

Informative Path Planning Toward Autonomous Real-World Applications

Brady Moon

CMU-RI-TR-25-13

May 2025

School of Computer Science
The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania

Thesis Committee:

Sebastian Scherer, Chair

David Wettergreen

Maxim Likhachev

Geoff Hollinger (Oregon State University)

*Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy*

Copyright © 2025 Brady Moon

To my wife and our children—Milo, April, and Cliff

Abstract

Gathering information from the physical world is critical for applications such as scientific research, environmental monitoring, search and rescue, defense, and disaster response. Autonomous robots provide significant advantages for information gathering, particularly in situations where human access is constrained, hazardous, or impractical. By leveraging intelligent algorithms, these robots can efficiently collect data, enhancing decision-making and accelerating insights. Informative Path Planning (IPP) plays a key role in maximizing the effectiveness of robotic information gathering by generating paths that optimize data collection while respecting operational constraints.

This thesis advances autonomous information gathering by addressing three key challenges: (1) solving IPP problems in high-dimensional spaces with complex sensor constraints, (2) incorporating real-world disturbances and risk into the planning framework, and (3) improving and leveraging world belief models. First, we introduce IA-TIGRIS, an adaptive, incremental sampling-based planner that efficiently computes long-horizon information-gathering paths while respecting vehicle motion constraints and non-trivial sensor footprints. We validate the planner through extensive simulations and real-world field tests on multiple unmanned aerial vehicle (UAV) platforms. Second, we develop a real-time wind field estimation method using onboard UAV measurements, a time-optimal path planner for UAVs in wind, and a deep learning-based energy risk assessment framework to quantify flight risk under uncertain environmental conditions. Third, we propose a new belief representation for search and tracking, along with planning approaches that incorporate human-inspired heuristics and predictive belief models.

The proposed methods are validated through a combination of simulation studies and extensive field deployments, contributing to the development of robust, real-world-ready autonomous systems. We demonstrate their effectiveness across diverse planning scenarios and on multiple UAV platforms, including both fixed-wing and multirotor systems. While the primary focus is on robotic information gathering, the underlying algorithms generalize to broader applications such as robotic exploration, active perception, target tracking, and multi-agent coordination.

Acknowledgments

I would first like to thank my advisor, Sebastian Scherer. His passion for robotic systems and his drive to get out into the field and test them have been a constant source of inspiration. I have always admired his well-rounded technical expertise and his ability to keep the focus on the story and impact of our work. His guidance has shaped not only the direction of my research, but also how I think about meaningful contributions in robotics.

I am grateful to my thesis committee for their feedback throughout this process. David, thank you for your guidance and for pushing me to pursue work that is both significant and novel. Max, I appreciate your emphasis on rigorous testing, which strengthened the technical foundation of my work. Geoff, your deep understanding of my work and your background in data gathering made your feedback especially valuable. Each of your insights has meaningfully shaped this dissertation.

I am deeply grateful to everyone in the AirLab—there was truly no better place to spend these formative years. Special thanks to Jay, whose friendship, mentorship, and collaborative spirit have been invaluable since the beginning. You generously shared countless opportunities with me at CMU, and I aspire to emulate your admirable practice of sharing recognition when many would claim credit for themselves. To my office mates in NSH 2204a—Huai, Jay, Anish, Joao, Andrew J., Satrajit, Jacky, Austin, and Wilkin—thank you for the stimulating conversations and camaraderie in our compact space. Satrajit, I particularly value our synergistic collaboration and friendship during your time at CMU. The AirLab has been home to many exceptional colleagues from whom I’ve learned immensely: Nayana, Seungchan, Cherie, Sagar, Arnav, Sourish, Adi, Andrew S., Chase, Parv, Junbin, Nikhil, Ian, Muqing, Junyi, and Micah, among many others. I’ve also had the privilege of mentoring talented undergraduate researchers like Narek, Wally, Aditya, Rohit, Ryan, Leo, Kaleb, and Rachel. My gratitude extends to the transformative RISS program and the opportunities to be involved. A special acknowledgment to John K. for his wealth of knowledge, problem-solving abilities, and willingness to help.

I would like to thank Rachel Burcin for her significant impact on my time at CMU. It’s difficult to put into words the countless ways my experience was shaped by her mentorship, advice, advocacy, and tireless efforts. Her influence has been truly pivotal,

and I hope to carry forward and emulate her remarkable ability to frame all challenges as opportunities. Thank you to Hoa for such a productive collaboration and for teaching me about areas of work that were new to me. I'm grateful to Graeme for your thoughtful insights and research instincts that helped guide my work. Costa, working with you on one of my first projects at CMU provided invaluable learning experiences from your multi-faceted career. I extend my appreciation to Matt Johnson-Roberson for your dedication to the RI community and your mentorship throughout my career journey. And to Zac, thank you for your feedback on the early stages of my research.

My research journey began with my undergraduate advisor, Cammy Peterson, at BYU. Thank you for patiently guiding me through the research fundamentals and setting me on this rewarding path. I am indebted to everyone at BYU who introduced me to this career I never knew existed before arriving on campus, and to the IMMERSE program for creating the opportunity to get started in research.

The Robotics Institute has been enriched by numerous colleagues and friends who have enhanced my experience through study groups, volunteer work, and shared spaces. I appreciate my cousin Benj for his ability to bring people together and selflessness. My gratitude extends to Ananya, Mrinal, Abbey, Maggie, Leo, Kate, Kevin, Josh, Victoria, Raunaq, Thiago, Rishi, Rebecca, Brian, Taylor, Easton, Jacob, Dan, Arun, Mitch, Bart, Mateo, Sam S., Mono, Rayna, Jay K., Akhil, Mohamad, Peter, Yorai, Vivian, Sam T., Shibo, Yuheng, Yafei, Seungjae, and many others whose names may not appear here but whose impact has been no less significant.

Working with the RI Climate Committee under the direction of Henny, Oliver, Sam, Tim, Matt, and others was a privilege, and witnessing the positive impact of our collective efforts to improve the RI community was deeply satisfying. I extend my appreciation to the exceptional RI staff members, Jess, B.J., Jean, Suzanne, Ashlyn, Brian, Janice, Nora, Debra, and Victor, whose dedication and support shape the culture of the RI. Finally, I thank the Eberly Center and the future faculty program, particularly Phoebe, for providing valuable professional development opportunities.

Most of all, I am deeply grateful to my family. To my parents, who instilled in me a deep love for learning, showed me the value of hard work in pursuit of meaningful goals, and taught me how to put the most important things in life first. To my wife, who has been my unwavering foundation through every challenge and triumph, thank you for sharing nearly a decade of cherished adventures with me. You are truly my everything. And to my children, you are the purest joy in my life and my greatest motivation. Every day, you remind me of what truly matters.

This work was funded by the National Science Foundation Graduate Research Fellowship under Grant No. DGE1745016, Office of Naval Research under Grant N00014-21-1-2110 and Contract No. N6833522C0179, and Department of Energy Grant under Grant No. DE-EE0008463. Their support is gratefully acknowledged.

Table of Contents

	Page
1 Introduction	1
1.1 Thesis Statement	2
1.2 Challenges	3
1.3 Thesis Outline	4
1.4 Summary of Contributions	5
2 Background	7
2.1 What is path planning?	7
2.2 Problem Definition	9
2.3 Approximations & Simplifications	10
2.4 Informative Path Planning Approaches	11
I Informative Path Planning	15
3 Informed, Sampling-based Algorithm for Informative Path Planning	17
3.1 Introduction	17
3.2 Background	20
3.3 Problem Definition	20
3.4 Methodology	21
3.5 Implementation	23
3.6 Testing Results	28
3.7 Conclusion	31
4 Incremental and Adaptive Sampling-Based Planner for Online Informative Path Planning	33
4.1 Introduction	34
4.2 Background	35
4.3 Problem Formulation	36
4.4 Proposed Approach	37
4.5 Simulation Evaluation & Results	47
4.6 Field Deployments	63

4.7	Conclusion	67
II	Disturbance and Risk Aware Path Planning	69
5	In-Situ Urban Wind Field Estimation	73
5.1	Introduction	73
5.2	Methodology	76
5.3	Field Validation	80
5.4	Conclusions and Future work	84
6	Time-Optimal Path Planning in a Constant Wind	85
6.1	Introduction	85
6.2	Background	87
6.3	Problem Definition	89
6.4	Methodology	89
6.5	Testing Results	97
6.6	Conclusion	99
7	CVaR-based Flight Energy Risk Assessment using a Deep Energy Model	101
7.1	Introduction	102
7.2	Problem Definition	104
7.3	Energy Modeling	105
7.4	Risk Modeling	109
7.5	Case Studies	111
7.6	Conclusions	114
III	Improving and Leveraging World Belief Models	115
8	Indoor Exploration using Predicted Maps	119
8.1	Introduction & Motivation	119
8.2	Background	120
8.3	Problem Definition	122
8.4	Previous Contributions in Our Research	122
8.5	Pathwise Information Gain	126
8.6	Approach	129
8.7	Experiments	135
8.8	Conclusion	141
9	Human-Inspired Indoor Exploration using Reinforcement Learning	143
9.1	Introduction	143
9.2	Background	145

9.3	Problem Statement	146
9.4	User Study on Indoor Exploration	147
9.5	Approach	151
9.6	Results	156
9.7	Conclusion	158
10	Unified-Objective Belief Model for Search & Track	159
10.1	Introduction	159
10.2	Related Works	161
10.3	Problem Statement	163
10.4	Background	164
10.5	Approach	168
10.6	Results	172
10.7	Conclusion and Future Work	176
11	Conclusion	179
11.1	Summary and Contributions	180
11.2	Future Directions	182
	Appendices	185
A	IPP Target Tracking Adaptation	187
B	Dubins Set Classification Correction	191
	Bibliography	195

List of Figures

2.1	Comparison of goal-oriented path planning objective functions	8
2.2	Comparison of non-goal-oriented path planning objective functions	9
3.1	Example informative path planning scenario	17
3.2	IPP for a missing hiker	18
3.3	Visualization of how the informed sample state is created from a Bayesian grid belief space.	23
3.4	Visualization of uniform vs. informed sampling	25
3.5	An example sensor model of (3.2) with blue being the true positive rate and red being the false positive rate.	26
3.6	A visualization of the edge reward approximation and variable definitions given a fixed forward-facing camera.	27
3.7	TIGRIS Monte Carlo simulation results plots	28
3.8	Example TIGRIS results in the testing environment	30
4.1	IA-TIGRIS deployed on a hexarotor UAV to map the location of cars . .	34
4.2	Overview of the IA-TIGRIS planning framework.	37
4.3	Visualization of the online replanning procedure	40
4.4	An illustration of the incremental planning and refinement process in IA-TIGRIS	41
4.5	An example of how the grid cells rewards are estimated given a planned trajectory.	46
4.6	Example results for IA-TIGRIS after a replanning step in our simulation environment.	48
4.7	IA-TIGRIS continuously replanning and refining the global path over the duration of the simulation.	49
4.8	Tree size vs computation time for 100 trials.	51
4.9	Examples showing how modifying the priority and time-based rewards effects behavior.	52
4.10	An isosurface plot showing the results of the global parameter search. .	53
4.11	A comparison of IA-TIGRIS when planning over a long horizon versus a short horizon.	54
4.12	Paths from various problem configurations.	55

4.13	A trajectory derived from an ergodic coverage objective.	57
4.14	The Monte Carlo simulation results for the planners.	60
4.15	A comparison of the methods based on the number of sampled prior clusters and the standard deviation of sampled prior clusters.	61
4.16	Google Earth screenshots illustrating the mission planning process and execution.	63
4.17	Hexarotor system (DJI M600 Pro) with onboard compute and camera. . .	64
4.18	Functional diagram of the DJI M600 Pro autonomy software.	64
4.19	The results for our hexarotor field deployment.	65
4.20	Fixed-wing platform used for autonomous flights with an onboard camera pitched at 10 degrees [189]	66
4.21	An example path generated for the fixed-wing platform conducting a large-area search for an object of interest.	67
5.1	A CFD wind field using the estimated inlet boundary condition at the test location	74
5.2	Wind field estimation pipeline overview	75
5.3	Data points (locations and inlet angles) for GPR training.	78
5.4	Wind field validation test site	80
5.5	The test UAV with the wind sensor.	81
5.6	The motion and bias corrected local wind angle and magnitude readings for the test run.	82
5.7	UAV flight path and wind field at test site	83
5.8	The estimated wind inlet angle and magnitude	84
6.1	An overview of our trochoidal method with an example problem.	86
6.2	An example optimal trochoidal RSL paths and its equivalent Dubins path in the wind frame.	88
6.3	An example problem setup with variable definition shown.	92
6.4	Randomly sampled candidate BSB trajectories for randomly selected initial and final states in a uniform wind-vector field.	96
6.5	The distribution of solutions across varying problem parameters	97
7.1	Path risk modeling pipeline	102
7.2	Dataset trajectory and sample data	106
7.3	Overview of the deep energy model.	107
7.4	Test Flights with the best and worst MAPE for b-TCN.	109
7.5	Illustrative example to show how CVaR is a more relevant risk metric .	110
7.6	Monte Carlo simulations for Case 1.	112
7.7	Comparison of CVaR risk contours for Case 2 under different UAV conditions.	114

8.1	MapEx pipeline overview.	123
8.2	Visualization of the PIPE information gain calculation.	126
8.3	Overview of PIPE planner pipeline.	128
8.4	Example map predictor output given an observed map.	130
8.5	Illustration of different raycast methods.	131
8.6	Computation time comparison of our optimized approach.	132
8.7	A visualization of the path visibility mask computation.	133
8.8	The seven indoor maps used to evaluate PIPE	136
8.9	Quantitative comparison of methods for each map size.	138
8.10	Qualitative comparison between sensor coverages estimated by PIPE and PW-NBV-2D.	140
8.11	Qualitative comparison of methods at time step $T = 2000$ on Map 6. . .	141
9.1	An overview of the development and research process for MapExRL. . .	144
9.2	User interface for the study	147
9.3	The set of test maps used in this work.	148
9.4	Three user runs and MapEx on Map 4	150
9.5	The MapExRL pipeline	151
9.6	Comparison of the six evaluated methods on Map 2 from the same start position.	157
10.1	An example scenario of a UAV searching for and tracking objects in a space.	160
10.2	An overview of the <i>UniSaT</i> pipeline.	163
10.3	A randomly sampled environment from the five characteristic scenarios and a random environment setup.	173
10.4	An example run of <i>UniSaT</i> at different time steps in our simulation environment.	175
10.5	The mean OSPA ² results of <i>UniSaT</i> and the baseline over 100 MC runs .	176
A.1	IPP for target tracking	188
B.1	Visualization of the Dubins set classification edge case	192
B.2	The Dubins set classification edge case path	192

List of Tables

3.1	Information reward ($M \pm SD$) with decreasing sparsity in the number of target belief centroids (n) in the environment.	29
4.1	Monte Carlo testing results given different budgets.	62
6.1	The corrected Dubins decision table	94
6.2	The distribution of time-optimal path types based on Monte-Carlo simulations for Bang-Straight-Bang trajectories.	98
6.3	The distribution of Dubins decision table blocks from the Monte-Carlo simulations.	99
7.1	Energy Model Results	107
8.1	Area Under the Curve (AUC) of IoU for each method.	139
8.2	Comparison of required time steps to reach 90% and 95% IoU with ground truth maps.	140
9.1	Results of the user study compared to MapEx.	149
9.2	Average and 95% confidence interval for reward and IoU on the test maps.	156
10.1	Results from 100 MC tests for each characteristic scenario.	174

Chapter 1

Introduction

Our world is driven by data—it is how we find answers to questions, answer scientific hypotheses, make policy decisions, create models, and predict future events. The importance of data has become more salient in the public’s mind as increasingly impressive learning models are continually being deployed in ways that are more public and leveraged by the everyday person. While we may see the effects of the data, we don’t think as often about how the data is collected.

Not all data can be gathered with equal amounts of effort. Digital data can be easily scraped and accumulated through simple computer scripts, but gathering data in the physical world requires significantly more time, resources, and risk. Some examples of data gathering in physical spaces include forestry management, agriculture monitoring, biological sciences, surveying, mapping, bridge inspection, wildlife monitoring, exploration, natural disaster response, national defense, reconnaissance, and wildfire tracking. In all of these examples, it is clear to see how having good data is key to understanding the problem and making informed decisions.

Having a human gather this data is often dangerous, slow, error prone, and tedious. Using robots in data gathering is a clear answer to many of these problems. Robots distance the human data gatherers away from the sources of danger, such as fire, radiation, precarious heights, animals, and chemicals. Robots can travel to areas a human cannot easily travel to, such as a faraway moons or the bottoms of ocean trenches. Robots can also be much more cost effective to gather with than humans, many times this is due to the fact that minimizing risk for humans is very costly.

The benefits of using robots in data gathering is greatly enhanced when robots can be autonomous rather than human teleoperated. Autonomous robots can free the operators to focus on other tasks, such as high level decision making, that require human involvement. This also gets rid of the bottleneck of having at least one operator

per robot, so the number of deployed robots can increase and lead to multi-robot coordination or even swarm strategies. Additionally, the robots can now venture into areas where there is limited or no communication to the ground station. The autonomy can also produce better actions or paths than a human could by leveraging algorithms and prior data.

Efficiently and optimally gathering data in the physical world with autonomous robots necessitates having path planning algorithms that take into account the amount and quality of data gathered over a given path. A way of formulating this problem is as an Informative Path Planning (IPP) problem which is to find the information maximizing path that does not violate a budget constraint. This work aims to reduce the amount of approximations and simplifications traditionally used when formulating the IPP problem in order to create paths that gather more information.

1.1 Thesis Statement

Autonomous information gathering can maximize information gain by integrating

-  **long-horizon** reasoning,
-  **environment dynamics**, and
-  **enhanced world belief models.**

Thesis Statement

For the purpose of this thesis:

- *Information gain* is considered within the context of informative path planning, where the objective is to maximize the information gathered while respecting budget constraints. The specific information objective depends on the problem formulation.
- *Long-horizon reasoning* refers to planning approaches that extend beyond immediate or myopic decision-making, considering future states and the accumulation of information over an extended trajectory.
- *Environment* includes the world belief model and disturbances encountered in field deployment.

- *World belief model* is the information the robot has about the world around it, whether past, present, or predicted. This model may come from prior data and integrate information from the perception pipeline. It provides data for the planner to make informed decisions.

1.2 Challenges

We break down the challenges into three areas that this thesis aims to address.

Challenge 1: *Solving IPP problems in high-dimensional spaces with non-trivial sensor constraints.*

Many approaches for IPP problems simplify finding a global solution by discretizing the configuration space and reducing the number of states to be as small as possible. To further simplify the problem, they typically also approximate the sensor model to be a circle of a fixed radius around the robot state. However, these simplifications can create a rather large performance gap when translating the planned path into the real world. The motion constraints of some robots can make it challenging to follow a 2D path on a grid. Additionally, many sensors of interest to data gather, such as electro-optical or infrared cameras do not gather information in a circle underneath the drone. Most lenses do not produce a circular sensor footprint, and cameras can be mounted or pointed in many configurations other than downward-facing. But adding this complexity also increased the complexity of the planning algorithm, and many methods can no longer be directly used.

Challenge 2: *Incorporating real-world disturbances and risk into the path-planning framework.*

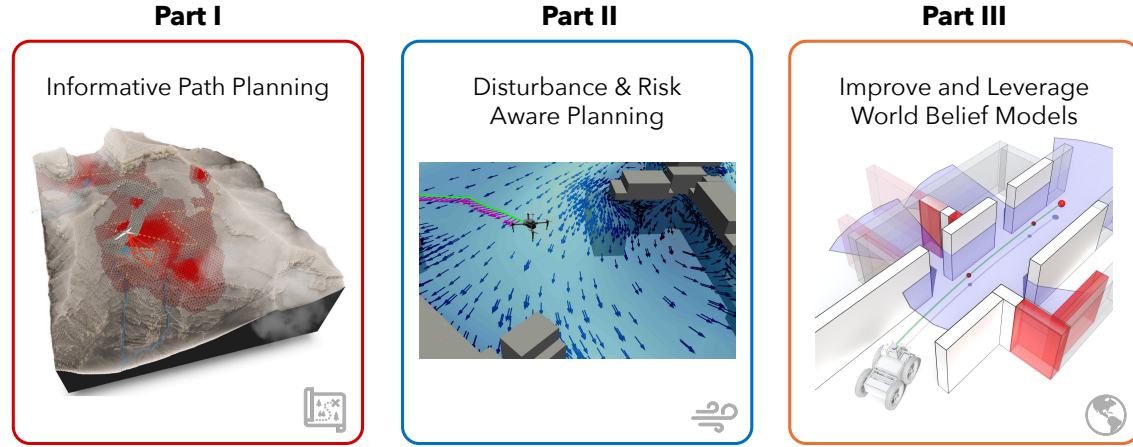
Real-world disturbances are often abstracted away when path planning for robots. However, fully ignoring the impact of disturbances can lead to infeasible, suboptimal, and risky paths. In this work, We specifically focus on the effect of wind fields on UAVs which are often just treated as a disturbance to be handled by the controller. We seek to answer (1) how can we estimate a wind field using in-situ measurements from a drone, specifically in complex urban environments, (2) how can we plan time-optimal paths in a wind field rather than wasting battery power fighting the wind with the controller or creating infeasible trajectories and (3) how can we estimate the risk of a given path in a wind field?

Challenge 3: *Improve and leverage world belief models.*

The objective function in IPP is tightly coupled to the chosen way to model the world belief and how it is modified with each observation. An example of a method used by many is using a Gaussian process model for the world and incorporating measurements using a Gaussian process regression. But this, along with other methods, has its pros

and cons and sometimes does not lend itself well to an objective function that matches well with the intended behavior of the path planner. There are clear opportunities to create world models that clearly map to objective functions that create intended behavior. Additionally, there are opportunities to leverage prior information maps to predict maps for better exploration in the information map.

1.3 Thesis Outline



This dissertation develops approaches to autonomous robotic data gathering that address the previously explained challenges in Section 1.2. Chapter 2 presents a background of the IPP problem formulation as well as the typical simplifications and approximations. The core components of this thesis are divided into three parts:

1. Part I: This thesis begins in Chapter 3 with our core informative path planning algorithm and a deeper introduction to informative path planning. We then present our planner, IA-TIGRIS, in Chapter 4 which takes the work in Chapter 3 and adapts it to be an informed sampling-based IPP planner that is both adaptive and iterative. These chapters work to address Challenge 1 as well as includes further modifications we have made to extend this work to target tracking and allowing for multiple sensor models. We demonstrate our approach on two different UAV platforms, with over 24 hours of deployment and 100 flights.
2. Part II: One of the takeaways from our work in Part I is the impact of real-world disturbances when deploying these systems in the real world. This part examines Challenge 2, focused on incorporating wind-fields into the path planning for a drone.

In Chapter 5, we present our work on estimating urban wind fields using the in-situ measurements onboard the drone. Using an estimated wind field, in

Chapter 6 we then examine how we can plan time-optimal paths within the wind field that reduce energy usage and ensure feasible trajectories. However, wind intrudes a high amount of uncertainty in regard to path deviations and total energy usage for path execution. In Chapter 7 we show how we can predict the energy usage and risk of a UAV flight in a wind field, leveraging our own drone energy usage dataset and Monte Carlo simulations.

3. Part III: Finally, we delve into Challenge 3 with two main questions: 1) how can we leverage prior information to improve data gathering, and 2) how to we reconcile two types of belief maps that have conditional relationships?

In Chapter 8, we look at the first question and see how we can use a global map predictor to expedite indoor map exploration. With our planner called *PIPE*, we show the importance of having a good approximation of information gain along the entire trajectory and how straight-forward approaches can sometimes lead to worse results than simple heuristics. We train and test our method on an indoor map dataset and show our method has state-of-the-art results.

From the previous work, it raises the question of how well the robot explores compared to human and what context does a human user to improve exploration. In Chapter 9, we show our insights from conducting a human user study where we present them with the same situation as presented in the previous chapter. Guided by insights from our user study, we introduce *MapExRL*, which improves robot exploration efficiency in structured indoor environments by enabling longer-horizon planning through reinforcement learning (RL) and global map predictions.

We then examine the second question in Chapter 10 and introduce *UniSaT* (Unified Search and Track), a novel unified-objective formulation for the search and track problem based on Random Finite Sets (RFS). Our approach models unknown and known objects using a combined generalized labeled multi-Bernoulli (GLMB) filter. The planner maximizes the mutual information of this unified belief model, creating balanced search and tracking behaviors.

At the conclusion, Chapter 11 provides a summary of this work and its contributions, as well as outlining potential future directions and finishing with concluding remarks.

1.4 Summary of Contributions

This thesis contributes to the field of autonomous information gathering by addressing key challenges in informative path planning (IPP), real-world disturbance adaptation, and world belief modeling. The primary contributions are as follows:

1. *Adaptive, Incremental, Online IPP Planner*: Development of **TIGRIS**, an informed sampling-based planner, and **IA-TIGRIS**, an adaptive and incremental planner that enables online replanning in dynamic environments. These planners efficiently handle high-dimensional spaces with motion constraints and non-uniform prior information distributions through informed sampling and incorporating efficient path-integrated reward estimation.
2. *Wind-Field Estimation, Trajectory Planning, and Risk Evaluation*: Creating effective methods for estimating urban wind fields using only UAV in-situ measurements, computing time-optimal UAV trajectories in wind, and producing coherent risk metrics for trajectories using our novel deep learning-based energy model.
3. *Information Gain Estimation*: Development of approaches for estimating information gain, including by leveraging map prediction. **UniSaT** enables search-and-track problems to be formulated with a single objective function, avoiding the need for user-tuned parameters and naturally balances exploration and exploitation. **TIGRIS** and **PIPE** introduce trajectory-aware information estimation methods, improving the performance of the informative path planners by more accurately estimating the effect of an chosen trajectory.
4. *Insights into Human Indoor Exploration*: A human user study that benchmarks existing robotic exploration methods. Leveraging the user-study findings, we develop **MapExRL**, a learning-based approach that integrates human-inspired exploration strategies to improve robot performance in structured environment for SOTA results.

We extensively validated the proposed methods in simulation and real-world experiments, with hardware demonstrations on multiple UAV platforms, including both multirotor and fixed-wing aircraft.

Chapter 2

Background

2.1 What is path planning?

Path planning or motion planning is usually defined as finding a set of states that leads a robot from a start state to a final state while minimizing a cost function, such as travel time. There are many types of path planners, each having a different cost function or objective.

For example, Fig. 2.1 shows a simple example of an aerial robot trying to get from $\mathbf{x}_s$ to $\mathbf{x}_f$. This type of goal-oriented path planning, meaning the robot is trying to get to a goal state, still has various objective functions that could be chosen that could lead to vastly different paths. In this example, we have shown some hypothetical trajectories for the shortest path (purple) that travels very closely to the obstacles, fastest path (orange) which chooses a smooth path, safest path (red) which stays far away from the obstacles but has a non-smooth path, and the energy-efficient path (green) which may take advantage of a wind field to glide to the final state in much longer path.

Each of these hypothetical paths are optimal given their objective function, even if one of them by themselves may look unintuitive as an outside observer without knowing the underlying objective function.

Another class of objective functions for path planning could be grouped as non-goal-oriented objective functions. This means that the main objective is not to get to an end state, but rather to minimize or maximize a separate objective. Three hypothetical examples are shown in Fig. 2.2. The first is a coverage path (green) which seeks to have the UAV travel to all areas of the map at a specified coverage resolution. However, this path may be too long for the UAV to be able to execute within on flight, having a limited range due to a max battery capacity. This introduces a budget

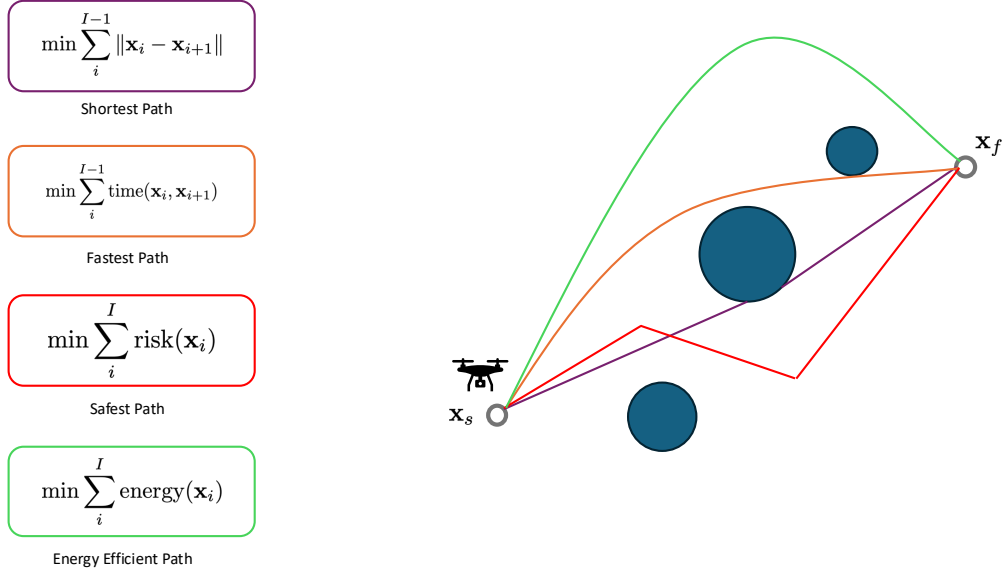


Figure 2.1: Comparison how different goal-oriented path planning objectives could lead to very different paths for a UAV to traverse to a goal position. The start state is marked with $\mathbf{x}_s$ and final state as $\mathbf{x}_f$.

constraint on the coverage path (orange). Looking at where the orange path covers, this may not fully achieve the intended result of the user. There may be areas of the space that are more important to cover than others, where each observation is not treated equally.

One way to adjust coverage would be to spend time in areas proportionally to their importance or expected total information gain, called ergodic coverage. This creates plans that try to both explore and exploit the information map. However, this approach does not directly maximize information gain or leverage the knowledge about the sensor configuration and sensor model. The strong assumptions about the information gain function, such as assuming information is gained at the same position of the robot, that time spent in a space is proportional to the information gain and that information gain is a linear function rather than having diminishing returns, can lead to undesired results.

This leads to informative path planning, where the objective is to directly maximize an objective such as information gain, subject to a budget constraint (e.g., time, distance, or energy). In the previous example, an informative path (blue) might be one that strategically explores the regions surrounding obstacles to gather the most valuable data. This formulation enables the planner to leverage prior knowledge about the environment, sensor configuration and model, and the robot's motion model to make efficient decisions about where to go next.

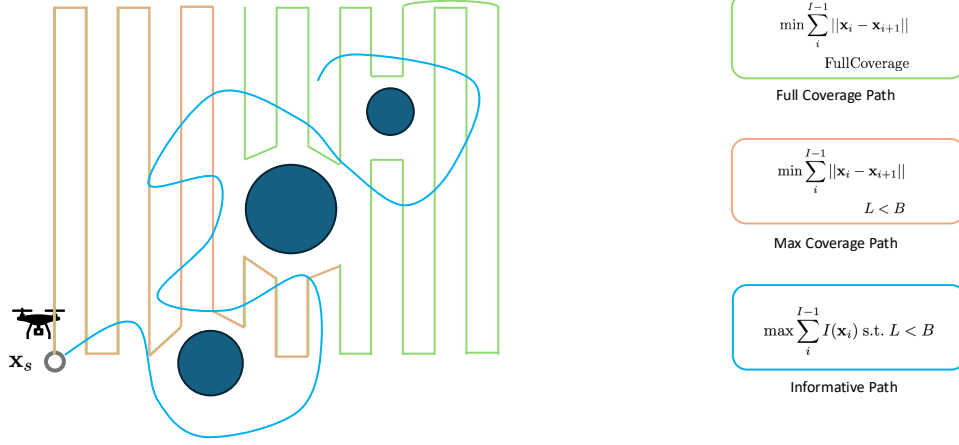


Figure 2.2: Comparison how different non-goal-oriented path planning objectives could lead to very different paths for a UAV to traverse to a goal position. The start state is marked with $\mathbf{x}_s$ and final state as $\mathbf{x}_f$.

These several example objective functions illustrate how small changes in the objectives and constraints of the path planning problem can lead to significant differences in the resulting paths, and consequently, in the execution outcomes. In some cases, the same path planning algorithm can be applied across different objective and constraint formulations. However, when the problem shifts from a goal-oriented path planning task to one that seeks to maximize an objective without a predefined endpoint, the complexity increases substantially. In the case of informative path planning, such problems are NP-Hard, requiring careful consideration of the methodologies used to solve them.

2.2 Problem Definition

The goal of IPP is to maximize information gain given a budget constraint. We define it as

$$\mathcal{T}^* = \arg \max_{\mathcal{T} \in \mathbb{T}} I_h(\mathcal{T}, b) \text{ s.t. } C(\mathcal{T}) \leq B$$

where $\mathcal{T}^*$ is the optimal trajectory, $\mathcal{T}$ is the set of feasible and collision-free trajectories, I is the information function, h is the sensor model, b is the world belief state, $C(\cdot)$ is the cost function, and B is the budget.

2.3 Approximations & Simplifications

Because IPP is a challenging NP-Hard optimization problem, it necessitates approximations, heuristics, and simplification in order to solve it in a timely manner. However, these modified problem definitions "tend to be limited to specific platforms or application domains." (Popović et al. [2020]). Additionally, many approximations lead to solutions that can perform arbitrarily poorly (Krause and Guestrin [2011], Singh et al. [2009]). The following are some of these simplifications and approximations:

- **Continuous vs. discrete search effort:** the IPP problem is much simpler when the "search effort" is modeled as a discrete action rather than a continuous sample Stone [1975]. An example of a true discrete search effort would be digging samples of soil. However, most sensor types are able to take unlimited samples and at a rather high rate. These would be sensors that measure temperature, humidity, wind, or take photos or video. Many methods model the problem where these sensor modalities only take samples at discrete states even though the traversal to the state would gather magnitudes more data than the one sample at the final state. An in between of discrete and continuous would be taking a subset of the actual high-rate samples during the traversal.
- **Simplified Sensor Model:** this includes both how the world belief model is updated as well as the dynamics, constraints, and reach of the sensor. A common simplification is modeling the sensor as a radius around the robot state and the world belief is updated within that radius. The only real sensor I can think of that would match this would be a fish eye camera pointing downward. The radius is sometimes an abstraction of lower level data gathering task that has been merged into one state.
- **Simplified Motion Model:** this includes both the dynamics and the state space of the robot. Many IPP methods plan in a two dimensional planar world. This can work fine either if the scale is large enough or the robot is agile or holonomic. But in some cases this simplification would make it hard to translate the plans into a traversable path in the real world.
- **One-step or Receding Horizon Planning:** planning over the entire budget can sometimes be too challenging and a simpler approach is just reasoning over the next action or over a reduced horizon. This can yield provably optimal local paths, but does not give global guarantees.
- **Planning Once:** if the objective of the planner is the model a static environment, many methods will choose to plan only once which justifies allowing for much longer planning times, as the planning would happen before deployment. Many

environments however aren't static and the single static plan wouldn't be able to take into account disturbances or new observations.

- **Reformulating as Coverage Planning:** in some cases where the budget allows full coverage of the space and all spaces give the same information gain, the solution is a coverage path. However, there are many cases where these conditions aren't met but a coverage path planner is still used. A variant of this is modeling the information gain at each step as proportional to the amount of time spent in the space. The coverage is then proportionally allocated based on the starting information gain and doesn't take into account the actual observation model.
- **Information Gain Simplifications:** calculating the information gain for a path is challenging because it is a prediction of the result for a given future robot action. We therefore often try to calculate and maximize the expectation of the information gain. But this is computation infeasible in many cases. This is when many use heuristic, approximations, or stand in metrics that act as surrogates to hopefully lead to the same effect as if using the actual information gain. Or a simpler world model is chosen that allows for calculating model uncertainty and then minimizing the uncertainty is the objective.

Each of the simplifications and approximations should be chosen carefully and fully justified, as they can limit the impact and scope of proposed works. The aim of this thesis is interrogate, prod, and explore these simplifications to find which should and can be relaxed to create IPP methodologies that can be applied in real-world applications.

2.4 Informative Path Planning Approaches

Robotic data gathering in physical spaces has been approached using various methodologies. The Informative Path Planning (IPP) problem can be formulated in multiple ways, including adaptive sampling and orienteering, each with its own unique problem constraints and variations.

