancy map obtained from a Clearpath Jackal robot. The initial navigation graph (Figure 7.2a) has edges radiating a few metres outwards from the robot’s initial location, in all directions except where the lidar’s line of sight is obstructed. Figure 7.2b shows a more developed navigation graph after a few actions have been taken, where the graph both extends further throughout the map and has more edges linking the initial set of waypoints.

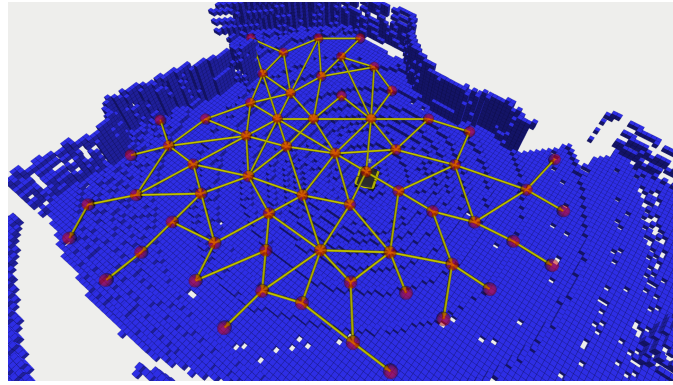
7.4.3 Choice of goal waypoint

The CHOOSEGOAL method of SafeEstMDP-Process consists of two main components. The first is the candidate selection process `GETCANDIDATES`, which returns a list of candidate goal states, in descending order of GP posterior variance evaluated at their respective locations. This step requires that any candidate $v \in V$ must have GP posterior variance above a given threshold η , i.e. $\text{Var}(p^{\mathcal{GP}}(f(v) \mid \mathcal{D})) \geq \eta$. The second is the goal selection process itself, which checks the candidates in batches, filtering out states which are not safely reachable and returnable with some required probability $p_{\min}$. When a batch has at least one candidate remaining after the filtering step, all remaining candidates are scored based on a scoring function $\text{score}^{\mathcal{D}}$, and the highest-scoring candidate (along with a policy to reach it) is returned.

To adapt this for SafeEstMDP-Map, we introduce a modified `GETCANDIDATES` routine, which is described in Algorithm 7.2. Here, the GP predictive variance threshold is replaced by a volumetric gain threshold η_{gain} , to reflect the mapping objective—that is, for a waypoint $v \in V$ to be considered as a candidate goal, we now require $\text{gain}(v) \geq$



(a) Initial navigation graph.



(b) Navigation graph after a few actions have been taken.

Figure 7.2: Navigation graph construction on a Clearpath Jackal robot simulated in Gazebo. Red spheres represent waypoints, yellow lines traversal actions.

η_{gain} . The *primary* candidate set of states (Line 2) can therefore be considered as equivalent to the candidate set returned by the original algorithm.

Algorithm 7.2 GETCANDIDATES (SafeEstMDP-Map)

Input: Visited waypoints visited , minimum safe probability $p_{\min}$, volumetric gain threshold η_{gain} , current waypoint v

Output: Candidate goal waypoint ordered sequence V'

- 1: $V^{\text{safe}} \leftarrow \{v' \in V \setminus \text{visited} \mid P(I[f(v')]) \in \mathcal{I}^{\text{safe}}(v') \mid \mathcal{D}) > p_{\min}\}$
 - 2: $V^{\text{prim}} \leftarrow \{v' \in V^{\text{safe}} \mid \text{gain}(v') \geq \eta_{\text{gain}}\}$
 - 3: $V^{\text{prog}} \leftarrow \{v' \in V^{\text{safe}} \setminus V^{\text{prim}} \mid \min_{v'' \in V^{\text{prim}}} \|v', v''\|_2 < \min_{\tilde{v} \in \text{visited}, v'' \in V^{\text{prim}}} \|\tilde{v}, v''\|_2\}$
 - 4: **return** $V' = V^{\text{prim}} \cup V^{\text{prog}}$ as an ordered set, with elements of V^{prim} first, ordered by decreasing $\text{gain}(v')$, and elements of V^{prog} after, ordered by increasing $\|v, v'\|_2$
-

However, we also introduce a new notion of *progression candidates*, which are returned alongside the primary candidates (Line 3). Progression candidates are waypoints which have volumetric gain below the threshold η_{gain} , but which would decrease the

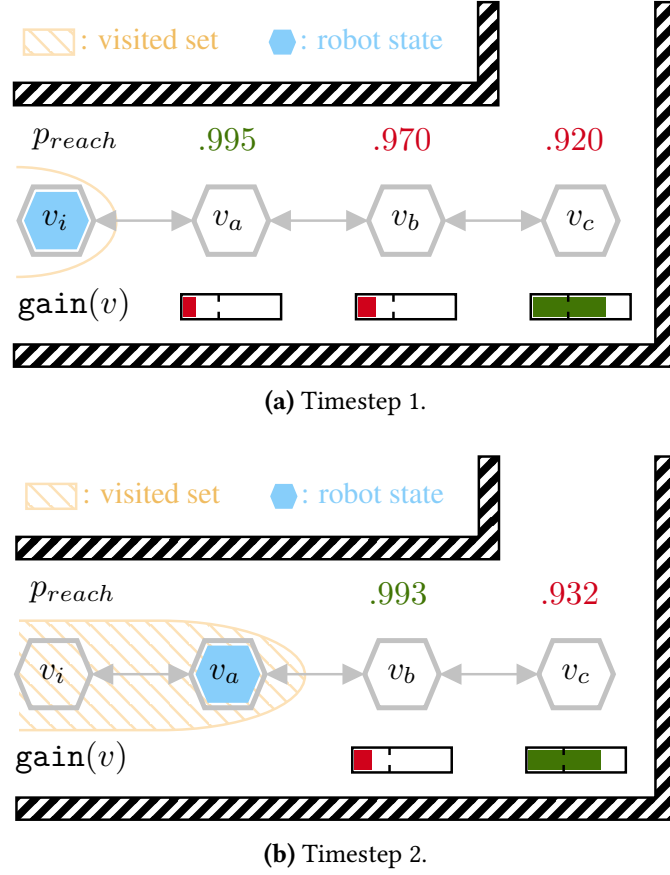


Figure 7.3: Robot progress towards high-information-gain states using progression candidates.

minimum distance between the visited set of states and any primary candidate state if visited. In other words, they are waypoints that progress the robot towards states with high volumetric gain. For most GP kernels that would be useful in this setting, taking measurements increasingly closer towards a primary candidate state will provide more information about the safety of that state. The addition of progression candidates therefore provides an additional mechanism for the robot to select goal states that might benefit the mapping process, by improving its coverage and providing additional information about the safety of primary candidates.

Figure 7.3 shows an example of a scenario in which the progression candidates are helpful. The robot is at the start of a row of (unexplored) waypoints along a corridor, ending in a 90° corner. At the first timestep (Figure 7.3a), the robot is at v_i . For this example, $p_{\min} = 0.99$. The waypoint v_c provides a view around the corner, so has a high volumetric gain and is a primary candidate. However, as it is several navigation

edges away from the robot, $p_{\text{reach}}(v_i, v_c) < p_{\text{min}}$: the robot will not choose v_c as a goal state. The waypoint v_a is safely reachable, but has volumetric gain below the minimum threshold so cannot be a primary candidate. However, v_a is a progression candidate, because visiting it would decrease the minimum distance between the visited set of states and v_c . At timestep 2, the robot has chosen v_a as its new goal and transitioned to it. Having taken a measurement at v_a , it is now more certain that v_b and v_c are safely reachable. It can then continue to choose v_b , which is now safely reachable with probability greater than p_{min} , as a new goal state and progress further towards v_c . Without the addition of progression candidates, the robot would be unable to move along the corridor in this manner, since at first none of the states would meet both the safe reachability criterion and the volumetric gain threshold.

7.4.4 Goal scoring function

We now introduce our modified scoring function, which is used in the final step of the CHOOSEGOAL process to select the candidate goal state to return. In SafeEstMDP-Process, the scoring function $\text{score}^{\mathcal{D}} : S^{\mathcal{D}} \times V \rightarrow \mathbb{R}$ accounts for three factors to assess how beneficial a state v would be to visit: the GP predictive variance at v , the expected cost to safely reach v from the current state s , and the safe reachability/returnability from v :

$$\begin{aligned} \text{score}^{\mathcal{D}}(s, v) = & \text{Var} \left(p^{\mathcal{GP}}(f(v) \mid \mathcal{D}) \right) \\ & \times \mathbb{E}_{\mathcal{M}^{\mathcal{D}}, s}^* \left(\text{cumul}_{-S^{\text{unsafe}}, \{(v, I) \mid (v, I) \in S^{\text{safe}}\}} \right)^{-\gamma_1} \\ & \times (p_{\text{reach}}(s, v) p_{\text{return}}(v) - p_{\text{min}}^2)^{\gamma_2}, \end{aligned} \quad (7.11)$$

where γ_1 and γ_2 provide the relative weightings of the components.

For SafeEstMDP-Map, we design a scoring function to encourage efficient exploration of the unknown space—a waypoint v is beneficial to observe if its volumetric gain is high, as this indicates it would likely be helpful in growing the robot’s map of the environment. Thus, we replace the GP variance component in Equation 7.11 with the volumetric gain term $\text{gain}(v)$ defined in Equation 7.9. This mirrors the functionality of the original scoring function, except with the exploration term now driven by increase

of volumetric gain, rather than minimisation of GP variance:

$$\begin{aligned} \text{score}^{\mathcal{D}}(s, v) &= \text{gain}(v) \\ &\times \mathbb{E}_{\mathcal{M}^{\mathcal{D}}, s}^* \left(\text{cumul}_{\neg S^{\text{unsafe}}, \{(v, I) | (v, I) \in S^{\text{safe}}\}} \right)^{-\gamma_1} \\ &\times (p_{\text{reach}}(s, v) p_{\text{return}}(v) - p_{\min}^2)^{\gamma_2}. \end{aligned} \quad (7.13)$$

7.5 Experiments

We present a series of experiments to demonstrate the performance of SafeEstMDP-Map. We first test the algorithm in a physics-based simulation, evaluating its performance against a set of baselines in a variety of environment configurations. We then also present results from an on-robot deployment in a physical environment, the Corsham Research Mine, integrating SafeEstMDP-Map with a complete robotic hardware and software stack. For all experiments, we solve the MDP via PRISM [230] using nested value iteration [225], and perform GP inference using GPy [231]. SafeEstMDP-Map is implemented in Python and C++, and executed using the robot operating system (ROS) [232].

7.5.1 Safety hazard model

In the experiments in this chapter, we consider the safety hazard of gamma radiation exposure, which can be harmful at high intensity to both robots and humans. The unknown process $f : X \rightarrow \mathbb{R}$ represents the radiation level at each location in the environment. Standard gamma radiation sensors only provide noisy, local measurements, which matches our locally observable safety formulation. We use the radiation layouts from Stephens *et al.* [12], which uses a combination of randomly-distributed point sources and Gaussian random fields to generate realistic scenarios.

Throughout this chapter, we consider a simple safety function which places an upper bound $\lambda_{\max} \in \mathbb{R}$ on the value of the unknown process:

$$\phi(s) = \begin{cases} 1 & \text{if } f(v) \leq \lambda_{\max} \\ 0 & \text{otherwise.} \end{cases} \quad (7.14)$$

In general, however, the SafeEstMDP formulation allows ϕ to be an arbitrary Boolean function over the state space of the PKSMDP. Stephens *et al.* [12] demonstrates SafeEstMDP-Process on a (simulated) AUV exploring a pre-mapped ocean environment,

where f represents unknown ocean currents, and the safety function considers the strength and direction of those currents. In this case, unsafe states are those where the depth of the water is too shallow, or states where the currents are strong enough in a direction that might push the UAV into neighbouring unsafe states.