2.4.1 Discrete Approaches to IPP

One common formulation of the IPP problem involves discretizing the continuous space into a fixed set of states, which allows the problem to be framed as an orienteering problem. Given a weighted undirected graph, orienteering entails determining a Hamiltonian path over a subset of nodes that maximizes total information gain while remaining within a budget constraint. This problem can be decomposed into two

main components: selecting a subset of nodes based on an optimization objective and solving a Traveling Salesman Problem (TSP) to find the shortest path among them. The specific optimization objective can be tailored to different tasks, such as environmental monitoring [Bottarelli et al., 2019] or general Constraint Satisfaction Problems (CSP) [Arora and Scherer, 2017].

However, orienteering-based IPP formulations face significant computational challenges. The problem is NP-hard [Singh et al., 2009], and finding near-optimal solutions efficiently requires approximation techniques. Many studies leverage mutual information-based heuristics to ensure submodularity in the information-gain function [Meliou et al., 2007]. This property enables diminishing returns: new observations provide more information when fewer observations have been made, allowing greedy algorithms [Chekuri and Pal, 2005] to approximate solutions with theoretical performance guarantees.

Despite these advances, discrete formulations struggle in large-scale and high-dimensional applications and can become intractable as the size of the search space increases [Binney et al., 2013]. Most existing implementations have been limited to relatively small search spaces, such as aquatic monitoring [Bottarelli et al., 2019], water quality assessment [Cheng et al., 2003], and chemical plume detection [Pang and Farrell, 2006]. In contrast, real-world applications such as search-and-rescue missions and large-scale dynamic target tracking demand algorithms that can scale effectively. While heuristics [Bottarelli et al., 2019], greedy methods [Chekuri and Pal, 2005, Singh et al., 2014], and Mixed Integer Programming (MIP) approaches [Vansteenwegen et al., 2011] have been explored, these methods often fail to capture complex interdependencies in information gain among sensing locations, limiting their applicability.

2.4.2 Receding Horizon Methods

An alternative approach to solving the IPP problem involves receding horizon-based methods. These strategies optimize an information-gain objective over a short lookahead window and rely on iterative replanning to approximate a globally optimal solution. Examples include temporal logic-based planning for safety-critical applications [Jones et al., 2013], Gaussian Process (GP)-based informative planning in uncertain environments [Sun et al., 2017], and adaptive lookahead methods that dynamically adjust the planning horizon to mitigate local optima [Yoo et al., 2016].

However, receding horizon planners are inherently myopic. Since they prioritize immediate information gain without fully leveraging prior environmental knowledge, they often struggle to find globally optimal solutions. Studies have shown that these planners can become trapped in locally optimal solutions unless the planning horizon is carefully tuned [Kailas, 2023].

2.4.3 Global Path Planning Approaches

To address these limitations, many approaches focus on computing globally optimal paths by discretizing the environment into a graph representation and solving a graph search problem [Binney et al., 2010, Singh et al., 2009, Zhu et al., 2021]. However, these approaches face computational bottlenecks. Frolov et al. [2014] found that many graph-based approaches perform only marginally better than simple lawnmower search patterns due to their inability to revisit areas effectively. Allowing for revisits is crucial for maximizing information gain when the reward function is submodular or when tasks require repeated measurements.

An alternative is to formulate IPP as a branch-and-bound problem, where suboptimal paths are pruned early to reduce computational overhead [Best and Fitch, 2016, Binney and Sukhatme, 2012]. While these techniques improve efficiency, computing tight bounds for pruning remains challenging, particularly in high-dimensional state spaces with uncertain environments. Gaussian Process (GP)-based methods have been proposed to estimate information gain efficiently, but they introduce additional computational constraints since inference in GPs scales cubically with the number of observations [Rasmussen and Williams, 2006]. Even with sparse approximations, long-horizon planning remains computationally intensive unless strong assumptions about stationarity are imposed on the covariance function.

2.4.4 Continuous Space Approaches

Rather than discretizing the environment, another class of IPP solutions operates in continuous space. Sampling-based methods [Hollinger, 2015, Hollinger and Sukhatme, 2014, Schmid et al., 2020] incrementally construct a tree by sampling points in the continuous domain. An anytime best path is selected based on the estimated information gain, subject to a budget constraint. This approach mitigates the scalability issues associated with graph-based methods while providing flexibility in large state spaces. However, naive implementations can still suffer from the curse of dimensionality. As the problem dimensionality grows, the volume of the search space expands exponentially, making exhaustive exploration impractical.

2.4.5 Toward Scalable and Efficient IPP

Given these limitations, an important question arises: how can we perform anytime informative path planning in large, high-dimensional spaces efficiently? Addressing this challenge requires balancing computational efficiency with the need for long-horizon planning.

In the following chapters, we explore these challenges in depth and present a solutions that builds on sampling-based planning while incorporating informed search strategies and adaptive refinement techniques to maximize efficiency in high-dimensional environments.

Part I

Informative Path Planning

Chapter 3

Informed, Sampling-based Algorithm for Informative Path Planning

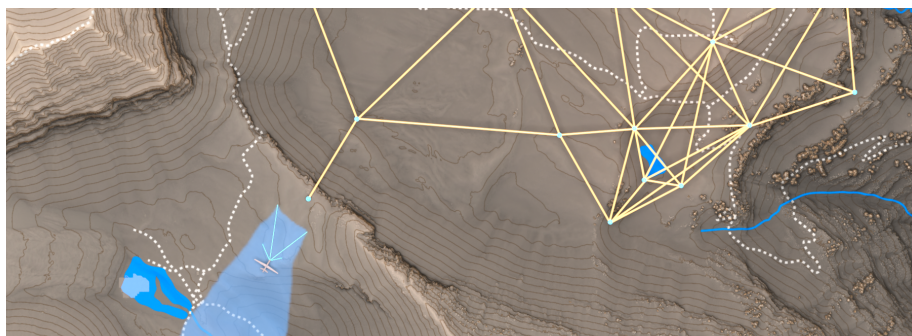


Figure 3.1: Example informative path planning scenario of searching over a mountainous terrain using a camera. Photo links to the overview video for this chapter.

In this chapter we delve deeper into the informative path planning problem and outline our first approach through the use of a sampling-based algorithm¹. We present our simulation results for single, global plans without online adaptation. This sets up the core foundation for the later work in Chapter 4, where we will extend this base algorithm to create a robust planner that is adaptive, incremental, and has been deployed and tested extensively on hardware systems.

3.1 Introduction

Real-world data gathering is an important problem being solved through autonomous robotic platforms. Some examples include spatiotemporal monitoring [Binney

¹This work was published in Moon et al. [2022].

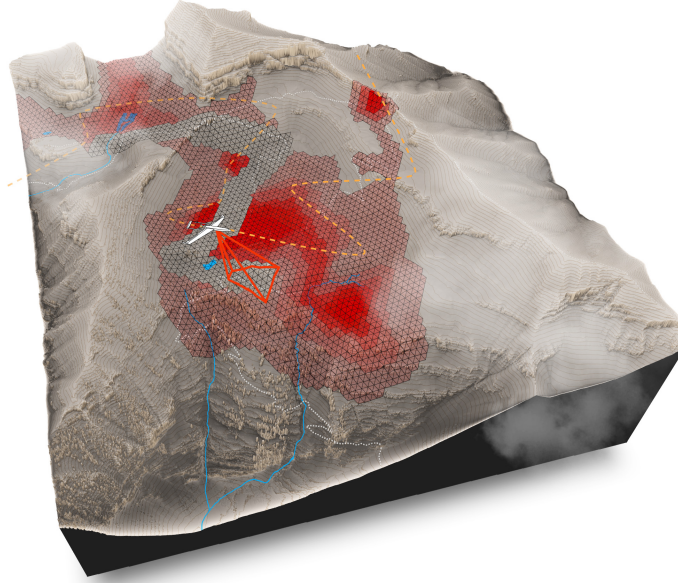


Figure 3.2: An example scenario for an informative path planning problem of searching for a missing hiker. The UAV plans a path of maximum information gain over a prior belief distribution of the hiker’s location.

et al., 2013], aerial surveillance [Cabreira et al., 2019, Moon and Peterson, 2018, Nguyen, 2015], estimating urban wind fields [Patrikar et al., 2020], and wildfire monitoring [Towhidul Islam and Hu, 2021]. The traditional methods when using humans to accomplish these tasks may be inefficient, infeasible, tedious, or too risky to carry out. Using autonomous robots to accomplish these tasks allows for a more efficient and safe alternative, as well as opening doors to new applications that were not formerly possible.

Each data gathering problem can present unique challenges depending on the type of data, method of collection, and sampling environment the robot must traverse, but the overall framework for solving the problem can be formulated in a unifying way. A typical way to formulate a data gathering problem is as an informative path planning (IPP) problem which involves finding paths that maximize information gathered while respecting budget constraints. These problems can be challenging to solve as they have been shown to be NP-hard [Singh et al., 2009] and even PSPACE-hard [Reif, 1979].

Many approaches [Alamdari et al., 2014, Arora and Scherer, 2017, Yu et al., 2014] attempt to find a global solution by discretizing the space and reducing the number of states to be as small as possible, such as planning on a grid or nodes on a graph. These solutions can be resolution complete, optimal for the given problem conditioning, but are challenging to apply to real-world scenarios and often give solutions that are clearly suboptimal for the situation. This can be due to having to reduce the size of the search space in order to get a solution quickly enough or the oversimplification of

the space which results in having to ignore important factors such as data gathered between the nodes on the graph.

The work by [Hollinger and Sukhatme \[2014\]](#) adapted common sampling-based motion planning algorithms to IPP with motion constraints and asymptotic optimality analysis. Their work quickly explores the space of possible paths to find a feasible and optimal solution and was shown to find optimal solutions faster than existing solvers. In the work in this chapter, we build upon their approach to create a sampling-based information gathering algorithm for IPP and present TIGRIS (Tree-based Information Gathering using Informed Sampling) which introduces two key improvements:

1. Perform informed sampling within a subset of the configuration space that influences the estimated information gain, allowing for planning in larger and higher dimensional search spaces by prioritizing the exploration of paths with high reward.
2. Include edge rewards using a novel edge cost formulation and demonstrate its advantage in spite of the extra computation.

We demonstrate that our algorithm works well at rapidly planning high information gain paths in high-dimensional and large spaces while being constrained to a relatively small budget relative to the size of the space. An example use-case of this algorithm is shown in [Fig. 3.2](#) where a fixed-wing UAV with a forward-facing image sensor is searching a region for a missing hiker. In such a situation, it is necessary for a path planner to be able to plan paths through areas of high potential reward in order to maximize the likelihood of collecting the most amount of relevant information while subject to a relatively small budget.

Additionally, in this chapter we also detail how we implement TIGRIS for our specific use case—searching for objects of interest using a fixed-wing UAV in an environment that is much larger than the available budget. We step through all the considerations that had to be made to successfully implement TIGRIS for this challenging problem including how to estimate edge rewards and the potential gain in information of a candidate path in the tree. We further detail our environment setup, belief representation, and how we implement our approach in a practical and feasible way and also release TIGRIS as an open source package.

The chapter is organized as follows: [Section 3.2](#) reviews the related works in this field and the contributions of the work in this chapter. [Section 3.3](#) lays out the problem definition. [Section 3.4](#) explains our proposed algorithm. [Section 3.5](#) details our approach to implementing TIGRIS for our particular use case and all the considerations that were made in our planning environment. [Section 3.6](#) lays out our testing framework and results compared to the baseline method.

3.2 Background

With the foundational background covered in Section 2.4, this chapter focuses on TIGRIS, our sampling-based approach for performing informative path planning in large, high-dimensional spaces. While prior approaches have explored receding horizon control, graph-based search, and sampling-based methods, these techniques face limitations in scalability or long-horizon reasoning.

A key challenge is balancing exploration and exploitation in high-dimensional environments while maintaining computational efficiency. Existing methods often struggle with computational tractability when applied to large-scale problems. Our approach builds upon sampling-based methods by incorporating informed sampling and edge-based reward estimation, allowing us to prioritize high-reward regions dynamically.

In this chapter, we present TIGRIS, detailing how it overcomes the limitations of prior work and enables efficient, scalable information gathering in complex environments. We analyze its performance in various scenarios and demonstrate its advantages over existing methods.

3.3 Problem Definition

As defined in Section 2.2, let $\mathcal{T}$ represent a sensor trajectory in $\mathbb{T}$ and $C(\mathcal{T})$ be the cost function for a given trajectory. Let $I(\mathcal{T})$ be the expected reward or information gain from a trajectory and B be the budget constraint.

We define the informative path planning problem as maximizing information gain subject to a budget constraint as in the following:

$$\mathcal{T}^* = \arg \max_{\mathcal{T} \in \mathbb{T}} I(\mathcal{T}) \text{ s.t. } C(\mathcal{T}) \leq B \quad (3.1)$$

where an optimal trajectory $\mathcal{T}^*$ is found that does not exceed the budget B .

In this formulation, the information reward function $I(\cdot)$ is not constrained to being modular and could also be time-varying modular or even submodular [Krause and Guestrin, 2011].

Algorithm 1: TIGRIS($\mathbf{x}_{start} \in X_{free}, B, T, \Delta, \mathcal{X}$)

```
1  $I_s \leftarrow I(\mathbf{x}_{start}); C \leftarrow 0;$ 
2  $n_{start} \leftarrow \{\mathbf{x}_{start}, I_s, C\};$ 
3  $V \leftarrow n_{start}; E \leftarrow \emptyset; V_{closed} \leftarrow \emptyset; G \leftarrow (V, E);$ 
4 while computation time  $< T$  do
5    $\mathbf{x}_{sample} \leftarrow \text{InformedSample}(\mathcal{X});$ 
6    $n_{nearest} \leftarrow \text{Nearest}(\mathbf{x}_{sample}, V \setminus V_{closed});$ 
7    $\mathbf{x}_{feasible} \leftarrow \text{Steer}(\mathbf{x}_{sample}, n_{nearest}, \Delta, X_{free});$ 
8   if  $\mathbf{x}_{feasible} \neq n_{nearest}$  then
9      $N_{near} \leftarrow \text{Near}(\mathbf{x}_{feasible}, V \setminus V_{closed});$ 
10    for  $n_{near} \in N_{near}$  do
11       $\mathbf{x}_{new}, e_{new} \leftarrow \text{Steer}(\mathbf{x}_{feasible}, n_{near}, \Delta, X_{free});$ 
12      if  $\mathbf{x}_{new} \neq n_{near}$  then
13         $I_{new} \leftarrow I(\mathbf{x}_{new}, e_{new}, n_{near});$ 
14         $C_{new} \leftarrow C_{n_{near}} + \text{Cost}(e_{new});$ 
15         $n_{new} \leftarrow \{\mathbf{x}_{new}, I_{new}, C_{new}\};$ 
16        if not  $\text{Prune}(n_{new})$  then
17           $E \stackrel{+}{\leftarrow} \{(n_{near}, n_{new})\};$ 
18           $V \stackrel{+}{\leftarrow} \{n_{new}\};$ 
19           $G \leftarrow (V, E);$ 
20          if  $C_{new} = B$  then
21             $V_{closed} \stackrel{+}{\leftarrow} \{n_{new}\};$ 
22 return  $\mathcal{T} \leftarrow \text{BestPath}(G)$ 
```

3.4 Methodology

On a high level, TIGRIS is similar to other sampling-based planners in that points are sampled within the search space and used to build out a tree. For the case of IPP, each node in the tree contains the state, path cost, and path information gain. At the end of the planning time, the node with the highest value of path information is returned. When searching over a large space while subject to a limited budget, our planner seeks to build high reward paths quickly through prioritizing the sampling of regions with high reward. This allows us to generate plans that outperform random sampling approaches throughout the planning time. The code for TIGRIS is described in Algorithm 1.

Let $\mathcal{X}$ represent the state space and $\mathbf{x}_{start}$ being the starting configuration within X_{free} . T represents the time allowed for planning and Δ is the extend distance when building the tree. For brevity, we will use the notation $X \stackrel{+}{\leftarrow} \{\mathbf{x}\}$ and $X \stackrel{-}{\leftarrow} \{\mathbf{x}\}$ to

represent the operations $X \leftarrow X \cup \{\mathbf{x}\}$ and $X \leftarrow X \setminus \{\mathbf{x}\}$ respectively.

The algorithm starts by calculating the information reward for the starting configuration $\mathbf{x}_{start} \in \mathcal{X}$ with $I(\cdot)$ and sets the cost to zero (Algorithm 1 Line 1). It then initializes the starting node n_{start} , adds it to the vertices set V , and initializes the edges set E and graph G (Algorithm 1 Line 2-3).

While the computation time has not exceeded the allowed planning time T , the tree will continue to be expanded. This begins with a new configuration $\mathbf{x}_{sample}$ being sampled using *InformedSample*($\cdot$) (Algorithm 1 Line 5), which in our case we use weighted sampling where the probability is directly proportional to the information reward. Details on how we implemented this sampling efficiently is found in Section 3.5.3.

The nearest node $n_{nearest} \in V \setminus V_{closed}$ to $\mathbf{x}_{sample}$ is found using the function *Nearest*($\cdot$) (Algorithm 1 Line 6). Efficient nearest neighbors search could be implemented using structures such as a k-d tree or R-tree.

A feasible point $\mathbf{x}_{feasible}$ is found by extending from the nearest node by a maximum cost Δ using the function *Steer*($\cdot$) (Algorithm 1 Line 7). This feasible point may have an edge cost less than Δ because the *Steer*($\cdot$) function will extend until a collision or the budget is reached. This is a subtle difference from [86] in that ours will not allow points to extend beyond the budget but will rather reduce the extended distance to create a path that uses the entire budget.

If the feasible point is not the same as the sample point, then near points $n_{near} \in V \setminus V_{closed}$ within a radius from $\mathbf{x}_{feasible}$ are found using *Near*($\cdot$) and set as N_{near} (Algorithm 1 Line 8-9). This radius could be predefined or set using a function like in Karaman and Frazzoli [2010] which defines a closed hypersphere of radius r_n where n is the number of nodes.

For each node n_{near} in the set N_{near} , and new state $\mathbf{x}_{new}$ and edge e_{new} is extended from n_{near} to $\mathbf{x}_{feasible}$ using the *Steer*($\cdot$) function of cost Δ (Algorithm 1 Line 11). If the new state $\mathbf{x}_{new}$ is not the same as $\mathbf{x}_{feasible}$ (meaning it is not trapped or in the closed set), then a new node n_{new} will be created. The information reward for the node is estimated using $I(\cdot)$ which calculates the reward for the trajectory up to the new state $\mathbf{x}_{new}$, including the new edge e_{new} and entire trajectory of edges and states preceding it. Depending on the modularity and temporal dependence of your information function, calculating the reward for a new node might be as simple as appending the additional information gain from the new state and edge or, alternatively, the entire trajectory up to that point may need to be calculated.

After calculating the estimated information reward, the cost of the new edge is found with *Cost*($\cdot$) and appended to the cost of the near node $C_{n_{near}}$. With the new

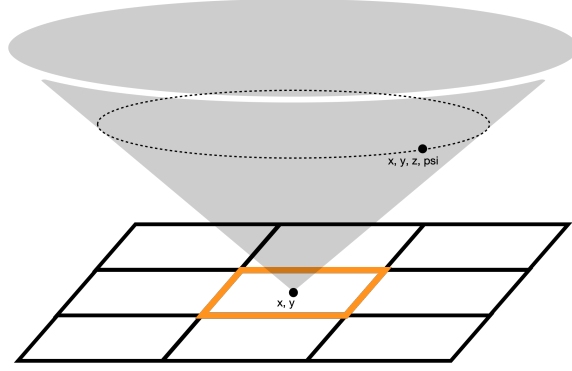


Figure 3.3: Visualization of how the informed sample state is created from a Bayesian grid belief space.

state, information, and cost, a new node is now created (Algorithm 1 Line 13-15). However, before adding this new node to the tree, it is first checked for if it should rather be pruned. The $Prune(\cdot)$ function examines nearby nodes to see if it is worth adding the new node. This could be a simple heuristic of seeing if a nearby node has a lower cost and higher reward, but this heuristic can have implications on finding the optimal solution. A variety of other heuristics can be used to reduce computational complexity, but, to ensure optimality, the upper bound on reward for the nodes would need to be used within the $Prune(\cdot)$ function.

If the node isn't pruned, the edge is added to the set E , the node is added to the set V , and the graph is updated (Algorithm 1 Line 17-19). If the cost of the new node is equal to the budget, then the node is also add to the closed set V_{closed} . Lastly, once the while loop is terminated when the max planning time is exceeded, the path with the highest information reward is returned.

3.5 Implementation

We implemented TIGRIS with the objective of reducing entropy with a bias toward increasing belief probabilities. Our data gathering is performed using a fixed wing UAV and a static forward facing camera. We discretize our space into grid cells with index i , and let X_i be the event that an object of interest lies within the cell at index i . Therefore $P(X_i)$ is the probability of the event, such as a lost hiker being in the cell. In the formulation of entropy reduction, the value within a cell could easily be any other metric that we want to reduce uncertainty for such as a model for radioactive strength in a space or a temperature field. We assume a prior belief over the map that we then leverage to plan a path that reduces the overall entropy.

Let $\mathbf{x} \in \mathbb{R}^3 \times SO(2)$ represent the state of a fixed wing UAV in inertial space.

Algorithm 2: InformedSample()

```
1  $x, y \leftarrow \text{WeightedSampler}();$   
2  $z \leftarrow \text{SampleZ}();$   
3  $\psi \leftarrow \text{Sample}\psi();$   
4  $\Delta_{xy} \leftarrow z \tan(\theta - v_{opt} \frac{\Theta_v}{2});$   
5  $x \leftarrow x - \Delta_{xy} \cos(\psi);$   
6  $y \leftarrow y - \Delta_{xy} \sin(\psi);$   
7 return  $x, y, z, \psi$ 
```

We assume constant curvature and connect states using Dubins paths. The sensor is statically mounted facing forward on the UAV with a downward pitch angle of θ and the heading matching the vehicle of ψ as seen in Fig. 3.6.

For the $\text{Prune}(\cdot)$ function, we chose an aggressive heuristic of not adding a new node if a nearby node has a lower cost and higher reward. For our nearest neighbor search we used the Open Motion Planning Library [Şucan et al., 2012]. Our function for $\text{InformedSample}(\cdot)$ is seen in Algorithm 2. We implemented weighted sampling with the probability of selection being proportional to information reward, however calculating the information reward for every state would be computationally infeasible.

To handle this, we compute the information reward for viewing each cell at an optimal range and then use that value for a weighted sample over cells (Algorithm 2 Line 1). We then sample uniformly within the subset of the space that is independent of the sampled cell, z and ψ (Algorithm 2 Line 2-3). The full sampled state can then be found using v_{opt} , which is a fraction reflecting the desired vertical placement of the target between the bottom of the frame and the optical axis in the image frame (Algorithm 2 Line 4-7). A visualization of this method is shown in Fig. 3.3.

The impact of informed sampling versus uniform sampling is shown in Fig. 3.4. We can see how the sampled points, shown as black dots, focus on the areas of higher entropy in the search space in comparison to uniform sampling. This results in a higher density of potential paths being evaluated in these areas. In this example, the resulting paths had higher reward when using informed sampling.

3.5.1 Sensor Model

A sensor model is necessary for belief map updates given sensor measurements. The function approximating the sensor performance could be derived through testing a given perception system and approximated with a high order polynomial or even be a lookup based on the testing data. However the method, it is important to ensure the model maps closely and accurately to the actual system, otherwise the search

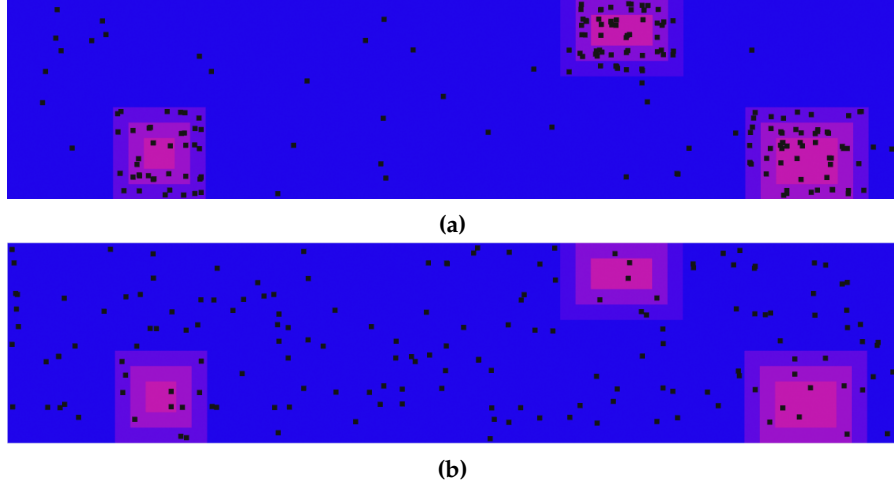


Figure 3.4: Visualization of sampled points (black dots) within a belief space. Blue areas have low entropy, while pink areas have high entropy. (a) Informed sampling across the belief space. (b) Uniform sampling across the belief space. You can see how the informed sampling focuses the search in areas of higher potential reward.

algorithm could lead to erroneous behaviors. There are many factors that could be included as inputs to a sensor model include target range, sensor velocity, sensor acceleration, shutter speed, sensing angle, altitude, target material type, weather, fog, lighting conditions, and target behavior.

In our implementation, we are modeling the performance of a perception system that uses an electro-optical sensor. Let the variable Z represent the event of a positive measurement. We model both the true positive rate $P(Z|X)$ and false positive rate $P(Z|\neg X)$ using (3.2), with the false positive rate just being the complement of the equation in our case, and r being the range of the object from the sensor and a , b , and c being parameters to fit the model to the performance of a perception pipeline. β is where the sensors performance drops off to where measurements do not change the belief space. Fig. 3.5 shows the plots of (3.2) and its complement.

$$f(r) = \begin{cases} \frac{1}{a+e^{b(r-c)}} & r \leq \beta \\ 0.5 & r > \beta \end{cases} \quad (3.2)$$

3.5.2 Information Reward Function

We base our information reward function on the reduction in entropy [Paull et al., 2014] from a new measurement. The full equation for Shannon entropy in our case is

$$H(X) = -P(X) \log P(X) - P(\neg X) \log P(\neg X)$$

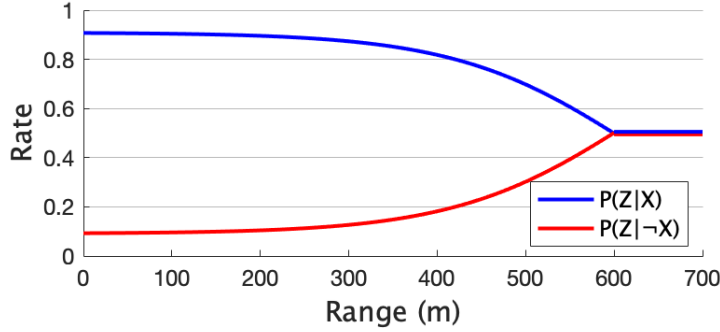


Figure 3.5: An example sensor model of (3.2) with blue being the true positive rate and red being the false positive rate.

Using Bayes' theorem to find $P(X|Z)$, the entropy reduction due to a positive measurement would be

$$\Delta H(X|Z) = H(X) - H(X|Z)$$

In order to prioritize reducing entropy in areas that increase our confidence in objects locations, we also add weights to the entropy reduction. R_p is the weight for increasing probabilities and R_n is for decreasing probabilities.

Ideally we would compute the expectation of the entropy reduction over the entire path, but computing this for all combinations of outcomes for each path segment would be computationally expensive. We follow the same approach as Hollinger [2015] of using the optimistic approximation of the expected reward to compute the upper bound of the reward. If $P(X) \geq 0.5$ then we assume a positive measurement Z and the converse if less than 0.5. The resulting reward function is the following:

$$R(X) = \begin{cases} R_p \Delta H(X|Z) & P(X) \geq 0.5 \\ R_n \Delta H(X|\neg Z) & P(X) < 0.5 \end{cases} \quad (3.3)$$

3.5.3 Estimating Edge Rewards

When calculating the information reward over a path, the sensor footprint can be projected onto the surface plane to find all cells within the footprint and the reward for each cell is calculated using (3.3) and summed. This step becomes more complicated when including edges in the reward calculation. Discretizing the edge into nodes would have imperfect overlaps due to a non-square sensor footprint, thus not including some of the grid cells in calculating the edge reward. To mitigate this issue, large overlaps of the sensor footprints would be needed but would thus lead to a large increase in the computation time for grid cell belief updates. Rather than choosing between the tradeoffs, we compute an approximation of the edge reward by finding

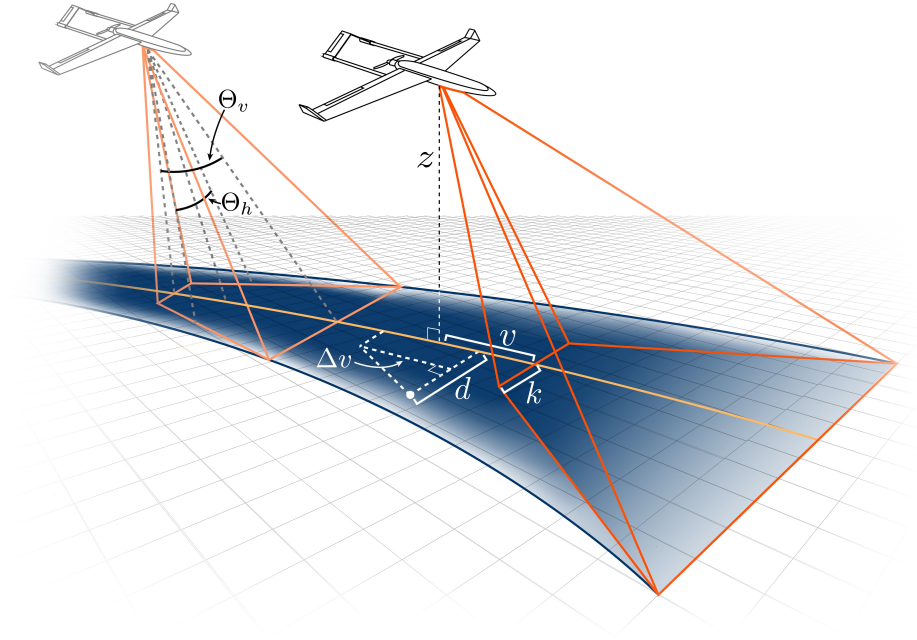


Figure 3.6: A visualization of the edge reward approximation and variable definitions given a fixed forward-facing camera.

all cells within the sliding sensor footprint along the edge and find the minimum distance the UAV will be to each of those cells. This ensures all cells within the edge are accounted for and each belief value is updated using a measurement when the sensor is closest to that cell.

The minimum range r_{min} of a point to our sensor is first found by calculating the perpendicular distance d of the point to the edge line. Let z be the altitude of the camera sensor above the ground, k be half of the length of the bottom edge of the projected camera footprint on the ground plane, $\Theta_v = VFOV$, and $\Theta_h = HFOV$. The distance L from the edge defines the distance a point will be to the sensor when it transitions from being on the bottom of the frame to the left or right edge of the image frame and is calculated using (3.4).

$$L = \begin{cases} k & \theta > \frac{\Theta_v}{2} \\ z \tan(\frac{\Theta_h}{2}) & \theta \leq \frac{\Theta_v}{2} \end{cases} \quad (3.4)$$

We then calculate r_{min} using (3.5) where $v = z \tan(\theta - \frac{\Theta_v}{2})$ and $\Delta v = \frac{d-L}{\tan(\theta)}$. The resulting r_{min} is used to for each cell belief update and reward calculation of cells that are being passed over. For cells at the end of edges or at nodes, euclidean distance is

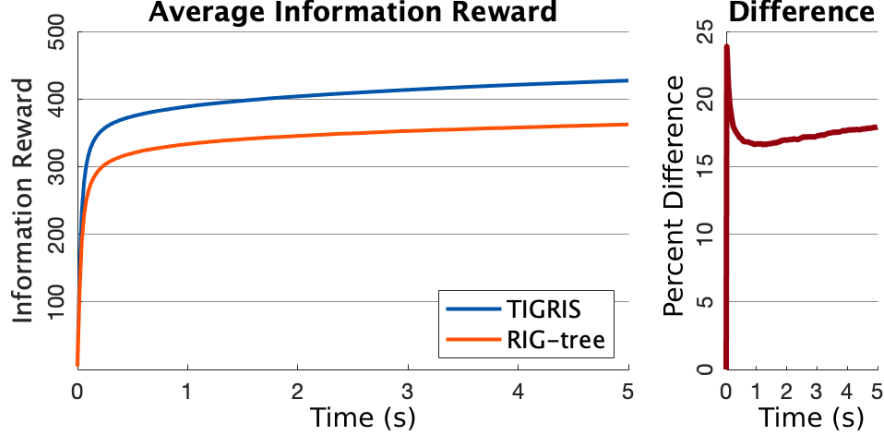


Figure 3.7: The results of 12000 Monte Carlo simulations. The blue line shows the average path information reward for the TIGRIS planner while the orange is for the RIG-tree. The graph on the right shows the difference in the two means.

used for the range. A visual representation of these calculations is seen in Fig. 3.6.

$$r_{min} = \begin{cases} \left\| \begin{bmatrix} d & v & z \end{bmatrix} \right\|_2 & \theta > \frac{\Theta_v}{2} \wedge d < L \\ \left\| \begin{bmatrix} d & 0 & z \end{bmatrix} \right\|_2 & \theta \leq \frac{\Theta_v}{2} \wedge d < L \\ \left\| \begin{bmatrix} d & v + \Delta v & z \end{bmatrix} \right\|_2 & \theta > \frac{\Theta_v}{2} \wedge d \geq L \\ \left\| \begin{bmatrix} d & \Delta v & z \end{bmatrix} \right\|_2 & \theta \leq \frac{\Theta_v}{2} \wedge d \geq L \end{cases} \quad (3.5)$$

3.6 Testing Results

We test our TIGRIS algorithm and the RIG-tree algorithm outlined in [Hollinger and Sukhatme \[2014\]](#), which was shown to find optimal solutions in similar testing environments. We used the same parameters for each algorithm and implemented both in C++. We also used the same information reward function when evaluating final trajectories for each algorithm. The simulation environment was a 5000x5000 meter area with grid cells of 50x50. The UAV had a budget $B = 6000$ and the camera had a pitch of $\theta = 65^\circ$.

The simulations were run on a 3.4 GHz Intel i7 processor with 16 GB of RAM. 12000 Monte Carlo simulations were carried out with 1-12 randomly placed target belief centroids. For each centroid, a random prior mean belief value and standard deviation were selected for a Gaussian kernel.

Algorithm	n = 1-3	n = 4-6	n = 7-9	n = 10-12
TIGRIS	314±132	423±99	469±86	502±78
RIG-tree [86]	235±121	351±108	411±98	452±91
Difference	79	71	58	51
Percent Diff	33.6%	20.5%	14.1%	11.1%

Table 3.1: Information reward ($M \pm SD$) with decreasing sparsity in the number of target belief centroids (n) in the environment.

The results for the Monte Carlo simulations are seen in Fig. 3.7. The plot shows the information reward of the best path averaged over the 12000 runs. For both methods a high quality path is quickly found and both seem to be approaching an asymptotic value. It is also clear to see the improvement in TIGRIS over the entire planning time. The mean and standard deviation of information reward for TIGRIS and RIG-tree at 5 seconds is 427.53 ± 126.82 and 362.38 ± 136.26 respectively, which is an 18.0% improvement. This give a P value of well less than 0.0001 for the single tail difference between the two means. It is interesting to note the right side of Fig. 3.7 which shows that the percent difference in the mean rewards continues to increase at a near linear rate after the initial jump.

Table 3.1 groups the results based on the number of target belief centroids that were in the testing environment. The data shows that as the density of the target belief centroids decreases the performance difference between the algorithms increases. Informed sampling becomes more of an advantage as it becomes harder for random sampling to find high-reward paths. This shows the strength of our method in sparse environments. Even at the lowest difference between the algorithms, the P-values of the difference between the mean information rewards were all far below 0.0001.