Following Snelson *et al.* [233], we use a log-warped GP regression, which is better suited to modelling the measurement noise of real-world radiation sensors, as well as capturing variation in the radiation field across multiple orders of magnitude [12]. As the radiation level does not affect transition dynamics in this setting, we need only define two intervals $\mathcal{I} = \{I_{safe}, I_{unsafe}\}$ where $I_{safe} = [0, \log(\lambda_{max})]$ and $I_{unsafe} = [\log(\lambda_{max}), \infty)$. If a non-log GP were used, the EstMDP structure would be identical, with equivalent intervals $I_{safe} = [0, \lambda_{max}]$ and $I_{unsafe} = [\lambda_{max}, \infty)$. The exploration GP models the 1D radiation level at each state: $p^{\mathcal{GP}} : \mathbb{R}^2 \rightarrow \text{Dist}(\mathbb{R})$. The GP input uses spatial coordinates $\{x, y\}$ in units of metres in a local coordinate frame.

7.5.2 Simulated domains

We evaluate the SafeEstMDP-Map algorithm in two simulated environments, each with three different radiation layouts. These experiments are run as full physics simulations in Gazebo [234], using ROS as the primary interface between modules. The two domains are shown in Figure 7.4, and are as follows:

1. *Reactor room* (Figure 7.4a): a simulated world representing a 20 m $\times$ 20 m nuclear reactor room, from Wright *et al.* [235].
2. *Research mine* (Figure 7.4b): A 35 m $\times$ 28 m map built from 3D SLAM in a real underground mine section at Corsham, UK.

7.5.3 Exploration of unknown environments

For this set of experiments, and those in the following section, we use $p_{\min} = 0.99$, goal selection weights $\gamma_1 = 1.5$ and $\gamma_2 = 0.8$, batch evaluation size $N = 30$, gain threshold $\eta_{\text{gain}} = 0.1$, and volumetric gain parameters $\alpha_{unk} = 20.0$, $\alpha_{free} = 1.0$, $\alpha_{occ} = 0.0$. The GP kernel is an RBF with lengthscale 2.0 m and variance 1.0. The GP observation noise $\sigma^2 =$

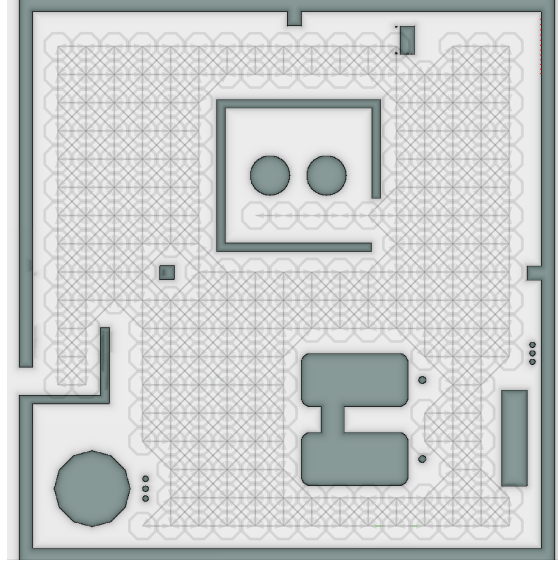
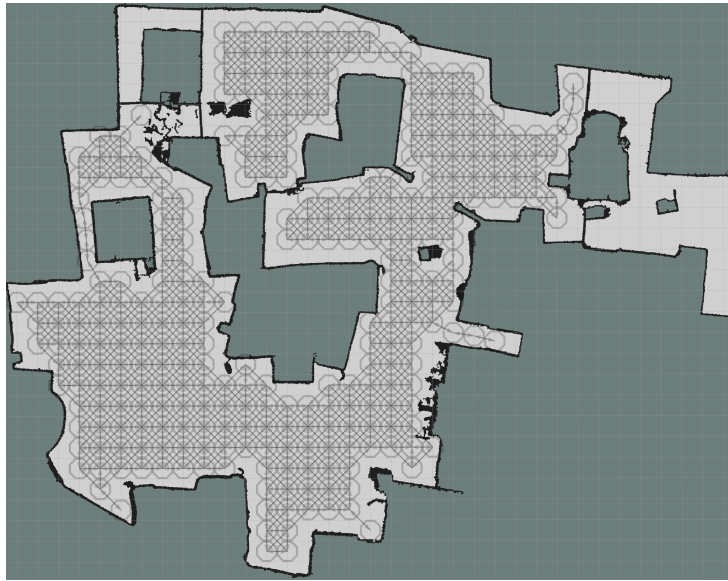
(a) Reactor room, 20 m $\times$ 20 m.(b) Research mine, 35 m $\times$ 28 m.

Figure 7.4: 2D occupancy maps of both simulated experiment domains. Note that the depicted topological maps are not used by SafeEstMDP-Map during exploration, and are shown only to indicate the approximate area of the map that is accessible to the robot.

0.0009, corresponding to a percentage observation noise $\sigma_{\%} \approx 3\%$ (see Stephens *et al.* [12], Equation 17). The robot’s initial position is randomly sampled from a set of 4 waypoints.

We evaluate our algorithm against two baselines. The first is a threshold-based algorithm which we denote ThresholdBasedExplorer (TB), which selects exploration goals using the same scoring function as SafeEstMDP-Map, but does so without an explicit hazard model. Instead, the TB baseline uses a threshold fraction $\varphi \in (0, 1)$ of the safety limit $\lambda_{\max}$ to prevent it from entering unsafe states. All states are initially presumed safe, but if the robot reaches a state at which the radiation level is more than $\varphi\lambda_{\max}$ on the way to a goal, it marks all adjacent *unvisited* states as unsafe. It then selects a new goal from the remaining safely reachable states, thereby turning back from the path to the previous goal. Since the reachability and returnability calculations (Equation 7.13) are now performed based on this deterministic categorisation of states as safe or unsafe, p_{reach} and p_{return} are now either 0 or 1 for all states. The second baseline, MeanPredictionExplorer (MP), again uses the same goal selection process as SafeEstMDP-Map, except with a limited version of the full EstMDP. Rather than weighting the EstMDP intervals based on the GP posterior at each state, this baseline instead assigns all probability mass to the interval containing the posterior mean value at each state. This baseline therefore classifies states as safe or unsafe based purely on the GP predictive mean, with no consideration of the uncertainty of these predictions. We evaluate SafeEstMDP-Map against MeanPredictionExplorer and ThresholdBasedExplorer with $\varphi = 0.3$ and $\varphi = 0.8$, simulating each configuration for 30 exploration runs.

The results in Figures 7.5 and 7.6 show that SafeEstMDP-Map is able to explore more effectively than the baselines across the variety of simulated configurations. The mean-prediction approach frequently results in unsafe termination, since without the full EstMDP model the safe reachability check is much less accurate, incorrectly classifying many unsafe states as safe. Of the threshold-based baselines, the more conservative one ($\varphi = 0.3$) is able to remain safe on almost every run. However, its exploration coverage on many of them is poorer than SafeEstMDP-Map’s, as it will tend to turn back as soon as its radiation reading begins to increase, whereas SafeEstMDP-Map’s radiation model allows it in some cases to continue forwards cautiously. On the other

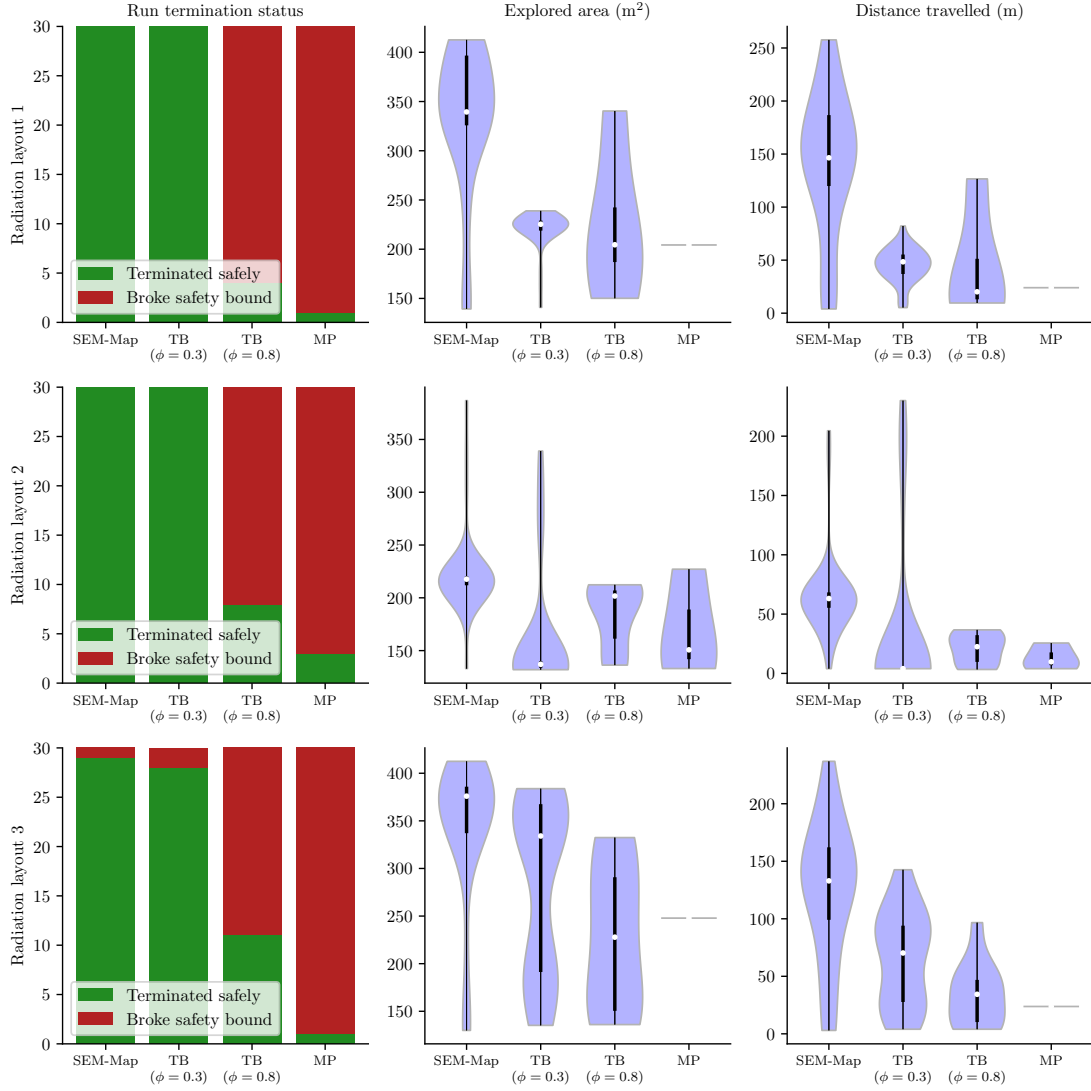


Figure 7.5: Experiments for SafeEstMDP-Map in the reactor room domain. Each algorithm is tested on three randomly generated radiation layouts, with each configuration repeated 30 times. Left: number of runs (out of 30) terminated successfully; middle: violin plots summarising the statistics over area explored; right: violin plots summarising the statistics over distance travelled.

hand, the more aggressive baseline ($\phi = 0.8$) explores more area, at the cost of far more runs terminating unsafely due to violating the safety specification.

SafeEstMDP-Map is able to use its GP model to explore more effectively across varied environments, pushing forwards when an unexplored area is likely to be safe, and remaining cautious when it is believed unsafe. It also scales well with map size, as the EstMDP model is usually not much larger than the underlying navigation graph—even when the map is large, most waypoints will have been visited or be safe with probability

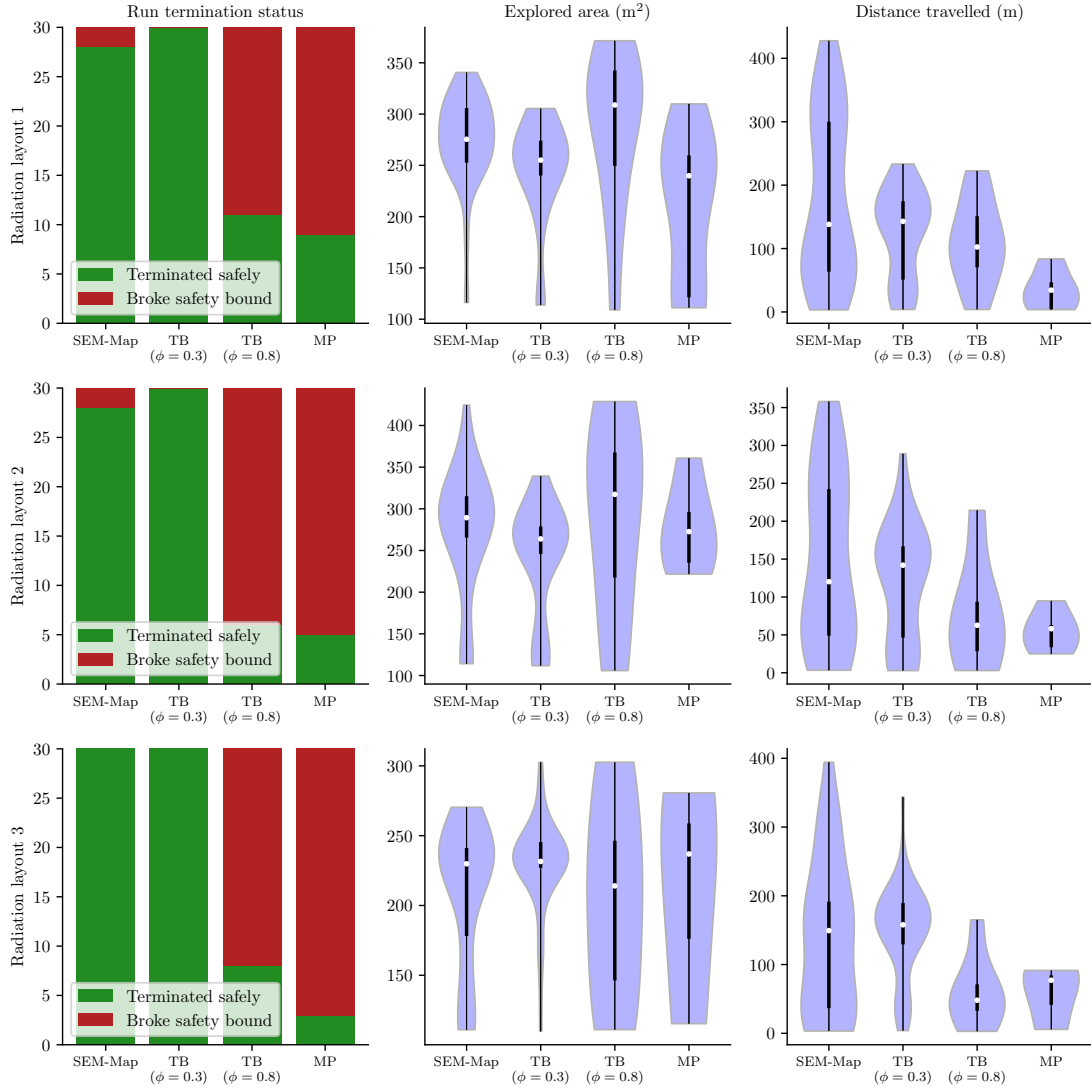


Figure 7.6: Experiments for SafeEstMDP-Map in the research mine domain for different radiation layouts. Each algorithm is tested on three randomly generated radiation layouts, with each configuration repeated 30 times. Left: number of runs (out of 30) terminated successfully; middle: violin plots summarising the statistics over area explored; right: violin plots summarising the statistics over distance travelled.

close to 1, so only the outer areas of the map will have significant uncertainty over radiation level. Consequently, the robot can evaluate dozens of goal choice waypoints per second during exploration, allowing for online planning with minimal delay, even in the research mine environment with maps of up to 150 waypoints. In some radiation layouts, SafeEstMDP-Map does still violate the safety constraint, although this is likely due to the GP model failing to accurately model the radiation field. This issue could be mitigated through further tuning or, more generally, applying domain knowledge in the design of the GP kernel.

7.5.4 Hardware and system configuration

For the real-world validation of our system, we conducted experiments on a Boston Dynamics Spot robot, shown with its sensor payload in Figure 7.7a. The robot’s on-board computer was an Intel NUC with an Intel i7-8559U CPU, a 256 GB SSD and 16 GB of RAM. For perception, the robot was equipped with an Ouster OS0-64 3D lidar with a full 360° field of view around the robot. Real-time localisation and mapping of the environment were provided by the VILENS SLAM system [236], and—as in the simulated experiments—interfaces between modules were through ROS.

In these experiments, we use simulated radiation sources due to the health and safety difficulties of working with real radiation. Simulating radiation in real-world experiments presents the additional challenge that the radiation simulator must know the ground truth position of the robot in order to accurately simulate its radiation readings. In fully simulated experiments, the ground truth robot position is readily available from the simulator. However, there was no external ground truth source (e.g. a motion-capture system) available in the Corsham Research Mine. To provide ground truth robot positioning, we therefore ran a separate instance of the VILENS SLAM system in localisation mode using a pre-built full coverage 3D map of the environment. Independently from the ground truth localisation system, an *online* VILENS instance provided SLAM for each experiment run. The navigation graph construction component (Algorithm 7.1) used the coordinate system and OctoMap established by the online SLAM system, and had no access to the pre-built environment map.

7.5.5 Deployment in a physical environment

In order to illustrate the behaviour that can be obtained from the SafeEstMDP-Map in the real world, this section first steps through a successful run of the algorithm in the Corsham Research Mine. The output from this particular run is shown in Figure 7.7. We then also comment on the exploration performance of the system across further deployments.

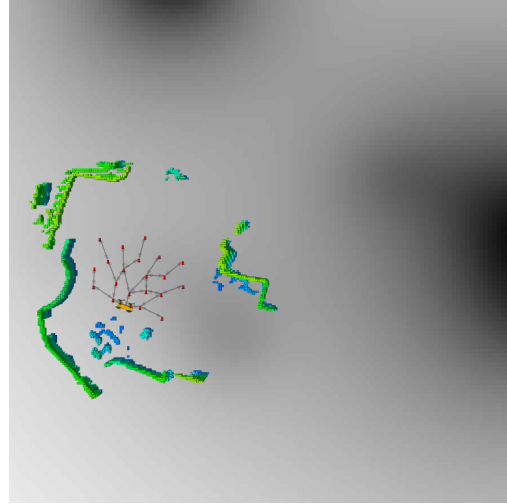
The robot started at the location indicated in Figure 7.7b, from which SafeEstMDP-Map constructed its initial topological map. The map at this point extended at most only a few metres from the robot's position, in directions in which line of sight was unobstructed. This area of the map had relatively low and uniform radiation levels, so the variation in the SafeEstMDP-Map scoring function (Equation 7.13) was dominated by the volumetric gain term. The robot's first plan was therefore to move up towards Location 2 in Figure 7.7d, since the nodes in this direction were slightly more open than those to the right, and were therefore expected to provide more information for growing the map.

Upon reaching Location 2, the robot found the corridor blocked. Additionally, now that its map of this area was more complete, there was very little unobserved space nearby, so all nearby nodes had volumetric gain values below the required threshold to act as exploration candidates. Therefore, SafeEstMDP-Map turned back towards its other unexplored frontier at Location 3, where the nodes still had high exploration scores and were considered safe.

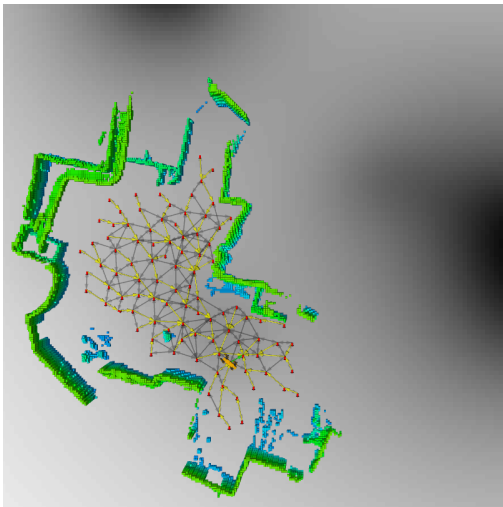
From Location 3, the robot was able to map the entire "room" located below it (Figure 7.7c), thus there was no more unexplored space there and no need to explore further downwards. Instead, the robot moved to the right, incrementally extending its map towards Location 4 as shown in the figure. It is worth noting here that Figure 7.7d does not show *every* goal selected by SafeEstMDP-Map during the run, but rather a simplified intuitive representation of the path followed—in reality, when moving into unexplored territory, the algorithm tends to plan only a few nodes ahead at a time, since far-away nodes cannot be known with high confidence to be safe.



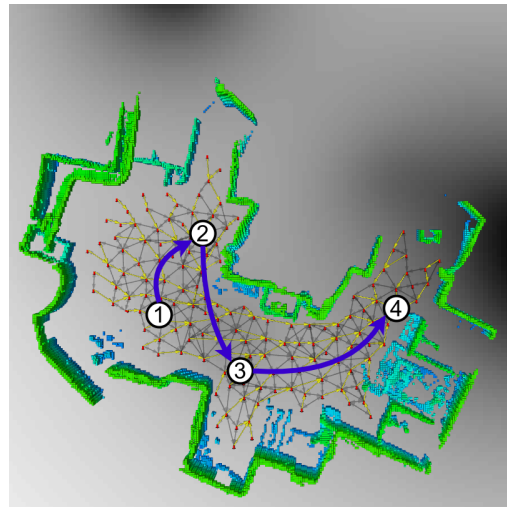
(a) Our Boston Dynamics Spot robot exploring the Corsham Research Mine in Wiltshire, UK.



(b) The map and topological graph constructed by SafeEstMDP-Map upon initialisation.



(c) The map and topological graph mid-way through the run.



(d) The final configuration and path followed by the robot during the run.

Figure 7.7: Progression of a single run of SafeEstMDP-Map in the Corsham Research Mine. (b-d) show a 3D occupancy map representing the robot's knowledge of the world around it (filtered for visual clarity), as well as a topological graph constructed online for planning and navigation. The grayscale background represents (ground truth) log radiation levels as a heatmap, with darker regions corresponding to higher levels.

As the robot approached Location 4, the radiation levels it measured began to increase, as indicated by the darker patch in Figure 7.7d. As this happened, SafeEstMDP-Map became more cautious, tending to generate plans that only moved the robot forwards one node at a time, or had it visit additional nodes near the frontier rather than pushing aggressively into the unknown region. Ultimately, when the robot did reach Location 4, the unvisited nodes ahead of it were deemed unsafe based on the GP model. The unvisited nodes in the previously explored areas of the map, on the other hand, were still safe to visit, but their volumetric gain values were below the required threshold. There were therefore no more nodes that could serve as candidates for SafeEstMDP-Map, so the run terminated at Location 4.

This is overall a successful run of SafeEstMDP-Map, exhibiting the desired behaviours of the algorithm. It balanced exploitation of volumetric gain information with safety, exploring more aggressively when confident about safety, and becoming more cautious as safety became less certain.

In total, three full runs of SafeEstMDP-Map were conducted in the Corsham Research Mine using the radiation layout shown in Figure 7.7. In all runs, the system was able to plan online and successfully explore the majority of the safely accessible area. With a total accessible area of approximately 400 m^2 , the system achieved consistent results of 348.31 m^2 , 321.93 m^2 and 345.07 m^2 of the environment explored, without violating the safety specification.

7.6 Discussion

SafeEstMDP-Map offers an efficient approach to safe exploration of completely unknown environments. The goal selection process balances safety and cost-to-reach with exploration of the environment, and combining this with a GP model of the safety hazard allows it to do this in a sensible way. In Section 7.5.3, we showed that rule-based approaches struggle to achieve this balance across different environments. Exploring too cautiously can be highly inefficient, resulting in inefficient exploration or failure to explore the full safely reachable area altogether. On the other hand, exploring too

aggressively will often result in violations of the safety constraint, which could be dangerous or costly in real-world applications.

The robustness of our algorithm could be bolstered significantly by introducing better mechanisms for hyperparameter tuning.