Fig. 3.8 shows one result within the simulation environment. The planned path by TIGRIS is seen in blue and has an information reward of 531.2 while the RIG-tree path is in pink and has an information reward of 407.2. The green arrows at each node show the ψ . The background color show the information reward for each cell with blue being low reward and red being high. The image on the bottom right shows the 3D view of the environment and helps visualize the viewing frustum and sensor footprint. From looking at the results, you can see how TIGRIS maximizes the amount of information reward with the given budget by viewing all of the highest reward regions, even multiple times.

It is also interesting to note the value of the best trajectory reward over time. Because the baseline did not include edge rewards, it would often miscalculate which trajectory was the best. Often a worse path would be selected over a better path

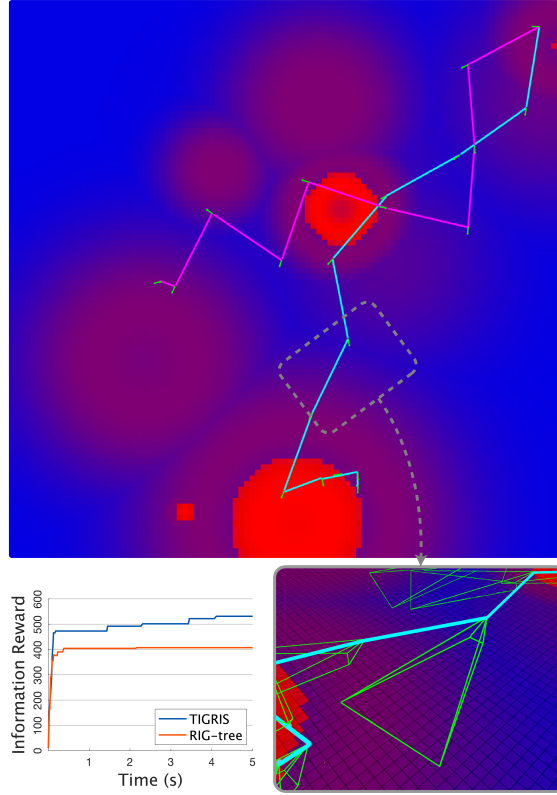


Figure 3.8: Example results in the testing environment. Red regions are areas of high reward and blue is low reward. The blue lines is the TIGRIS solution and the pink line is RIG-tree. The bottom right image shows a 3D view of the environment, and the bottom left graph shows the best path reward of the planners over time.

because an area of high reward under an edge was not accounted for. A graph of the best path information reward versus time can be seen in the bottom left of Fig. 3.8. The best path reward for the baseline decreased at some points in the beginning of the planning. Even with the additional computation time due to edge reward calculations, testing showed TIGRIS found better paths by using our edge information reward approximation. Another benefit to adding edge rewards is it allows for larger extend distances while still being able to closely approximate the information reward over the trajectory. This means TIGRIS can plan in large spaces with a less dense tree than the baseline. Without the edge rewards, calculating the trajectory reward with only the nodes would be a poor approximation of the path reward and would not reflect the reality that information is also gathered between the nodes.

3.7 Conclusion

This chapter presents a novel informed sampling-based algorithm for informative planning called TIGRIS. Our method is tested using Monte Carlo simulations and is shown to have a statistically significant improvement over a comparable baseline. TIGRIS does well at quickly finding paths that maximize the information reward and reducing entropy over the belief space.

As the budget increases to be large relative to the search space, the performance of TIGRIS would be expected to degrade as the problem approaches that of a coverage planner and the density of the tree would need to be intractably high. This leads us to the work in the next chapter, where we introduce replanning on the fly and path refinement.

Chapter 4

Incremental and Adaptive Sampling-Based Planner for Online Informative Path Planning

In the previous chapter, we introduced the TIGRIS algorithm which was a sampling-based planner for informative path planning. Its key contributions were informed sampling and its edge reward approximation method. However, this algorithm was only run once to create a single global plan.

To effectively adapt to real-time data collection, informative path planning must be computed online and be responsive to new observations. In this chapter, we present IA-TIGRIS, an incremental and adaptive sampling-based informative path planner that can be run efficiently with onboard computation. Our approach leverages past planning efforts through incremental refinement while continuously adapting to updated world beliefs. We additionally present detailed implementation and optimization insights to facilitate real-world deployment, along with an array of reward functions tailored to specific missions and behaviors. Extensive simulation results demonstrate IA-TIGRIS generates higher-quality paths compared to baseline methods. We validate our planner on two distinct hardware platforms: a hexarotor UAV and a fixed-wing UAV, each having unique motion models and configuration spaces. Our results show up to a 41% improvement in information gain compared to baseline methods, suggesting significant potential for deployment in real-world applications.¹

¹Project website and video for the work in this chapter can be found at ia-tigris.github.io, and the associated paper is Moon et al. [2025]

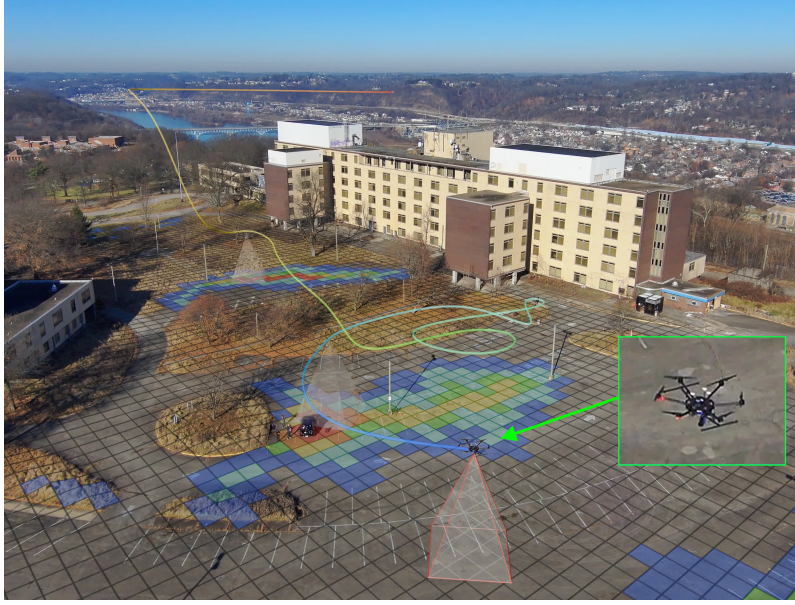


Figure 4.1: IA-TIGRIS deployed on a hexarotor UAV to map the location of cars in the environment. The visualization shows a probability grid representing the prior belief of car locations and a representative path generated by our planner. The algorithm runs entirely onboard, continuously refining and adapting paths based on updated world beliefs from sensor observations.

4.1 Introduction

Robots play a vital role in gathering information from the physical world, supporting a wide range of applications such as scientific research [De Almeida et al., 2021, McCammon and Hollinger, 2021, Shah et al., 2020], environmental monitoring [Barbedo, 2019, Christensen, 2015, Kaufmann et al., 2021, Lim et al., 2023, Patrikar et al., 2020], search and rescue operations [Alsamhi et al., 2022, Bashyam and Guggenheim, 2019], and disaster response efforts [Bailon-Ruiz et al., 2022, Bejiga et al., 2017, Mohsan et al., 2023]. By employing intelligent algorithms, robotic systems enhance the efficiency of data collection, provide valuable insights, and support well-informed decision-making processes. These autonomous robots provide unparalleled advantages in situations where human access is constrained, dangerous, or logistically challenging [Dunbabin and Marques, 2012]. Moreover, the scalability and speed of autonomous information gathering robots significantly enhances the rate of information gathering by not capping the number of robots based on the number of human operators.

Informative path planning (IPP) seeks to create intelligent paths for robots to execute actions that maximize their inherent potential and advantages in data gathering situations, while not violating budget constraints on the system or mission. This approach has many advantages over most coverage-based approaches which often don't take into account the flight time of the drone or prior information about the

space and don't adapt to learned information during flight. However, solving the IPP problem is computationally expensive, being at least NP-hard even in a discrete space [Singh et al., 2009], leading to methods that only plan over a short horizon or are static during the path execution.

In the work in this chapter, we introduce IA-TIGRIS (Incremental and Adaptive Tree-based Information Gathering using Informed Sampling), a sampling-based informative path planner that incrementally builds its search tree and adapts to new information online by efficiently updating and building the set of possible trajectories. This leads to global paths that are more robust to path execution disturbances, adapt to changes in the information map, and are higher quality than a single plan due to continual online refinement. Fig. 4.1 demonstrates IA-TIGRIS deployed on a hexarotor drone for an IPP application. Our key contributions are the following:

- A novel incremental and adaptive sampling-based planner for IPP that dynamically plans online using the updated belief map. We provide key implementation details, insights, and optimizations that enable efficient trajectory evaluation and updates during incremental planning.
- Thorough testing and evaluation of our method against other baselines, as well as ablations and analysis on the components of our algorithm.
- Hardware validation on two distinct UAV platforms with different motion constraints and configuration spaces, demonstrating versatility and real-world applicability.

The chapter is organized as follows: Section 4.2 gives a detailed overview of related works and the contributions of the work in this chapter. Section 4.3 presents the problem formulation. Our proposed approach is detailed in Section 4.4. The section outlines the planner's design and its integration with system components to enable adaptive and efficient informative path planning for real-world applications. Section 4.5 provides our detailed experimental evaluation and results of the approach. The results of our field deployments are summarized in Section 4.6, and Section 4.7 offers the chapter conclusion and future work.

4.2 Background

Numerous works have worked to address these drawbacks by formulating the task of information gathering in continuous space and introducing adaptive sampling-based algorithms. Examples of some of these works include Arora et al. [2019], Bender et al. [2013], Das et al. [2013], Hollinger [2015], Ma et al. [2017], Marchant and Ramos

[2014], Schmid et al. [2020]. These methods employ sampling-based approaches, which involves the selection and inclusion of new states in continuous space into a tree structure. As the tree grows, the optimal path—maximizing information gain while respecting budget constraints—is continuously refined and the best path is selected. However, these approaches face significant challenges in expansive spaces, where increasing dimensionality causes the search space to grow substantially relative to regions of high information reward.

In order to address these challenges, in Chapter 3 we introduced a sampling-based approach, TIGRIS, for IPP in large and high-dimensional search spaces. The method performs informed sampling within the continuous space and incorporates edge information gain during reward estimation, efficiently generating global paths that optimize information gathering within budget constraints.

While TIGRIS demonstrated effective global planning, the approach generated static plans that remained unchanged during robot execution. Our proposed algorithm, IA-TIGRIS, also uses a sampling-based framework but continuously optimizes and refines plans online as the robot acquires new information. This adaptive refinement is crucial for long-horizon planning scenarios where optimal paths are difficult to determine in a single planning cycle. Through improvements in tree-building logic and novel belief space node embeddings, IA-TIGRIS achieves sufficient computational efficiency for onboard deployment. We validate our method through extensive simulation testing and demonstrate efficient online planning capabilities on hardware across two distinct UAV systems.

4.3 Problem Formulation

This work addresses the challenge of maximizing data gathered by a robot within the specified budget constraints of the system, a problem formally known as the IPP problem, as we defined in Section 2.2 as

$$\mathcal{T}^* = \arg \max_{\mathcal{T} \in \mathbb{T}} I(\mathcal{T}) \text{ s.t. } C(\mathcal{T}) \leq B.$$

We let $\mathcal{T}$ represent the sensor trajectory within the set of feasible trajectories $\mathbb{T}$, and $C(\mathcal{T})$ be the cost function for a given trajectory. An example of potential costs would be trajectory length, energy usage, or time and would not exceed the budget constraint B . Lastly, $I(\mathcal{T})$ is the reward or information gain from a trajectory, and an optimal trajectory $\mathcal{T}^*$ is found that maximizes the information gain and does not exceed the budget B .

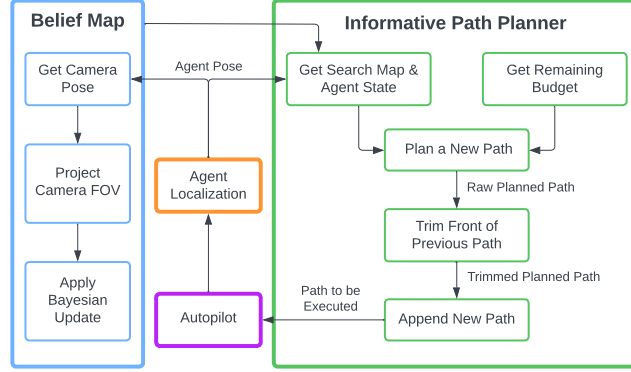


Figure 4.2: Overview of the IA-TIGRIS planning framework. As the robot is executing the current path, the belief map is updated using the current robot position. Whenever a new plan is initiated, the current robot position, belief map, and remaining budget are passed to the planner. The new path is merged into the previous path and given to the autopilot for execution.

4.4 Proposed Approach

This section details our proposed approach for informative path planning online using incremental and adaptive methods. A high level overview of our framework is shown in Fig. 4.2. Subsection 4.4.1 provides a thorough explanation of the core algorithm, including the main planning loop and the graph update procedure, elucidating the steps involved in sampling, steering, and constructing the tree of potential paths. Subsection 4.4.2 discusses the strategy to enable adaptive replanning with the presented algorithm. Finally, Subsection 4.4.3 explores the techniques used to incrementally update the path, enhancing the efficiency and responsiveness of the planning process. Subsections 4.4.4 and 4.4.5 explain how the belief space and observation model integrate into the planner, and Subsection 4.4.6 presents our reward models.

4.4.1 Algorithm Overview

The IA-TIGRIS algorithm comprises two main components: the main planning loop (Algorithm 3) and the graph update procedure (Algorithm 4). Let the notation $X \leftarrow^+ \{\mathbf{x}\}$ and $X \leftarrow^- \{\mathbf{x}\}$ represent the operations $X \leftarrow X \cup \{\mathbf{x}\}$ and $X \leftarrow X \setminus \{\mathbf{x}\}$ respectively. The IA-TIGRIS planning loop takes as input the agent start pose, x_{start} , the state space $\mathcal{X}$, the remaining budget B , the remaining allowed planning time T , the extend distance for tree building Δ , and the radius for neighbor selection R .

At the beginning of planning, the set of vertices V is empty. The initial information I_s at the start pose x_{start} is calculated by $I(\cdot)$, and the initial cost C is set to 0. The tuple n_{start} is initialized and added to the set of vertices V , while the set of edges E

Algorithm 3: IA-TIGRIS

Input : $\mathbf{x}_{start}$: Agent start pose
 $\mathcal{X}$: State space
 B : Budget
 T : Allowed planning time
 Δ : Extend distance for tree building
 R : Radius for neighbor selection

Output: $\mathcal{T}$: Planned path

```
1 if  $V = \emptyset$  then
2    $I_s \leftarrow I(\mathbf{x}_{start}); C \leftarrow 0$ 
3    $n_{start} \leftarrow \{\mathbf{x}_{start}, I_s, C\}$ 
4    $V \leftarrow n_{start}; E \leftarrow \emptyset; V_{closed} \leftarrow \emptyset; G \leftarrow (V, E)$ 
5 else
6    $\text{UPDATEGRAPH}(\mathbf{x}_{start}, B)$ 
7 while computation time  $< T$  do
8    $\mathbf{x}_{sample} \leftarrow \text{INFORMEDSAMPLE}(\mathcal{X})$ 
9    $n_{nearest} \leftarrow \text{NEAREST}(\mathbf{x}_{sample})$ 
10   $\mathbf{x}_{feas}, e_{feas} \leftarrow \text{STEER}(n_{nearest}, \mathbf{x}_{sample}, \Delta, X_{free})$ 
11   $\text{ADDPPOSETOGRAPH}(\mathbf{x}_{feas}, e_{feas}, n_{nearest}, \Delta, R, B)$ 
12   $N_{near} \leftarrow \text{NEAR}(\mathbf{x}_{feas}, R, V \setminus V_{closed})$ 
13  for  $n_{near} \in N_{near}$  do
14    if  $n_{near} \neq \mathbf{x}_{feas}$  and  $n_{near} \neq n_{nearest}$  then
15       $\mathbf{x}_{new}, e_{new} \leftarrow \text{STEER}(n_{near}, \mathbf{x}_{feas}, \Delta, X_{free})$ 
16      if  $\mathbf{x}_{new} \neq n_{near}$  then
17         $\text{ADDPPOSETOGRAPH}(\mathbf{x}_{new}, e_{new}, n_{near}, \Delta, R, B)$ 
18 return  $\mathcal{T} \leftarrow \text{BESTPATH}(G)$ 
```

and the closed set of vertices V_{closed} are initialized as empty. The graph G is initialized as $G \leftarrow (V, E)$.

The Algorithm 3 then loops until the computation time exceeds the allowed planning time T . During each iteration of the loop, a sample x_{sample} is drawn from the state space $\mathcal{X}$ using the INFORMEDSAMPLE function which is a weighted sampler based on the reward for viewing a state in the belief space. This is done by first conducting the weighted sample in the belief space and then uniformly sampling in the subset of the configuration space that views the sampled belief space state while respecting desired observation constraints. Informed sampling focuses paths on areas of higher reward rather than uniformly sampling in the search space.

The nearest node $n_{nearest}$ in the graph G to x_{sample} is identified, and the algorithm attempts to steer from $n_{nearest}$ to x_{sample} within a specified distance Δ while checking for collisions, resulting in a feasible pose x_{feas} and an edge e_{feas} . In contrast with TIGRIS in Chapter 3, the new pose and edge are immediately evaluated to be added to the graph G using the ADDPOSETOGRAPH function, allowing the neighbor selection

Algorithm 4: ADDPOSETOGRAPH

Input : $\mathbf{x}_{new}$: New pose
 e_{new} : Edge from n_{parent} to $\mathbf{x}_{new}$
 n_{parent} : Parent node in graph
 Δ : Extend distance for tree building
 R : Radius for neighbor selection
 B : Budget

```
1  $I_{new} \leftarrow I(\mathbf{x}_{new}, e_{new}, n_{parent})$ 
2  $C_{new} \leftarrow C_{n_{parent}} + Cost(e_{new})$ 
3  $n_{new} \leftarrow \{\mathbf{x}_{new}, I_{new}, C_{new}\}$ 
4 if not PRUNE( $n_{new}$ ) then
5    $E \leftarrow^+ \{e_{new}\}$ 
6    $V \leftarrow^+ \{n_{new}\}$ 
7    $G \leftarrow (V, E)$ 
8   if  $C_{new} = B$  then
9      $V_{closed} \leftarrow^+ \{n_{new}\}$ 
```

radius R to be less than the extend distance Δ .

The algorithm then searches for nearby nodes N_{near} within a radius R of x_{feas} that are not in the closed set V_{closed} . For each neighbor n_{near} in N_{near} , if n_{near} is distinct from x_{feas} and $n_{nearest}$, the algorithm attempts to steer from n_{near} to x_{feas} . If the new pose x_{new} is distinct from n_{near} , the pose and edge evaluated to be added to the graph G using the ADDPOSETOGRAPH function. This neighborhood expansion structure is different than TIGRIS in that it now allows for neighboring nodes to extend toward x_{feas} even if it did not add an edge connecting to $n_{nearest}$.

After the computation time is exhausted, the algorithm extracts the best path T from the graph G using the BESTPATH function. This function simply returns the path with the highest information reward in the graph.

The ADDPOSETOGRAPH function, outlined in Algorithm 4, handles the addition of a new pose x_{new} and edge e_{new} to the graph G . The information I_{new} for the new pose is calculated, and the cumulative cost C_{new} is computed by adding the cost of the edge e_{new} to the cost of n_{parent} . A new node n_{new} is created as a tuple containing the new pose, information, and cumulative cost. The node n_{new} is then checked by the PRUNE function. The pruning function checks nearby nodes within a radius to see if they contain a better solution. Because most of the nodes in the tree are not full paths, the upper bound on their rewards are compared. For more aggressive pruning, heuristics can be used for this comparison. In our case we use the heuristic of checking if a nearby node has a lower cost and higher current reward, but this heuristic can have implications on finding the optimal solution. If n_{new} passes the pruning check, the edge e_{new} and node n_{new} are added to the sets E and V , respectively, and graph

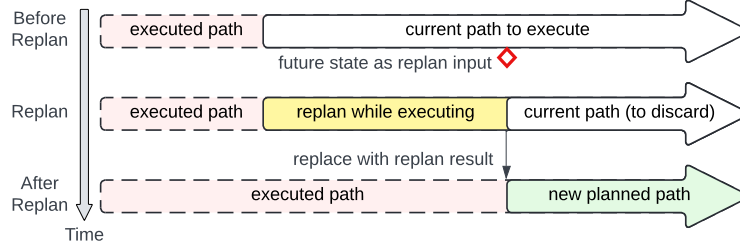


Figure 4.3: Visualization of the online replanning procedure. The future robot state is used as the starting point for the next plan. The newly-planned path is merged with the previous path.

G is updated. If the cumulative cost C_{new} equals the budget B , the new node n_{new} is added to the closed set V_{closed} .

4.4.2 Adaptive Replanning

For environments where the world is static and measurements are deterministic, global plans could be static and created once before flight. Having a single, static plan would also work well in situations when onboard computation is restricted, there is high confidence the global path is optimal, the executed path will be perfectly followed, and the predicted information gain will be close to the actual information gain. However, there are many cases where the plan would need to be adjusted due to disturbances and modified due to new information. When disturbances cause deviations from an expected observation, the path can be replanned during path execution to re-observe locations and adjust the path according to the current information map.

The adaptability of IA-TIGRIS is due to its ability to continuously replan the global trajectory online. The updated paths take into account the current world belief updated by all observations. This brings robustness, as executed paths may not follow the planned path due to disturbances and lead to differences in the planned versus expected observations.

The Algorithm 3 described above is a planning procedure that generates a path from the starting configuration. To replan adaptively during flight, a starting configuration is chosen that is time T in the future along the previously-planned path. Similarly, the budget B is updated based on remaining flight time based on the current battery levels or mission constraints. The new path is merged with the previous path, keeping any portion of the previous path before the starting pose of the new path. This process is visualized in Fig. 4.3.

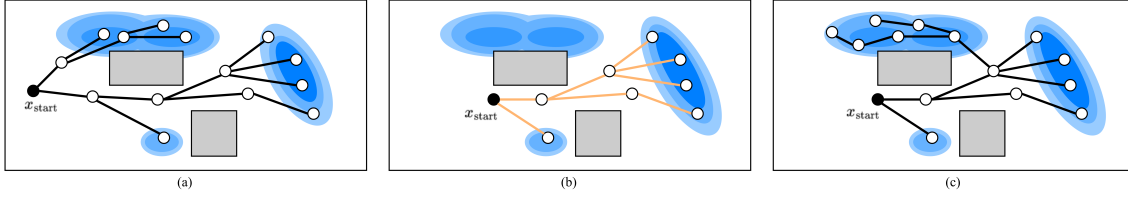


Figure 4.4: An illustration of the incremental planning and refinement process in IA-TIGRIS. (a) The global planner first generates long-horizon paths to maximize predicted information gain. (b) At the start of each new planning cycle, infeasible portions of the tree, based on the executed trajectory and the current plan, are pruned. The remaining tree is then updated by recomputing information gain and cost based on the latest belief state and available budget, which is significantly more efficient than rebuilding the tree from scratch. (c) The tree is further expanded and refined for the duration of the planning cycle, after which the process repeats.

4.4.3 Incremental Planning

When replanning online, discarding the previous planning graph wastes valuable information and forces the planner to start from scratch. By using an incremental approach, the planner can reuse the previous planning efforts, improving efficiency and enabling the next global plan to refine the earlier one. This often results in higher-quality paths. Even when the budget and belief map remain unchanged between replans, many scenarios—such as large budgets or small areas relative to the budget—can benefit from incremental refinement. In challenging planning situations, this approach allows the planner to iteratively expand the search tree, continuing to explore and evaluate potential paths.

To enable incremental planning, our method recycles as much of the previously-built graph as possible, rather than create a new graph whenever a plan is requested. Even when the belief map and budget have changed, recycling the previous graph offers significant benefits. The previous graph has already been validated for collisions and adheres to the vehicle’s motion model, allowing the planner to build upon prior work and save computational resources while maintaining accuracy.

To reuse the previous tree, Algorithm 3 first checks if the vertex set V is empty. If it is not empty, then rather than initializing the graph, it calls the function `UPDATEGRAPH`. This function first finds where the starting point for the next plan lies within the previous plan. When a matching node is found in the previous plan, all nodes and edges before that node are pruned away. The remaining portions of the graph are checked for the budget constraint and the information gain up to each node are updated based on the current information map. This is done by efficiently traversing the tree recursively from the starting node working down to all leaf nodes. The planner then uses the updated graph to plan as normal. A high-level overview of this process is shown in Fig. 4.4.

4.4.4 Belief Space Representation and Node Embedding

In informative path planning, the information within a space can be represented using various models, such as Gaussian mixture models, Gaussian process regressions, grids, or voxels. This belief space can encode different types of information. For example, it might represent the probability of an object of interest being located in specific regions of the space. The choice of belief space formulation is important, as it directly influences the planning process by guiding search efforts and behaviors. An effective representation for a specific system, application, and planner balances computational complexity, memory requirements, and the ability to perform operations on the belief model.

In this work, we adopt a grid-based representation of the belief space. Specifically, the search space is discretized into a grid of cells, where each cell i is associated with a binary random variable X_i indicating the presence ($X_i = 1$) or absence ($X_i = 0$) of an object of interest. The probability $P(X_i)$ represents our belief about an object being located in cell i . This grid-based representation is computationally straightforward to update with new or predicted observations using a Bayesian framework and an appropriate sensor model.

Belief Space Node Embedding

The nodes in the planning tree maintained by the IA-TIGRIS algorithm store the necessary information to represent the evolving belief space. The information gain up to a given node is computed based on the change in entropy within the grid cells observed by the agent’s sensor footprint during traversal from the root node to the current node.

A straight-forward approach to calculating the information gain for a given node consists of traversing the tree from the root all the way to the given node. It is necessary to evaluate the entire trajectory because the information gain can be not modular and previous observations affect the information gain of future observations. Calculating the additional information gain contributed by a new node and edge requires knowledge of the belief state of the search map at the previous node. However, this approach of calculating information gain starting at the root node is computationally inefficient and slow. Let n be the number of cells in the search map, m be the number of nodes in the tree, k be the number of nodes in a trajectory, and j be the upper bound of cells updated by a node in the tree. The temporal complexity of this method is $\mathcal{O}(kj)$ and the memory complexity is $\mathcal{O}(n)$.

One approach to speeding up computation is to store the entire search map within each node. In this approach, the information gain for a new node is simply the sum

of the parent node’s information gain and the contribution of the new node. This method results in a temporal complexity of $\mathcal{O}(j)$ for calculating the information gain of a new node and a memory complexity of $\mathcal{O}(nm)$. While feasible for small maps, this method becomes impractical for large-scale maps relevant to the domains and applications considered in this work due to its high memory requirements.

To efficiently compute the information gain for a new node, we introduce a node embedding strategy that eliminates the need for a full traversal of the planning tree from the root to the current node. Instead of storing the complete belief state in each node, we use a sparse representation that records only belief updates. A hash map efficiently tracks modifications to belief probabilities, reducing memory complexity while preserving the ability to compute information gain incrementally. This allows for efficient calculations of the information gain for a new node while maintaining computational efficiency without sacrificing accuracy.

The temporal complexity of calculating the information gain for a new node is now $\mathcal{O}(j)$, and the overall memory complexity of this belief space representation is only $\mathcal{O}(jm + n)$. As long as $j \ll n$, this approach is much more memory-efficient than the naive approach of storing the complete belief state in each node. Each node in our tree now contains the cumulative information gain along the path, current state of the agent, path cost, and edge path.

While implementing this embedded method, we observed the significant impact on computation time from selecting efficient keys for the hashmap. Using strings as keys resulted in substantial overhead due to frequent string-to-integer conversions. Even transitioning from a two-integer key to a single-integer key yielded a notable improvement in performance and is the approach we use in our implementation through a simple index-to-key conversion. Given that cell value lookup operation occurs frequently, optimizing its efficiency was highly impactful for accelerating the expansion rate of the search tree.

By incorporating this belief space node embedding strategy, our planning framework efficiently maintains an accurate representation of the evolving belief space within the search tree. This enables the planner to greatly reduce computational overhead and create higher quality paths.

4.4.5 Sensor Modeling

The updates to our belief map are dependent on the sensor model, which captures the performance characteristics of the perception system employed. There are many environmental factors that affect the performance of a sensor, such as visibility and range, which can be incorporated into this model.

For our approach, we formulate the sensor model as a piecewise function and is implemented as a lookup table. For a given range r and binary measurement Z , this table maps the range of the object from the sensor to the corresponding true positive rate (TPR) $P(Z|X, r)$ and true negative rate (TNR) $P(\bar{Z}|\bar{X}, r)$ of an object detection model. A typical range-based detection model will have better performance at close ranges and then tapers off as the distance increases.

By maintaining a lookup table of these sensor models, we can easily incorporate different sensor modalities and their corresponding capabilities into the planning framework. The sensor model can be derived from actual test data and easily integrated into the framework rather than having to fit a function to the results.

The sensor model for belief map updates is a crucial component of the overall planning framework, as it directly affects the information gain computed for each node in the IA-TIGRIS tree. Accurately modeling the sensor’s performance characteristics enables the planner to make informed decisions, resulting in more effective exploration and search strategies that align with actual sensor capabilities. Conversely, inaccuracies in the sensor model can lead to overconfidence in observed areas or unnecessary revisits to regions that have already been sufficiently searched.

Multiple Sensor Models

Through our work with project sponsors, we found that sensor models can be influenced by a range of factors depending on the use case. While our models primarily account for range, other factors, such as terrain type or object class, can also play a significant role. To incorporate this variability, we enabled per-grid-cell specification of the sensor model, allowing users to tailor priors based on domain expertise or available prior information.

4.4.6 Information Reward Models

The information reward function $I(\cdot)$ used in our framework is not constrained to being modular and could be time-varying or even submodular, depending on the specific requirements of the application. In this work, we have implemented an objective of reducing the overall entropy of the belief space. As defined in Section 4.4.4, our belief space represents the probability of the presence of an object of interest in a cell. The reward for one cell $I(X_i)$ is calculated by finding the difference between the initial entropy and the final entropy after updating the belief with the measurements and sensor model defined in Section 4.4.5.

The Shannon entropy of a cell is calculated by

$$H(X_i) = -P(X_i) \log P(X_i) - P(\bar{X}_i) \log P(\bar{X}_i) \quad (4.1)$$

where X_i represents the event that the cell is occupied and $\bar{X}_i$ denotes its complement, the event that the cell is free.

The entropy reduction due to a single positive measurement would be calculated as

$$\Delta H(X_i|Z) = H(X_i) - H(X_i|Z)$$

where $H(X_i)$ is given by (4.1), and $H(X_i|Z)$ is the entropy conditioned on the sensor measurement Z .

The final information gain for a trajectory $I(\mathcal{T})$ would be

$$I(\mathcal{T}) = \sum_i^I H(X_{i,init}) - H(X_{i,final})$$

where $X_{i,init}$ is the initial belief state of cell i , and $X_{i,final}$ is the final belief state.

Information Reward Estimation

When adding a new node in the planning tree, the approach for estimating the information gain may vary depending on the modularity and temporal dependence of the information function. In the case of a modular and temporally independent function, the reward for a new node can be simply appended to the cumulative reward along the path. Alternatively, if the information function exhibits non-modularity or temporal dependence, the entire trajectory up to the current node may need to be re-evaluated to determine the appropriate reward. Because we do not yet know the future measurements for each cell, the information gain for the planned path must be estimated.

One option would be to compute the expectation of the information gain given all possible future measurements. However, this can be computationally expensive and infeasible. We avoid this by following the approach of [Hollinger \[2015\]](#) and TIGRIS and use an optimistic approximation of the expected reward. Specifically, we assume a positive measurement Z if $P(X) \geq 0.5$, and a negative measurement otherwise. This leads to the following cell reward function for a single measurement:

$$I(X_i) = \begin{cases} \Delta H(X_i|Z) & P(X_i) \geq 0.5 \\ \Delta H(X_i|\bar{Z}) & P(X_i) < 0.5 \end{cases} \quad (4.2)$$

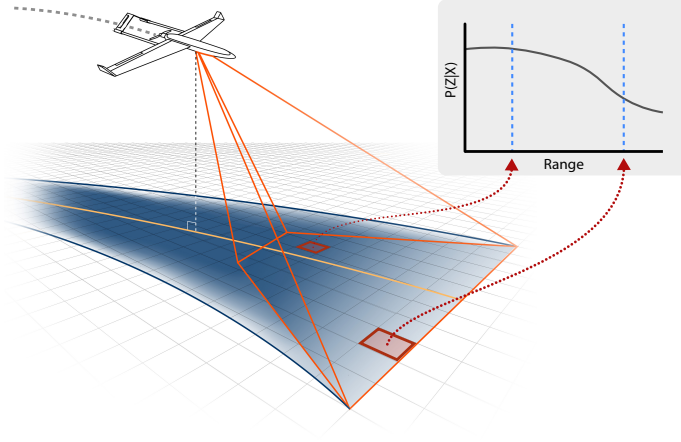


Figure 4.5: An example of how the grid cells rewards are estimated given a planned trajectory. The cells that are closer to the center of the trajectory have a smaller minimum distance than cells farther from the center of the trajectory, leading to a larger change in entropy for the closer cells.

To compute the information reward along a path, the sensor footprint is projected onto the surface plane to determine all the cells it contains. The reward for each cell is calculated using (4.2) and then aggregated. This process becomes more intricate when incorporating graph edges into the reward computation. Discretizing the trajectory between nodes into individual views creates imperfect overlaps due to the non-square sensor footprint, potentially omitting some grid cells from the edge reward calculation. Mitigating this issue requires significant overlap between sensor footprints, which, in turn, greatly increases the computation time for updating grid cell beliefs. Rather than selecting a discretization resolution for the trajectory, we compute a conservative approximation of the edge reward by updating all cells within the view of edge trajectory, using the closest viewing distance along the path for each cell. This approach ensures that all cells along the edge are accounted for, with each belief value updated using a measurement taken when the sensor is closest to that cell. A visualization of this reward estimation is shown in Fig. 4.5.

Priority-dependent Rewards

In the case where certain areas of an environment or spatial information are more important than others, we introduced the ability to weight the information reward by a priority scalar, p_i , representing the priority of cell i . This approach provides finer control over the UAV behavior by allowing the entropy of different cells to be weighted unequally. Assigning a higher priority to one prior over another results in paths that favor observations of the higher-weighted prior. For example, this is useful when one type of object is more important than another, and the prior spatial distributions are concentrated in distinct areas of the search space (as demonstrated in Section 4.5.3).

Time-dependent Rewards

The IPP formulation maximizes total information gain over the entire mission without explicitly relating the reward to the timing of observations—rewards are time-independent. However, in some applications, operators may prefer prioritizing areas of high information gain earlier in the trajectory rather than later.

We introduce the ability to create time-dependent reward that decreases the reward value for all cells over the mission duration, resulting in plans that prioritize visits to areas of high information gain early in the plan. This works especially well when paired with priority-dependent rewards, as introduced in Section 4.4.6. The UAV will be biased toward visiting the higher priority priors early in the plan, and the operator can even roughly dictate the order to visit areas through differing priority weights and using the time-dependent reward decay function.

In our use cases, we implement a simple reward decay function

$$\Gamma(t) = \begin{cases} \gamma & t \geq \gamma_t \\ \beta t + 1 & t < \gamma_t \end{cases}$$

where $\Gamma(t) \in [\gamma, 1]$ and γ is the floor value of the decay function. β is the decay rate and $\gamma_t = (\gamma - 1)/\beta$ is the time where the function output becomes the floor value γ . This gives a linear decay until the floor value.

Combining our decay function $\Gamma(t)$ and cell priority values p_i creates a priority and time dependent reward function

$$I(X_i, t) = \begin{cases} p_i \Gamma(t) \Delta H(X_i | Z_i) & P(X_i) \geq 0.5 \\ p_i \Gamma(t) \Delta H(X_i | \bar{Z}_i) & P(X_i) < 0.5 \end{cases}$$

4.5 Simulation Evaluation & Results

To rigorously evaluate the performance and benefits of IA-TIGRIS, we first conduct tests in a simulation environment. These experiments include ablations to assess the impact of key design decisions, as well as Monte Carlo evaluations comparing IA-TIGRIS against baseline methods.

4.5.1 Simulation Framework

We use a simplified simulator, similar to the one used in Chapter 3 and shown in Fig. 4.6, to enable rapid development and validation of our approach. The simulation

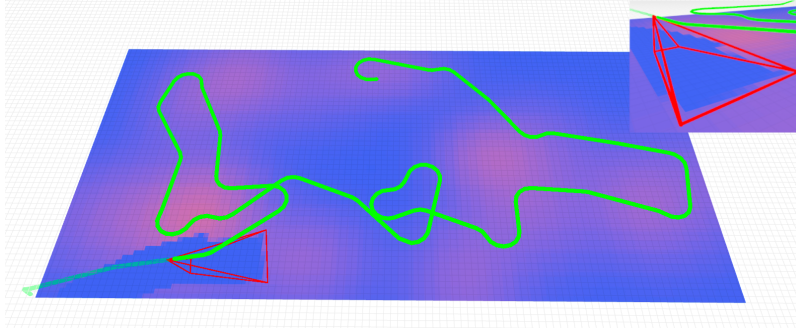


Figure 4.6: Example results for IA-TIGRIS after a replanning step in our simulation environment. Pink regions represent regions of high entropy while blue regions are areas of low entropy. The red projection shows the camera frustum while the green line represents the planned path.

employs a constant-velocity motion model for the fixed-wing UAV. To account for the effects of banking on observations, we compute the orientation change between consecutive waypoints. If this change exceeds a threshold, the UAV is considered to be banking, and the belief space is not updated during this period. This reflects the fact that the camera is primarily pointed toward the sky during banking and is also affected by increased motion blur.

When the UAV reaches a commanded waypoint within a specified Euclidean distance threshold, the next waypoint in the list becomes the new commanded waypoint. During replanning, a newly generated path is merged with the existing path, incorporating the updated trajectory starting from the beginning of the new path. A simple proportional (P) controller is used to regulate heading and altitude, guiding the UAV between waypoints.

The UAV commanded speed is 25 m/s, with a planning time of 10 seconds and a the flight altitude of 50 m. The turning radius is 100 m, and the onboard camera is pitched 30 degrees from horizontal with a 36.9-degree field of view. The belief space grid cells are 30 m, and the total budget for flight distance is 15,000 m. To accelerate testing, the simulation runs at twice the real-time speed, with results subsequently rescaled to match real-time conditions.

Fig. 4.7 presents four snapshots from a simulation. The planned path is shown in green, the executed path in yellow, the robot’s current position as an orange diamond, and the starting position as a red circle. Early in the simulation, the planned trajectory makes two passes over the larger clusters of high-entropy regions, which represent areas of high uncertainty. The search tree explores a wide range of paths across these regions to maximize information gain, refining its evaluation over time. As the planning cycle progresses, the search tree is iteratively updated through tree recycling, which improves computational efficiency by retaining and modifying previous solutions rather than replanning from scratch. At $t = 180$, the plan adapts by doubling back to

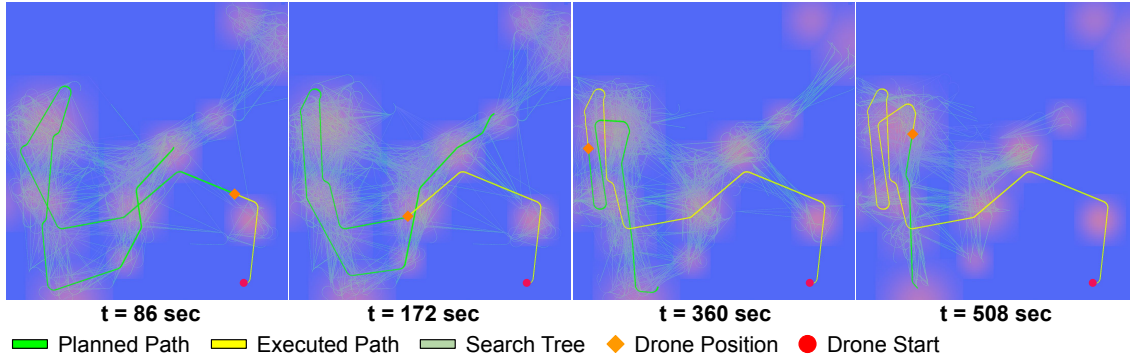


Figure 4.7: IA-TIGRIS continuously replanning and refining the global path over the duration of the simulation. These snapshots show how the path is refined online to maximize information gain. The search tree is also visualized and shows how the plans focus on the areas of high information gain and take into account the remaining budget.

the middle-left region, due to finding a more rewarding path. Toward the end of the simulation, the effect of the budget constraint becomes apparent, as the search tree no longer extends into the upper-right corner of the search area.

4.5.2 Metrics

The primary metric for evaluating the success of our plan is the information gain along the executed path, quantified as the change in total entropy of the belief space from the initial to the final state. To facilitate interpretation and comparison across tests, we report this as the percent reduction in entropy. Initially, the entropy reduction is zero, as no observations have been made. As the robot gathers information, the entropy decreases, reflecting an improved understanding of the environment. This reduction continues to accumulate throughout the simulation as new observations refine the information map.

4.5.3 IA-TIGRIS Ablations

We conduct ablation experiments to evaluate the impact of incremental and adaptive planning, belief space node embeddings, priority and time dependent rewards, and planner parameters.

Incremental and Adaptive Planning Evaluation

We first evaluate the efficiency of recycling a search tree compared to building a new one for each planning cycle. For IA-TIGRIS to be both adaptive and incremental,

reusing and recycling previous planning efforts must be computationally efficient. As explained in Section 4.4.3, each planning begins by pruning irrelevant trajectories, checking budget constraints for remaining nodes, and updating information gain based on the current belief map. If these updates consume too much planning time, there will be insufficient time left to refine and extend the trajectory.

Our implementation of IA-TIGRIS demonstrates that recycling a previously built tree is significantly more efficient than constructing a new tree from scratch. For example, a tree that takes 10 seconds to build requires less than 0.2 seconds to update under our planner settings and test environment. This efficiency gain allows the entire tree to be recycled and extended with the remaining planning time, even if none of the previous plan was executed. By avoiding the computational overhead of trajectory generation, neighborhood search, and collision checking, tree recycling results in faster updates and enables the planner to generate improved paths within the same time constraints.

To quantify the impact of tree recycling and replanning, we conducted 50 tests in the simulation framework described in Section 4.5.1. Without replanning, the average reduction in entropy was 20.4%. Introducing replanning without tree recycling improved this to 23.1%, a 13.2% increase in performance. Even in the absence of environmental disturbances, replanning improves path quality by addressing mismatches between estimated and actual information gain, particularly as the planning horizon shortens due to decreasing budget. Incorporating incremental planning with tree recycling further increased the reduction in entropy to 25.4%, a 24.5% improvement compared to replanning without tree recycling. This highlights IA-TIGRIS’s to refine and improve the previous global plan.

As expected, the benefits of replanning and refinement become more pronounced when planning time is reduced. When the allowed planning time is reduced to 3 seconds, the performance difference between no replanning and replanning with tree recycling increases to 25.8%. This gap would widen further for even shorter planning times, where initial plans tend to be of lower quality. However, as planning time decreases, a larger portion of the time is allocated to recycling previous plans rather than expanding the tree, leaving less time for refinement. This introduces a tradeoff between path quality and responsiveness; more frequent replanning improves adaptability to new information, while longer planning cycles allow for more extensive trajectory refinement.

Belief Space Node Embedding Analysis

We evaluated the impact of belief space embedding on the efficiency of our planning. Calculating the information gain for a trajectory can take up a disproportionate amount

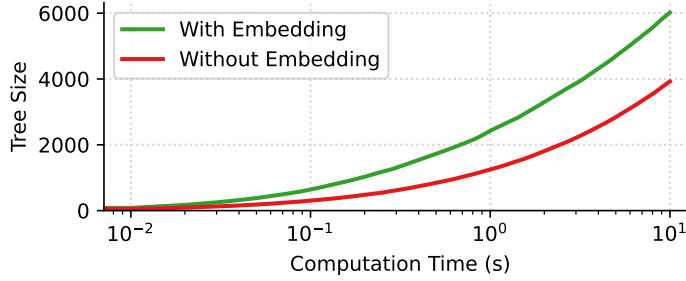


Figure 4.8: Tree size vs computation time for 100 trials.

of the planning time if not implemented well. We found that in our typical testing environments, using our belief space embedding in the tree nodes reduced the time to compute the information gain by around 80-95%. This increase in speed only resulted in an additional ~ 20 MB of memory usage for the search tree.

Fig. 4.8 shows the tree size of the first plan vs time for 100 trials on random simulation environments. Due to the reduction in time to add each new node to the planning tree, the tree size is much larger at any point during planning.

Our embedding also has a profound impact on the speed of recycling the search tree between plans. Previously the information gain at each node would be calculated individually when recycling the tree, with the nodes toward the base of the tree getting evaluated many times. Now, the tree can be updated recursively from the root to the leaves without evaluating the information at any node more than once. When tested in our typical environments, this cut down the time to rebuild a tree from $\sim 180.93 \mu\text{s}$ per node to $\sim 15.48 \mu\text{s}$ per node.

We also evaluate the impact of belief space embedding by using Google Benchmark [Google, 2024] on an extremely large environment. For these benchmark tests, the number of nodes in the trajectory was $k = 7$, the number of grid cells in the search space was $n = 39,841,344$, and the number of grid cells updated by the new node was $j = 11,984$. The time to evaluate the information gain for a new node decreased from 43 ms to 1.9 ms, a 96% improvement. This led to the size of the search trees to on average be more than double in size.

Priority and Time Dependent Rewards Evaluation

To illustrate the impact of priority-dependent and time-dependent rewards, we conduct a simulation with the results shown in Fig. 4.9. To make the effect of flight paths more clear, the camera is pitched down so that the flight path coincides more closely with what is observed by the sensor. The example scenario has three clusters of uncertainty in the search space.

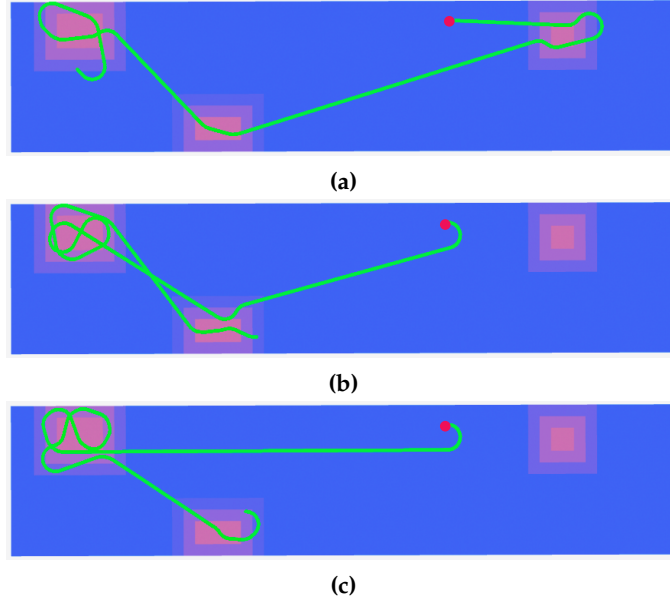


Figure 4.9: Examples showing how modifying the priority and time-based rewards effects behavior. The starting point is shown in red and the planned path in green. (a) The base case with equal priorities. (b) A higher priority on the far left prior causes the drone to spend more of its budget observing that space. (c) With time-based rewards, the drone visits the high priority prior as quickly as possible to maximize reward.

Fig. 4.9(a) shows the base case where three areas of uncertainty have equal priority. The planned path visits the closest cluster first and then travels to the left side, viewing all three clusters. Fig. 4.9(b) demonstrates when the cluster on the far left is adjusted to have a higher priority. The path no longer visits the cluster on the right in favor of saving the budget to expend more effort in searching the higher-priority region. When adding a time-based reward along with the priority, Fig. 4.9(c) shows how the path first visits the high-priority prior before moving on to the other prior on the left. The time-based reward causes the order of observations to affect the total reward.

Planner Global Parameter Search

We investigate the effects of the extend distance Δ , the near radius R used in $\text{NEAR}(\cdot)$, and the pruning radius used in $\text{PRUNE}(\cdot)$ within IA-TIGRIS, as outlined in Section 4.4.1. To do this, we perform a global parameter search across 25 randomly generated environments. Each environment was $5 \text{ km} \times 5 \text{ km}$, with a prior belief map consisting of a random number of Gaussian distributions with varying locations, means, and standard deviations. The planning loop was given a 10-second planning time, and the agent continued to observe the space until budget was expended.

Using percent reduction in entropy as the performance metric, we visualize the

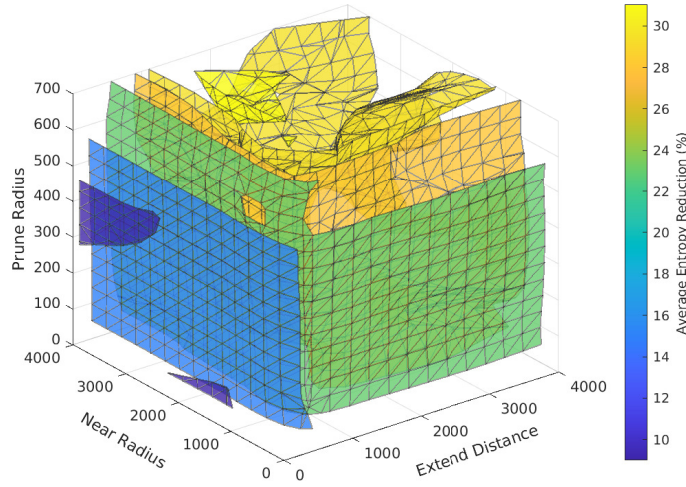


Figure 4.10: An isosurface plot showing the results of the global parameter search. Each surface shows given parameters that result in similar average percent of entropy reduction over a set of tests.

results in an isosurface plot, shown in Fig. 4.10, where higher percentages indicate greater information gain. The results suggest that, for the sampled environments, planner performance improves as the extend distance and near radius are larger. If one is much smaller than the other, performance greatly reduced. However, the relationship between pruning radius and performance is less clear, though a slight performance increase is observed as the pruning radius grows.

The pruning radius primarily affects the planned path length at the start of a test. A very small pruning radius results in paths shorter than the 15 km budget, likely because a denser tree limits tree depth. Similarly, reducing the extend distance below 500 m leads to shorter initial plans, suggesting that both parameters play a role in determining tree expansion.

We find that as the extend distance increases, maintaining a sufficiently large near radius is crucial for sustaining performance. If the near radius drops below the extend distance, performance declines. Meanwhile, the pruning radius can be set relatively high without significant performance loss, and doing so allows for longer planned paths. Additionally, the CPU performance of a given system may influence the optimal parameter settings, suggesting the need for adjusting parameters based on computational resources.

The parameter with the highest performance in our testing had the combination of an extend distance and near radius that balanced expansion while still creating a dense tree, and a large pruning radius to act as a stronger heuristic for high-quality paths. For our subsequent simulation testing, we selected an extend distance of 1500 m, near radius of 1500 m, and a pruning radius of 600 m.

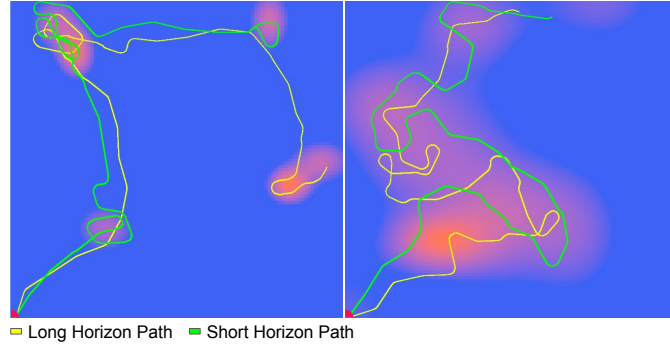


Figure 4.11: A comparison of IA-TIGRIS when planning over a long horizon versus a short horizon. Long-horizon planning enables IA-TIGRIS to allocate the entirety of its budget efficiently and is able to view all of the high-entropy clusters in the example on the left. In the example on the right, short-horizon planning leads to less smooth paths with slightly higher information gain, as the high-entropy areas are connected and have lower entropy the farther they are from the starting position.

Long-Horizon Planning

To demonstrate the effects of long and short horizon planning, we ran a simulation on two different maps with varying information distributions as seen in Fig. 4.11, where the pink region have high entropy and blue regions have low entropy. The environment on the left side of the figure shows a distribution with clusters of information scattered across the space while the environment on the right shows a denser distribution where the various areas of high entropy overlap. For short-horizon planning, IA-TIGRIS considers a planning horizon of up to 5000 m, rather than planning over the entirety of the remaining budget. The long-horizon planning starts with the entire budget of 15000 m and uses the entire remaining budget for every new plan in the planning process.

For the clustered information map on the left, the long horizon path search over more clusters and achieves a higher overall information gain compared to the short horizon path. This is because the short-horizon planner takes myopic actions, consuming the majority of its budget on the first two clusters. By the time it reaches the third cluster, it exhausts its budget and is unable to reach the last cluster. In contrast, the long horizon planner strategically balances its budget across all four clusters, enabling higher information gain.

For the dense information map on the right, the long-horizon planner generates a smoother path and extends farther into the space compared to the short-horizon planner. However, the long horizon planner allocates a smaller portion of its budget on the highest information gain area near the start position. As a result, the short-horizon planner achieves a slightly better total information gain by prioritizing immediate rewards and concentrating its budget on the highest information cluster close to the

starting position.

From these two maps, we observe that the long-horizon planner excels in environments with clustered information where reasoning over long horizons is essential to avoid myopic choices and balance the allocation of budget. The short-horizon planner does perform well in environments with densely distributed information, where following local gradients of information gain can lead to effective solutions. These findings suggest that the impact of the planning horizon on system performance can be environment-dependent. In our testing framework and applications, planning over the entire budget typically yields the best results, as our prior information maps often feature clustered regions of high information gain.

Effect of Problem Configuration on Paths

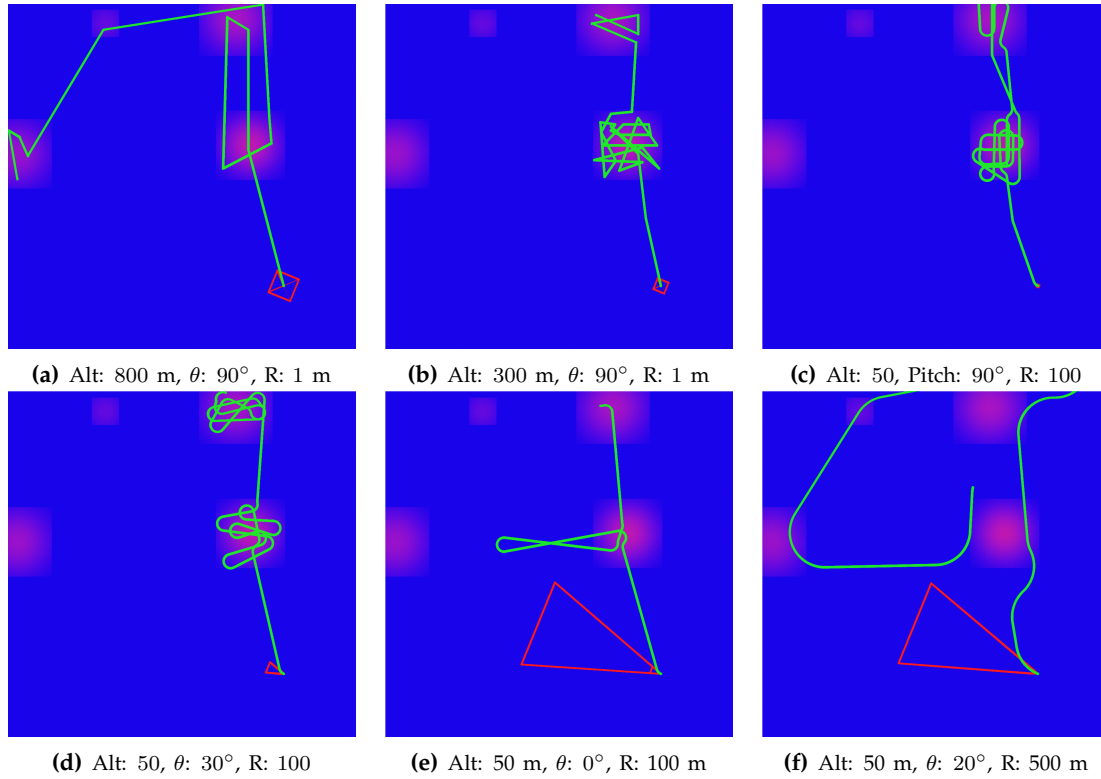


Figure 4.12: Paths from varying the altitude, camera pitch (θ), turning radius (R), and sensor model.

To illustrate the flexibility of our planner to different robot motion models, plan configurations, sensor models, and sensor configurations, and show the effects on the resulting plans in Fig. 4.12. This environment is 5 km square and the UAV is given a budget of 15 km. We evaluate configurations such as a quadrotor vs fixed-wings (varying the turning radius), camera pitched forward vs down, high altitude vs low

altitude, and a sensor requiring multiple vs single observations. The visualized planned paths in green are just the initial plan before execution and would be expected to improve as the paths are refined and remaining budget decreases. In the randomly sampled belief space, we see four Gaussians of higher uncertainty, with all four being in the upper half of the search space. The two highest-uncertainty Gaussians are on the right, a medium-uncertainty Gaussian is on the left, and a small, low-uncertainty Gaussian is located in the top-left corner. The agent starts in the bottom right, and the camera footprint is visualized in red.

In Fig. 4.12(a), we show a simulated quadrotor flying at a high altitude with its camera oriented downward. The planned path spans all four prior distributions, with paths spaced to provide effective coverage given the sensor’s footprint width. In Fig. 4.12(b), the quadrotor flies at a lower altitude, resulting in a smaller sensor footprint, and has a sensor model with a lower TPR and TNR. The planner responds by allocating the available budget to focus on the two priors with the highest uncertainty. Fig. 4.12(c) shows the case of an even lower altitude and a fixed-wing UAV with a turning radius of 100 m, where the sensor footprint is only slightly larger than the individual grid cells in the belief map. Because cell beliefs are only updated when fully contained within the sensor footprint, the resulting paths align orthogonally to the orientation of the belief maps. This behavior highlights the planner’s ability to reason explicitly about sensor geometry and its effect on belief updates.

In Fig. 4.12(d) we simulate the fixed-wing UAV flying at 50 m but now the camera is pitched down 30 degrees. The sensor is now able to see more and is able to cover the two largest priors better than the previous example. In Fig. 4.12(e) we test with zero pitch on the camera and cut off the sensor footprint at 2 km. This creates a much different path than before. We now can no longer use the planned path as a proxy for roughly what will be seen, as all of the sensed area is not even underneath the location of the UAV. Though this path may seem not effective, in fact *it observes every prior fully*, and is able to do so with less than 8 km of budget. Most of our testing on fixed-wing UAVs have results similar to this example where it is challenging to evaluate the quality of a path by just looking at where the UAV traveled; where the UAV is located does not coincide with information being gathered in the same place. Fig. 4.12(f) is at the same altitude as the previous test, but has the camera pitched down 20 degrees and the vehicle has a turning radius of 500 m, forcing the UAV to make paths that are more sweeping in nature rather than observing each prior before moving on.

As an example comparison, Fig. 4.13 shows an ergodic coverage path generated for the same prior belief map. Ergodic coverage aims to guide the UAV to spend time in different regions in proportion to their expected information content. Trajectories are typically computed using trajectory optimization methods such as gradient descent,

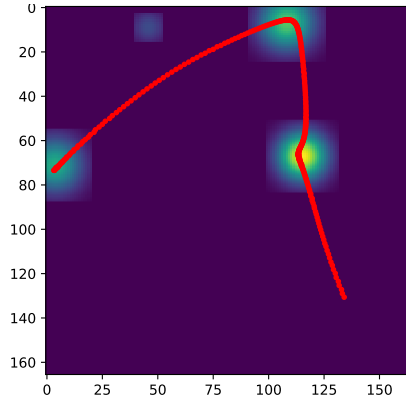


Figure 4.13: A trajectory derived from an ergodic coverage objective.

starting from an initial trajectory guess. In the figure, we observe that the UAV slows down over the centers of the priors to dwell longer in high-uncertainty areas. The ergodic objective promotes a balance between exploration and exploitation through a spectral decomposition of the belief space, weighting different spatial frequencies based on their relative importance and a hand-tuned weighting function.

In contrast, the informative path planning approach used here **directly maximizes the expected information gain as the planning objective**. Rather than matching a spatial distribution of expected information, the planner reasons explicitly about how specific actions—given the camera configuration, sensor model, flight altitude, and sensor field of view—will contribute to belief updates. This allows for paths that more tightly align with the sensing task’s desired outcomes. As shown previously, accounting for sensor characteristics and environmental factors can significantly impact the effectiveness and efficiency of the resulting trajectories.

While ergodic coverage offers a principled way to promote broad exploration and coverage of information-rich areas, it does so indirectly by encouraging time allocation patterns rather than maximizing information gain itself. Additionally, ergodic coverage typically abstracts away detailed sensor and vehicle models, which can limit its ability to fine-tune behavior for specific sensing tasks. In applications where accurate modeling of the sensing process is critical to mission success, explicitly optimizing for information gain may provide greater flexibility, adaptability, and overall performance.²

²While ergodic coverage provides a principled framework for balancing exploration and exploitation, it has several limitations for information gathering tasks. It typically assumes a linear relationship between time spent in a region and information gain, rather than modeling discrete observations with diminishing returns. As a result, slowing down does not necessarily yield more independent or useful measurements. The sensor model, such as footprint, field of view, and orientation, is also abstracted away, which can lead to inefficiencies when a single observation suffices or when sensing is highly directional. Ergodic coverage guides motion using an expected information density as a proxy rather than modeling the sensing process directly. The balance between exploration and exploitation is tuned through spectral weighting, which

4.5.4 Baseline Planners

To evaluate the performance of IA-TIGRIS, we compare it against several baseline methods. In the following section, we describe these baselines, their implementation details, and their respective advantages and limitations, particularly in the context of information gathering in large, high-dimensional search spaces. The simulation framework and vehicle parameters remain consistent across all planners, and each method is allowed to replan during testing.

Monte-Carlo Tree Search

Monte Carlo Tree Search (MCTS) can be a powerful technique for finding feasible and optimal paths in complex environments. It is a heuristic search algorithm that builds a search tree incrementally through repeated simulations. At each iteration, it selects a node to explore based on a selection policy (often the Upper Confidence Bound or UCB1 algorithm), expands the tree by adding possible actions from that node, runs a simulation from the newly added node, and updates the statistics of nodes along the path traversed during the simulation.

The UCB1 (Upper Confidence Bound) algorithm is a technique commonly used in the context of multi-armed bandit problems and Monte Carlo Tree Search (MCTS) for balancing exploration and exploitation. It helps in selecting actions or nodes that are likely to yield high rewards while also exploring less-frequented options to gather more information about their potential rewards.

We formulate our UCB score in the following manner:

$$UCB_{\text{node}} = \frac{I(X_{\text{node}})}{\alpha} + C \times \sqrt{\frac{\ln(N_{\text{tree}})}{N_{\text{node}}}}$$

Here $I(X_{\text{node}})$ denotes the estimated information gain from the node, α denotes the normalization factor which is given by $\frac{B}{v_{\text{desired}}}$, B being the maximum planning budget and v_{desired} being the desired speed of our UAV. C denotes the exploration weight, and N_{tree} denotes the number of visits to the tree root node while N_{node} denotes the number of times the present node has been visited.

may not correspond intuitively to sensing performance. Moreover, spectral approximations inherently spread observations across the space, which can be inefficient under tight budgets. Trajectory optimization methods like gradient descent are sensitive to initial conditions and prone to local minima [Miller et al., 2016], and handling obstacles often requires relaxing ergodicity constraints and introducing additional mechanisms such as control barrier functions. Overall, when detailed sensor modeling and efficient use of limited budgets are critical, directly optimizing expected information gain may offer greater flexibility and effectiveness.

After selecting a candidate node, if it has been visited before, it is expanded by applying motion primitives to generate child nodes, growing the tree. Unvisited nodes skip this step. Following expansion, either the unvisited candidate node or one of its children is selected for the simulation phase, where the future values of nodes along the path are estimated to update the total potential information gain. This informs the selection policy in subsequent iterations. Once planning time is exhausted, the path with the highest information gain is returned.

While MCTS is probabilistically guaranteed to converge to the optimal path [Dam et al., 2022], it is constrained to actions within a predefined set of motion primitives. Its reliance on random sampling to estimate the future value of nodes can result in poor approximations, particularly in environments with sparse, localized pockets of high information gain. This limitation is especially pronounced in large search areas or scenarios with large budgets constraints, where estimating future node values becomes increasingly expensive. As a result, in such scenarios, MCTS is often implemented with a finite planning horizon, which can restrict its ability to account for long-term consequences or dependencies in the environment.

Greedy

For the greedy planner, we iterated through each cell within the search bounds and calculated the reward for a given cell i as $g_i = R(X_i)/d_i$ where $R(X_i)$ is given through (4.2) and d_i represents the Euclidean distance between the current position the robot at the current time t and the closest viewpoint to the cell. To compute this viewpoint, the yaw between the current pose of the robot and the intersected cell is first calculated. Using the robot’s sensor configuration and this yaw, x and y coordinates are calculated that view the cell at the desired flight altitude. With this formulation, the planner prioritizes regions with a high ratio of entropy to distance. This can lead to locally optimal choices that contradict with paths that lead to higher information gain over the entire trajectory.

Random

The random planner operates by iteratively sampling points within the defined search bounds and calculating the minimum-cost path to observe each sampled point. This process is repeated until the available budget is fully expended. The random planner does not utilize any prior information about the environment or target distribution. Additionally, it does not optimize the sequence of actions, instead treating each sampled point independently without considering the global structure of the search problem. This simplicity allows the random planner to highlight the performance

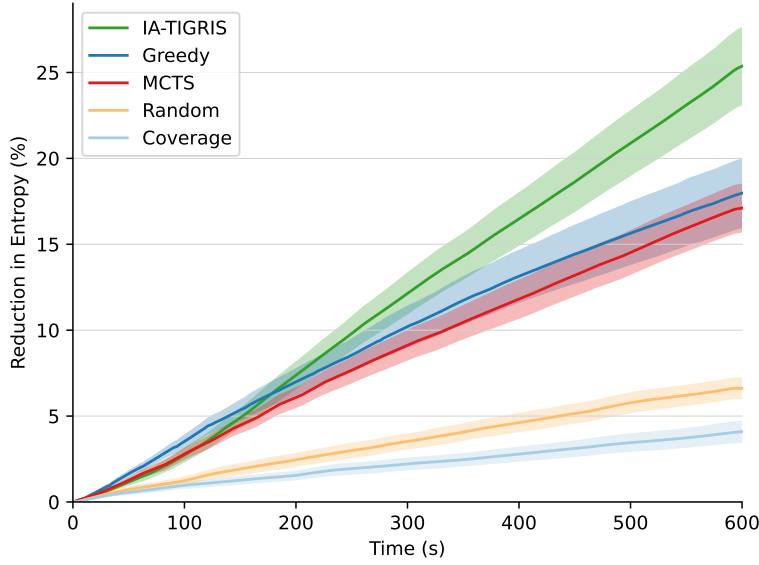


Figure 4.14: The Monte Carlo simulation results for the planners. The plots show the average percent reduction in entropy over the course of the simulations, and the shading shows the 95% confidence intervals. IA-TIGRIS outperforms all of the baselines.

benefits of more sophisticated methods by providing a lower-bound comparison for evaluation.

Coverage

The coverage planner generates a plan that systematically covers the entire search space using a straightforward lawn-mower pattern. The spacing between each pass is set to match the width of the projected observation footprint at 20% from the bottom, ensuring that no grid cells are missed. This spacing also maintains a distance that enables high-quality sensor measurements. However, due to the size of the search spaces considered, the coverage planner spends significant time surveying empty regions. This approach results in inefficient use of the budget, as it prioritizes full coverage with safe sensor overlap, even in areas with little or no valuable information. While simple and robust, this method highlights the tradeoff between exhaustive coverage and efficient, targeted exploration.

4.5.5 Tests and Analysis

To evaluate the efficacy of IA-TIGRIS against baseline methods, we perform Monte Carlo testing and analyze the impact of the prior and budget on the performance of each method. In all test cases, rewards are calculated using (4.2), and horizon lengths

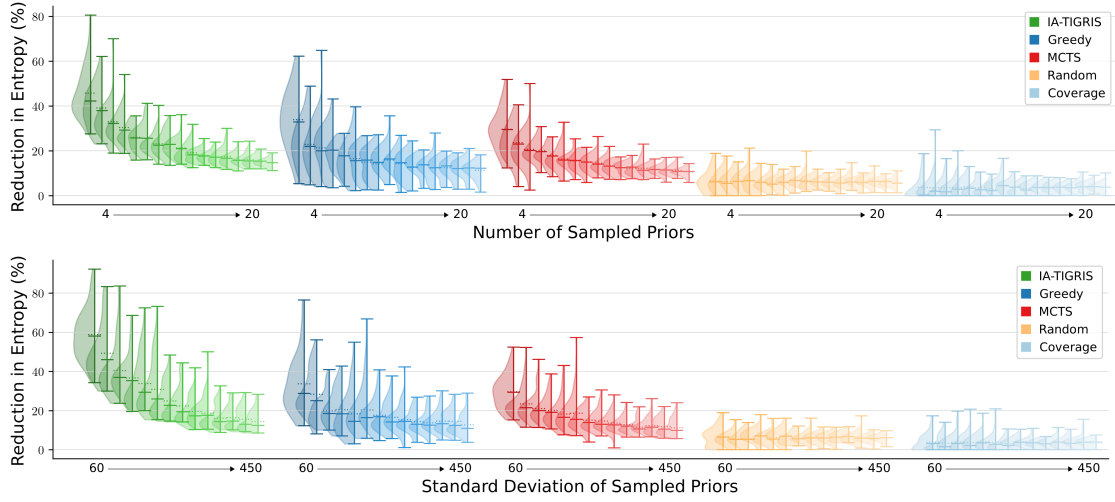


Figure 4.15: A comparison of the methods based on the number of sampled prior clusters and the standard deviation of sampled prior clusters. IA-TIGRIS is most effective compared to the baselines when there is high variation in the search space. As the search space prior information becomes more evenly spread out, the performance gap between the methods tends to decrease.

are set to match the full budget. The tests are conducted on an Intel Xeon CPU E5-2620 v4 @ 2.10GHz, ensuring consistent computational conditions across all evaluations.

Monte Carlo Testing

Our simulated testing environment is a 5000×5000 m square with Gaussian-distributed prior information randomly placed throughout the search space. The number of prior clusters was sampled uniformly between $[4, 20]$, with standard deviations between $[60, 450]$, and maximum value between $[0.05, 0.5]$.

The results of 100 Monte Carlo tests are shown in Fig. 4.14. IA-TIGRIS clearly outperforms the other method methods, achieving nearly a 40% greater reduction in entropy than the next best method. Early in the simulation, the greedy method initially gains information more quickly, as expected, but this does not translate to better long-term performance. Since our method optimizes for total information gain, it generates paths that maximize information collection over the entire budget. MCTS performed slightly worse than the greedy approach.

The random paths slightly outperformed the coverage paths. This is likely because the lawnmower strategy requires sufficient overlap between passes to avoid missing areas, and its long straight paths often lead to redundant observations due to the UAV's forward-facing camera. Changing the heading of the UAV is beneficial to viewing more of the search space, which may explain why random paths performed better.

	5000 m	10000 m	15000 m	30000 m	60000 m
IA-TIGRIS	9.41 $\pm$ 1.0	18.28 $\pm$ 1.8	25.36 $\pm$ 2.3	41.08 $\pm$ 2.9	58.85 $\pm$ 2.9
Greedy	6.99 $\pm$ 0.8	13.10 $\pm$ 1.5	17.97 $\pm$ 2.0	30.00 $\pm$ 2.3	49.38 $\pm$ 3.5
MCTS	6.06 $\pm$ 0.7	11.80 $\pm$ 1.1	17.11 $\pm$ 1.4	30.21 $\pm$ 2.2	48.68 $\pm$ 2.7
Random	2.19 $\pm$ 0.3	4.29 $\pm$ 0.7	6.61 $\pm$ 0.6	17.50 $\pm$ 1.2	22.47 $\pm$ 1.4
Coverage	1.58 $\pm$ 0.3	2.82 $\pm$ 0.4	4.09 $\pm$ 0.7	12.04 $\pm$ 1.9	16.77 $\pm$ 2.4

Table 4.1: Monte Carlo testing results given different budgets. The values are the average percent reduction in entropy and the 95% confidence bounds. IA-TIGRIS had the best performance for all budgets.