Our experiments used manual tuning of the scoring function parameters (Equation 7.13) in order to achieve effective exploration behaviour. However, with a small amount of data from previous runs, it would be possible to select these parameters automatically, or even to adapt them online as the robot gains a better understanding of its environment. Similarly, the use of more specialised GP kernels could also improve modelling accuracy, and therefore overall exploration performance. This could be done either by directly incorporating domain knowledge into the kernel design, or by introducing techniques such as deep kernel learning [55] to learn more expressive kernel functions from data.

The incremental graph construction algorithm from Section 7.4.2 implicitly assumes a planar environment, in which the robot can navigate purely in the horizontal plane. This is, of course, a highly limiting assumption which would preclude the use of the algorithm in many real-world environments where true 3D navigation is required. However, the SafeEstMDP-Map algorithm itself is modular, and compatible with any approach for incremental graph construction. Our approach could be extended to a broader set of challenging environments by integrating it with a more sophisticated, terrain-aware graph-based exploration algorithm (e.g. [122]).

7.7 Summary

We have presented the SafeEstMDP-Map algorithm for safe exploration and mapping of unknown environments in the presence of a GP-modelled hazard. We have addressed **C4 Limited Priors**, extending previous work on safe exploration to remove reliance on a pre-existing topological map of the environment. This paves the way for monitoring missions in completely unknown environments, enabling a robot to autonomously map the safely reachable area without putting itself at risk. The work in this chapter also contributes towards addressing **C1 Environment Modelling**, reinforcing the

effectiveness of GPs for modelling continuous environment processes in robot missions. Our experiments validated the exploration approach in simulation, and demonstrated a successful deployment on a real robot.

8

Conclusion

Contents

8.1 Summary of contributions	136
8.2 Future work	138

8.1 Summary of contributions

This thesis has contributed planning algorithms for long-term robot monitoring missions in complex and uncertain environments, geared towards applications such as ecological monitoring and precision agriculture. In doing so, we have addressed a number of the challenges that arise when planning in this context. Here, we summarise the contributions of our work to addressing the four challenges outlined in Section 1.2.

C1 Environment Modelling In Chapter 4, we formalised the problem of long-term monitoring for robots over multiple deployments as a Multi-Mission PKSMDP. We introduced a belief-based MCTS planning approach for this setting, using a GP model of an unknown spatiotemporal environment process. We demonstrated that, with a well-chosen GP kernel, such a model can enable planning in this setting, making effective use of the limited and noisy data available to the robot. Our planning approach incorporates a routine for learning GP hyperparameters online between missions, allowing the robot

to adapt its model to environments about which it has little prior knowledge. Further, we employed a novel batch evaluation method for GP-based rewards in MCTS, enabling faster rollouts to mitigate the high computational demand of trial-based planning using GPs. In Chapter 5, we scaled up our planning capabilities using a hierarchical approach, and demonstrated that our GP framework remains viable in larger domains. Throughout this thesis, we used GPs to model bioactivity in coral reefs, wildfire hotspots, and gamma radiation fields. The diversity of these applications demonstrates the versatility of GP models for planning in diverse environments.

C2 Resource Constraints In Chapter 5, we addressed planning for long-term robot monitoring missions under resource constraints, where the robot was able to replenish its resources at depots to enable uninterrupted autonomous operation. We introduced a hierarchical approach to this planning problem, where the high-level planner handled resource management and goal selection, and the low-level planner handled action selection to execute the chosen goals. We demonstrated empirically that the flexibility of allowing a robot to manage its own resources was beneficial to its mission outcomes, allowing it to adapt its active hours based on when the most reward could be gained. We also saw that the hierarchical approach enabled effective planning over longer horizons, and that—with some additional computational cost—this enabled the robot to achieve better mission outcomes. In Chapter 6, we also integrated a more realistic robot battery model into our planner, in order to better reflect the constraints of real-world operations.

C3 Stochastic Interactions The dynamics of mobile robots operating in unstructured environments are complex, which can make it difficult to accurately model the outcomes of actions. In Chapter 6, we extended the work from Chapter 5 to consider stochastic models of action durations and resource replenishment. We developed a chance-constrained planning approach, demonstrating that principled consideration of uncertainty could enable robust planning in the face of stochastic dynamics. Empirically, we showed that explicit consideration of uncertainty allowed our planner to uphold the chance constraint while still achieving comparable reward to a baseline planner. In environments with

higher degrees of stochasticity, we showed that explicitly incorporating uncertainty into planning became increasingly important.

C4 Limited Priors In Chapter 7, we addressed the challenge of safe robot exploration and mapping of environments with unknown hazards. This is an essential capability to enable robot deployments in new environments, removing the need for prior knowledge of the environment, which can be costly or difficult to obtain. We presented the SafeEstMDP-Map algorithm, which enables a robot to autonomously map the full safely reachable region of an environment, while remaining safe with respect to some GP-modelled hazard. The output of this exploration is a topological map of the safely reachable area, which can be used for subsequent deployments in the same environment. We demonstrated empirically in simulation that our approach can enable effective exploration in the presence of hazardous radiation, striking a balance between exploration and safety. Additionally, in Chapters 4, 5, and 6, we integrated GP hyperparameter learning into our planning algorithms, making them more adaptable and reducing their reliance on prior knowledge of the environment features of interest.

8.2 Future work

The work presented in this thesis opens up a variety of avenues for future research into planning for long-term robot monitoring. Although we have addressed each of the challenges outlined in Section 1.2, there are still many directions in which the boundaries of each could be pushed further. We outline some of the more promising ones below.

Scaling Gaussian processes Throughout this thesis, we used GPs to model continuous environment processes based on noisy observations. One limitation of our approach is that both the memory required for storing the dataset, and the computational demand of GP inference, grow without bound as the size of the dataset increases. While this did not pose a problem for the experiments presented in this thesis, it does in principle limit the scalability of our approach. As discussed in Section 3.1.4, techniques such as sparse approximations and windowing could be used to mitigate this issue, allowing for a limit

to be placed on the allowable size of the dataset used for GP inference. With a mechanism to cap the memory and computation demands of GP inference, the planner presented in Chapter 6 would be able to operate indefinitely, enabling true persistent monitoring. A study into the effectiveness of these techniques in the context of long-term monitoring performance over time would be a valuable extension to our work.

Stochastic transition models In Chapter 6, we introduced a stochastic model of action durations, in order to better reflect the complexity of robot dynamics in the real world. This meant that the outcomes of actions were described by distributions, rather than fixed values. However, we still assumed that actions were guaranteed to produce the intended result when executed, which is often a limiting assumption in robotics applications. An AUV navigating underwater, for example, might be swept off course by a strong current, ending up in a different location entirely than the one it intended to reach [12]. Or, a ground robot might find a path unexpectedly blocked by an obstacle, resulting in a failed action. Incorporating these outcomes into the probabilistic transition function would allow the planner to reason about them explicitly, resulting in more robust performance in challenging environments. Moreover, the transition function could be learned online, allowing the robot to adapt to changes in its environment over time.

Multi-robot monitoring Much of the literature on long-term monitoring, surveillance and coverage planning focuses on multi-robot teams, since teams can more easily cover large environments. The work presented in this thesis has focused on addressing the single-robot problem, and extending this work to multi-robot teams represents a natural next step. This would involve addressing several additional challenges, such as decentralised communication, coordination, and task allocation. However, in doing so, it could enable operation at much larger scales, and with an additional layer of redundancy to mitigate the costs associated with failure of a single robot.

References

- [1] J. Liu and R. K. Williams, “Monitoring over the long term: Intermittent deployment and sensing strategies for multi-robot teams,” in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2020, pp. 7733–7739.
- [2] A. Botta, P. Cavallone, L. Baglieri, G. Colucci, L. Tagliavini, and G. Quaglia, “A review of robots, perception, and tasks in precision agriculture,” *Applied Mechanics*, vol. 3, no. 3, pp. 830–854, 2022.
- [3] S. Griffith and C. Pradalier, “Survey registration for long-term natural environment monitoring,” *Journal of Field Robotics*, vol. 34, no. 1, pp. 188–208, 2017.
- [4] J. Dong, J. G. Burnham, B. Boots, G. Rains, and F. Dellaert, “4D crop monitoring: Spatio-temporal reconstruction for agriculture,” in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2017, pp. 3878–3885.
- [5] R. Bailon-Ruiz, A. Bit-Monnot, and S. Lacroix, “Real-time wildfire monitoring with a fleet of UAVs,” *Robotics and Autonomous Systems*, vol. 152, p. 104 071, 2022.
- [6] M. S. Couceiro, D. Portugal, J. F. Ferreira, and R. P. Rocha, “SEMFIRE: Towards a new generation of forestry maintenance multi-robot systems,” in *2019 IEEE/SICE International Symposium on System Integration (SII)*, IEEE, 2019, pp. 270–276.
- [7] F. Afghah, A. Razi, J. Chakareski, and J. Ashdown, “Wildfire monitoring in remote areas using autonomous unmanned aerial vehicles,” in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, IEEE, 2019, pp. 835–840.
- [8] G. Muscato, F. Bonaccorso, L. Cantelli, D. Longo, and C. D. Melita, “Volcanic environments: Robots for exploration and measurement,” *IEEE Robotics & Automation Magazine*, vol. 19, no. 1, pp. 40–49, 2012.
- [9] M. Dunbabin and L. Marques, “Robots for environmental monitoring: Significant advancements and applications,” *IEEE Robotics & Automation Magazine*, vol. 19, no. 1, pp. 24–39, 2012.
- [10] A. Stephens, B. Lacerda, and N. Hawes, “Planning for long-term monitoring missions in time-varying environments,” in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2024.

- [11] A. Stephens, B. Lacerda, and N. Hawes, “Hierarchical planning for resource-constrained long-term monitoring missions in time-varying environments,” in *European Conference on Artificial Intelligence (ECAI)*, IOS Press, 2024.
- [12] A. Stephens, M. Budd, M. Staniaszek, B. Casseau, P. Duckworth, M. Fallon, N. Hawes, and B. Lacerda, “Planning under uncertainty for safe robot exploration using Gaussian process prediction,” *Autonomous Robots*, vol. 48, no. 7, pp. 1–29, 2024.
- [13] B. Kuipers and Y.-T. Byun, “A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations,” *Robotics and autonomous systems*, vol. 8, no. 1-2, pp. 47–63, 1991.
- [14] US Geological Survey, *ESRI World Imagery*, <https://earthexplorer.usgs.gov/>, Accessed: 2024-10-02, 2020.
- [15] C. K. Williams and C. E. Rasmussen, *Gaussian processes for machine learning*. MIT Press Cambridge, MA, 2006, vol. 2.
- [16] J. Hensman, N. Fusi, and N. D. Lawrence, “Gaussian processes for big data,” in *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*, 2013, pp. 282–290.
- [17] R. Bellman, “A Markovian decision process,” *Journal of mathematics and mechanics*, pp. 679–684, 1957.
- [18] T. M. Moldovan and P. Abbeel, “Safe exploration in Markov decision processes,” in *Proceedings of the 29th International Conference on Machine Learning*, ser. ICML’12, Edinburgh, Scotland: Omnipress, 2012, pp. 1451–1458, ISBN: 9781450312851.
- [19] M. Turchetta, F. Berkenkamp, and A. Krause, “Safe exploration in finite Markov decision processes with Gaussian processes,” *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 29, 2016.
- [20] M. Budd, B. Lacerda, P. Duckworth, A. West, B. Lennox, and N. Hawes, “Markov decision processes with unknown state feature values for safe exploration using Gaussian processes,” in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2020, pp. 7344–7350.
- [21] A. Fern, S. Natarajan, K. Judah, and P. Tadepalli, “A decision-theoretic model of assistance,” *Journal of Artificial Intelligence Research*, vol. 50, pp. 71–104, 2014.
- [22] L. Feng, C. Wiltche, L. Humphrey, and U. Topcu, “Synthesis of human-in-the-loop control protocols for autonomous systems,” *IEEE Transactions on Automation Science and Engineering*, vol. 13, no. 2, pp. 450–462, 2016.