We also conducted Monte Carlo tests where either the number of prior clusters or their standard deviation was held constant to analyze how variations in the information map affect planner performance. The results, shown in Fig. 4.15, include two cases: the upper figure fixes the number of priors, while the lower figure fixes their standard deviation. All other agent and simulation parameters remained unchanged.

Across these tests, the performance gap between IA-TIGRIS and the baselines widens as the number and standard deviation of the Gaussian priors decrease. When entropy is more uniformly distributed across the search space, simpler methods perform reasonably well within the given budget. However, when information is concentrated in sparse, distinct regions, longer-horizon planning becomes essential. In such cases, IA-TIGRIS demonstrates a significant advantage by effectively reasoning about the budget and prioritizing high-value regions.

Budget Analysis

To evaluate the impact of budget constraints on performance, we conducted additional tests beyond our initial Monte Carlo experiments, evaluating budgets of 5000 m, 10000 m, 30000 m, and 60000 m. Table 4.1 summarizes the average entropy reduction across these budgets.

IA-TIGRIS consistently achieved the highest entropy reduction across all budget constraints, with a statistically significant margin over alternative methods. Greedy generally ranked second but was slightly outperformed by MCTS at the 30000 m budget level. Greedy and MCTS exhibited comparable performance throughout the tests, with their results closely tracking each other. Consistent with our previous findings, Random and Coverage methods yielded the lowest results.

Among the tested methods, only IA-TIGRIS and MCTS explicitly incorporate budget constraints into their planning algorithms. Notably, at lower budgets (5000 m and 10000 m), these methods achieved higher entropy reduction compared to the equivalent time steps (200 s and 400 s) in the 15000 m budget scenario shown in Fig. 4.14. This

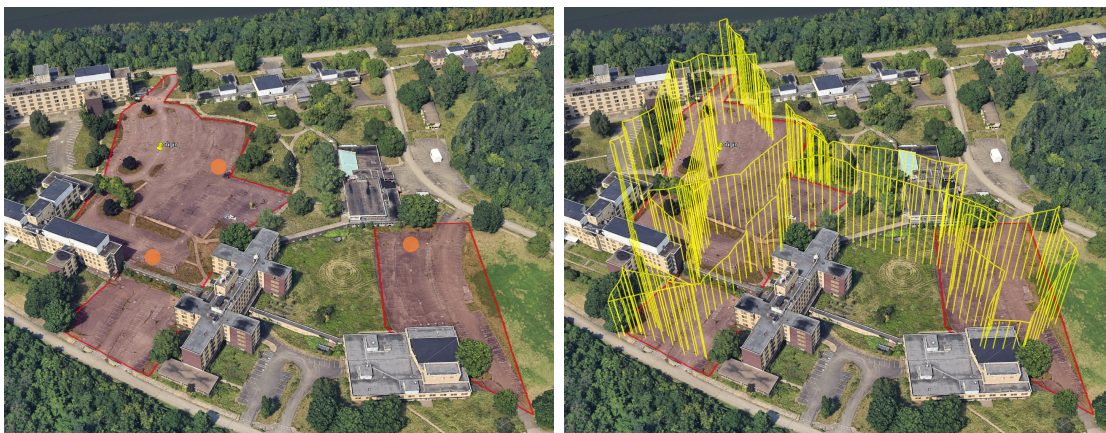


Figure 4.16: Google Earth screenshots illustrating the mission planning process and execution. Left: Areas of high entropy targeted for search are highlighted in red, representing regions with a binary occupied/unoccupied probability of 0.2. Three points of particular interest, each assigned a 0.5 probability, are marked in orange. Right: The executed drone flight path (yellow) shows the optimized path for maximum information gain across the search space.

improved performance stems from IA-TIGRIS’s optimization of total path reward under budget constraints, contrasting with the myopic next-best-action approach of the greedy method. The remaining methods—Greedy, Random, and Coverage—maintain consistent behavior regardless of budget constraints, as their planning strategies do not account for resource limitations.

The performance gap between IA-TIGRIS and the next-best method varied with budget size, showing margins of 34.6%, 39.5%, 41.1%, 36.0%, and 19.2% in ascending budget order. This gap widened through the first three budget levels as problem complexity increased, before declining significantly at higher budgets. This performance pattern suggests that implementing a planning horizon could enhance efficiency by limiting tree search depth, enabling the planner to prioritize path quality optimization over exhaustive space exploration.

4.6 Field Deployments

4.6.1 Hexarotor Deployment

The first field experiment that we present uses a hexarotor drone to cover an urban area shown in Fig. 4.1. We designed this field experiment to simulate classifying where cars are within a search area. Hence, we set the plan request to focus on parking lots at the field test site (Fig. 4.16, left), with the addition of three chosen grid cells within the parking lots being marked as having a higher uncertainty. The plan request boundaries and priors were created with GPS coordinates in Google Earth, exported as

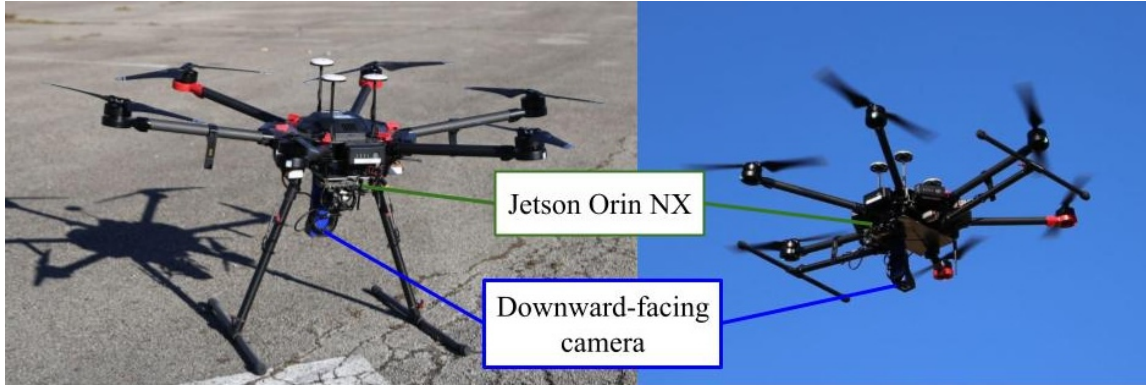


Figure 4.17: Hexarotor system (DJI M600 Pro) with onboard compute and camera. Left image shows drone on the ground, right image shows drone in flight.

kml files, and then converted into our plan request message format.

The following sections details the hardware, autonomy, and experimental results for our hexarotor deployments.

Hardware System

The hardware consists of the DJI M600 Pro, shown in Fig. 4.17, along with the physical sensing and onboard computer payload. The DJI M600 Pro contains a flight controller that handles pose estimation and position-based control. The DJI M600 Pro’s flight controller also handles teleoperation if human intervention is necessary. Beneath the drone’s base, we mount a custom hardware payload. That payload consists of an onboard computer, a Jetson Xavier, to run the autonomy software shown in Fig. 4.18. The payload also contains a downward-facing a camera for sensing the environment. The camera is a Seek S304SP thermal camera. The camera intrinsics are used to calculate the frustum’s intersection with the search map’s cells in IA-TIGRIS.

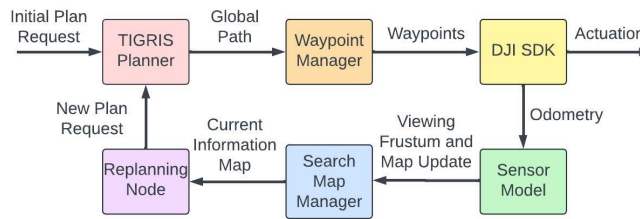


Figure 4.18: Functional diagram of the DJI M600 Pro autonomy software.

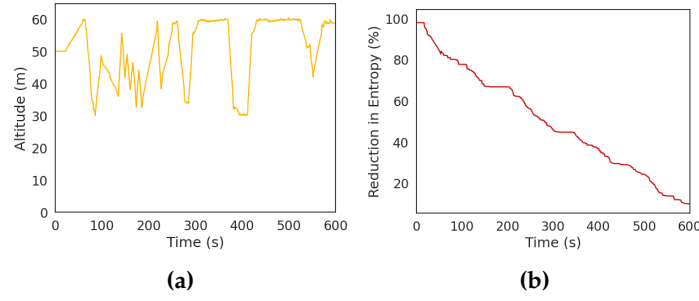


Figure 4.19: The results for our hexarotor field deployment. (a) Plot of flown altitude over time, showing large variation throughout the experiment. (b) Reduction in entropy percentage over time of field experiment.

Autonomy System

Fig. 4.18 illustrates the functional system diagram for the real world field test on the DJI M600. The user specifies the initial plan request prior to takeoff. The TIGRIS planner makes an initial plan on that plan request and sends a global path to the waypoint manager. The waypoint manager tracks the current waypoint within the plan and sends the next waypoint to the DJI software development kit, which then sends actuation commands to the motors. The position of the drone is used to calculate the distance from the drone to the ground and sends that distance parameter to the sensor model. The sensor model's true positive and false positive rate is used to calculate the per-cell entropy updates in the search map manager. The search map manager publishes the current information map, and the replanning node sends an updated plan request to the IA-TIGRIS planner every ten seconds.

The drone started at an altitude of 50 m above the origin of the reference frame. The informed sampler in IA-TIGRIS was set to add states at altitudes of either 30 m or 60 m, creating a trade-off between observation area and detector accuracy. The budget was 2000 m, the planning horizon was 600 m, and the planning time was 10 seconds.

Experimental Results

The right image of Fig. 4.16 shows the path selected by IA-TIGRIS in the search area. The figure highlights how the planner dynamically adjusts altitudes over time to balance coverage and sensing resolution, maximizing information gain. Higher altitudes allow for broader area coverage, while lower altitudes provide more detailed observations where needed. Additionally, the planner prioritizes revisiting the three regions of higher uncertainty, recognizing the need for repeated observations reduce entropy. This adaptive strategy ensures that uncertain areas receive sufficient attention to improve the belief map. As a result, the entropy of the information map decreases



Figure 4.20: Fixed-wing platform used for autonomous flights with an onboard camera pitched at 10 degrees [189]

to near zero by the end of the mission, as shown in Fig. 4.19(b), indicating that the planner has effectively gathered the necessary information. This behavior demonstrates the planner’s ability to optimize sensing actions, balancing altitude selection, revisit frequency, and exploration to maximize mission success.

4.6.2 Fixed-wing Deployments

Our proposed approach was extensively tested on the fixed-wing AlareTech TL-1 UAV, shown in Fig. 4.20. The UAV is equipped with an onboard camera pitched at 10 degrees, which introduces a more challenging planning problem due to the non-holonomic motion model and the camera’s field of view. Over more than 20 flight hours and 100 flights running IA-TIGRIS, we validated our approach with the objective to search for objects of interest in a large search space across a variety of test scenarios, including different terrain types, varying environmental conditions, and diverse target distributions. An example mission from these tests is shown in Fig. 4.21. In this scenario, the planner was given the search bounds and a designated high-priority region. The resulting flight path prioritized revisiting the high-priority area twice, optimizing sensor use and ensuring maximum information gain. This strategy led to the successful detection of the object of interest, with its estimated position marked by the red dot in the figure.

The map on the upper right in Fig. 4.21 shows the information map after plan execution was complete. Due to the UAV’s limited budget, the upper right and lower left corners of the map are not searched by the agent. The budget is instead utilized to search over the area of higher priority two times. Compared to the paths in Fig. 4.16, we observe that the paths for the fixed wing are smoother and have a larger turning radius, demonstrating how IA-TIGRIS respects the motion constraints of the vehicle. We can also see the effect of wind on the path execution, where the flown path shown in green deviates from the planned path shown in yellow. This illustrates the

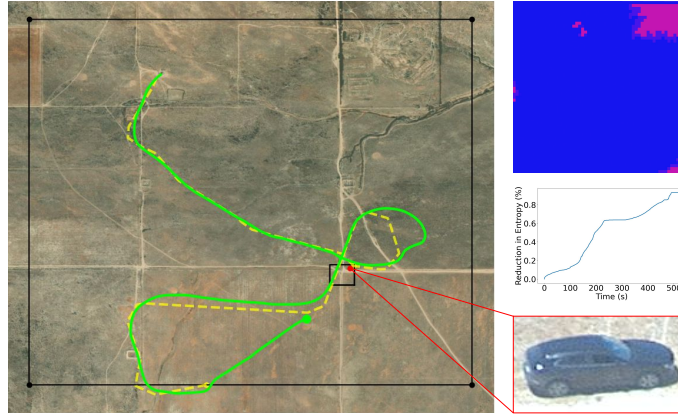


Figure 4.21: An example path generated for the fixed-wing platform conducting a large-area search for an object of interest. The larger black rectangle denotes the search bounds, while the smaller black rectangle highlights a region of higher uncertainty. The red dot marks the estimated position of the detected object based on image detections. The upper-right map displays the information state after planning is complete, while the middle plot shows the percent change in entropy over mission time. The flown path illustrates a balance between allocating resources to the high-priority region and exploring other areas within the search space.

importance of online planning in the cases where this deviation is large or would accumulate over the course of a longer mission and cause the expected observed area to be much different than actual observed area.

4.7 Conclusion

This chapter presents a novel sampling-based IPP planner for autonomous robots that is both adaptive and incremental. Our planner, IA-TIGRIS, is shown to be effective and computationally efficient, making it suitable for onboard deployment. We conduct a thorough evaluation of individual planner components, providing generalizable insights for other IPP frameworks. Extensive simulation testing against baseline methods highlights the planner’s advantages, and we further validate its effectiveness through demonstrations on two robotic platforms.

Future work includes extending this approach to multi-agent deployments in semi-urban environments for object search and search-and-rescue operations. Further improvements could involve batch sampling and parallel processing to enhance computational efficiency, automatic parameter and horizon selection based on information map characteristics, and integrating trajectory optimization for smoother paths. Additionally, expanding the framework to support planning with a gimbaled camera would enable more flexible and targeted sensing capabilities.

Part II

Disturbance and Risk Aware Path Planning

Taking the research in robotics from theoretical works and simulations into the real world of field robotics brings about challenges that are often missed or set aside to simplify the problem. Deploying robotic platforms in the field often necessitates taking into account additional disturbances and challenges introduced by the environment. For example, the perception pipeline could be affected by changing lighting conditions, glare, and precipitation, while the path planning and controls are affected by disturbances and changing environmental factors such as wind [Kulkarni et al., 2020].

In Part I, we focused on the informative path planning problem and presented our planner IA-TIGRIS. We ended with showing the field results for IA-TIGRIS on a fixed-wing UAV. When we look at Fig. 4.21, we can see how the executed path in green deviates rather far from the planned path in yellow. This is due to the disturbances caused by wind. For aerial robots, wind can have a large impact on the performance of your system and even dictate the operating window. Wind introduces additional risks and constraints, especially if your system requires low or no wind.

Part II of this thesis contains three chapters that complement each other to be able to first estimate the wind field, then plan time-optimal paths within that wind field, and lastly be able to quantify the impact on energy usage and risks for the given flight path.

The section summaries and contributions are as follows:

- Chapter 5 focuses on how to **estimate urban wind fields** using in-situ measurements³. Computational Fluid Dynamics (CFD) simulations can help provide a wind field estimate, but their accuracy depends on the knowledge of the distribution of the inlet boundary conditions. This chapter provides a real-time methodology using a Particle Filter (PF) that utilizes wind measurements from a UAV to solve the inverse problem of predicting the inlet conditions as the UAV traverses the flow field. A Gaussian Process Regression (GPR) approach is used as a surrogate function to maintain the real-time nature of the proposed methodology. Real-world experiments with a UAV at an urban test-site prove the efficacy of the proposed method. The flight test shows that the difference in mean inlet ground truth and estimated angles were between -3.7° and 1.3° and the difference in mean magnitudes were between -0.2 m/s and 0.020 m/s. A video overview can be viewed at <https://youtu.be/U4XdYgSJRZM>.
- Chapter 6 explores how to quickly compute **time-optimal paths between two states for UAVs in a wind field**⁴. Time-optimal path planning in high winds for a turning rate constrained UAV is a challenging problem to solve. Previous works have used trochoidal path segments, which consist of straight and maximum-rate

³This work was included in Patrikar [2020] and published in Patrikar et al. [2020].

⁴This work has been published in RA-L as Moon et al. [2023].

turn segments, as optimal extremal paths in uniform wind conditions. Current methods iterate over all candidate trochoidal trajectory types and choose the time-optimal one; however, this exhaustive search can be computationally slow. In this chapter we present a method to decrease the computation time. We achieve this via a geometric approach to reduce the candidate trochoidal trajectory types by framing the problem in the air-relative frame and bounding the solution within a subset of candidate trajectories. This method reduces overall computation by 37.4% compared to pre-existing methods, freeing up computation for other onboard processes and can lead to significant total computational reductions when solving many trochoidal paths. When used within the framework of a global path planner, such as IA-TIGRIS, faster state expansions help find solutions faster or compute higher-quality paths. We also release our open-source codebase as a C++ package. The codebase can be found at <https://github.com/castacks/trochoids> and a video overview of our method can be found at <https://youtu.be/qOU5gl7JshI>.

- Chapter 7 examines how to **predict the energy usage and risk of a UAV flight**, especially under environmental uncertainty such as from wind⁵. Energy management is a critical aspect of risk assessment for UAV flights, as a depleted battery during a flight brings almost guaranteed vehicle damage and a high risk of human injuries or property damage. Predicting the amount of energy a flight will consume is challenging as routing, weather, obstacles, and other factors affect the overall consumption. We develop a deep energy model for a UAV that uses Temporal Convolutional Networks to capture the time varying features while incorporating static contextual information. Our energy model is trained on a real world dataset and does not require segregating flights into regimes. We illustrate an improvement in power predictions by 29% on test flights when compared to a state-of-the-art analytical method. Using the energy model, we can predict the energy usage for a given trajectory and evaluate the risk of running out of battery during flight. We propose using Conditional Value-at-Risk (CVaR) as a metric for quantifying this risk. We show that CVaR captures the risk associated with worst-case energy consumption on a nominal path by transforming the output distribution of Monte Carlo forward simulations into a risk space. Computing the CVaR on the risk-space distribution provides a metric that can evaluate the overall risk of a flight before take-off. Our energy model and risk evaluation method can improve flight safety and evaluate the coverage area from a proposed takeoff location. A video overview and our codebase are available at <https://youtu.be/PHXGigqilOA> and <https://git.io/cvar-risk>

⁵This work was published in Choudhry et al. [2021]. My contributions were mainly focused in overall methodology design, wind and risk modeling, implementation, testing, and analysis

Chapter 5

In-Situ Urban Wind Field Estimation

5.1 Introduction

Enabling autonomous unmanned aerial vehicles (UAVs) to estimate wind patterns is crucial if they are to realize their potential in dense urban landscapes. Owing to their small size, low speeds, and close proximity to obstacles, the safety and operational performance of UAVs flying within cities is significantly affected by the prevailing wind conditions [Cybyk et al., 2014, Galway et al., 2011]. An estimate of the wind field can provide planning and decision algorithms with necessary information to compute safer [Kothari et al., 2014] and energy efficient paths [Ware and Roy, 2016].

UAVs are uniquely positioned to act as a source of in-situ wind measurements. The work in this chapter explores the possibility of using these local wind estimates to predict a global wind field. Wind flow estimation to enable autonomous dynamic soaring [Lawrance and Sukkariéh, 2010] has predominantly focused on estimating high altitude, clutter-free wind fields. Methods using Gaussian Process Regression [Lawrance and Sukkariéh, 2010], polynomial parameterization of the wind field [Langelaan et al., 2012], and Weibull probability density function with Prandtl's power law relationship [Rodriguez et al., 2016] have been presented. Low altitude urban wind fields by comparison are more complicated and difficult to predict.

Computational Fluid Dynamics (CFD) has been presented as a promising solution to calculate the low altitude urban wind flow patterns. Techniques using Reynolds-Averaged Navier–Stokes (RANS) solvers have gained prominence over others, mainly due to their computational advantage. Using CFD results for UAV path planning has been explored in some works [Galway et al., 2011, Orr et al., 2005], but they lack a comprehensive analysis on how close the simulation models match the real world conditions. One notable effort is made by Ware and Roy [2016] who validated the

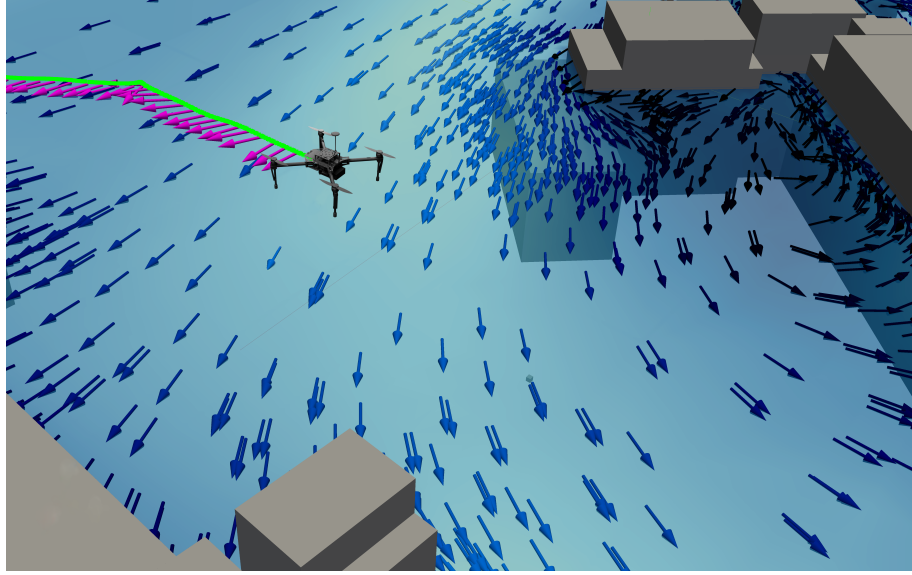


Figure 5.1: A CFD wind field using the estimated inlet boundary condition at the test location, along with the path of the UAV (green), and the corrected wind measurements (magenta) from the onboard anemometer.

CFD simulation results using in-situ measurements from ground stations. The work however uses historical data for estimating the boundary conditions which may not lead to sufficiently accurate wind field predictions. Another notable effort uses high resolution wind models to provide inlet conditions to predict CNN-based wind-fields [Achermann et al., 2019].

Improving the predictive accuracy of CFD simulations is an active field of research [Ricci et al., 2020]. Previous work on Uncertainty Quantification (UQ) [García-Sánchez et al., 2014] has proven that uncertainties in inlet conditions have a higher effect on the accuracy of CFD results than other parameters such as aerodynamic roughness. Other works [Wise et al., 2018] that compared CFD results with real world data have argued that even for minor changes in the inlet wind direction the simulated flow patterns can change considerably, to the extent that planning decisions might be influenced. It is thus important to have a reliable estimate of the inlet conditions. One obvious solution is to use wind sensors at the periphery of the urban area or a tall structure within the city. However, measurements from these sources can introduce considerable error as often times it is difficult to determine the errors between the incoming wind at the boundary and measured wind conditions. Sousa et al. [2018] have addressed this problem of predicting the distribution of the boundary conditions (inlet speed and angle) by solving an inverse problem of estimating inlet conditions using an ensemble Kalman filter (EnKF) and given in-situ measurements from static sensors. Their subsequent work provides real world validation Sousa and Gorlé [2019] of the methods.

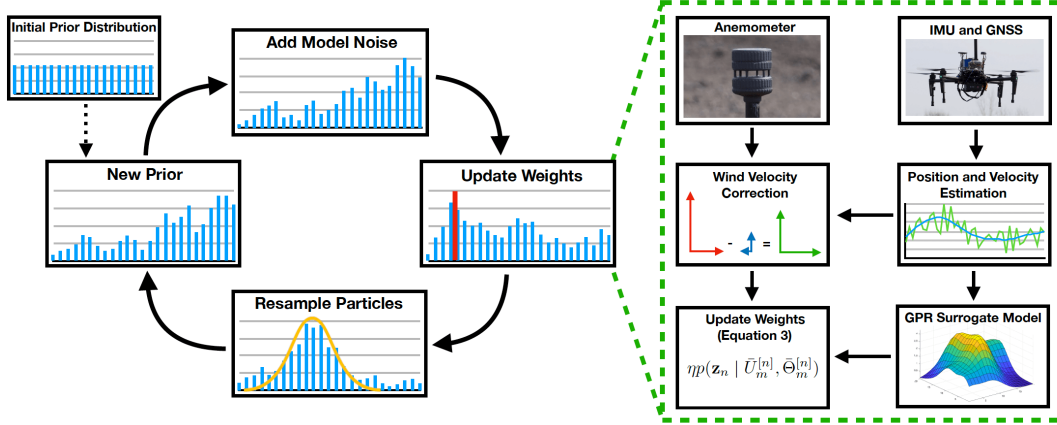


Figure 5.2: Wind measurements from an onboard anemometer are corrected for motion and bias. These measurements are then used in a Particle Filter framework to provide a posterior distribution on the inlet conditions by comparing with estimates from a Gaussian Process based forward propagation model.

This chapter explores the possibility of using UAV-based sequential measurements using an onboard anemometer to solve the inverse problem of estimating the inlet boundary conditions using a particle filter (PF) based approach¹. A PF is known to provide a better estimate of the posterior belief for low dimensional problems as it samples directly from the distribution as opposed to an EnKF which has a tendency to provide approximations to the posterior that are too close to a Gaussian distribution [Stordal et al. \[2011\]](#). UAVs have an ability to fly in the roughness layer [\[Orr et al., 2005\]](#) that is just above the Urban Canopy Layer (UCL) and thus can potentially give a much better estimate of the inlet conditions affecting the UCL. Our setup also avoids the need of setting up a static wind measurement network, which may prove prohibitive because of cost and maintenance hurdles. As wind enters the city, it flows between structures and form a distinctive pattern. The core idea of our approach is to identify this pattern using the onboard measurements and then solve the inverse problem to estimate the inlet conditions. If the estimated boundary conditions are used as an input for running the CFD simulation, the resulting field should most closely resemble the actual wind field. The contributions of the work in this chapter are three fold:

1. We present a particle filter approach to solve the inverse problem of estimating the boundary conditions given sequential in-situ wind measurements from a UAV.
2. We present a domain specific treatment to circumvent the problem of running multiple forward simulations using Gaussian process regression to maintain the real-time onboard nature of the methodology.

¹The choice of anemometer is for ease of prototyping but any method that provides accurate local wind measurements from onboard sensors can be used [\[Marino et al., 2015, Niedzielski et al., 2017\]](#)

3. We present real world test results that prove the efficacy of the presented methodology.

The sections are organized as follows: Section 5.2 provides details on the proposed methodology along with the mathematical formulations. Section 5.3 provides details on the field trials and the results. Section 5.4 presents the discussions and conclusions.

5.2 Methodology

This section details the methodology to estimate the boundary conditions given measurements from the UAV. Care was taken that the onboard algorithms are computationally tractable for any onboard computer with modest specifications. We assume the UAV to be operating at a constant altitude and consider a 2D wind field. The goal is to find the distribution on the wind inlet angle Θ and wind inlet magnitude U . For the case under consideration, the local wind angle θ and the local reduced velocity $u_r = \frac{u}{U}$ are only a function of the inlet wind angle. The methodology is summarized in Fig. 5.2.

5.2.1 Mathematical Formulations

Mathematically, the problem can be stated as: Given a set of measurements $\mathbf{z}_i = \{\theta_i, u_i\}$ for $i \in [0, N]$ from the UAV such that θ_i represents the local wind angle, and u_i represents the local wind magnitude at the i^{th} data-point, find the belief $bel(\Theta_{0:i}, U_{0:i}) = p(\Theta_{0:i}, U_{0:i} \mid \mathbf{z}_{1:i})$ for $i \in [0, N]$ where Θ_i is the wind inlet boundary angle and U_i is the wind inlet magnitude after i measurements. To solve for this belief, we will perform a domain specific derivation of a particle filter approach [Thrun et al. \[2000\]](#).

Using the Bayes Rule we can write:

$$\begin{aligned} bel(\Theta_{0:i}, U_{0:i}) &= p(\Theta_{0:i}, U_{0:i} \mid \mathbf{z}_{1:i}) \\ &= \eta p(\mathbf{z}_i \mid \Theta_{0:i}, U_{0:i}, \mathbf{z}_{1:i-1}) p(\Theta_{0:i}, U_{0:i} \mid \mathbf{z}_{1:i-1}) \end{aligned} \quad (5.1)$$

Using the Markov property for conditional independence:

$$\begin{aligned} bel(\Theta_{0:i}, U_{0:i}) &= \eta p(\mathbf{z}_i \mid \Theta_i, U_i) p(\Theta_{0:i}, U_{0:i} \mid \mathbf{z}_{1:i-1}) \\ &= \eta p(\mathbf{z}_i \mid \Theta_i, U_i) p(\Theta_{0:i-1}, U_{0:i-1} \mid \mathbf{z}_{1:i-1}) \\ &\quad p(\Theta_i, U_i \mid \Theta_{0:i-1}, U_{0:i-1}, \mathbf{z}_{1:i-1}) \\ &= \eta p(\mathbf{z}_i \mid \Theta_i, U_i) p(\Theta_{0:i-1}, U_{0:i-1} \mid \mathbf{z}_{1:i-1}) \\ &\quad p(\Theta_i, U_i \mid \Theta_{i-1}, U_{i-1}) \end{aligned} \quad (5.2)$$

Thus, for the m^{th} particle, the weight can be calculated as

$$w_m = \eta p(\mathbf{z}_i | \Theta_i, U_i) \quad (5.3)$$

This probability distribution can be represented as a multivariate normal distribution:

$$p(\mathbf{z}_i | \Theta_i, U_i) = \frac{\exp -\frac{1}{2}(\mathbf{z}_i - \bar{\mathbf{z}}_i)^T \Sigma^{-1}(\mathbf{z}_i - \bar{\mathbf{z}}_i)}{\sqrt{(2\pi)^2 |\Sigma|}} \quad (5.4)$$

Where

$$\bar{\mathbf{z}}_i = f(\Theta_i, U_i, \mathbf{X}_i) \quad (5.5)$$

where $\mathbf{X}_i = \{x_i, y_i\}$ are the coordinates of the UAV when the measurement $\mathbf{z}_i$ was taken. This function is the forward model and requires running the CFD solver, but given that it is a computationally expensive operation, we instead use a computationally cheaper but less accurate surrogate function. The details are further discussed in Section 5.2.3

Algorithm 5: Particle Filter

Input: $\mathbf{z}_{1:N}$, $[\{x_1, y_1\}, \dots, \{x_N, y_N\}]$

- 1 $(U_{0:M}^{[0]}, \Theta_{0:M}^{[0]}) \leftarrow \text{UniSample}([0, U_{max}], [0, 2\pi], M)$
- 2 **for** $n \leftarrow 1$ **to** N **do**
- 3 $(\bar{U}_{0:M}^{[n]}, \bar{\Theta}_{0:M}^{[n]}) = (U_{0:M}^{[n-1]}, \Theta_{0:M}^{[n-1]}) + \mathcal{N}(0, \sigma)$
- 4 **for** $m \leftarrow 1$ **to** M **do**
- 5 $w_m = \eta p(\mathbf{z}_n | \bar{U}_m^{[n]}, \bar{\Theta}_m^{[n]})$
- 6 **end**
- 7 $(U_{0:M}^{[n]}, \Theta_{0:M}^{[n]}) \leftarrow \text{ReSample}(\bar{U}_{0:M}^{[n]}, \bar{\Theta}_{0:M}^{[n]}, w_{0:M})$
- 8 **end**

5.2.2 Particle Filter

The particle filter implementation is detailed in Algorithm 5. M particles are initialized from a uniform distribution between minimum and maximum values. Each particle is a tuple of inlet angle and magnitude. The algorithm has three major components. The propagation step is the first step (Line 3), which in this case only involves adding zero mean Gaussian noise to the particles. This helps to avoid premature convergence. The next step (Line 5) involves calculating the weight of each particle based on how likely the particle represents the actual value of the inlet conditions. This weight is calculated for each particle using (5.5) and (5.4). Finally based on the calculated weights, a re-sampling algorithm chooses particles for the

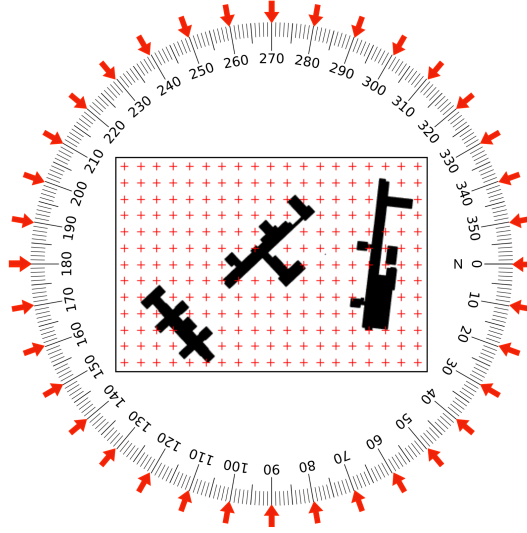


Figure 5.3: Data points (locations and inlet angles) for GPR training.

next iteration based on the weighted probability distribution. The resampling process accounts for the difference of the target and the proposal distribution. To improve performance, we use a low-variance resampling strategy [Thrun et al., 2000] to sample the distribution, as it helps preserve the diversity if samples have same importance factors and is a more systematic way for re-sampling than the independent random sampler. The sampler operates by choosing a random number between $[0, M^{-1}]$ and then repeatedly adds M^{-1} to the random number while selecting particles that correspond to the resulting summation. For the sake of brevity, the complete algorithm is not presented, and the reader is directed to Table 4.4 Thrun et al. [2000] for a detailed explanation.

5.2.3 Surrogate function

Calculating (5.5) can prove to be a computationally expensive process, as it involves running a forward pass of the CFD simulation for the particular inlet condition and probing the location for the wind angle and magnitude. One way to avoid this issue, is to use a surrogate function to approximate these values. Jorge Sousa et al. Sousa et al. [2018] use a set of CFD simulations to construct polynomial chaos expansions (PCE) for the quantities of interest. For our use case, the PCE method gave a less accurate result, as the function variables also include the location along with the inlet angle. We decided to use a Gaussian Process Regression (GPR) to model the surrogate functions, as they have been shown to better represent spatial data. Two GPR models are trained: one for the local wind angles $\hat{f}_\theta$ and one for local wind reduced velocity

$\hat{f}_{u_r}$. Both are functions of the inlet angle and probe location. Thus (5.5) is replaced by:

$$\begin{aligned}\bar{u}_i &= \hat{f}_{u_r}(\Theta_i, \mathbf{X}_i)U_i \\ \bar{\theta}_i &= \hat{f}_{\theta}(\Theta_i, \mathbf{X}_i)\end{aligned}\tag{5.6}$$

The datapoints used to train the GPR models were obtained by running multiple CFD simulations with regularly spaced inlet angles and recording data on an equally spaced grid. The data-points are visually represented in Fig. 5.3. The performance of the GPR models is discussed in Section 5.3.2.

5.2.4 Wind Measurements and Correction

Wind measurements are obtained from an ultrasonic anemometer onboard the UAV. The anemometer records the wind angle and magnitude with respect to the moving drone. Previous works that have used anemometers for onboard wind measurements [Bruschi et al., 2016, Garibaldi Castillo and Londner, 2010, Thorpe et al., 2018] have reported that while the wind angle measurements were found to be accurate for all flight conditions, the magnitude measurements show a bias that reports a higher wind magnitude than expected. We found that the effect is more pronounced at lower UAV speeds. Thus, in order to record reliable wind measurements we need to remove the magnitude bias in addition to correcting for the UAV motion. The correction is given by:

$$\begin{aligned}w_N &= (w_m^{[raw]} - \alpha) \cos(w_\theta^{[raw]}) - V_N \\ w_E &= (w_m^{[raw]} - \alpha) \sin(w_\theta^{[raw]}) - V_E \\ w_m^{[corrected]} &= \sqrt{w_N^2 + w_E^2} \\ w_\theta^{[corrected]} &= \arctan(w_E, w_N)\end{aligned}\tag{5.7}$$

where V_N and V_E represent the inertial speed of the UAV in North and East directions, and w_N and w_E represent the corrected wind components in North and East directions. α is the bias correction factor. The value of the bias correction factor was hand tuned using historical data of similar setups [Bruschi et al., 2016] and nil wind observations.