- [23] C. Costen, M. Rigter, B. Lacerda, and N. Hawes, “Shared autonomy systems with stochastic operator models,” *International Joint Conferences on Artificial Intelligence Organization*, 2022.
- [24] A. Singh, A. Krause, and W. J. Kaiser, “Nonmyopic adaptive informative path planning for multiple robots,” in *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI)*, 2009, pp. 1843–1850.
- [25] J. Binney and G. S. Sukhatme, “Branch and bound for informative path planning,” in *2012 IEEE International Conference on Robotics and Automation*, IEEE, 2012, pp. 2147–2154.
- [26] Z. W. Lim, D. Hsu, and W. S. Lee, “Adaptive informative path planning in metric spaces,” *The International Journal of Robotics Research*, vol. 35, no. 5, pp. 585–598, 2016.
- [27] C. Street, B. Lacerda, M. Mühlig, and N. Hawes, “Multi-robot planning under uncertainty with congestion-aware models,” in *Adaptive Agents and Multi-Agent Systems*, 2020.
- [28] D. Claes, F. Oliehoek, H. Baier, and K. Tuyls, “Decentralised online planning for multi-robot warehouse commissioning,” in *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, 2017, pp. 492–500.
- [29] F. Faruq, D. Parker, B. Lacerda, and N. Hawes, “Simultaneous task allocation and planning under uncertainty,” in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2018, pp. 3559–3564.
- [30] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, “Planning and acting in partially observable stochastic domains,” *Artificial intelligence*, vol. 101, no. 1-2, pp. 99–134, 1998.
- [31] S. Thrun, “Probabilistic robotics,” *Communications of the ACM*, vol. 45, no. 3, pp. 52–57, 2002.
- [32] T. D. Barfoot, *State estimation for robotics*. Cambridge University Press, 2024.
- [33] M. A. Osborne, S. J. Roberts, A. Rogers, S. D. Ramchurn, and N. R. Jennings, “Towards real-time information processing of sensor network data using computationally efficient multi-output Gaussian processes,” in *2008 international conference on information processing in sensor networks (IPSN 2008)*, IEEE, 2008, pp. 109–120.
- [34] M. Budd, P. Duckworth, N. Hawes, and B. Lacerda, “Bayesian reinforcement learning for single-episode missions in partially unknown environments,” in *Conference on Robot Learning*, PMLR, 2023, pp. 1189–1198.

- [35] R. Bellman, “Dynamic programming and Lagrange multipliers,” *Proceedings of the National Academy of Sciences*, vol. 42, no. 10, pp. 767–769, 1956.
- [36] L. Kocsis and C. Szepesvári, “Bandit based Monte-Carlo planning,” in *European Conference on Machine Learning*, Springer, 2006, pp. 282–293.
- [37] D. Auger, A. Couetoux, and O. Teytaud, “Continuous upper confidence trees with polynomial exploration–consistency,” in *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2013, Prague, Czech Republic, September 23–27, 2013, Proceedings, Part I 13*, Springer, 2013, pp. 194–209.
- [38] Z. Sunberg and M. Kochenderfer, “Online algorithms for pomdps with continuous state, action, and observation spaces,” in *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 28, 2018, pp. 259–263.
- [39] A. Couëtoux, J.-B. Hoock, N. Sokolovska, O. Teytaud, and N. Bonnard, “Continuous upper confidence trees,” in *Learning and Intelligent Optimization: 5th International Conference, LION 5, Rome, Italy, January 17–21, 2011. Selected Papers 5*, Springer, 2011, pp. 433–445.
- [40] J. R. Norris, *Markov chains*. Cambridge university press, 1998.
- [41] N. Golding and B. V. Purse, “Fast and flexible Bayesian species distribution modelling using Gaussian processes,” *Methods in Ecology and Evolution*, vol. 7, no. 5, pp. 598–608, 2016.
- [42] M. Ingram, D. Vukcevic, and N. Golding, “Multi-output Gaussian processes for species distribution modelling,” *Methods in ecology and evolution*, vol. 11, no. 12, pp. 1587–1598, 2020.
- [43] W. J. Wright, K. M. Irvine, T. J. Rodhouse, and A. R. Litt, “Spatial Gaussian processes improve multi-species occupancy models when range boundaries are uncertain and nonoverlapping,” *Ecology and Evolution*, vol. 11, no. 13, pp. 8516–8527, 2021.
- [44] M. V. Talluto, K. Mokany, L. J. Pollock, and W. Thuiller, “Multifaceted biodiversity modelling at macroecological scales using Gaussian processes,” *Diversity and Distributions*, vol. 24, pp. 1492–1502, 10 Oct. 2018, ISSN: 14724642. DOI: [10.1111/ddi.12781](https://doi.org/10.1111/ddi.12781).
- [45] L. Bottarelli, M. Bicego, J. Blum, and A. Farinelli, “Orienteering-based informative path planning for environmental monitoring,” *Engineering Applications of Artificial Intelligence*, vol. 77, pp. 46–58, 2019.

- [46] P. Duckworth, B. Lacerda, and N. Hawes, “Time-bounded mission planning in time-varying domains with semi-MDPs and Gaussian processes,” in *Conference on Robot Learning*, PMLR, 2021, pp. 1654–1668.
- [47] L. L. Duan, X. Wang, J. P. Clancy, and R. D. Szczesniak, “Joint hierarchical Gaussian process model with application to personalized prediction in medical monitoring,” *Stat*, vol. 7, 1 2018, ISSN: 20491573. DOI: [10.1002/sta4.178](https://doi.org/10.1002/sta4.178).
- [48] S. N. Wood, *Generalized additive models: an introduction with R*. Chapman and Hall/CRC, 2017.
- [49] D. Heckerman, “A tutorial on learning with Bayesian networks,” *Learning in graphical models*, pp. 301–354, 1998.
- [50] L. Uusitalo, “Advantages and challenges of Bayesian networks in environmental modelling,” *Ecological modelling*, vol. 203, no. 3-4, pp. 312–318, 2007.
- [51] P. A. Aguilera, A. Fernández, R. Fernández, R. Rumí, and A. Salmerón, “Bayesian networks in environmental modelling,” *Environmental Modelling & Software*, vol. 26, no. 12, pp. 1376–1388, 2011.
- [52] M. Bauer, M. Van der Wilk, and C. E. Rasmussen, “Understanding probabilistic sparse Gaussian process approximations,” *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 29, 2016.
- [53] J. Quinonero-Candela and C. E. Rasmussen, “A unifying view of sparse approximate Gaussian process regression,” *The Journal of Machine Learning Research*, vol. 6, pp. 1939–1959, 2005.
- [54] A. Damianou and N. D. Lawrence, “Deep Gaussian processes,” in *Artificial intelligence and statistics*, PMLR, 2013, pp. 207–215.
- [55] A. G. Wilson, Z. Hu, R. Salakhutdinov, and E. P. Xing, “Deep kernel learning,” in *Artificial intelligence and statistics*, PMLR, 2016, pp. 370–378.
- [56] H. Salimbeni and M. Deisenroth, “Doubly stochastic variational inference for deep Gaussian processes,” *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 30, 2017.
- [57] M. Havasi, J. M. Hernández-Lobato, and J. J. Murillo-Fuentes, “Inference in deep Gaussian processes using stochastic gradient Hamiltonian Monte Carlo,” *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 31, 2018.
- [58] H. Liu, Y.-S. Ong, X. Shen, and J. Cai, “When Gaussian process meets big data: A review of scalable gps,” *IEEE transactions on neural networks and learning systems*, vol. 31, no. 11, pp. 4405–4423, 2020.

- [59] K. Jakkala, “Deep Gaussian processes: A survey,” *arXiv preprint arXiv:2106.12135*, 2021.
- [60] M. Popović, J. Ott, J. Rückin, and M. J. Kochenderfer, “Learning-based methods for adaptive informative path planning,” *Robotics and Autonomous Systems*, p. 104 727, 2024.
- [61] I. M. R. Fernández, C. E. Denniston, D. A. Caron, and G. S. Sukhatme, “Informative path planning to estimate quantiles for environmental analysis,” *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 10 280–10 287, 2022.
- [62] S. Choudhury, N. Gruver, and M. J. Kochenderfer, “Adaptive informative path planning with multimodal sensing,” in *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 30, 2020, pp. 57–65.
- [63] Y. Cao, Y. Wang, A. Vashisth, H. Fan, and G. A. Sartoretti, “CAtnIPP: Context-aware attention-based network for informative path planning,” in *Conference on Robot Learning*, PMLR, 2023, pp. 1928–1937.
- [64] E. Galceran and M. Carreras, “A survey on coverage path planning for robotics,” *Robotics and Autonomous systems*, vol. 61, no. 12, pp. 1258–1276, 2013.
- [65] C. S. Tan, R. Mohd-Mokhtar, and M. R. Arshad, “A comprehensive review of coverage path planning in robotics using classical and heuristic algorithms,” *IEEE Access*, vol. 9, pp. 119 310–119 342, 2021.
- [66] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, “NeRF: Representing scenes as neural radiance fields for view synthesis,” *Communications of the ACM*, vol. 65, no. 1, pp. 99–106, 2021.
- [67] A. Yu, V. Ye, M. Tancik, and A. Kanazawa, “Pixelnerf: Neural radiance fields from one or few images,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 4578–4587.
- [68] J. Rückin, L. Jin, and M. Popović, “Adaptive informative path planning using deep reinforcement learning for UAV-based active sensing,” in *2022 International Conference on Robotics and Automation (ICRA)*, IEEE, 2022, pp. 4473–4479.
- [69] C. E. Denniston, O. Peltzer, J. Ott, S. Moon, S.-K. Kim, G. S. Sukhatme, M. J. Kochenderfer, M. Schwager, and A.-a. Agha-mohammadi, “Fast and scalable signal inference for active robotic source seeking,” in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2023, pp. 7909–7915.
- [70] G. Hitz, E. Galceran, M.-È. Garneau, F. Pomerleau, and R. Siegwart, “Adaptive continuous-space informative path planning for online environmental monitoring,” *Journal of Field Robotics*, vol. 34, no. 8, pp. 1427–1449, 2017.

- [71] S. Yanes Luis, M. Perales Esteve, D. Gutiérrez Reina, and S. Toral Marín, “Deep reinforcement learning applied to multi-agent informative path planning in environmental missions,” in *Mobile Robot: Motion Control and Path Planning*, Springer, 2023, pp. 31–61.
- [72] K. C. Vivaldini, T. H. Martinelli, V. C. Guizilini, J. R. Souza, M. D. Oliveira, F. T. Ramos, and D. F. Wolf, “Uav route planning for active disease classification,” *Autonomous robots*, vol. 43, pp. 1137–1153, 2019.
- [73] M. Popović, T. Vidal-Calleja, G. Hitz, J. J. Chung, I. Sa, R. Siegwart, and J. Nieto, “An informative path planning framework for UAV-based terrain monitoring,” *Autonomous Robots*, vol. 44, no. 6, pp. 889–911, 2020.
- [74] J. Ott, E. Balaban, and M. J. Kochenderfer, “Sequential Bayesian optimization for adaptive informative path planning with multimodal sensing,” in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2023, pp. 7894–7901.
- [75] C. E. Denniston, G. Salhotra, A. Kangaslahti, D. A. Caron, and G. S. Sukhatme, “Learned parameter selection for robotic information gathering,” in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2023, pp. 10 519–10 526.
- [76] Y. Wang, Y. Wang, Y. Cao, and G. Sartoretti, “Spatio-temporal attention network for persistent monitoring of multiple mobile targets,” in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2023, pp. 3903–3910.
- [77] G. A. Hollinger and G. S. Sukhatme, “Sampling-based robotic information gathering algorithms,” *The International Journal of Robotics Research*, vol. 33, no. 9, pp. 1271–1287, 2014.
- [78] G. Flaspohler, V. Preston, A. P. Michel, Y. Girdhar, and N. Roy, “Information-guided robotic maximum seek-and-sample in partially observable continuous environments,” *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3782–3789, 2019.
- [79] R. Marchant, F. Ramos, S. Sanner, *et al.*, “Sequential Bayesian optimisation for spatial-temporal monitoring,” in *UAI*, Citeseer, 2014, pp. 553–562.
- [80] S. Bai, F. Chen, and B. Englot, “Toward autonomous mapping and exploration for mobile robots through deep supervised learning,” in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2017, pp. 2379–2384.