5.2.5 Summary

The methodology assumes that an accurate 3D model of the environment is available, and the UAV has a sensor suite capable of reliably measuring the local wind angle and magnitude. The steps are as follows:

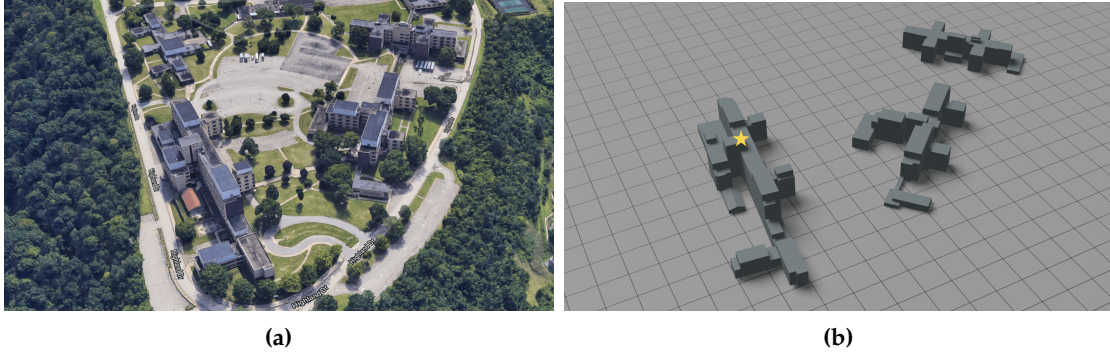


Figure 5.4: (a) Image of the field validation test site. (b) 3D model of test site. Yellow star represents the location of the ground truth weather station on the top of the building.

1. Given the 3D model, run multiple forward RANS CFD simulations to collect data points (Fig. 5.3) for training the surrogate GPR models represented in (5.6).
2. As the UAV is flying in the flow field, collect the raw wind measurements and correct them using (5.7).
3. Use the offline trained surrogate models and the corrected wind measurements to estimate the inlet conditions using Algorithm 5.
4. Use the inlet conditions (and the distribution) to select the correct wind-field estimate precomputed from a external high-fidelity CFD solver.

5.3 Field Validation

This section gives details on the test setup and the hardware used for field validation of the algorithm. Discussion on the experiments and results follows. Field trials involve autonomously flying a predetermined sequence of waypoints and estimating the inlet conditions. For validation, the estimated inlet conditions are compared to a static wind sensor mounted on a rooftop nearby. The static wind sensor is measuring the free-stream wind magnitude and direction.

5.3.1 Setup

Our chosen test site is located on an elevated piece of land with around ten buildings, all ranging in heights of 5 to 35 meters (H_{max}). The buildings are abandoned and only authorised personnel are allowed within the area. The complex of buildings effectively replicates a typical urban environment and will allow for complex interactions of the



Figure 5.5: The test UAV with the wind sensor.

wind and buildings. Because the test site is elevated from the surroundings, it provides an ideal environment to minimize disturbances to the inlet conditions.

An image and model of the test site are seen in Fig. 5.4(a) and 5.4(b) respectively. The size of the site is roughly 300×250 meters. To serve as a ground truth for the inlet conditions, a weather station was placed in the location marked with a star in Fig. 5.4(b) on top of the tallest building. Given the proximity of the ground weather station to the testing site and the elevated nature of land, the weather station can provide accurate measurements of inlet angles.

Our UAV test platform is a DJI M100 quadrotor with a FT Technologies FT205EV Lightweight Ultrasonic Wind Sensor. This wind sensor is ideal for mounting on a flying platform because of its low weight and integrated magnetometer. To minimize disturbances from the quadrotor propellers, the sensor is mounted on a 40 cm long carbon fiber pole. Fig. 5.5 shows the UAV with the sensor. The DJI SDK fuses GPS and IMU data to provide the position and velocity estimates.

For our static weather station we used a Maximet GMX500 which has a resolution of 0.01 m/s and an accuracy of $\pm 3\%$. From the station we recorded wind heading and wind magnitude, as well as the UTC time, using a single board computer. The inlet estimates from the UAV and the static measurements are synced using the GPS time on the static sensor. To reduce the high frequency noise, all the sensor measurements are passed through a median filter before using them in the algorithms.

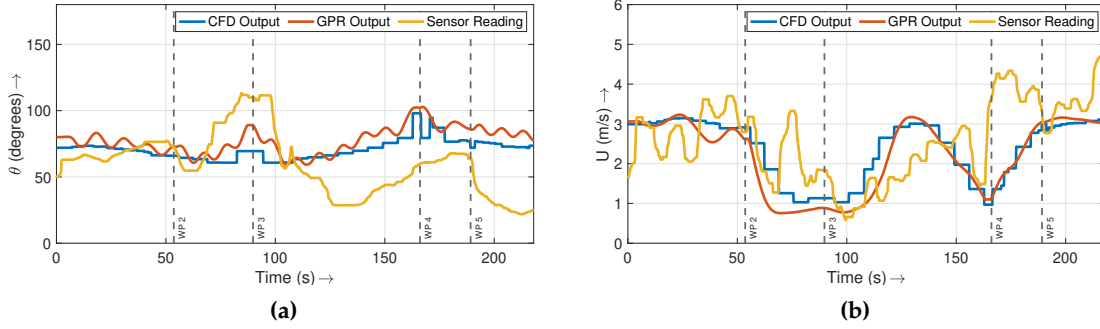


Figure 5.6: The motion and bias corrected local wind angle (a) and magnitude (b) readings for the test run. Also shown is the CFD and GPR output for the same locations as the test run using the mean of the inlet ground truth measurements.

5.3.2 Preprocessing

In order to successfully test the methodology, GPR models need to be trained to estimate the wind angles and reduced velocities given the inlet angles for the area under consideration. To generate the training data, we run CFD simulations on regularly spaced inlet angles spanning from 0° to 360° with a step size of 10° . To sample points spatially, a regularly spaced 20 m resolution grid in X and Y was used as probe locations. Fig. 5.3 shows these points in red. Care was taken to remove points that fall within structures. CFD simulations were carried out using the OpenFoam SIMPLE solver [Weller et al., 1998]. The mesh was created using SnappyHexMesh. The CFD assumes incompressible flows, and the values are recorded at steady state. A standard $\kappa - \epsilon$ model is used to model the turbulence characteristics.

The GPR models use a radial basis function (RBF) with a length scale of 10 as the kernel function. The training is carried out using the `sklearn` library [Pedregosa et al., 2011]. To test the model performance, we plot both the angle and reduced magnitude predictions against CFD results in Fig. 5.6(a) and 5.6(b) respectively.

To identify which areas in the map are the most informative, the correlation between local wind angles and inlet angles was evaluated across the test space. For each point in a five-meter spaced grid, a Pearson correlation coefficient was calculated between the local wind angles and inlet angles. Fig. 5.7(a) shows the correlation coefficients across the test space. This shows how areas which are often blocked from the wind will not correlate well to the inlet angle, and taking measurements there would not be as beneficial for inlet condition estimation. In contrast, there are also areas with high correlation, which are usually areas where the wind will be less hindered regardless of the inlet conditions. In the case of our test site, the center has the highest correlation, as well as the spaces farther out from the buildings.

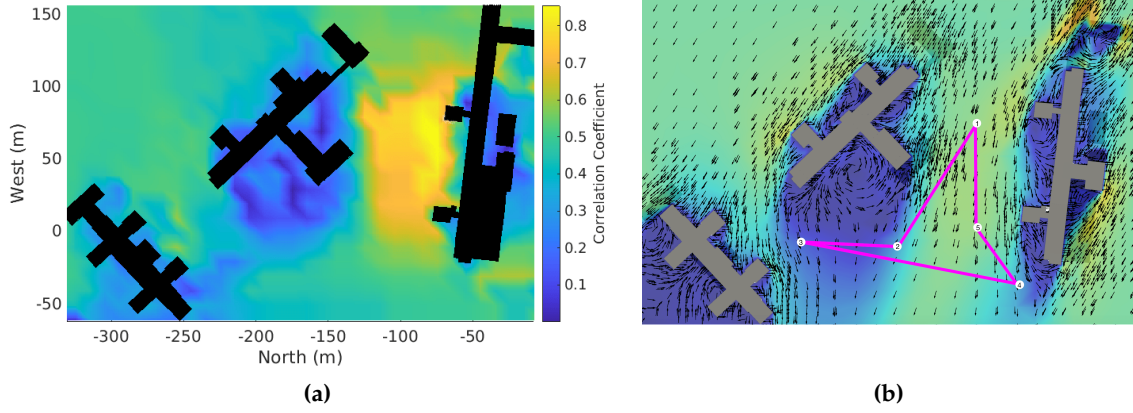


Figure 5.7: (a) Correlation between local wind angles and inlet wind angles. (b) The path followed autonomously by the UAV at the test site. The waypoints are numbered in the order they were visited. The arrows show the wind field flow directions, while the color gradient represent the magnitudes.

5.3.3 Experiments

For the experiments, a set of waypoints for the UAV path were selected based on the correlation graph. The resulting path can be seen in Fig. 5.7(b). The commanded altitude of the path was 20 meters AGL, and it was flown at a commanded inertial speed of 2 m/s . The path was flown autonomously, and the corrected wind angle and speed are shown in Fig. 5.6(a) and 5.6(b) respectively. The data from the the weather station was recorded concurrently for comparison. In order to match the training of the surrogate function, the wind measurements while the UAV was on the ground, taking off, and landing were not used in the particle filter, but rather only the measures when the UAV was at the commanded flight path altitude.

5.3.4 Results

The estimated inlet angle and magnitude for a test run with moderate wind speeds are plotted in Fig. 5.8(a) and 5.8(b) respectively. The figures show that the estimated inlet conditions are in agreement with the values seen by the onboard anemometer. The figures also show the mean and one standard deviation of the ground truth measurements. As comparison, the cumulative means and one standard deviation of the PF output is plotted. It is clear that the mean of the PF outputs converge to the mean of the ground truth measurements. The 95% confidence interval for the difference between the mean estimated inlet angle and mean ground truth angle is -3.680° and 1.250° . The 95% confidence interval for the difference between the mean estimated inlet magnitude and mean ground truth magnitude is -0.206 m/s and 0.020 m/s . Zero lies

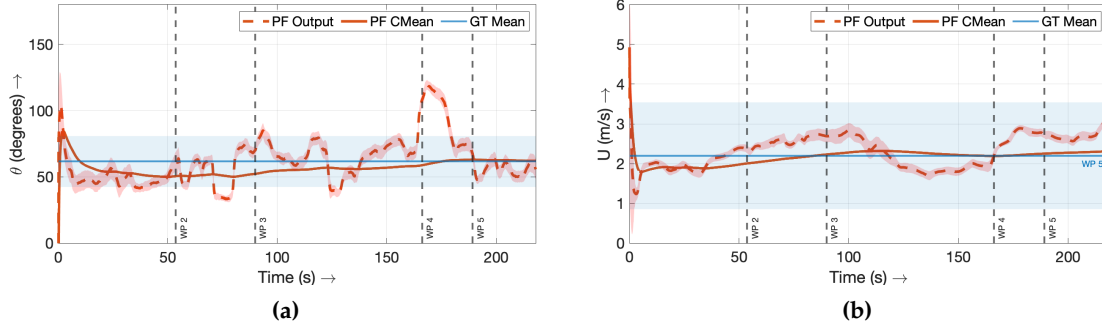


Figure 5.8: The wind inlet angle (a) and magnitude (b) estimated (and one std. deviation) from the particle filter and the mean inlet magnitude (and one std. deviation) measured by the ground truth weather station.

within the bounds of both confidence intervals, thus showing the statistical significance of these results. With unoptimized code and $M = 1000$, each pass of the PF takes 277 milliseconds on an Intel NUC8i5BEK with a processor base frequency of 2.30 GHz.

5.4 Conclusions and Future work

This chapter presents a method to use onboard wind measurements from a UAV to solve the inverse problem of estimating the inlet conditions to improve CFD-based wind field estimations. The work in this chapter shows that it is possible to reliably use an anemometer on a moving UAV platform as sources of local wind measurements. The measurements are then used in a particle filter to provide a posterior distribution of the inlet conditions. A Gaussian Process Regression model is used as a surrogate function to achieve real-time performance. The method is implemented on a multirotor platform and the results are shown to be consistent with the measurements from a static roof mounted weather station.

While the initial results are encouraging, more comprehensive testing at low and high wind conditions needs to be undertaken to fully evaluate the efficacy of the approach. The current approach considers a constant altitude 2D wind field. Extension to 3D will involve training the surrogate GPRs with an additional variable which might have a negative impact on the training accuracy. Replacing GPRs with a CNN-based surrogate model [Achermann et al., 2019] might be a possible solution. Future iteration would also involve replacing the anemometer with an algorithm that uses onboard sensors (GNSS, IMU, ..) to improve accuracy and enable wider commercial adoption.

Chapter 6

Time-Optimal Path Planning in a Constant Wind

The previous chapter established that it is reasonable for a UAV to have a wind field estimation onboard that can then be leveraged for downstream tasks—one of those being path planning. Ignoring wind in path planning can lead to less robust results, suboptimal paths, infeasible paths, or even the loss of the vehicle due to collisions. Incorporating wind information is therefore critical for ensuring the UAV can efficiently and safely navigate the environment.

6.1 Introduction

One common approach to connecting two points on a plane, such as when building a tree of trajectories with IA-TIGRIS, is to use a time-optimal path under a curvature-constrained motion model. As shown by [Dubins \[1957\]](#) and proven by [Johnson \[1974\]](#), the time-optimal solution in a wind-free environment consists of a sequence of maximum curvature turns and straight segments, known as a Dubins path. This solution is simple, quick to compute, and, despite lacking curvature continuity, has seen widespread use in robotic applications such as Autonomous Underwater Vehicles (AUV) [[Wang et al., 2017](#)] and Uncrewed Aerial Vehicles (UAV) [[Owen et al., 2015](#)]. However, these classic Dubins paths do not account for the influence of environmental flows like wind or ocean currents, which can significantly alter the time-optimal trajectory.

To address this, we now explore how incorporating wind into the planning process changes the time-optimal solutions, and how we can adapt traditional Dubins-style path planning to account for wind effects. By doing so, we aim to generate more robust and efficient paths that minimize traversal time under wind disturbances.

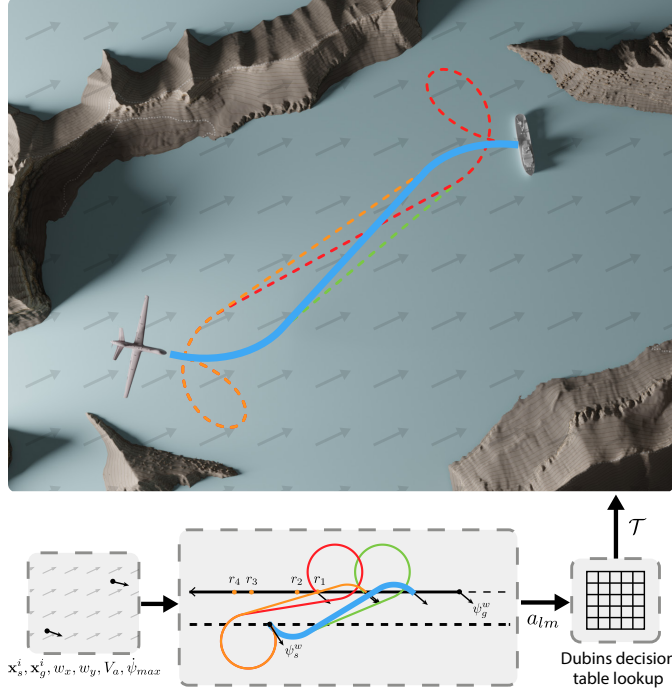


Figure 6.1: An overview of our method with an example problem. The input conditions are used to find the quadrant pair for the solution to then lookup the set of candidate trajectories in the Dubins decision table. In this case, the only candidate trajectory is LSR, thus reducing the problem to solving only one out of the four potential trajectories. The bottom middle section shows the four extremal paths transformed into $\mathcal{F}^w$, where the goal state is moving in the negative x direction with the same speed as the wind. It is clear to see that the optimal LSR path is the first to intersect with the goal state. The upper image shows the four potential extremal paths in $\mathcal{F}^i$, with the blue LSR path being optimal.

The work in this chapter describes a framework for finding time-optimal curvature-constrained path planning for vehicles in a uniform flow field. We consider a kinematic model for a UAV flying at a constant altitude and constant air-relative speeds in a steady uniform wind. This research could also extend to AUVs or autonomous surface vehicles (ASVs).

This work utilizes the framework provided by [Techy and Woolsey \[2009\]](#) to quickly compute the analytical solution for two of the four possible Bang-Straight-Bang trajectories while performing a costly global root-finding search using the Newton-Raphson method for the remaining trajectories. Iterating through solving all four trajectories and choosing the most-time optimal path is a computationally inefficient process. Through this work, we explore a solution to this problem by only needing to solve a subset of the possible Bang-Straight-Bang trajectories.

To reduce the number of trajectories we need to solve for, we utilize the work of [Shkel and Lumelsky \[2001\]](#) which identifies the optimal Bang-Straight-Bang trajectory

for a Dubins path, in cases where the states are not close together, based on the distance between initial and final states as well as their angles. Through their analysis and proofs, the quadrants of the states act as a lookup in a decision table to determine the optimal trajectory type. In cases where multiple potential trajectory types exist for given initial conditions, they created switching functions to narrow down the set of trajectories to a single trajectory.

We propose a novel approach that first formulates the problem as the goal state moving at a uniform velocity in the opposite direction of the wind flow similar to [Mcgee et al. \[2005\]](#). We then compute the transition points where the initial or final quadrants change and find between which transition points the solution lies. Because the quadrants of the initial and final points are constant between the transition points, we can utilize the decision table presented above to predict the type of trajectory we can expect to be optimal. We then use the approach presented in [Techy and Woolsey \[2009\]](#) to compute the analytical or numerical solutions for the BSB trajectories depending on the type. This method allows us to improve overall computational efficiency and speed. This is the state-of-the-art solution, to the best of our knowledge, for computing the most time-optimal trajectories in a uniform flow field while constrained by a dynamics model. We have also released our codebase as a package for others to use, benchmark, test, and improve.

The following sections are organized as follows: Section 6.2 specifies the background and related works, Section 6.3 outlines the problem definition, Section 6.4 explains our methodology in depth, Section 6.5 presents our results, and Section 6.6 gives the conclusion and future work.

6.2 Background

Several approaches have been used to perform path-planning in uniform wind conditions, given some dynamic curvature constraints, namely, vehicle velocity and turning radius.

In the absence of wind conditions, the work by [Shkel and Lumelsky \[2001\]](#) presents an approach for planning time-optimal paths using Dubins Curves, as were first formulated by Dubins in [Dubins \[1957\]](#). However, instead of having to iterate over all the feasible trajectories (four in the case if the distance between the start and goal position is greater than four times the turning radius of the vehicle), they present a decision table wherein they can predict the types of feasible trajectories based on the start and goal configurations of the UAV. Based on the quadrants in which the initial orientation and the final orientation of the UAV, they can reduce the number of candidate trajectories to below four for each block of the decision table. This table

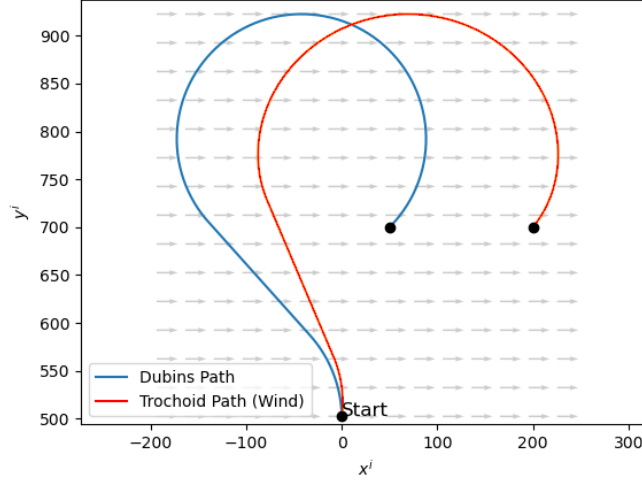


Figure 6.2: An example optimal trochoidal RSL paths and its equivalent Dubins path in the wind frame.

produces sixteen combinations of possible quadrants, which can be represented in the form a_{lm} where l is the quadrant number of the initial and m is the quadrant number of the final orientation. A switching function then chooses a single trajectory from the feasible candidate trajectory in the decision table blocks. A modified version of the decision table can be found in Table 6.1 within the methodology section.

Some of the previous works have used a frame transformation to align the axis with the wind. McGee et al. [2005] and McGee and Hedrick [2007] use this frame transformation to augment the problem into a moving-goal problem, wherein the velocity of the goal point is equal and opposite to the velocity of the wind. Finding intersection point is referred to as the “Rendezvous Problem.” The problem states that there are six admissible Dubins Paths, namely RSR, LSL, RSL, LSR, RLR, and LRL, wherein “R” represents a maximum turn-rate right turn, “L” represents a maximum turn-rate left turn and “S” is a Straight Line. The solutions to this problem can either be solvable or unsolvable. If the problem is solvable, the paths and their lengths can be computed for all eight admissible paths using a standard numeric root-finding approach such as the Newton-Raphson method. The path with the shortest distance is chosen as the optimal path. Fig. 6.2 shows a Dubins Path augmented into a trochoidal path when a wind vector is applied.

Additionally, the work by Techy and Woolsey [2009] also uses the same frame-alignment mentioned above. However, a key difference is that of the six admissible trajectory types, they can find closed-form solutions to two trajectory types, namely, RSR and LSL. To find the remaining four paths, it proposes using a standard numerical global root-finding approach such as Newton-Raphson. The optimal path is then found

by computing the lengths of the above six paths and choosing the path with the shortest length.

Another optimal control approach was also proposed by Bakolas and Tsiotras [2010] wherein it tries to solve the Markov-Dubins problem and the Zermelo's navigation problem (which was also formulated in McGee and Hedrick [2007]) in order to find the shortest path to a target while avoiding obstacles and respecting the kinematic constraints of the vehicle. A key difference between the method proposed in the work in this chapter and that proposed by McGee and Hedrick [2007] was that this work proposes the use of backward dynamic programming to compute the optimal control policy and value function for vehicles with Dubins' type kinematics operating in the presence of wind or currents. In contrast, McGee and Hedrick [2007] uses the Pontryagin's maximum principle to compute the optimal control input for the kinematic airplane model.

6.3 Problem Definition

Let the system dynamics be defined by the equations

$$\begin{aligned}\dot{x}^i(t) &= V_a \cos(\psi^i(t)) + w_x \\ \dot{y}^i(t) &= V_a \sin(\psi^i(t)) + w_y \\ \dot{\psi}^i(t) &= u(t)\end{aligned}$$

where the state is given by $\mathbf{x} = [x, y, \psi]^\top$, i denotes the inertial frame $\mathcal{F}^i$, t is time, V_a is the vehicle airspeed, w_x is the wind speed in the x axis, w_y is the wind speed in the y axis, and u is the control input.

The objective is to find the control inputs that yield the minimum time path between an initial state $\mathbf{x}_s^i = [x_s^i, y_s^i, \psi_s^i]^\top$ and final state $\mathbf{x}_g^i = [x_g^i, y_g^i, \psi_g^i]^\top$ given the control constraint $\|u\| < \dot{\psi}_{max}$. As was shown in McGee et al. [2005], Techy and Woolsey [2009], the solution to this optimization problem is one of the six extremal paths.

6.4 Methodology

On a high level, the trochoidal path planner leverages the Dubins set classification work of Shkel and Lumelsky [2001] to reduce the solutions space of the moving goal state problem formulation of McGee and Hedrick [2007] and then solves the candidate solutions using the methods of Techy and Woolsey [2009] wherein an analytical solver is used to find the solution to the RSR and LSL trajectories and a global root finding

Algorithm 6: Trochoid($\mathbf{x}_s^i, \mathbf{x}_g^i, w_x, w_y, V_a, \dot{\psi}_{max}$)

```
1 Set  $\mathbf{x}_s^w, \mathbf{x}_g^w$  using (6.1-6.4);
2  $\Gamma \leftarrow \emptyset$ ;
3  $\delta\theta_\alpha \leftarrow \text{CalcuDeltaTheta}(\psi_s^w)$ ;
4  $\delta\theta_\beta \leftarrow \text{CalcDeltaTheta}(\psi_g^w)$ ;
5  $\Gamma \leftarrow \{\text{CalcTransitionPoints}(\psi_s^w, \delta\theta_\alpha)\}$ ;
6  $\Gamma \leftarrow \Gamma \cup \{\text{CalcTransitionPoints}(\psi_g^w, \delta\theta_\beta)\}$ ;
7 for  $r_n \in \Gamma$  do
8   if  $\text{CheckSegment4R}(r_n)$  then
9      $\mathcal{T} \leftarrow \{\text{RSR}, \text{RSL}, \text{LSL}, \text{LSR}\}$ ;
10    break;
11     $l_p \leftarrow \text{DubinsPath}(r_n)$ ;
12     $t_{start} = l_p/V_a$ ;  $t_{goal} = r_n/V_w$ ;
13    if  $t_{start} \leq t_{goal}$  then
14      Compute  $\theta_q$  using (6.6);
15      break;
16 if  $\theta_q$  is empty then
17   Compute  $\theta_q$  as angle between  $r_N$  and  $\pi$ ;
18  $\alpha_q = \psi_s^w - \theta_q$ ;  $\beta_q = \psi_g^w - \theta_q$ ;
19  $a_{lm} \leftarrow \text{FindQuadrants}(\alpha_q, \beta_q)$ ;
20  $\mathcal{T} \leftarrow \text{DecisionTable}(a_{lm})$ ;
21 for each candidate trochoidal trajectory in  $\mathcal{T}$  do
22   Solve for  $u$  using (6.7-6.21);
23   if  $\text{PathTime}(u) < \text{PathTime}(u_{best})$  then
24      $u_{best} \leftarrow u$ 
25 return  $u_{best}$ 
```

approach is used to find the solution of the RSL and LSR trajectories. An overview of our method is outlined in Algorithm 6.

The first step is to align the x axis with the wind, transforming the frame from the inertial frame $\mathcal{F}^i$ to the wind frame $\mathcal{F}^w$. The wind magnitude can be calculated at $V_w = \sqrt{v_x^2 + v_y^2}$, and the angle, ψ_w , can thus be computed as

$$\psi_w^i = \tan^{-1} \left(\frac{v_y}{v_x} \right) \quad (6.1)$$

Let ψ_s^w and ψ_g^w be the wind start and goal angles defined as

$$\psi_s^w = \psi_s^i - \psi_w^i \quad (6.2)$$

$$\psi_g^w = \psi_g^i - \psi_w^i \quad (6.3)$$

The x and y elements of the start and goal states are transformed from $\mathcal{F}^i$ to $\mathcal{F}^w$ with the rotational matrix

$$R_i^w = \begin{bmatrix} \cos(\psi_w^i) & \sin(\psi_w^i) \\ -\sin(\psi_w^i) & \cos(\psi_w^i) \end{bmatrix} \quad (6.4)$$

The problem can now be viewed as finding the minimum time intersection point of the vehicle and goal state, with the wind being removed from the vehicle system dynamics and viewing the goal state as moving in the negative x direction with the velocity V_w . The updated system dynamics of the vehicle would be

$$\dot{x}^w(t) = V_a \cos(\psi^w(t))$$

$$\dot{y}^w(t) = V_a \sin(\psi^w(t))$$

$$\dot{\psi}^w(t) = u(t)$$

and the goal state position at time t would be

$$x_g^w(t) = x_g^w - tV_w$$

$$y_g^w(t) = y_g^w$$

$$\psi_g^w(t) = \psi_g^w$$

With the problem formulated in this manner, connecting any two *static* points in $\mathcal{F}^w$ can be treated as a Dubins problem, as there is no wind acting on the vehicle. A Dubins path can be solved with just the distance between the two states and the orientation of the start and goal states relative to the axis connecting the two states. We define this axis-aligned frame as $\mathcal{F}^d$ shown in Fig. 6.3. To find the Dubins path between the static start state and a goal state parameterized by a given t , the angles ψ_s^w and ψ_g^w are transformed to the Dubins frame, $\mathcal{F}^d$ by first computing the angle, θ , between the two points with

$$\theta(t) = \tan^{-1} \left(\frac{y_g^w(t) - y_s^w}{x_g^w(t) - x_s^w} \right)$$

The initial and the final angles in $\mathcal{F}^d$, α and β , can be computed as

$$\alpha(t) = \psi_s^w - \theta(t)$$

$$\beta(t) = \psi_g^w - \theta(t)$$

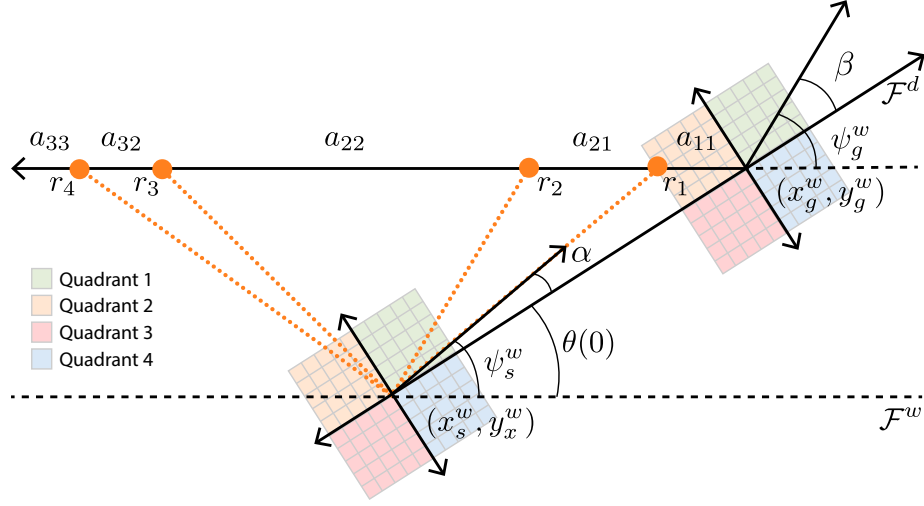


Figure 6.3: An example problem setup with variable definition shown. In this example, there are four transition points which is the maximum. Depending on V_a , w_x , w_y , and $\dot{\psi}_{max}$, the solution would lie within one of the quadrant pair sections.

With α , β , and the distance d between the states, the Dubins path between the states can be found. If $d > 4R$, where R is the turning radius, we can directly look up the optimal extremal path type based on the quadrants of α and β [Shkel and Lumelsky, 2001]. Taking the length of this path, the time of traversal for the vehicle would be $t_{start} = l_p/V_a$ where l_p is the length of the path. The time for the goal state to reach the state at the speed of V_w would be $t_{goal} = t$. If t_{start} is equal to t_{goal} , then this would be the rendezvous time and state.

As the goal-point moves in the direction opposite to the wind, the θ changes along with both $\alpha(t)$ and $\beta(t)$. We can divide the goal state trajectory into sections based on the points along the trajectory where $\alpha(t)$ or $\beta(t)$ transitions into new quadrants. Each section defines a period where the quadrants of $\alpha(t)$ and $\beta(t)$ are constant. If we can find which section of the path contains the solution, we can constrain the solution space based on the quadrants of $\alpha(t)$ and $\beta(t)$.

Proposition 1. *If the solution lies between the transition points r_n and r_{n-1} , then the set of candidate optimal trajectories $\mathcal{T}$ is the block in the Dubins decision table associated with any α or β angle derived from a point between r_n and r_{n-1} .*

Proof. Let the quadrant of α be l , the quadrant of β be m , and the quadrant pair a_{lm} denotes the block of row l and column m in the Dubins decision table. Any transition point r_n denotes a change of l or m . Therefore between r_n and r_{n-1} , l and m stay constant. Therefore a_{lm} is the same for all α and β between the transition points. Any of such values can be used to find a_{lm} and the corresponding set of candidate optimal trajectories $\mathcal{T}$ from the block in the decision table. $\square$

We define the point where $\alpha(t)$ or $\beta(t)$ transitions to a new quadrant as r_n , where $n \in N$, N is the total number of transition points, and Γ is the ordered set of transition points.

The first quadrant transition for $\alpha(t)$ is at $\theta_{\alpha 1} = \delta\theta_\alpha + \theta(0)$ where $\delta\theta_\alpha$ is the rotation for $\alpha(t)$ to transition to a new quadrant and is positive when y_g^w is positive and negative when y_g^w is negative. This is due to $\theta(t)$ rotating in the positive direction when the goal state $\mathbf{x}_g^w(0)$ is above the start state $\mathbf{x}_s^w$ and rotating in the negative direction when the goal state is below the start state.

If the rotation from $\theta(0)$ to $\theta_{\alpha 1}$ doesn't cross the x axis, then the transition point can be computed by

$$r_n = x_g^w - x_s^w - \frac{y_g^w - y_s^w}{\tan(\theta_{\alpha 1})} \quad (6.5)$$

which is the distance from $\mathbf{x}_g^w(0)$ where $\mathbf{x}_g^w(t)$ intersects with the line from $\mathbf{x}_s^w$ in the $\theta_{\alpha 1}$ direction. There is no intersection point if $\theta_{\alpha 1}$ is directed away from the goal state trajectory.

The following transition point angle $\theta_{\alpha 2}$ is $\frac{\pi}{2}$ from $\theta_{\alpha 1}$ in the same direction as $\delta\theta_\alpha$. Just as before, if $\theta_{\alpha 2}$ does not cross the x axis, then the next transition point is calculated with (6.5) using $\theta_{\alpha 2}$. This process is then repeated for $\beta(t)$ to find its transition points. An example configuration can be seen in Fig. 6.3. There can be at most 4 total transition points, as a rotation of $\frac{\pi}{2}$ from either of the latter points would lead to an angle pointing away from the goal state trajectory.

The transition points are then iterated over to check if the solution lies before that point. First, the segment between the transition point and the previous one is checked to satisfy the $d > 4R$ condition. If the check is satisfied, the time it takes to go from the goal state to the transition point r_n is computed by $t_{goal} = r_n/V_w$. Similarly, the time for the vehicle to travel from the start state to the transition point is also computed by finding the Dubins path between the points and then solving for the time with $t_{start} = l_p/v$, where l_p is the path length. If $t_{start} < t_{goal}$, it means that the solution lies between the transition points. To find the quadrants for the section, we compute the angle of the midpoint between the transition points with

$$\theta_q = \tan^{-1} \left(\frac{y_g - y_s}{x_g - x_s - \frac{r_n + r_{n-1}}{2}} \right) \quad (6.6)$$

Using the midpoint rather than the transition point ensures that the lookup of the quadrants does not lie on a boundary.

If the solution does not lie between any of the transition points ($t_{start} > t_{goal}$ for all transition points), then the solution lies beyond the last transition point. If that

$Q_s \backslash Q_g$	1	2	3	4
1	RSL	RSR RSL LSR	RSR LSR	LSR RSL RSR
2	LSL RSL LSR	LSL RSL RSR	RSR	RSR RSL
3	LSL LSR	LSL	RSR LSR LSL	RSR LSR RSL
4	RSL LSR LSL	LSL RSL	LSL LSR RSL	LSR

Table 6.1: The above Dubins decision table is adapted from [Shkel and Lumelsky \[2001\]](#) with corrections to $\{a_{12}, a_{21}, a_{34}, a_{43}\}$. "R" represents a right-turn segment, "S" represents a straight segment, and "L" represents a left-turn segment. Q_s is the quadrant of the start state, and Q_g is the quadrant of the goal state.

happens, rather than computing the angle of the midpoint between transition points, we directly compute the θ_q as halfway between the last transition point and π .

After finding θ_q , the associated initial and final angles α_f and the β_f are found using the following equation:

$$\begin{aligned}\alpha_q &= \psi_s^w - \theta_q \\ \beta_q &= \psi_g^w - \theta_q\end{aligned}$$

We can then find the quadrant l of α_q and the quadrant m of β_q , creating the pair a_{lm} . We check the corresponding block in the Dubins decision table and let $\mathcal{T}$ represent the set of candidate trajectories within that block.