- [81] E. Zwecher, E. Iceland, S. R. Levy, S. Y. Hayoun, O. Gal, and A. Barel, “Integrating deep reinforcement and supervised learning to expedite indoor mapping,” in *2022 International Conference on Robotics and Automation (ICRA)*, IEEE, 2022, pp. 10 542–10 548.
- [82] J. A. Caley, N. R. Lawrance, and G. A. Hollinger, “Deep learning of structured environments for robot search,” *Autonomous Robots*, vol. 43, pp. 1695–1714, 2019.
- [83] G. Georgakis, B. Bucher, A. Arapin, K. Schmeckpeper, N. Matni, and K. Daniilidis, “Uncertainty-driven planner for exploration and navigation,” in *2022 International Conference on Robotics and Automation (ICRA)*, IEEE, 2022, pp. 11 295–11 302.
- [84] B. Hepp, D. Dey, S. N. Sinha, A. Kapoor, N. Joshi, and O. Hilliges, “Learn-to-score: Efficient 3d scene exploration by predicting view utility,” in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 437–452.
- [85] *MODIS Collection 6 Hotspot / Active Fire Detections MCD14ML distributed from NASA FIRMS*, <https://earthdata.nasa.gov/firms.doi:10.5067/FIRMS/MODIS/MCD14ML>, Accessed: 2024-03-27, 2024.
- [86] M. J. Kochenderfer, T. A. Wheeler, and K. H. Wray, *Algorithms for decision making*. MIT Press, 2022.
- [87] S. Choudhury, M. Bhardwaj, S. Arora, A. Kapoor, G. Ranade, S. Scherer, and D. Dey, “Data-driven planning via imitation learning,” *The International Journal of Robotics Research*, vol. 37, no. 13-14, pp. 1632–1672, 2018.
- [88] Z. Liu, M. Deshpande, T. Qi, D. Zhao, R. Madhivanan, and A. Sen, “Learning to explore (l2e): Deep reinforcement learning-based autonomous exploration for household robot,” in *Robotics: Science and Systems*, 2023.
- [89] C. H. Papadimitriou and J. N. Tsitsiklis, “The complexity of Markov decision processes,” *Mathematics of operations research*, vol. 12, no. 3, pp. 441–450, 1987.
- [90] D. Silver and J. Veness, “Monte-Carlo planning in large POMDPs,” *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 23, 2010.
- [91] W. Lee, N. Rong, and D. Hsu, “What makes some pomdp problems easy to approximate?” *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 20, 2007.
- [92] J. Pineau, G. Gordon, and S. Thrun, “Anytime point-based approximations for large pomdps,” *J. Artif. Int. Res.*, vol. 27, no. 1, pp. 335–380, 2006, ISSN: 1076-9757.
- [93] M. Lauri, D. Hsu, and J. Pajarinen, “Partially observable Markov decision processes in robotics: A survey,” *IEEE Transactions on Robotics*, vol. 39, no. 1, pp. 21–40, 2022.

- [94] A. Somani, N. Ye, D. Hsu, and W. S. Lee, “DESPOT: Online POMDP planning with regularization,” *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 26, 2013.
- [95] P. Chaudhari, S. Karaman, D. Hsu, and E. Frazzoli, “Sampling-based algorithms for continuous-time pomdps,” in *2013 American Control Conference*, IEEE, 2013, pp. 4604–4610.
- [96] S. Mahadevan, “Partially observable semi-Markov decision processes: Theory and applications in engineering and cognitive science,” in *AAAI Fall Symposium on Planning with Partially Observable Markov Decision Processes*, Orlando, FL, USA, 1998, pp. 113–120.
- [97] S. Omidshafiei, A.-A. Agha-Mohammadi, C. Amato, and J. P. How, “Decentralized control of partially observable Markov decision processes using belief space macro-actions,” in *2015 IEEE international conference on robotics and automation (ICRA)*, IEEE, 2015, pp. 5962–5969.
- [98] S. Omidshafiei, A.-A. Agha-Mohammadi, C. Amato, S.-Y. Liu, J. P. How, and J. Vian, “Graph-based cross entropy method for solving multi-robot decentralized pomdps,” in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2016, pp. 5395–5402.
- [99] A. Wachi, Y. Sui, Y. Yue, and M. Ono, “Safe exploration and optimization of constrained mdps using Gaussian processes,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, 2018.
- [100] J. Fink and V. Kumar, “Online methods for radio signal mapping with mobile robots,” in *2010 IEEE International Conference on Robotics and Automation*, IEEE, 2010, pp. 1940–1945.
- [101] A. Viseras and R. Garcia, “DeepIG: Multi-robot information gathering with deep reinforcement learning,” *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 3059–3066, 2019.
- [102] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. De Freitas, “Taking the human out of the loop: A review of bayesian optimization,” *Proceedings of the IEEE*, vol. 104, no. 1, pp. 148–175, 2015.
- [103] P. Morere, R. Marchant, and F. Ramos, “Sequential Bayesian optimization as a POMDP for environment monitoring with UAVs,” in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2017, pp. 6381–6388.
- [104] R. Marchant and F. Ramos, “Bayesian optimisation for informative continuous path planning,” in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 6136–6143. DOI: [10.1109/ICRA.2014.6907763](https://doi.org/10.1109/ICRA.2014.6907763).

- [105] M. O. Duff, *Optimal Learning: Computational procedures for Bayes-adaptive Markov decision processes*. University of Massachusetts Amherst, 2002.
- [106] A. Guez, D. Silver, and P. Dayan, “Scalable and efficient Bayes-adaptive reinforcement learning based on Monte-Carlo tree search,” *Journal of Artificial Intelligence Research*, vol. 48, pp. 841–883, 2013.
- [107] M. Pecka and T. Svoboda, “Safe exploration techniques for reinforcement learning—an overview,” in *Modelling and Simulation for Autonomous Systems: First International Workshop, MESAS 2014, Rome, Italy, May 5-6, 2014, Revised Selected Papers 1*, Springer, 2014, pp. 357–375.
- [108] Y. Sui, A. Gotovos, J. Burdick, and A. Krause, “Safe exploration for optimization with Gaussian processes,” in *International conference on machine learning*, PMLR, 2015, pp. 997–1005.
- [109] Y. Sui, V. Zhuang, J. Burdick, and Y. Yue, “Stagewise safe Bayesian optimization with Gaussian processes,” in *ICML*, PMLR, 2018.
- [110] T. S. Badings, A. Abate, N. Jansen, D. Parker, H. A. Poonawala, and M. Stoelinga, “Sampling-based robust control of autonomous systems with non-Gaussian noise,” 2022.
- [111] A. Rutherford, P. Duckworth, N. Hawes, and B. Lacerda, “Motion planning in uncertain environments with rapidly-exploring random Markov decision processes,” in *2021 European Conference on Mobile Robots (ECMR)*, IEEE, 2021.
- [112] B. Yamauchi, “Frontier-based exploration using multiple robots,” in *Proceedings of the second international conference on Autonomous agents*, 1998, pp. 47–53.
- [113] B. Yamauchi, A. Schultz, and W. Adams, “Mobile robot exploration and map-building with continuous localization,” vol. 4, Institute of Electrical and Electronics Engineers Inc., 1998, pp. 3715–3720, ISBN: 078034300X. DOI: [10.1109/ROBOT.1998.681416](https://doi.org/10.1109/ROBOT.1998.681416).
- [114] L. Freda and G. Oriolo, “Frontier-based probabilistic strategies for sensor-based exploration,” in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, IEEE, 2005, pp. 3881–3887.
- [115] A. Akbari and S. Bernardini, “Informed autonomous exploration of subterranean environments,” *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7957–7964, 2021.

- [116] M. Tranzatto, M. Dharmadhikari, L. Bernreiter, M. Camurri, S. M. K. Khattak, F. Mascarich, P. Pfreundschuh, D. Wisth, S. Zimmermann, M. Kulkarni, *et al.*, “Team CERBERUS wins the DARPA Subterranean Challenge: Technical overview and lessons learned,” 2022.
- [117] T. Rouček, M. Pecka, P. Čížek, T. Petříček, J. Bayer, V. Šalanský, T. Azayev, D. Heřt, M. Petrлік, T. Báča, *et al.*, “System for multi-robotic exploration of underground environments CTU-CRAS-NORLAB in the DARPA Subterranean Challenge,” *Journal of Field Robotics*, vol. 2, 2022.
- [118] N. Hudson, F. Talbot, M. Cox, J. Williams, T. Hines, A. Pitt, B. Wood, D. Frousheger, K. Lo Surdo, T. Molnar, *et al.*, “Heterogeneous ground and air platforms, homogeneous sensing: Team CSIRO Data61’s approach to the DARPA Subterranean Challenge,” *Journal of Field Robotics*, vol. 2, 2022.
- [119] B. Morrell, R. Thakker, À. Santamaria Navarro, A. Bouman, X. Lei, J. Edlund, T. Pailevanian, T. S. Vaquero, Y. L. Chang, T. Touma, *et al.*, “NeBula: Team CoSTAR’s robotic autonomy solution that won Phase II of DARPA Subterranean Challenge,” *Field robotics*, vol. 2, pp. 1432–1506, 2022.
- [120] S. Scherer, V. Agrawal, G. Best, C. Cao, K. Cujic, R. Darnley, R. DeBortoli, E. Dexheimer, B. Drozd, R. Garg, *et al.*, “Resilient and modular subterranean exploration with a team of roving and flying robots,” *Journal of Field Robotics*, vol. 2, 2022.
- [121] J. Bayer and J. Faigl, “On autonomous spatial exploration with small hexapod walking robot using tracking camera Intel RealSense T265,” in *2019 European Conference on Mobile Robots (ECMR)*, IEEE, 2019.
- [122] T. Dang, M. Tranzatto, S. Khattak, F. Mascarich, K. Alexis, and M. Hutter, “Graph-based subterranean exploration path planning using aerial and legged robots,” *Journal of Field Robotics*, vol. 37, pp. 1363–1388, 8 2020.
- [123] J. Williams, S. Jiang, M. O’Brien, G. Wagner, E. Hernandez, M. Cox, A. Pitt, R. Arkin, and N. Hudson, “Online 3D frontier-based UGV and UAV exploration using direct point cloud visibility,” in *IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, IEEE, 2020, pp. 263–270.
- [124] T. Dang, S. Khattak, F. Mascarich, and K. Alexis, “Explore locally, plan globally: A path planning framework for autonomous robotic exploration in subterranean environments,” in *2019 19th International Conference on Advanced Robotics (ICAR)*, IEEE, 2019, pp. 9–16.