During the implementation and testing of the Dubins decision table based on [Shkel and Lumelsky \[2001\]](#), inconsistencies between the decision table and result were seen, particularly in the no-wind cases. This error was validated by computing and comparing all six admissible paths in the Dubins set. We found that the decision table sometimes does not give the actual shortest path for quadrant pair $a_{lm} \in \{a_{12}, a_{21}, a_{34}, a_{43}\}$. These four cases are equivalent with symmetry, and we describe the issue in detail within the Appendix B. A corrected version of the Dubins decision table is also provided in Table 6.1.

After finding $\mathcal{T}$, we solve the trochoid paths within the set using the method described in [Techy and Woolsey \[2009\]](#) and choose the one within minimum time, but we now no longer have to iterate over all candidate solutions in most cases. As seen in Table 6.1, all blocks have less than four path types, with some having only one.

Only when the solution is less than $4R$ do all path types have to be checked.

For paths of type RSR or LSL, the solutions can be found analytically using the values

$$x_{s0} = x_s^w - \frac{V_a}{\delta_1 \omega \sin(\psi_s^w)} \quad (6.7)$$

$$y_{s0} = y_s^w + \frac{V_a}{\delta_1 \omega \cos(\psi_s^w)} \quad (6.8)$$

$$x_{g0} = x_g^w - \frac{V_a}{\delta_2 \omega \sin(\delta_2 \omega t_{2\pi}) + \psi_g^w} - V_w t_{2\pi} \quad (6.9)$$

$$y_{g0} = y_g^w + \frac{V_a}{\delta_2 \omega \cos(\delta_2 \omega t_{2\pi}) + \psi_g^w} \quad (6.10)$$

and the point of exit of the first bank t_1 and the point of entry to the second bank t_2 for LSL and RSR trajectories can be found with

$$\gamma = \tan^{-1} \left(\frac{y_{g0} - y_{s0}}{x_{g0} - x_{s0} + V_w \frac{\psi_s^w - \psi_g^w + 2k\pi}{\delta_2 \omega}} \right) \quad (6.11)$$

$$t_1 = \frac{t_{2\pi}}{\delta_1 2\pi} \left(\sin^{-1} \left(\frac{V_w}{V_a} \sin(\gamma) \right) + \gamma - \psi_s^w \right) \quad (6.12)$$

$$t_2 = t_1 + \frac{\psi_s^w - \psi_g^w + 2k\pi}{\delta_2 \omega}, k \in \{-3, -2, -1, 0, 1, 2\} \quad (6.13)$$

where $\delta_1 \in \{-1, 1\}$ and $\delta_2 \in \{-1, 1\}$ result in a left or right turn for the first and second bank, $\omega = |\dot{\psi}_{max}|$, $t_{2\pi} = 2\pi/\omega$, and $\psi_s^w - \psi_g^w$ is modulo 2π .

If the solution is of RSL or LSR, they can be found using a numerical root-finding technique such as the Newton-Raphson solver to find the roots of the equation

$$f(t_1) = E \cos(\delta_1 \omega t_1 + \psi_s^w) + F \sin(\delta_1 \omega t_1 + \psi_s^w) - G \quad (6.14)$$

where,

$$E = V_a \left(V_w \frac{\delta_1 - \delta_2}{\delta_1 \delta_2 \omega} - (y_{g0} - y_{s0}) \right) \quad (6.15)$$

$$F = V_a \left(x_{g0} - x_{s0} + V_w \left(t_1 \left(\frac{\delta_1}{\delta_2} - 1 \right) + \frac{\psi_s^w - \psi_g^w + 2k\pi}{\delta_2 \omega} \right) \right) \quad (6.16)$$

$$G = V_w (y_{g0} - y_{s0}) + \frac{V_a^2 (\delta_2 - \delta_1)}{\delta_1 \delta_2 \omega} \quad (6.17)$$

and then using t_1 to find t_2 using (6.13). Because there are multiple solutions, each has to be checked to find the optimal trajectory.

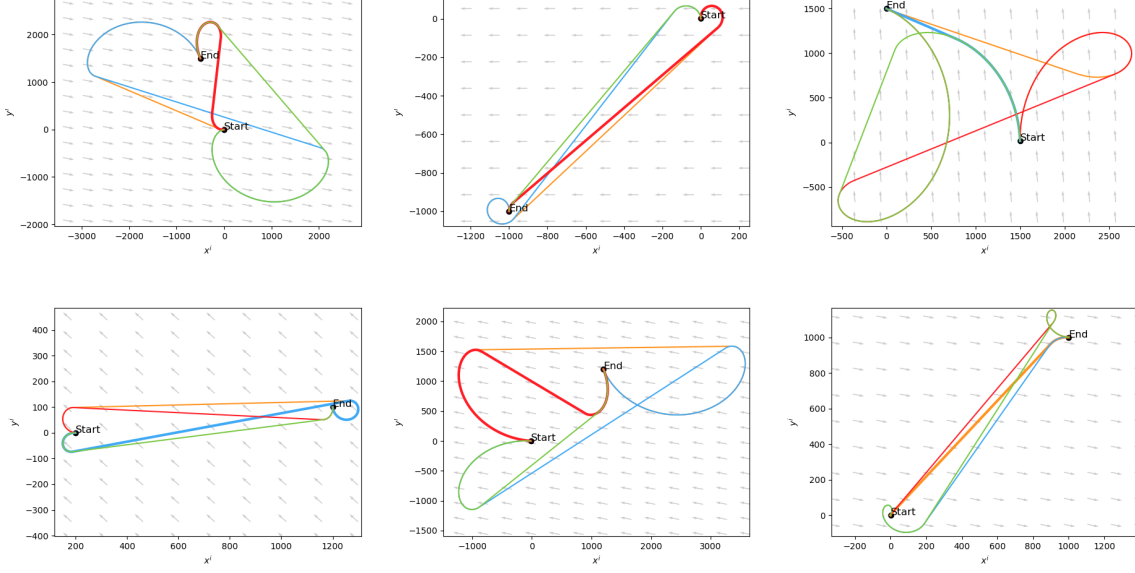


Figure 6.4: Randomly sampled candidate BSB trajectories for randomly selected initial and final states in a uniform wind-vector field. RSL is shown as red, RSR as orange, LSL as green, and LSR as blue. The bolded path in each subfigure is the optimal one.

After finding t_1 and t_2 , the trochoidal paths can be constructed using

$$x_s^w(t) = \frac{V_a}{\delta_1 \omega} \sin(\delta_1 \omega t + \psi_s^w) + V_w t + x_{s_0} \quad (6.18)$$

$$y_s^w(t) = \frac{V_a}{\delta_1 \omega} \cos(\delta_1 \omega t + \psi_s^w) + y_{s_0} \quad (6.19)$$

$$x_g^w(t) = \frac{V_a}{\delta_2 \omega} \sin(\delta_2 \omega t + \psi_g^w) + V_w t + x_{g_0} \quad (6.20)$$

$$y_g^w(t) = \frac{-V_a}{\delta_2 \omega} \cos(\delta_2 \omega t + \psi_g^w) + y_{g_0} \quad (6.21)$$

where $t \in [0, t_1]$ for the first two equations to construct the first banking segment, and $t \in [t_2, t_{2\pi}]$ for the last two equations to construct the second banking segment. For derivations of (6.7-6.21) and more in-depth explanations, see [Techy and Woolsey \[2009\]](#).

The C++ package for our method is found at <https://github.com/castacks/trochoids>. The README contains instructions for how to build, run, and test the codebase. Since the target application for this planner is UAVs, we also wanted to be able to handle changes in altitude between start and goal states. To do this we simply added a linear interpolation of the altitude between the start and goal states for all points in the path. Other methods could include only changing altitude on the straight segment or adding helical paths at the end or beginning to climb or descend.

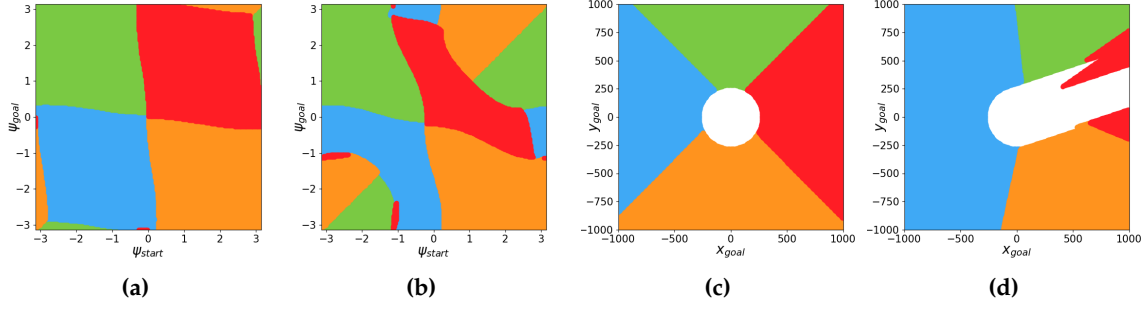


Figure 6.5: The first two figures show the distribution of solutions across varying ψ_s^i and ψ_g^i values and illustrate the difference between cases with wind speed zero (6.5(a)) and with wind speed 12.65 m/s (6.5(b)). The last two figures show the distribution of the solutions across varying start and goal states with wind speed zero (6.5(c)) and wind speed 12.65 m/s (6.5(d)). RSL is shown as red, RSR as orange, LSL as green, and LSR as blue.

6.5 Testing Results

We validate our proposed method by performing Monte-Carlo simulations. The start and goal poses x_s^i, y_s^i, x_g^i and y_g^i were randomly sampled from a uniform distribution in the range $[-1000, 1000]$. Similarly, ψ_s^i, ψ_g^i , and ψ_w^i were uniformly sampled in the range $[0, 2\pi]$. Additionally, the wind speed V_w was randomly sampled in the range $[1, 15]$ m/s, and the turning radius R_0 was sampled in the range $[10, 1000]$ m. The UAV airspeed was set as $V_a = 20$ m/s throughout all tests. Fig. 6.4 shows six randomly sampled trajectories.

The average run-time over 10,000 sampled parameters for Bang-Straight-Bang trajectories on a 3.40 GHz CPU was 1.4224 ms with a standard deviation of 8.006×10^{-5} ms as compared to the baseline method, where two analytical solves and two numerical solves were iterated over, and the shortest path length selected [Techy and Woolsey \[2009\]](#). This baseline method had an average solve time of 2.2711 ms with a standard deviation of 2.97077×10^{-5} ms. The testing results yielded a 37.4% improvement over the baseline method, and the P value for the single tail difference between the two means is well less than 0.0001. We found that 91.73% of the 10,000 cases satisfy the $d > 4R_0$ condition mentioned in Section 6.4, which determines the optimal trajectories are of type Bang-Straight-Bang.

Table 6.2 shows the distribution of the Bang-Straight-Bang trajectories for the 10,000 runs, and the optimal candidate trajectory types are evenly spread out at approximately 25%. The distribution for the quadrant pairs of the tests within the Dubins decision table can be seen in Table 6.3. The overall distribution of trajectories within a specific block for the 10,000 randomly sampled cases was also somewhat equivalent, showing full coverage of the testing domain.

Path Type	Distribution
LSL	25.8%
LSR	24.88%
RSL	24.6%
RSR	24.68%

Table 6.2: The distribution of time-optimal path types based on Monte-Carlo simulations for Bang-Straight-Bang trajectories.

We also visualize and compare the distributions of the optimal trajectory types for Dubins and trochoidal paths over various conditions in Fig. 6.5. For these figures, $V_a = 20$ m/s, $V_w = 12.65$ m/s, $\psi_w = 0.322$ rad, and $R_0 = 70$ m. Fig. 6.5(a) shows the optimal Dubins trajectories, where the wind is ignored, for $x_s^i = 0$, $y_s^i = 0$, $x_g^i = 470$ and $y_g^i = 0$, and the initial and goal angles ψ_s^i and ψ_g^i are sampled from $[-\pi, \pi]$. The distribution of the trajectory types for wind cases can be seen in Fig. 6.5(b), depicting a shift in the decision boundaries for the wind cases. Similarly, start and goal angles were held constant at $\psi_s^i = \pi/4$ and $\psi_g^i = 3\pi/4$ and the start and goal positions were sampled from a range of $[-1000, 1000]$ m with the same R_0 and V_w as mentioned above. The distribution for Dubins trajectories (ignoring the wind) can be seen in Fig. 6.5(c), whereas the distribution of trochoidal trajectories can be seen in Fig. 6.5(d). The white hole in the center indicates states that do not satisfy the $d > 4R_0$ condition and thus have not been included in the plot. The shift in trajectory type distributions from the effects of wind shows why the same switching functions from Shkel and Lumelsky [2001] cannot be used directly for choosing the optimal trajectories.

To validate the robustness of this approach, we tested 5 million randomly sampled start states, goal states, wind speeds, and turning radii and compared the trajectory outputs to the numerical solution as well as the analytical solution when available. Through this rigorous testing, we found the aforementioned error in the Dubins decision table, which occurred in only 3 out of a million test cases. Fixing this error necessitated adding an additional numerical solve to four quadrant pair cells. Adding these extra numerical solves affected the bounds of the maximum time improvements of our method. With the baseline having two numerical solves for every solution, we were initially able to reduce the average number of numerical solves to 1 using the original decision table, achieving approximately a 50% improvement from the baseline. However, to make this approach more robust, fixing the error in the decision table caused the average number of numerical solves to increase to 1.25, giving a maximum theoretical speed reduction of 37.5%.

Decision table block	Distribution
a_{11}	5.26%
a_{12}	5.48%
a_{13}	5.46%
a_{14}	4.96%
a_{21}	5.51%
a_{22}	5.40%
a_{23}	5.44%
a_{24}	5.30%
a_{31}	5.99%
a_{32}	5.37%
a_{33}	5.78%
a_{34}	5.72%
a_{41}	5.54%
a_{42}	5.79%
a_{43}	5.16%
a_{44}	5.32%

Table 6.3: The distribution of Dubins decision table blocks from the Monte-Carlo simulations.

6.6 Conclusion

Based on the extensive testing results, the proposed method to compute time-optimal paths for curvature-constrained vehicles in a uniform wind outperforms baseline methods. The proposed approach reduces computation time, which is particularly important for real-time autonomous navigation systems such as UAVs, AUVs, or ASVs, especially in scenarios with limited computational capacities onboard. The future direction of this work is to explore fully implementing the switching functions for each quadrant block in the decision table rather than iterating over the potential trajectories in each cell.

Chapter 7

CVaR-based Flight Energy Risk Assessment using a Deep Energy Model

In the previous chapters, we focused on estimating wind fields and planning time-optimal paths in wind-affected environments to ensure robust UAV navigation. However, another critical challenge for UAV operations, particularly in long-duration missions, is managing energy consumption. Limited battery capacity remains one of the major obstacles to large-scale commercial deployment of UAVs [Murray and Chu, 2015], despite recent advancements in improving battery energy density [Kerman et al., 2017, Li, 2019] and energy management systems [Boukoberine et al., 2019]. While these innovations provide some relief, the reality is that battery limitations continue to pose a significant operational risk.

This risk is exacerbated by the difficulty in accurately estimating the factors that influence UAV energy consumption, such as wind conditions, air density, payload, UAV design, and airspeed [Thibbotuwawa et al., 2018]. Furthermore, the inherent uncertainty in these factors adds an additional layer of complexity when calculating energy usage during flight. To address this challenge, this chapter presents a method for quantifying flight risk based on UAV energy consumption estimates, taking into account these various contributing factors such as wind fields. For the purposes of this chapter, we focus on multirotor UAVs, though the method can be generalized to other platforms as well.

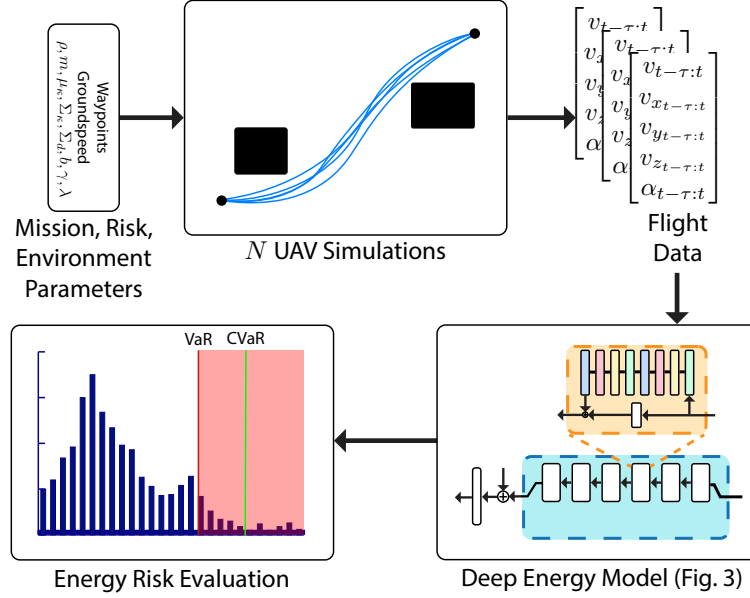


Figure 7.1: For a given mission, risk profile, and environmental uncertainty, our algorithm uses a deep energy model in a Monte Carlo forward simulation to generate a energy distribution that informs a CVaR-based risk assessment.

7.1 Introduction

While fixed-wing UAV systems benefit from decades-long research in energy models for fixed-wing crewed aircraft, multirotors can leverage some of the research in rotary-wing aircraft but generally lack such a comprehensive research background by the virtue of the recent development and more complicated physical interactions of UAVs.

A recent assessment [Zhang et al., 2021] found that while there are multiple attempts at formulating energy model for UAVs, the model estimates differ by a factor of 3-5. The paper attributes this discrepancy to the numerous assumptions in modeling the underlying dynamics of various UAV configurations. The authors of this work argue that the lack of a standardized empirical dataset makes it difficult to compare individual models. Broadly, the prevalent energy models can be grouped into two major categories [Prasetya et al., 2019]. The first are the analytical models that refer to the first-principle based models that rely on established helicopter theories to calculate the energy consumption. The parameter values for these models are either calculated using historical data [Kirschstein, 2020] or by using a regression-based approach with empirical flight tests [Liu et al., 2017]. One drawback of using analytical models is that they inherently make simplifying assumptions while formulating the equations. Rotor-Rotor interactions, crosswind effects, and the effect of higher order motions on the propulsion system are often ignored. On the other hand, end-to-end approaches

completely avoid constructing any explicit models but directly use data-driven methods to regress parameters. The regression is often done by fitting predetermined polynomials to empirical data [Abeywickrama et al., 2018, Maekawa et al., 2017]. Within this paradigm, Prasetia et al. [2019] have presented an approach with multiple polynomial regressions to handle different maneuvers. While the results look promising, the proposed method is complicated and involves generating the movement labels in the dataset to identify various flight maneuvers. The aforementioned methods also do not explicitly take into account external factors like wind or air density which limits their accuracy. A deep learning based model is more expressive than a simple polynomial model. It has been shown recently [Bai et al., 2018] that compared to other temporal architectures, TCNs are more flexible, stable and less memory intensive.

While risk assessment for aerial robots operating in real world is a relatively new research domain, techniques previously developed for performing risk assessments in the domain of applied finance [Rockafellar et al., 2000, Shapiro et al., 2014] can be leveraged to perform similar analysis for robot operations. Majumdar and Pavone [2020] argue that the traditional risk metrics in robotics like mean-variance may misrepresent the risk associated with a stochastic process. They propose using more informative metrics like Conditional-Value-at-Risk (CVaR) which has been used in some previous works as an optimization constraint to limit the worst-case risk exposure in robotics applications [Hakobyan et al., 2019, Sharma et al., 2020].

In this chapter, we propose the use of CVaR as a risk metric to quantify the risk associated with worst-case energy consumption. Focusing the analysis on the tail-end of the distribution gives a better idea of the associated worst-case energy use. The energy distribution is obtained by using a Temporal Convolutional Network (TCN) based energy consumption model. The model is trained, validated, and tested on a subset of the energy usage dataset collected using more than 10 hours of real-world flying time [Rodrigues et al., 2020, 2021]. A Monte-Carlo (MC) forward simulation setup is used to capture the stochasticity of the environment and the UAV dynamics model to capture their effects on the energy consumption. A CVaR value is then generated by mapping the energy distribution to a risk distribution using a user-defined risk profile and battery capacity. Novel case studies using high-fidelity Computational Fluid Dynamics (CFD) wind data using models of real urban areas provide a glimpse on the insights our risk assessment offers.

The major contributions of the work in this chapter are as follows:

- We propose a deep learning based end-to-end model for energy estimation. The model achieves state-of-the-art results for total energy as well as power consumption estimation on a public dataset.
- We present a stochastic simulation framework that performs MC runs to provide

a distribution on the energy usage given a planned path and environmental conditions.

- We provide a CVaR-based risk assessment framework that uses a mapping between the energy distribution to a risk-space using a user-defined risk profile and battery capacity.

The sections are organized as follows: Section 7.3 provides details on the energy model and its results. Section 7.4 provides details on the methods of calculating risk using the energy model and a Monte-Carlo setup. Section 7.5 gives two case studies to demonstrate applications of the risk method. Section 7.6 presents the discussions and conclusions.

7.2 Problem Definition

Let $x_t \in \mathbb{R}^3 \times SO(3)$ represent the state of the UAV in inertial space at time t . Let $\xi : [0, t_f] \mapsto \mathbb{R}^3 \times SO(2)$ be a given nominal vehicle trajectory with yaw that the UAV follows using a controller $\pi : (x_t, \xi) \mapsto u_t$ under system dynamics as defined in (7.1), where d is the additive noise in system dynamics and κ is the environmental context related to features like ambient wind.

$$\ddot{x}_t = f(x_t, u_t, \kappa) + d \quad (7.1)$$

This work assumes that the disturbances lie within the set manageable by the controller, the planned trajectory is feasible, and real-time trajectory modification is not necessary.

Given battery usable capacity b , random variable for energy consumption $e \in E$, where $E \in \mathbb{R}_+^1$ is energy-space, our objective is to provide a pre-flight energy-based risk assessment by calculating a suitable risk metric on a empirically derived risk distribution $p(r)$, $r \in R$ where $R \in \mathbb{R}^1$ is the risk-space. Incorporating collision risk is left for future work.

7.2.1 Approach

We address the problem of finding the risk distribution and the corresponding metric by first finding an energy distribution $J = \{e|\kappa, d\}$ and then transforming it into risk-space using a user-based risk profile $G(e, b) : E \mapsto R$. To do this, we first construct a power consumption model $Q(\eta_{t-\tau:t})$, where $\eta_{t-\tau:t}$ is a set of derived features from state history $x_{t-\tau:t}$ and κ . We then use N MC forward simulations to find the energy distribution of a UAV while flying the trajectory ξ , under a pre-defined distribution of

environmental noise $\kappa \sim N(\mu_\kappa, \Sigma_\kappa)$ (epistemic uncertainty) and a distribution of system dynamics noise $d \sim N(0, \Sigma_d)$ (aleatoric uncertainty), using (7.2).

$$e = \int_0^{t_f} Q(\eta_{t-\tau:t}) dt \quad (7.2)$$

The MC generates a energy histogram with M bins which is then transformed from energy-space to a risk-space using the risk-profile $G(e, b)$ to approximate $\hat{p}_N(r)$. The risk-probability for r in a given bin R_l is given by (7.3).

$$\hat{p}_N(r) = \frac{M}{N} \sum_{i=1}^N I(G(e, b) \in R_l) \quad (7.3)$$

The risk-distribution can now be analysed using metrics like Conditional Value-at-Risk (CVaR) which the authors argue is a relevant metric in that it effectively captures the risk posed by the tail-end distribution. The CVaR value can then be used as part of a comprehensive evaluation that combines other risk factors such as risk to property and people on the ground to aid in a go/no-go pre-flight decision. An overview of our approach can be seen in Fig. 7.1.

7.3 Energy Modeling

This section details the construction of the power consumption model Q . In order to estimate the power consumed by the UAV, we introduce an end-to-end model that takes time varying features and invariant features, and estimates the power consumption of the UAV at each time step. This is done by framing the problem as a multivariate time-series regression problem.

7.3.1 Energy Data Description

We use a publicly available dataset [Rodrigues et al. \[2020, 2021\]](#) which contains power and state measurements for 209 flights on a DJI Matrice M100 drone. 195 of these flights were performed while following a fixed triangular trajectory (see Fig. 7.2b) with varying altitudes, speeds, and payloads, while the remainder of 10 flights (referred to as random flights) followed a different flight path (see Fig. 7.2c) for unique testing data. We split the flights as 60:20:20 for the train, validation and test sets of flights, while making sure that the 10 random path flights were only in the test set. We used a total of 114 training flights (4 hours and 4 minutes flight time), 38 validation flights (1 hour and 24 minutes flight time) and 37 test flights (1 hour and 23 minutes flight time).

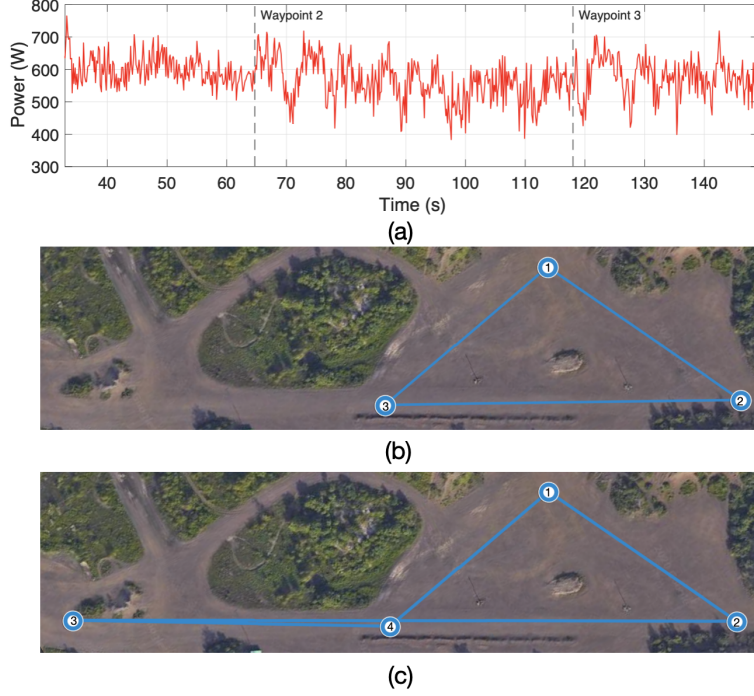


Figure 7.2: Figure shows the dataset used for training the energy model. The top image (a) shows power output for the GPS route (b) that is used for the majority of the dataset. The bottom image (c) shows the route used to create unique variations for the testing set.

time).

7.3.2 Energy Model Architecture

Fig. 7.3 provides an overview of the deep energy model. We construct and split η_t as the time varying features η_t^v and time invariant features η^i . The time varying features $\eta_t^v = \{v_t, v_{x_t}, v_{y_t}, v_{z_t}, \alpha_t\}$ are airspeed v_t , its components in body-frame v_{x_t} and v_{y_t} , vertical speed v_{z_t} , and the angle of attack α_t ¹. The time-invariant features $\eta^i = \{\rho, m\}$ are the density of air ρ , and the mass of the payload m . A vector Φ_t containing temporal embedding of features is obtained from the temporal encoder.

The contextual information is then added to Φ_t by concatenating time-invariant features to create an input $\Phi'_t = \Phi_t \oplus \eta^i$, containing both temporal as well as contextual information. Φ'_t is then fed into a dense layer to estimate the power consumption of the UAV at time t . There are two reasons we separated the time-varying and fixed inputs. The first is we noticed that separating the inputs improves the convergence rates, and second is that it makes the model modular.

¹The angle of attack is approximated to pitch angle as the dataset does not include explicit angle of attack information.

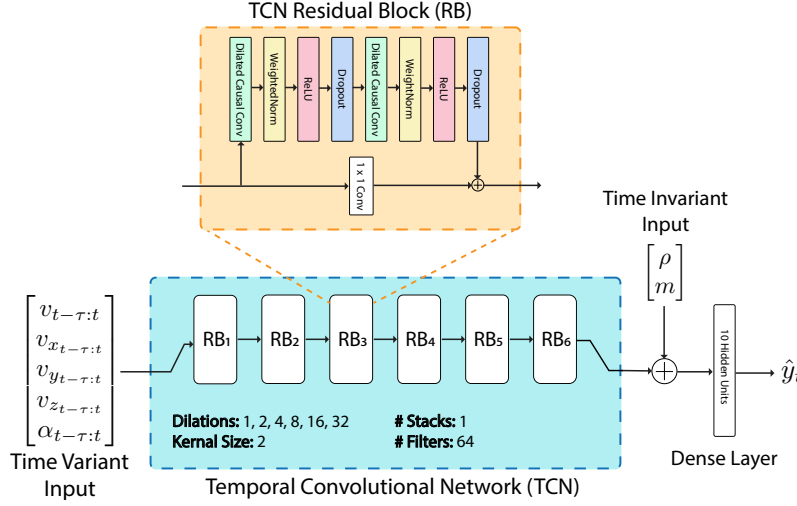


Figure 7.3: Overview of the deep energy model. The time varying features are passed into the TCN, whereas the time invariant features are concatenated to the output of the TCN and passed through a dense layer. The hyperparameter values correspond to b-TCN.

Model	Random Flights		Test Flights		# params
	MAPE(%)	RE(%)	MAPE(%)	RE(%)	
Liu [110]	14.05	9.16	12.70	7.41	5
b-LSTM	13.58	10.35	11.47	7.57	84,649
s-LSTM	15.35	11.65	12.36	8.24	5,833
b-TCN	10.38	7.38	9.06	5.12	76,073
s-TCN	10.36	7.56	9.13	5.52	5,225

Table 7.1: Energy Model Results

In order to encode the time-varying information, we chose a TCN-based [Oord et al., 2016] architecture. In the interest of comparative analysis, we also tested LSTMs [Hochreiter and Schmidhuber, 1997] for the temporal modeling. For the TCNs, we specifically used the implementation as described in Bai et al. [2018]. TCNs have been shown to perform at least at or better than LSTMs in a variety of temporal modeling tasks with added advantages including lower memory requirements, flexible receptive field sizes, and stable gradients.

7.3.3 Model Experiments

We perform a hyperparameter search for both TCNs and LSTMs. For TCNs, we vary the number of filters in each causal convolution unit between $\{16, 32, 64, 128\}$, the

kernel size between $\{2, 3, 4\}$, the number of residual block stacks between $\{1, 2\}$, and the number of layers between $\{3, 4, 5, 6\}$. We use the Adam optimizer with a learning rate of 0.002 and make use of exponential dilation [Bai et al., 2018]. We selected the top two models LSTM and TCN models based on their performance on the validation set and evaluate them on the test set. The results of which are in Table 7.1 and are further discussed in Section 7.3.5.

7.3.4 Metrics

We use the mean absolute percentage error (MAPE) to measure the performance of the models on the power estimation task. The MAPE for a flight i , is defined in (7.4), where $t = [0, t_f]$, and $y_t^{(i)}$ and $\hat{y}_t^{(i)}$ are the true and predicted value of power for flight i at time step t , respectively.

$$\text{MAPE}_i = \frac{1}{t_f^{(i)}} \sum_{t=0}^{t_f^{(i)}} \left| \frac{y_t^{(i)} - \hat{y}_t^{(i)}}{y_t^{(i)}} \right| \times 100\% \quad (7.4)$$

For measuring the model performance on the energy estimation task, we first divide the flight into sections, $s \in \{1, \dots, S^{(i)}\}$, based on the yaw angle. We then calculate the energy consumption in each section, $e_s^{(i)}$, and use these to calculate the adjusted relative error for flight i , RE_i in (7.5), where $\hat{e}_s^{(i)}$ is the predicted energy for each section.

$$\text{RE}_i = \frac{1}{S^{(i)}} \sum_{s=1}^{S^{(i)}} \left| \frac{e_s^{(i)} - \hat{e}_s^{(i)}}{e_s^{(i)}} \right| \times 100\% \quad (7.5)$$

This metric forces the model to accurately predict total energy consumed in all portions of the flight and not have errors from different parts of a flight cancel out.

7.3.5 Energy Model Results

Based on their performance on the validation set, we selected the top two LSTM and TCN models. The *b*-TCN model is the bigger TCN model with 64 filters, kernel size 2, 1 residual block stack, and 5 layers. The *s*-TCN model is the smaller TCN model with 16 filters, kernel size 2, 1 residual block stack, and 5 layers. Both *b*-LSTM and *s*-LSTM, had 3 hidden layers with 0 dropout and were trained using RMSprop optimizer. *b*-LSTM had 64 hidden units, whereas *s*-LSTM had 16 hidden units.

From Table 7.1, we see that the TCNs consistently outperform the LSTMs on the power as well as energy estimation tasks. Additionally, note that even though the *s*-TCN is about 93% smaller than the *b*-TCN, the performance of both is similar. This



Figure 7.4: Test Flights with the best and worst MAPE for b-TCN. The bar graphs in the bottom right corners represent the total energy estimate over the entire flight.

shows that the TCN architecture lends itself better to model compression and hence might be better for edge applications by using less memory and faster inference. However, it should be noted that the s-TCN takes considerably more number of epochs to train. Fig. 7.4 shows time-series plots of the best and worst flights for the b-TCN. We can see that the model is able to follow the peaks and troughs of power consumption.

7.4 Risk Modeling

7.4.1 Wind Modeling

An important factor in predicting energy consumption for a UAV is accounting for the effects of the wind field. In order to accurately simulate the energy usage of a given trajectory, an accurate wind model must be included in the environmental context κ for the system dynamics of (7.1). For this work, wind is broken down into two components: constant wind and turbulence.

The constant wind is the averaged wind field which is temporally static but spatially varying for a given environment. Given an accurate model of the environment, the current wind field can be found using the wind inlet angle Θ and magnitude U by using CFD. CFD has been shown to be an effective method in calculating the low altitude urban wind flow patterns for UAV flight [Galway et al., 2011, Orr et al., 2005, Patrikar et al., 2020, Ware and Roy, 2016]. However, wind fields are not temporally constant because the inlet conditions are not steady and therefore need to be defined

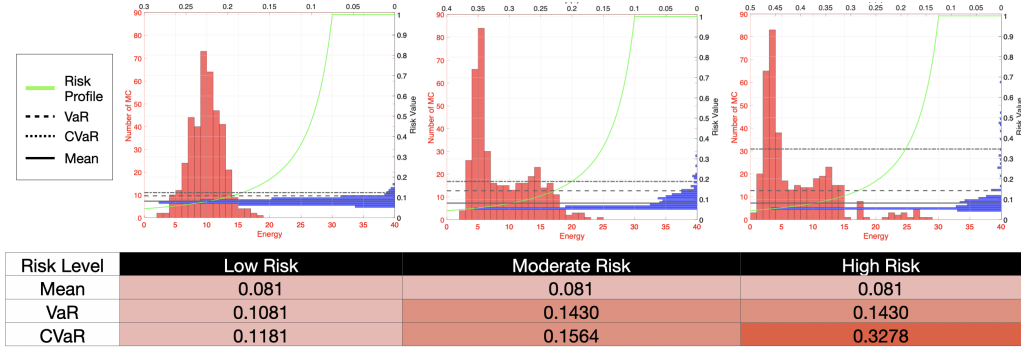


Figure 7.5: Illustrative example to show how CVaR is a more relevant risk metric than mean and VaR as it captures the tail-end distribution better.

as a distribution. The mean and standard deviations of the inlet angle and magnitude are given as $\mu_{\kappa} = \{\mu_{\Theta}, \mu_U\}$ and $\Sigma_{\kappa} = \text{diag}(\sigma_{\Theta}, \sigma_U)$ respectively and can come from historical data spanning over years or even an hour depending on the application.

In addition to the constant wind, turbulence is added using the Dryden turbulence model [DoD, 1997]. Previous works [Abichandani et al., 2020, Allison et al., 2018, Beard and McLain, 2012, Waslander and Wang, 2009] have shown that this model can be adapted for simulation of wind turbulence for small UAV.

7.4.2 Monte Carlo Simulations

Given μ_{κ} , Σ_{κ} , and Σ_d , we run N MC simulations to create a set of possible vehicle trajectories given the distribution of conditions. The simulated vehicle follows the trajectory ξ with the controller π and the dynamics defined in the previous section.

7.4.3 Risk Modeling

As defined in Section 7.2, $G(e, b)$ represents the user-defined operation risk profile. In this work, as a test case, $G(e, b)$ can be chosen as in (7.6).

$$G(e, b) = \exp\left(\frac{\gamma}{(\max\{b - e, \lambda\})}\right) - 1 \quad (7.6)$$

where $\gamma, \lambda > 0$ are constants. Such a choice of $G(e, b)$ takes into account the fact that as the energy consumption nears the battery capacity, the risk increases exponentially until it reaches a limit value.