- [125] C. Cao, H. Zhu, H. Choset, and J. Zhang, “Exploring large and complex environments fast and efficiently,” in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2021, pp. 7781–7787.
- [126] S.-K. Kim, A. Bouman, G. Salhotra, D. D. Fan, K. Otsu, J. Burdick, and A.-a. Aghamohammadi, “PLGRIM: Hierarchical value learning for large-scale exploration in unknown environments,” in *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 31, 2021, pp. 652–662.
- [127] V. Verma, M. W. Maimone, D. M. Gaines, R. Francis, T. A. Estlin, S. R. Kuhn, G. R. Rabideau, S. A. Chien, M. M. McHenry, E. J. Graser, *et al.*, “Autonomous robotics is driving Perseverance rover’s progress on Mars,” *Science Robotics*, vol. 8, no. 80, eadi3099, 2023.
- [128] P. Testor, B. De Young, D. L. Rudnick, S. Glenn, D. Hayes, C. M. Lee, C. Pattiaratchi, K. Hill, E. Heslop, V. Turpin, *et al.*, “OceanGliders: A component of the integrated GOOS,” *Frontiers in Marine Science*, vol. 6, p. 422, 2019.
- [129] A. B. Phillips, R. Templeton, D. Roper, R. Morrison, M. Pebody, P. M. Bagley, R. Marlow, E. Chaney, J. Burris, A. Consensi, *et al.*, “Autosub Long Range 1500: A continuous 2000 km field trial,” *Ocean Engineering*, vol. 280, p. 114 626, 2023.
- [130] M. Wakchaure, B. Patle, and A. Mahindrakar, “Application of ai techniques and robotics in agriculture: A review,” *Artificial Intelligence in the Life Sciences*, vol. 3, p. 100 057, 2023.
- [131] K. Mateja, W. Skarka, M. Peciak, R. Niestrój, and M. Gude, “Energy autonomy simulation model of solar powered UAV,” *Energies*, vol. 16, no. 1, p. 479, 2023.
- [132] R.-A. Peloquin, D. Thibault, and A. L. Desbiens, “Design of a passive vertical takeoff and landing aquatic UAV,” *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 381–388, 2016.
- [133] S. A. H. Mohsan, M. A. Khan, F. Noor, I. Ullah, and M. H. Alsharif, “Towards the unmanned aerial vehicles (UAVs): A comprehensive review,” *Drones*, vol. 6, no. 6, p. 147, 2022.
- [134] L. Chen, P. Wu, K. Chitta, B. Jaeger, A. Geiger, and H. Li, “End-to-end autonomous driving: Challenges and frontiers,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.
- [135] L. Kunze, N. Hawes, T. Duckett, M. Hanheide, and T. Krajník, “Artificial intelligence for long-term robot autonomy: A survey,” *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 4023–4030, 2018.

- [136] N. Hawes, C. Burbridge, F. Jovan, L. Kunze, B. Lacerda, L. Mudrova, J. Young, J. Wyatt, D. Hebesberger, T. Kortner, *et al.*, “The STRANDS project: Long-term autonomy in everyday environments,” *IEEE Robotics & Automation Magazine*, vol. 24, no. 3, pp. 146–156, 2017.
- [137] K. Trainum, R. Tunis, B. Xie, E. Hauser, *et al.*, “Robots in assisted living facilities: Scoping review,” *JMIR aging*, vol. 6, no. 1, e42652, 2023.
- [138] T. G. Dietterich, “Hierarchical reinforcement learning with the MAXQ value function decomposition,” *Journal of artificial intelligence research*, vol. 13, pp. 227–303, 2000.
- [139] C. Diuk, A. L. Strehl, and M. L. Littman, “A hierarchical approach to efficient reinforcement learning in deterministic domains,” in *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, 2006, pp. 313–319.
- [140] K. Erol, J. Hendler, and D. S. Nau, “HTN planning: Complexity and expressivity,” in *AAAI*, vol. 94, 1994, pp. 1123–1128.
- [141] K. Erol, J. Hendler, and D. S. Nau, “Complexity results for HTN planning,” *Annals of Mathematics and Artificial Intelligence*, vol. 18, no. 1, pp. 69–93, 1996.
- [142] O. Obst, “Using a planner for coordination of multiagent team behavior,” in *International Workshop on Programming Multi-Agent Systems*, Springer, 2005, pp. 90–100.
- [143] B. Hayes and B. Scassellati, “Autonomously constructing hierarchical task networks for planning and human-robot collaboration,” in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2016, pp. 5469–5476.
- [144] C. Hogg, U. Kuter, and H. Munoz-Avila, “Learning hierarchical task networks for nondeterministic planning domains,” in *Twenty-First International Joint Conference on Artificial Intelligence*, 2009.
- [145] R. S. Sutton, D. Precup, and S. Singh, “Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning,” *Artificial intelligence*, vol. 112, no. 1-2, pp. 181–211, 1999.
- [146] N. Gopalan, M. Littman, J. MacGlashan, S. Squire, S. Tellex, J. Winder, L. Wong, *et al.*, “Planning with abstract Markov decision processes,” in *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 27, 2017, pp. 480–488.

- [147] L. P. Kaelbling and T. Lozano-Pérez, “Hierarchical task and motion planning in the now,” in *2011 IEEE International Conference on Robotics and Automation*, IEEE, 2011, pp. 1470–1477.
- [148] J. Binney, A. Krause, and G. S. Sukhatme, “Optimizing waypoints for monitoring spatiotemporal phenomena,” *The International Journal of Robotics Research*, vol. 32, no. 8, pp. 873–888, 2013.
- [149] A. Singh, F. Ramos, H. D. Whyte, and W. J. Kaiser, “Modeling and decision making in spatio-temporal processes for environmental surveillance,” in *2010 IEEE International Conference on Robotics and Automation*, IEEE, 2010, pp. 5490–5497.
- [150] J. M. Santos, T. Krajník, and T. Duckett, “Spatio-temporal exploration strategies for long-term autonomy of mobile robots,” *Robotics and Autonomous Systems*, vol. 88, pp. 116–126, 2017.
- [151] T. M. Sears and J. A. Marshall, “Mapping of spatiotemporal scalar fields by mobile robots using Gaussian process regression,” in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2022, pp. 6651–6656.
- [152] K.-C. Ma, L. Liu, H. K. Heidarsson, and G. S. Sukhatme, “Data-driven learning and planning for environmental sampling,” *Journal of Field Robotics*, vol. 35, no. 5, pp. 643–661, 2018.
- [153] K. Tiwari, V. Honoré, S. Jeong, N. Y. Chong, and M. P. Deisenroth, “Resource-constrained decentralized active sensing for multi-robot systems using distributed Gaussian processes,” in *2016 16th International Conference on Control, Automation and Systems (ICCAS)*, IEEE, 2016, pp. 13–18.
- [154] X. Lan and M. Schwager, “Rapidly exploring random cycles: Persistent estimation of spatiotemporal fields with multiple sensing robots,” *IEEE Transactions on Robotics*, vol. 32, no. 5, pp. 1230–1244, 2016.
- [155] K.-C. Ma, Z. Ma, L. Liu, and G. S. Sukhatme, “Multi-robot informative and adaptive planning for persistent environmental monitoring,” in *Distributed Autonomous Robotic Systems: The 14th International Symposium*, Springer, 2018, pp. 285–298.
- [156] L. V. Nguyen, S. Kodagoda, R. Ranasinghe, and G. Dissanayake, “Information-driven adaptive sampling strategy for mobile robotic wireless sensor network,” *IEEE Transactions on Control Systems Technology*, vol. 24, no. 1, pp. 372–379, 2015.
- [157] K. H. Low, J. M. Dolan, and P. Khosla, “Adaptive multi-robot wide-area exploration and mapping,” in *Proceedings of the 7th International Joint Conference on Autonomous agents and Multiagent systems-Volume 1*, 2008, pp. 23–30.

- [158] Y. Kantaros, B. Schlotfeldt, N. Atanasov, and G. J. Pappas, “Sampling-based planning for non-myopic multi-robot information gathering,” *Autonomous Robots*, vol. 45, no. 7, pp. 1029–1046, 2021.
- [159] V. Suryan and P. Tokekar, “Learning a spatial field in minimum time with a team of robots,” *IEEE Transactions on Robotics*, vol. 36, no. 5, pp. 1562–1576, 2020.
- [160] S. Nayak, M. Greiff, A. Raghunathan, S. Di Cairano, and A. P. Vinod, “Data-driven monitoring with mobile sensors and charging stations using multi-arm bandits and coordinated motion planners,” in *2024 American Control Conference (ACC)*, IEEE, 2024, pp. 299–305.
- [161] A. Krause, A. Singh, and C. Guestrin, “Near-optimal sensor placements in Gaussian processes: Theory, efficient algorithms and empirical studies,” *Journal of Machine Learning Research*, vol. 9, no. 2, 2008.
- [162] P. Vansteenwegen, W. Souffriau, and D. Van Oudheusden, “The orienteering problem: A survey,” *European Journal of Operational Research*, vol. 209, no. 1, pp. 1–10, 2011.
- [163] J. Yu, M. Schwager, and D. Rus, “Correlated orienteering problem and its application to persistent monitoring tasks,” *IEEE Transactions on Robotics*, vol. 32, no. 5, pp. 1106–1118, 2016.
- [164] J. Yu, J. Aslam, S. Karaman, and D. Rus, “Anytime planning of optimal schedules for a mobile sensing robot,” in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2015, pp. 5279–5286.
- [165] X. Lan and M. Schwager, “Planning periodic persistent monitoring trajectories for sensing robots in Gaussian random fields,” in *2013 IEEE international conference on robotics and automation*, IEEE, 2013, pp. 2415–2420.
- [166] S. C. Pinto, S. B. Andersson, J. M. Hendrickx, and C. G. Cassandras, “Multi-agent infinite horizon persistent monitoring of targets with uncertain states in multi-dimensional environments,” *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 10 963–10 968, 2020.
- [167] A. Jones, M. Schwager, and C. Belta, “Information-guided persistent monitoring under temporal logic constraints,” in *2015 American Control Conference (ACC)*, IEEE, 2015, pp. 1911–1916.
- [168] A. B. Asghar and P. Tokekar, “Energy-aware route planning for a battery-constrained robot with multiple charging depots,” *CoRR*, vol. abs/2310.00834, 2023. DOI: [10.48550/ARXIV.2310.00834](https://doi.org/10.48550/ARXIV.2310.00834). arXiv: [2310.00834](https://arxiv.org/abs/2310.00834). [Online]. Available: <https://doi.org/10.48550/arXiv.2310.00834>.

- [169] M. Baum, J. Dibbelt, A. Gemsa, D. Wagner, and T. Zündorf, “Shortest feasible paths with charging stops for battery electric vehicles,” in *Proceedings of the 23rd SIGSPATIAL international conference on advances in geographic information systems*, 2015, pp. 1–10.
- [170] S. Pourazarm and C. G. Cassandras, “Optimal routing of energy-aware vehicles in transportation networks with inhomogeneous charging nodes,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 8, pp. 2515–2527, 2017.
- [171] I. Kucukoglu, R. Dewil, and D. Cattrysse, “The electric vehicle routing problem and its variations: A literature review,” *Computers & Industrial Engineering*, vol. 161, p. 107 650, 2021.
- [172] G. Y. Handler and I. Zang, “A dual algorithm for the constrained shortest path problem,” *Networks*, vol. 10, no. 4, pp. 293–309, 1980.
- [173] S. Storandt, “Quick and energy-efficient routes: Computing constrained shortest paths for electric vehicles,” in *Proceedings of the 5th ACM SIGSPATIAL international workshop on computational transportation science*, 2012, pp. 20–25.
- [174] E. Altman, *Constrained Markov decision processes*. Routledge, 2021.
- [175] J. D. Isom, S. P. Meyn, and R. D. Braatz, “Piecewise linear dynamic programming for constrained pomdps,” in *AAAI*, vol. 1, 2008, pp. 291–296.
- [176] J. Lee, G.-H. Kim, P. Poupart, and K.-E. Kim, “Monte-Carlo tree search for constrained POMDPs,” *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 31, 2018.
- [177] F. Blahoudek, T. Brázdil, P. Novotný, M. Ornik, P. Thangeda, and U. Topcu, “Qualitative controller synthesis for consumption Markov decision processes,” in *International Conference on Computer Aided Verification*, Springer, 2020, pp. 421–447.
- [178] N. D. Wallace, H. Kong, A. J. Hill, and S. Sukkarieh, “The orienteering problem with replenishment,” in *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)*, IEEE, 2020, pp. 973–978.
- [179] A. Y. Lam, Y.-W. Leung, and X. Chu, “Electric vehicle charging station placement: Formulation, complexity, and solutions,” *IEEE Transactions on Smart Grid*, vol. 5, no. 6, pp. 2846–2856, 2014.
- [180] T. Kundu and I. Saha, “Charging station placement for indoor robotic applications,” in *2018 IEEE international conference on robotics and automation (ICRA)*, IEEE, 2018, pp. 3029–3036.

- [181] Q. Cui, Y. Weng, and C.-W. Tan, “Electric vehicle charging station placement method for urban areas,” *IEEE Transactions on Smart Grid*, vol. 10, no. 6, pp. 6552–6565, 2019.
- [182] D. Aksaray, K. Leahy, and C. Belta, “Distributed multi-agent persistent surveillance under temporal logic constraints,” *IFAC-PapersOnLine*, vol. 48, no. 22, pp. 174–179, 2015.
- [183] N. Nigam, S. Bieniawski, I. Kroo, and J. Vian, “Control of multiple uavs for persistent surveillance: Algorithm and flight test results,” *IEEE Transactions on Control Systems Technology*, vol. 20, no. 5, pp. 1236–1251, 2011.
- [184] J. Scherer and B. Rinner, “Short and full horizon motion planning for persistent multi-uav surveillance with energy and communication constraints,” in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2017, pp. 230–235.
- [185] S. Seyedi, Y. Yazıcıoğlu, and D. Aksaray, “Persistent surveillance with energy-constrained UAVs and mobile charging stations,” *IFAC-PapersOnLine*, vol. 52, no. 20, pp. 193–198, 2019.
- [186] N. Mathew, S. L. Smith, and S. L. Waslander, “Multirobot rendezvous planning for recharging in persistent tasks,” *IEEE Transactions on Robotics*, vol. 31, no. 1, pp. 128–142, 2015.
- [187] X. Lin, Y. Yazıcıoğlu, and D. Aksaray, “Robust planning for persistent surveillance with energy-constrained UAVs and mobile charging stations,” *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4157–4164, 2022.
- [188] S. Eiffert, N. D. Wallace, H. Kong, N. Pirmarzashti, and S. Sukkarieh, “A hierarchical framework for long-term and robust deployment of field ground robots in large-scale farming,” in *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)*, IEEE, 2020, pp. 948–954.
- [189] M. Tomy, B. Lacerda, N. Hawes, and J. L. Wyatt, “Battery charge scheduling in long-life autonomous mobile robots,” in *2019 European Conference on Mobile Robots (ECMR)*, IEEE, 2019, pp. 1–6.
- [190] B. J. Ayton and B. C. Williams, “Vulcan: A Monte Carlo algorithm for large chance constrained MDPs with risk bounding functions,” *arXiv preprint arXiv:1809.01220*, 2018.
- [191] R. J. Moss, A. Jamgochian, J. Fischer, A. Corso, and M. J. Kochenderfer, “ConstrainedZero: Chance-constrained POMDP planning using learned probabilistic failure surrogates and adaptive safety constraints,” in *International Joint Conference on Artificial Intelligence (IJCAI)*, 2024.

- [192] M. Ono, M. Pavone, Y. Kuwata, and J. Balaram, “Chance-constrained dynamic programming with application to risk-aware robotic space exploration,” *Autonomous Robots*, vol. 39, pp. 555–571, 2015.
- [193] D. Mitchell, N. Chakraborty, K. Sycara, and N. Michael, “Multi-robot persistent coverage with stochastic task costs,” in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2015, pp. 3401–3406.
- [194] C. G. Grlj, N. Krznar, and M. Pranjic, “A decade of UAV docking stations: A brief overview of mobile and fixed landing platforms,” *Drones*, vol. 6, no. 1, p. 17, 2022.
- [195] Y. Liu, “A laser intensity based autonomous docking approach for mobile robot recharging in unstructured environments,” *IEEE Access*, vol. 10, pp. 71 165–71 176, 2022.
- [196] A. M. Yazdani, K. Sammut, O. Yakimenko, and A. Lammas, “A survey of underwater docking guidance systems,” *Robotics and Autonomous systems*, vol. 124, p. 103 382, 2020.
- [197] S. McCammon, N. Aoki, T. A. Mooney, and Y. Girdhar, “Adaptive online sampling of periodic processes with application to coral reef acoustic abundance monitoring,” in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2022, pp. 11 671–11 678.
- [198] J. Jackson, C. S. Lawson, C. Adelmant, E. Huhtala, P. Fernandes, R. Hodgson, H. King, L. Williamson, K. Maseyk, N. Hawes, *et al.*, “Short-range multispectral imaging is an inexpensive, fast, and accurate approach to estimate biodiversity in a temperate calcareous grassland,” *Ecology and Evolution*, vol. 12, no. 12, e9623, 2022.
- [199] N. D. Goodman, T. D. Ullman, and J. B. Tenenbaum, “Learning a theory of causality,” *Psychological review*, vol. 118, no. 1, p. 110, 2011.
- [200] M. K. Ho, D. Abel, T. L. Griffiths, and M. L. Littman, “The value of abstraction,” *Current opinion in behavioral sciences*, vol. 29, pp. 111–116, 2019.
- [201] N. Srinivas, A. Krause, S. M. Kakade, and M. W. Seeger, “Information-theoretic regret bounds for Gaussian process optimization in the bandit setting,” *IEEE transactions on information theory*, vol. 58, no. 5, pp. 3250–3265, 2012.
- [202] E. Staaterman, M. B. Ogburn, A. H. Altieri, S. J. Brandl, R. Whippo, J. Seemann, M. Goodison, and J. E. Duffy, “Bioacoustic measurements complement visual biodiversity surveys: Preliminary evidence from four shallow marine habitats,” *Marine Ecology Progress Series*, vol. 575, pp. 207–215, 2017.

- [203] T.-H. Lin, T. Akamatsu, F. Sinniger, and S. Harii, “Exploring coral reef biodiversity via underwater soundscapes,” *Biological Conservation*, vol. 253, p. 108 901, 2021.
- [204] M. B. Kaplan, T. A. Mooney, J. Partan, and A. R. Solow, “Coral reef species assemblages are associated with ambient soundscapes,” *Marine Ecology Progress Series*, vol. 533, pp. 93–107, 2015.
- [205] R. Stöckli, E. Vermote, N. Saleous, R. Simmon, and D. Herring, “The blue marble next generation-a true color earth dataset including seasonal dynamics from modis,” *Published by the NASA Earth Observatory*, 2005.
- [206] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*. MIT Press, 2022.
- [207] *SolarXOne: Autonomous UAV Technology*, <https://xsun.fr/autonomous-drone/>, Accessed: 2024-04-09, 2023.
- [208] T. Wang, “Poisson-disk sampling: Theory and applications,” in *Encyclopedia of Computer Graphics and Games*, Springer, 2024, pp. 1424–1431.
- [209] N. Owen-Smith, G. Hopcraft, T. Morrison, S. Chamaillé-Jammes, R. Hetem, E. Bennitt, and F. Van Langevelde, “Movement ecology of large herbivores in African savannas: Current knowledge and gaps,” *Mammal Review*, vol. 50, no. 3, pp. 252–266, 2020.
- [210] I. Stenius, J. Folkesson, S. Bhat, C. I. Sprague, L. Ling, Ö. Özkahraman, N. Bore, Z. Cong, J. Severholt, C. Ljung, *et al.*, “A system for autonomous seaweed farm inspection with an underwater robot,” *Sensors*, vol. 22, no. 13, p. 5064, 2022.
- [211] V. H. Johnson, A. A. Pesaran, and T. Sack, “Temperature-dependent battery models for high-power lithium-ion batteries,” National Renewable Energy Lab.(NREL), Golden, CO (United States), Tech. Rep., 2001.
- [212] C. Eisentraut, H. Hermanns, and L. Zhang, “On probabilistic automata in continuous time,” in *2010 25th Annual IEEE Symposium on Logic in Computer Science*, IEEE, 2010, pp. 342–351.
- [213] M. Mansouri, B. Lacerda, N. Hawes, and F. Pecora, “Multi-robot planning under uncertain travel times and safety constraints,” in *International Joint Conferences on Artificial Intelligence*, International Joint Conferences on Artificial Intelligence Organization, 2019.
- [214] C. Azevedo, B. Lacerda, N. Hawes, and P. Lima, “Long-run multi-robot planning under uncertain action durations for persistent tasks,” in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2020, pp. 4323–4328.

- [215] V. A. Ziparo, L. Iocchi, M. Leonetti, and D. Nardi, “A probabilistic action duration model for plan selection and monitoring,” in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2010, pp. 4716–4721.
- [216] G. Cielniak, G. Das, P. From, and M. Hanheide, “Discrete event simulations for scalability analysis of robotic in-field logistics in agriculture – a case study,” 2018.
- [217] A. Thummler, P. Buchholz, and M. Telek, “A novel approach for phase-type fitting with the em algorithm,” *IEEE Transactions on dependable and secure computing*, vol. 3, no. 3, pp. 245–258, 2006.
- [218] H. Frank, “Shortest paths in probabilistic graphs,” *operations research*, vol. 17, no. 4, pp. 583–599, 1969.
- [219] F. Fu and H. Dong, “Targeted optimal-path problem for electric vehicles with connected charging stations,” *Plos one*, vol. 14, no. 8, e0220361, 2019.
- [220] H. Hõimoja, A. Rufer, G. Dziechciaruk, and A. Vezzini, “An ultrafast ev charging station demonstrator,” in *International Symposium on Power Electronics, Electrical Drives, Automation and Motion*, IEEE, 2012, pp. 1390–1395.
- [221] H. Jin, S. Lee, S. H. Nengroo, and D. Har, “Development of charging/discharging scheduling algorithm for economical and energy-efficient operation of multi-ev charging station,” *Applied Sciences*, vol. 12, no. 9, p. 4786, 2022.
- [222] S. Summers and J. Lygeros, “Verification of discrete time stochastic hybrid systems: A stochastic reach-avoid decision problem,” *Automatica*, vol. 46, no. 12, pp. 1951–1961, 2010.
- [223] J. Kemeny, J. Snell, and A. Knapp, *Denumerable Markov chains*, 2nd. Springer-Verlag, 1976.
- [224] F. Teichteil-Königsbuch, “Stochastic safest and shortest path problems,” in *AAAI*, 2012.
- [225] B. Lacerda, F. Faruq, D. Parker, and N. Hawes, “Probabilistic planning with formal performance guarantees for mobile service robots,” *The International Journal of Robotics Research*, vol. 38, no. 9, pp. 1098–1123, 2019.
- [226] A. Elfes, “Using occupancy grids for mobile robot perception and navigation,” *Computer*, vol. 22, no. 6, pp. 46–57, 1989.
- [227] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, “Octomap: An efficient probabilistic 3d mapping framework based on octrees,” *Autonomous robots*, vol. 34, pp. 189–206, 2013.

- [228] C. Stachniss, G. Grisetti, and W. Burgard, “Information gain-based exploration using rao-blackwellized particle filters,” in *Robotics: Science and systems*, vol. 2, 2005, pp. 65–72.
- [229] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, “Curiosity-driven exploration by self-supervised prediction,” in *International conference on machine learning*, PMLR, 2017, pp. 2778–2787.
- [230] M. Kwiatkowska, G. Norman, and D. Parker, “Prism 4.0: Verification of probabilistic real-time systems,” in *Computer Aided Verification: 23rd International Conference, CAV 2011, Snowbird, UT, USA, July 14-20, 2011. Proceedings 23*, Springer, 2011, pp. 585–591.
- [231] GPy, *GPy: A Gaussian process framework in Python*, <http://github.com/SheffieldML/GPy>, since 2012.
- [232] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y. Ng, *et al.*, “ROS: An open-source robot operating system,” in *ICRA workshop on open source software*, Kobe, Japan, vol. 3, 2009, p. 5.
- [233] E. Snelson, Z. Ghahramani, and C. Rasmussen, “Warped Gaussian processes,” *Advances in Neural Information Processing Systems (NeurIPS) (NeurIPS)*, vol. 16, 2003.
- [234] N. Koenig and A. Howard, “Design and use paradigms for Gazebo, an open-source multi-robot simulator,” in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, IEEE, vol. 3, 2004, pp. 2149–2154.
- [235] T. Wright, A. West, M. Licata, N. Hawes, and B. Lennox, “Simulating ionising radiation in Gazebo for robotic nuclear inspection challenges,” *Robotics*, vol. 10, no. 3, p. 86, 2021.
- [236] D. Wisth, M. Camurri, and M. Fallon, “VILENS: Visual, inertial, lidar, and leg odometry for all-terrain legged robots,” *IEEE Transactions on Robotics*, 2022.