The MC simulations defined in the previous section can be used to generate the distribution for e which in turn can be used to generate the risk probability density

function $p(r)$ by normalizing the output of $G(e, b)$. From here on we follow the formulation proposed by [Rockafellar et al. \[2000\]](#) to define Conditional Value-at-Risk (CVaR).

The Conditional Value-at-Risk at level ν or CVaR_ν gives the expected risk of the worst $(1 - \nu)$ quantile of the distribution $G(e, b)$. Intuitively CVaR_ν gives us an idea on the behaviour of the worst $(1 - \nu)$ quantile cases by providing the expected risk in the cases that are “beyond” Value-at-Risk at level ν (VaR_ν) risk.

$$\text{CVaR}_\nu = \frac{1}{1 - \nu} \int_{G(e, b) \geq \text{VaR}_\nu} G(e, b) p(r) dr \quad (7.7)$$

While the mathematical superiority of CVaR_ν as a *coherent risk metric* [Majumdar and Pavone \[2020\]](#) is useful in formulating convex CVaR_ν optimization problems [Rockafellar et al. \[2000\]](#), it also helps to intuitively see how VaR_ν and CVaR_ν help provide information about a particular risk distribution profile. Fig. 7.5 shows a deliberately constructed example that highlights how the different metrics—mean, VaR_ν , and CVaR_ν —capture the risk profile. The three figures show three different energy MC outputs in red. These outputs are then transformed into risk space by using (7.6) with coefficients $\gamma = 6.93$, $\lambda = 10$, $b = 40$ J. This in turn gives us $p(r)$ or the risk *p.d.f* in blue. VaR_ν and CVaR_ν values are calculated at $\nu = 0.95$. As can be seen, while intuitively it is clear that moving to the right of the three examples, the MC energy output relates to a higher risk flight because of a longer tail, both the mean and VaR_ν are not able to capture this behaviour.

7.5 Case Studies

In order to illustrate potential applications of the presented work, two case studies will be given. The first case study shows how the energy risk for a proposed flight trajectory would be evaluated and could be used before the fly or no-fly command is given. The second case study is an evaluation of all potential flight paths from a given takeoff location to evaluate the coverage area for a proposed takeoff location.

7.5.1 Case 1:

Before a UAV takes off to carry out a nominal path, it would be prudent to evaluate the risk of depleting the battery during flight. Using our proposed method of evaluating risk, the pilot in command could calculate the distribution of necessary energy to carry out the flight path and its CVaR_ν . The environmental conditions for the MC simulations would model the conditions during the proposed flight window.

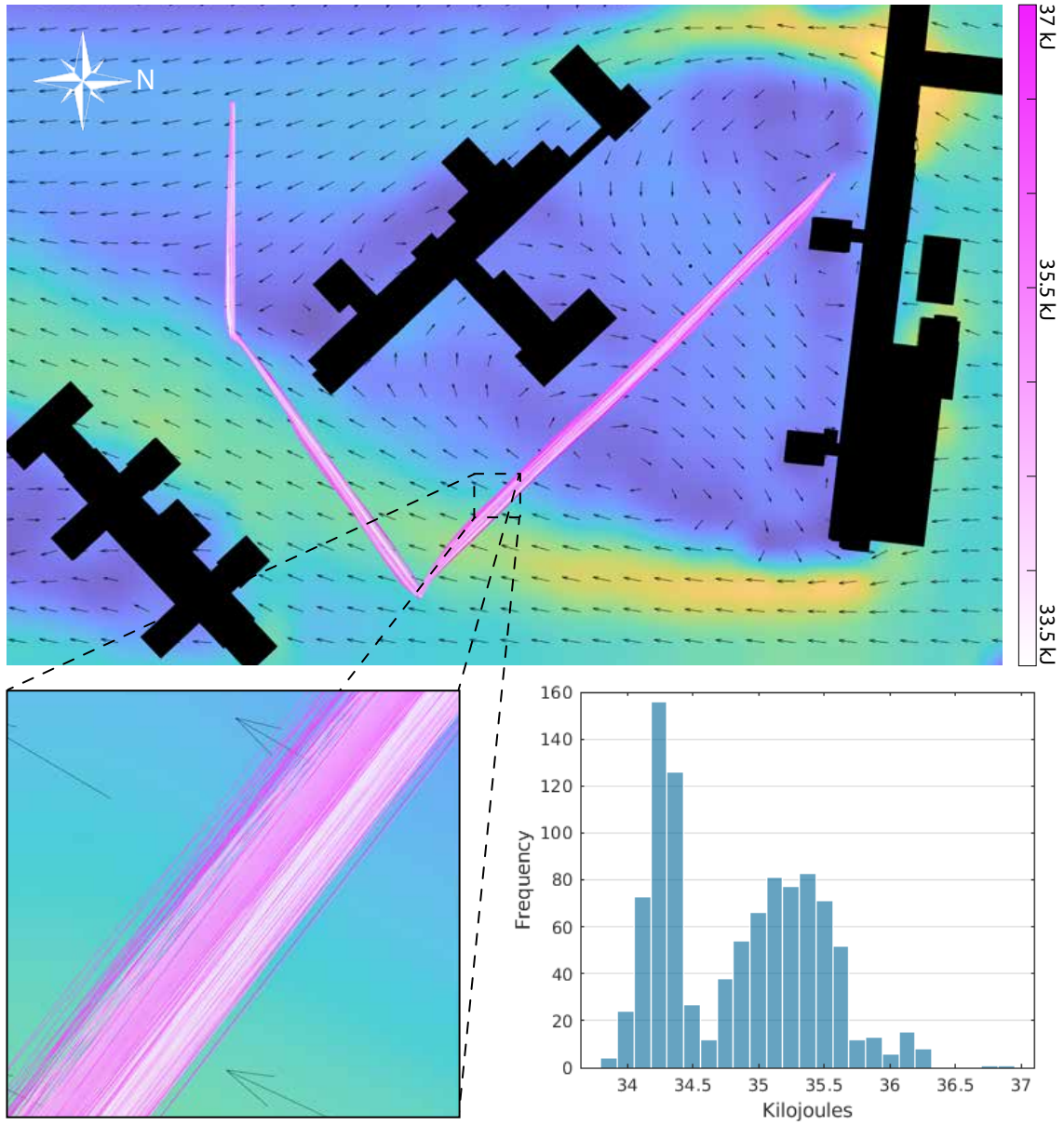


Figure 7.6: Monte Carlo simulations for Case 1. Each flight is represented with a pink line and is shaded according to the total energy consumption. The distribution of the total energy consumption for the flights is shown in the histogram.

To demonstrate this case, real environmental data is used from a test site with around ten buildings ranging in heights of 5 to 35 meters. This area is similar to a typical urban environment with non-uniform wind fields. The area is around 300×250 meters. The wind field model was calculated using CFD, and the wind inlet conditions were captured using a weather station on the tallest building in the area. The window of time for this test was February 19, 2020 from 4:20-4:22 PM. The inlet conditions were $\mu_\kappa = \{-2.53^\circ, 3.14m/s\}$, $\Sigma_\kappa = diag(28.47^\circ, 1.55m/s)$

Fig. 7.6 shows the 1000 MC simulations and the predicted amount of energy for the trajectory. The mean energy consumption was 34.86 kJ. With $b = 369.36$ kJ, $\lambda = 92.34$, $\gamma = 64$, and $\nu = .95$, the $CVaR_\nu$ value is 0.2117. This value reflects how risky this proposed flight path is and would be compared against the allowable risk determined by the operator or company.

7.5.2 Case 2:

In many applications of a UAV, the vehicle will take off and return to the same location every time. In these cases it could be useful to evaluate the coverage area for that particular location. This knowledge could influence the placement of launching location and optimize the number of base stations necessary to ensure full coverage of a region.

The testing environment for this case is a 3300×2130 meter area of a major metropolitan city. This city has a higher density of buildings compared to Case 1. The wind field model for this city came from a high resolution CFD simulation. One starting location is chosen at a point close to the center of the map, and over 1000 goal locations are randomly sampled from the space within a given radius. A BIT* planner [Gammell et al., 2015] finds a path connecting the start and goal states. For this example we used yearly wind statistics as the inlet conditions in order to evaluate the coverage area over a long period of time. Flights for each path are then simulated and evaluated using our risk assessment method.

Fig. 7.7(a) shows a contour plot of the $CVaR_\eta$ to reach each location in the region. Natural corridors lead to lower energy consumption and thus lower risk values. In contrast, regions of the map with dense building placements leads to longer paths, higher energy consumption, and higher risk. With a given risk value limit, the coverage area can be calculated for the given environment conditions.

With an adjusted risk profile to reflect a lower usable battery capacity, the contour plot of the $CVaR_\nu$ values changes dramatically, as can be seen in Fig. 7.7(b). The risk values steadily increase when traversing farther away from the takeoff location and even approaches a risk value of 1.

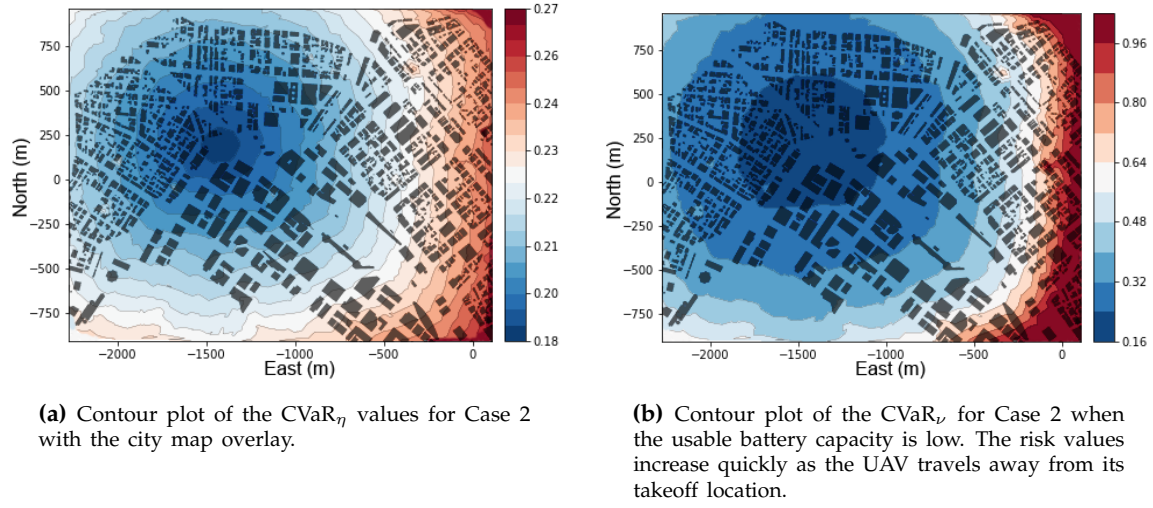


Figure 7.7: Comparison of CVaR risk contours for Case 2 under different UAV conditions.

7.6 Conclusions

The work in this chapter presents a novel, open-source, state-of-the-art, deep-learning based energy model trained on a publicly available dataset to quantify the energy risk for multirotor UAVs. MC forward simulations are used to find a energy distribution which is transformed into a risk-space. CVaR is used to conduct a risk assessment on the risk-space distribution. Case studies using real world data highlight how the proposed framework can be applied.

While the proposed deep energy model applies to various multirotor configurations, we present results only for a quadrotor. Our tests focus on delivery-style missions, excluding aggressive, evasive, or acrobatic maneuvers. Despite these limitations, we demonstrate that CVaR-based risk assessment effectively analyzes multirotor energy consumption.

Part III

Improving and Leveraging World Belief Models

In Parts I and II, we explored various world belief models, either by estimating them—such as in the case of wind field estimation—or by leveraging them for planning and risk quantification. IPP is about maximizing the information gain in our world belief model, but how do we quantify this information gain and what does it represent? Bruyninckx et al. [2021] define world models as “the information the robot has about the world around it, and that needs to be shared between several activities. There can be multiple ‘versions’ of the ‘world’ present at the same time, for example, past, present, and future states.” Expanding on this idea, Sakagami et al. [2023] describe the inferred world belief as playing the role of a world model by integrating information from perception pipelines and providing relevant data for planning activities. In Part III, we examine Challenge 3 to improve and leverage these world belief models.

In Chapter 8, we will present our work that shows how we leverage prior information about the structure of the space to improve data gathering. We present our previous contributions in this area, where we use a global map predictor to expedite indoor map exploration. We then introduce *PIPE*, highlighting the importance of accurately approximating information gain along the entire trajectory. To evaluate our method, we train and test it on an indoor map dataset, showing that it achieves state-of-the-art results.

In Chapter 9, we extend our work on indoor map exploration by drawing insights from human performance on the same tasks. We investigate how humans explore effectively and what strategies contribute to their success. To gain these insights, we conduct a user study in which participants navigate the same scenarios presented in the previous chapter. Building on these findings, we introduce *MapExRL*, a method that enhances robot exploration efficiency in structured indoor environments by incorporating reinforcement learning (RL) and global map predictions for longer-horizon planning.

In our final chapter, Chapter 10, we address the challenge of reconciling two types of belief maps with conditional relationships: one representing where objects might be in a space and the other tracking known objects. To tackle this, we introduce *UniSaT* (Unified Search and Track), a novel formulation of the search and track problem based on Random Finite Sets (RFS). Our approach models both unknown and known objects within a unified framework using a generalized labeled multi-Bernoulli (GLMB) filter. The planner then optimizes mutual information over this unified belief model, enabling a balanced trade-off between search and tracking behaviors.

Chapter 8

Indoor Exploration using Predicted Maps

8.1 Introduction & Motivation

In the previous chapters, our informative path planners operated within a known configuration space, where UAVs were provided with predefined keep-out zones, elevation maps, and obstacles, assuming most of the space was collision-free. However, this assumption does not hold in exploration planning, where the environment is only partially known. In such cases, both the configuration space and the belief space associated with the objective are initially unknown and must be discovered jointly. This creates significant challenges, not only in generating long-horizon collision-free paths but also in accurately evaluating their potential information gain.

Exploration is an important problem in robotics, with applications ranging from indoor search and rescue operations [Sampedro et al., 2019] to planetary missions [Schilling and Jungius, 1996] and outdoor monitoring [Hitz et al., 2014]. The primary objective of exploration is to plan paths in an unknown environment to maximize understanding of the environment within a limited budget. This objective may involve building an accurate map [Yamauchi et al., 1998], identifying objects of interest [Chaplot et al., 2020], and finding traversable routes for robot teams [Zhang et al., 2022]. The key challenge in exploration is choosing how to navigate to maximize information gain, which requires reasoning over unobserved space due to uncertain environment geometry, occlusions, sensor range limits, and sensor noise.

Many real-world environments possess inherent structure, predictability, and repeatability of environment geometry. For example, indoor environments such as offices or hospitals consist of repeated rooms and corridors, where observing partial views of

the space can provide valuable information for predicting the overall layout. However, most existing approaches have difficulty leveraging this predictability for more efficient exploration. For instance, conventional frontier-based exploration [Best et al., 2024, Yamauchi, 1997] do not directly reason over the predictable nature of the environment. Instead, they rely on heuristics such as “visit closest area first”.

To leverage such structural predictability, recent work has explored using deep learning techniques to predict unknown regions of maps, and use such prediction to improve exploration. For instance, IG-Hector [Shrestha et al., 2019] uses a predicted map to estimate how much area can be observed from a viewpoint. However, such viewpoint scoring heuristic is sensitive to the accuracy of the predicted map. UPEN [Georgakis et al., 2022] instead leverages such uncertain map prediction behavior, by explicitly predicting map uncertainty and then going towards these areas. However, this method does not consider sensor coverage or visibility, which do not reflect actual sensor acquisition. Therefore, our key in this chapter is to jointly reason over what the robot can observe and its uncertainty to more accurately estimate the potential information gain for an action.

8.2 Background

8.2.1 Robot Exploration

Exploring an unknown environment and finding paths to construct a map are core challenges in robotics research. Traditional robotic exploration approaches use the concept of frontiers [Yamauchi, 1997], the boundaries between known and unknown space. Robots typically select frontiers to visit using distance-based heuristics [Yamauchi, 1997] or more advanced techniques, such as viewpoint selection and graph-search strategies [Best et al., 2022]. Another category of approaches leverages information theory, where robots plan actions to maximize information gain [Bai et al., 2016, Bourgault et al., 2002]. Various information gain metrics have been proposed, including maximizing 2D sensor coverage [Deng et al., 2020, González-Banos and Latombe, 2002], optimizing 3D volumetric information gain [Bircher et al., 2016, Corah and Michael, 2021, Delmerico et al., 2018], and reducing entropy or uncertainty [Vallvé and Andrade-Cetto, 2014]. Other methods take a topological or graph-based approach, modeling the environment as a structured representation to improve scalability in large spaces [Cao et al., 2021, Kim et al., 2013]. In this work, we adopt an information gain-based approach that combines sensor coverage and map uncertainty reduction to guide exploration.

8.2.2 Map Prediction for Exploration

Recent studies in robotic exploration go beyond using only the currently observed area for path planning; instead, they leverage structural predictability to generate broader map predictions that guide planning. A prominent approach involves using image inpainting techniques to infer unobserved regions of a top-down occupancy map [Chang et al., 2007, Ericson and Jensfelt, 2024, Georgakis et al., 2022, Luperto and Amigoni, 2022, Ramakrishnan et al., 2020, Saroya et al., 2020, Sharma et al., 2023, Shrestha et al., 2019, Tan et al., 2024, Tao et al., 2024, Zwecher et al., 2022], as summarized in a survey by [Tan and Nejat, 2022]. Traditional methods, such as those using Hough line features to find similar maps in pre-built datasets [Luperto and Amigoni, 2022], may struggle with complex geometries. Deep learning-based approaches offer a more expressive alternative, enabling more accurate map predictions.

Several studies integrate deep learning-based map predictors with reinforcement learning-based planners for exploration [Ramakrishnan et al., 2020, Tao et al., 2024, Zwecher et al., 2022]. However, these approaches are typically constrained to short-horizon decisions, such as predicting only a few meters ahead [Ramakrishnan et al., 2020] or selecting a single step in a particular direction [Tao et al., 2024, Zwecher et al., 2022]. Other works combine deep-learning map prediction techniques with classical exploration planning methods, allowing for more explicit long-horizon planning without requiring additional data to learn planning policies [Ericson and Jensfelt, 2024, Georgakis et al., 2022, Luperto and Amigoni, 2022, Luperto et al., 2021, Shrestha et al., 2019].

A key challenge in using predicted maps for exploration is defining an effective information gain metric. One approach estimates information gain by predicting sensor coverage at different viewpoints [Ericson and Jensfelt, 2024, Luperto and Amigoni, 2022, Luperto et al., 2021, Shrestha et al., 2019]. However, this method is highly sensitive to errors in the predicted map. Alternatively, UPEN [Georgakis et al., 2022] accounts for map uncertainty by generating an ensemble of predictions and using their variance to guide exploration toward uncertain areas. While this approach captures uncertainty, it does not explicitly consider potential sensor coverage, which is critical for maximizing actual information gain.

To address these limitations, the work in this chapter jointly reasons over predicted sensor coverage and map uncertainty to define a *probabilistic information gain* metric. This formulation seamlessly integrates with classical exploration frameworks [Yamauchi, 1997], enabling the generation of highly informative exploration paths.

8.3 Problem Definition

We define the indoor robot exploration problem as a robot navigating a 2D indoor environment $\mathcal{E} \subset \mathbb{R}^2$ to build a map of the environment. The robot’s state at time t is given by $\mathbf{x}_t = [x_t, y_t]$, where $x_t, y_t \in \mathbb{Z}$. It perceives its surrounding using a noise-free 2D LiDAR sensor with a range of λ and updates an (observed) occupancy grid map O_t , where each cell is classified as free, occupied, or unknown. The sensor collects l evenly distributed samples per scan over a full 360° field of view. At each time step, the robot moves Δ by selecting an action $a \in \mathcal{A}$, where $\mathcal{A}$ is a movement on an 8-connected grid. The objective of exploration is to determine a feasible path $\{\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_T\}$ that constructs an accurate map representation within a fixed time T .

We define the indoor robot exploration problem as a robot navigating a 2D indoor environment $\mathcal{E} \subset \mathbb{R}^2$ to build a map of the environment. The robot’s state at time t is given by $\mathbf{x}_t = [x_t, y_t]$, where $x_t, y_t \in \mathbb{Z}$. The robot moves by selecting an action $a_t \in \mathcal{A}$, where $\mathcal{A}$ consists of movements on an 8-connected grid. At time $t = 0$, the robot has no prior knowledge of the environment and must incrementally construct a map through exploration.

The robot perceives its surroundings using a noise-free 2D LiDAR sensor with range λ , which collects l evenly distributed rays spanning a full 360° field of view. At each time step, the robot receives a new LiDAR scan o_t and updates an occupancy grid map O_t , a top-down representation of the environment where each cell is classified as free, occupied, or unknown. The map O_t accumulates all LiDAR observations $o_{1:t}$ up until time t . Additionally, the robot may generate a predicted map P_t using a global prediction module, which we describe in Section 8.6.1.

The objective of exploration is to determine a feasible trajectory $\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_T$ that maximizes the understanding of the environment within a fixed time budget T . This understanding can be evaluated using various metrics, but we define our primary metric as the pixel-wise accuracy of P_t (i.e. the IoU of the predicted occupied cells).

8.4 Previous Contributions in Our Research

In our previous work toward this problem, we proposed MapEx [Ho et al., 2025], a novel exploration framework that uses predicted maps to form probabilistic sensor model as information gain metric. MapEx generates multiple predicted maps from the observed information, from which we compute mean and variance maps. We take the variances of predicted maps and estimated visible area into consideration to compute the information gain of a given viewpoint for exploration planning. We tested MapEx

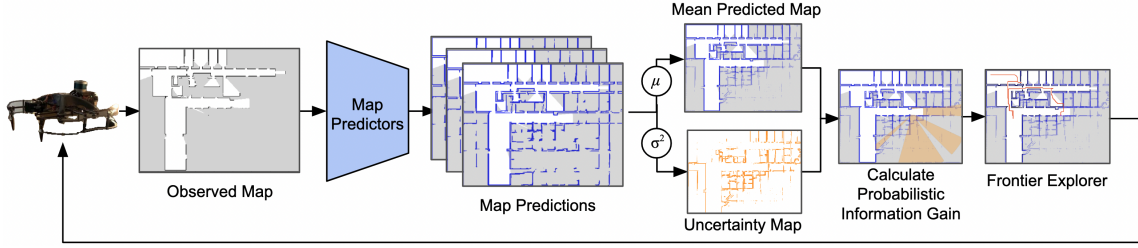


Figure 8.1: MapEx estimates *probabilistic information gain* from predicted maps to use in a frontier explorer. Information gain for each frontier is estimated based on the mean and variance of the predicted maps. The frontier with the highest reward is set as the next goal.

with the real-world KTH floor plan dataset [Aydemir et al., 2012], demonstrating superior exploration performance and topological understanding by 12.4% to SOTA predicted-map based exploration methods and by 25.4% to nearest-frontier method.

In summary, our contributions were as follows:

- A new robot exploration framework that uses predicted maps to form probabilistic sensor model as information gain metric for exploration planning.
- We used the information gain metric to augment frontier-based exploration, which achieves improved exploration results on the real-world KTH floor plan dataset over SOTA methods that uses map predictions (IG-Hector [174], UPEN [75]).
- We demonstrated that the predicted maps produced by MapEx have improved global topological understanding and higher utility for downstream path planning.

8.4.1 MapEx Approach

To efficiently explore structured indoor environments, we leveraged predicted maps to identify areas with high probabilistic information gain. The MapEx pipeline, illustrated in Fig. 8.1, augments a frontier-based explorer for efficient exploration. When selecting the next frontier, the robot first generates a predicted map P_t based on the current observed occupancy grid O_t (Section 8.6.1). Additionally, it generates an ensemble of predictions $\{P_{i,t}\}$, from which it computes the variance map V_t . The frontiers are extracted from the boundary between known and unknown cells in O_t , similar in Yamauchi [1997]. We cluster frontiers using connected components, filter out small ones, and use the centroids of the remaining clusters as final frontiers. We then compute information gain I of each frontier by summing the values of all pixels in the variance map V_t that fall within the visibility mask ν . In other words,

$$I = \sum_{(x_k, y_k)} V_t[x_k, y_k], \quad \forall (x_k, y_k) \in \nu. \quad (8.1)$$

Algorithm 7: MapEx Exploration Planner

Input: environment $\mathcal{E}$, time budget T , start pose $\mathbf{x}_0$

Output: Observed map O_T , predicted map P_T

```
1 for  $t \leftarrow 1$  to  $T$  do
2   observation  $o_t \leftarrow \mathbf{x}_t$ , update occupancy grid  $O_t$  ;
3   if waypoint  $\Psi$  is not available then
4     prediction  $P_t \leftarrow F(O_t)$ ,  $P_{i,t} \leftarrow F_i(O_t)$  // (8.2), (8.3)
5     variance map  $V_t \leftarrow P_{i,t}$  // (8.4)
6     extract frontiers  $\mathcal{F} \leftarrow O_t$  ;
7     for frontier  $f$  in  $\mathcal{F}$  do
8       visibility mask  $\nu \leftarrow \text{probabilistic raycast}(f, P_{i,t})$  ;
9       info gain  $I \leftarrow \sum_{(x_k, y_k) \in \nu} V_t(x_k, y_k)$  score frontier
10       $f.\text{score} \leftarrow I / \|\mathbf{x}_t - f\|_2$  ;
11   waypoint  $\Psi \leftarrow$  choose frontier with maximum score ;
12    $a_t \leftarrow$  local planner( $\Psi$ ), transitions to  $\mathbf{x}_{t+1}$  ;
```

This information gain I estimates the potential reduction in uncertainty, not just by considering the size of observable area, but also the variance it will cover. To discourage backtracking, we divide each frontier score by the Euclidean distance from the current pose of the robot to each frontier, so that if the scores are similar, the robot prefers to select a closer frontier. Once the waypoint is chosen, a local path is generated using an A* local planner, and the robot moves along the path. The pseudocode of the algorithm is provided in Algorithm 7.

8.4.2 Results, Insights, and Challenges

All methods were tested on 10 held-out floor plans in the KTH dataset [Aydemir et al., 2012], over 1000 timesteps. The robot starts at the 4 corners of each floor plan, resulting in a total of 40 {floor plan, start pose} initial conditions. The robot is equipped with a LiDAR with a range of 20m and 2500 samples per scan. For the map predictor, we fine-tuned with 2367 images in the training set, with a batch size of 4 for 40 epochs.

We selected three baselines for comparison, from which one is a classical frontier-based approach and two are representative map prediction-based exploration methods.

- *Nearest-Frontier* [206]: A classical exploration approach that visits frontiers with lowest euclidean distance.
- *IG-Hector* [174]: Map-prediction based approach that uses the predicted map to estimate a viewpoint’s sensor coverage and use it as information gain metric.

- *UPEN* [75]: Map-prediction based approach that uses an ensemble of map predictions to estimate map prediction uncertainty that is used as exploration heuristic.

We evaluate *MapEx* using three metrics: coverage, predicted map quality, and topological understanding. *Coverage* is the percentage of the map observed by the robot. *Predicted Map Quality* is assessed by the intersection over union (IoU) of the predicted and ground truth occupancy maps, specifically for the occupied class within the building footprint. A higher IoU indicates better prediction accuracy. *Topological Understanding* (TU) evaluates how the exploration supports downstream path planning. It is measured by planning an A* path from a start to goal location using the predicted map and calculating success rates based on path collisions or goal achievement. Success is tested over 100 random goal locations per map.

MapEx outperformed *Nearest* by 33.3% in coverage and 22.9% in IoU, *UPEN* by 16.2% in coverage and 9.9% in IoU, and *IG-Hector* by 15.5% in coverage and 12.3% in IoU. *Nearest* focuses on close frontiers, leading to low coverage and IoU, while *UPEN* achieves initial gains but plateaus in low information areas. *IG-Hector* prioritizes sensor coverage but struggles due to poor map prediction early on. *MapEx* excels by addressing both sensor coverage and uncertainty. In TU, *MapEx* surpasses *Nearest* by 20.1%, *UPEN* by 9.6%, and *IG-Hector* by 11.1%, showing 25.4% better overall performance than *Nearest* and 12.4% better than other map-prediction-based baselines.

What we found was *UPEN* focuses on variance but ignores sensor coverage, leading it to explore areas with high variance near the center of the maps, failing to traverse the many corridors and rooms. In contrast, *MapEx* explores the center initially but quickly shifts to more informative regions, such as hallways, based on high sensor coverage estimates. *IG-Hector*, which prioritizes sensor coverage, struggles early due to poor map predictions, leading to inefficient exploration of tight spaces. *MapEx*, however, explores more informative regions, yielding a higher-quality predicted map. Additionally, *MapEx*'s frontier scoring effectively balances sensor coverage and uncertainty, selecting frontiers with both high coverage and medium uncertainty, or prioritizing high-uncertainty areas when necessary.

One of the challenges with *MapEx* is that it sometimes makes suboptimal decisions because it does not fully account for the total information gain an action can provide. In its current form, *MapEx* calculates the information gain for an action by considering only the visibility at the frontier, normalized by the Euclidean distance. However, this overlooks the fact that information is also gathered along the path to the frontier itself. Since the robot follows an A* path to the frontier, rather than a straight-line Euclidean path, this method doesn't accurately reflect the total information gain from traversing the entire path, nor does it accurately quantify the cost. To address this limitation, we aim to improve this approach by more accurately estimating the coverage along the

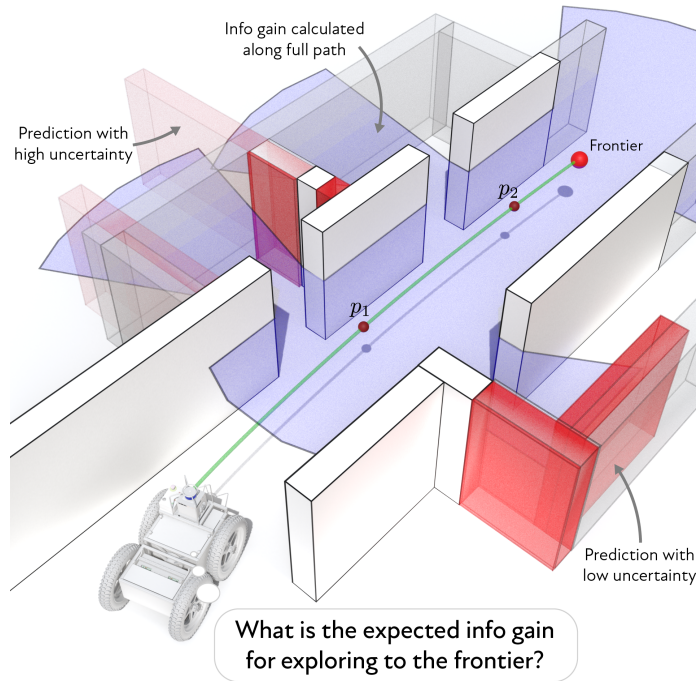


Figure 8.2: Our planner, **PIPE**, enables a robot to explore spaces efficiently by reasoning over the expected information gain over the entire path. This calculation leverages a map predictor for calculating both the expected visibility as well as increase in map quality.

path and its cost, which will help in making better decisions about which frontier to prioritize next.

8.5 Pathwise Information Gain

A common approach to computing predicted information gain for an exploration action is to estimate the expected gain at possible next waypoints. For example, [Bircher et al. \[2016\]](#) iteratively selects viewpoints that maximize the volume of unmapped, visible voxels within the sensor’s field of view. These methods, which evaluate information gain at specific waypoints, can be referred to as *pointwise* information gain-based approaches. In contrast, a more comprehensive and accurate strategy is to adopt a *pathwise* information gain metric, which optimizes planned paths based on total information gain accumulated along the robot’s trajectory. This better matches reality, where most robotic sensors continuously gather information, not merely at waypoints. Prior works have explored different aspects of pathwise information gain. In informative path planning, [Binney et al. \[2013\]](#) proposed a broader formulation that accounts for information gain not only at individual graph nodes but also sensor measurements along the edges, such as temperature or depth, rather than modeling

broader sensor coverage along the path. More specialized approaches, such as [Liu et al. \[2022\]](#), focus on modeling sensor coverage along a path.

In the work in this chapter, we focus on pathwise sensor coverage, a subset of pathwise information gain that considers the sensor’s ability to observe and map the environment along a planned trajectory. Pathwise integration of sensor coverage presents several challenges. First, it is often computationally expensive; for example, estimating camera or LiDAR coverage at every point along a path requires significant computation. Finding efficient integration method remains a non-trivial problem. Second, information gain can be overestimated, especially when undiscovered obstacles or occlusions exist beyond the robot’s current field of view. Third, these estimation errors can accumulate over the trajectory, leading to compounding inaccuracies and suboptimal planning.

To address these challenges, we first focus on efficiently computing integrated information gain along a planned trajectory. We propose an approach that effectively estimates sensor coverage along the path by insightfully leveraging computational geometry, significantly reducing computational overhead. Second, to mitigate overestimation and error accumulation, we integrate map prediction into the robot’s raycasting for sensor coverage estimation. Prior research such as [Georgakis et al. \[2022\]](#), [Katyal et al. \[2019\]](#), [Shrestha et al. \[2019\]](#) and MapEx have shown that deep learning models can predict maps beyond observed areas, enhancing exploration in various ways. We specifically use such predictive models to refine sensor coverage estimates by generating plausible map predictions ahead of the robot. To the best of our knowledge, our work is the first to integrate map prediction-based exploration with a pathwise information gain metric, enabling more efficient exploration planning.

Combining these two aspects, we propose **Pathwise Information Gain with Map Prediction for Exploration (PIPE)** planner¹. **PIPE** generates predictions of map beyond observed areas, computes cumulative sensor coverage along planned trajectories, and leverages predictive uncertainties as an information gain metric for planning. We validate **PIPE** on a real-world floorplan dataset, and show that it outperforms SOTA baselines across multiple evaluations.

The contributions of the work in this chapter are as follows:

- We propose an indoor robot exploration planner, **PIPE**, that estimates information gain using path-integrated cumulative sensor coverage and map predictions.
- We propose a fast method for estimating expected pathwise info gain by leveraging computational geometry.
- We show that **PIPE** outperforms state-of-the-art indoor exploration baselines on

¹Associated paper: [Baek et al. \[2025\]](#)

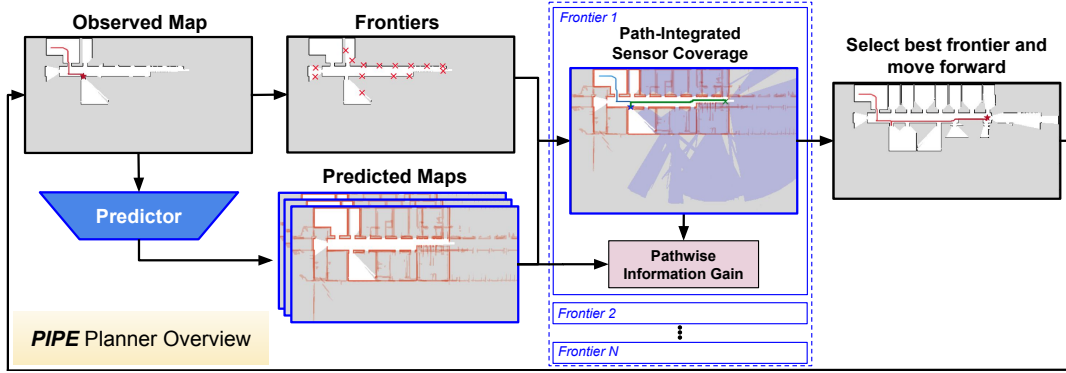


Figure 8.3: Overview of *PIPE* planner pipeline. The robot extracts frontiers from its observed occupancy grid map and generates predicted maps. For each frontier, *PIPE* estimates path-integrated sensor coverage and generates a visibility mask for the full path. It computes pathwise information gain by summing the pixelwise variance of predictions within the visibility mask. The robot then selects the frontier with the highest information gain for exploration planning.

a real-world floorplan dataset in both budget-constrained and full exploration.

8.5.1 Related Work

In path planning, reasoning about a robot’s viewpoint is a common approach. Estimating the amount of information a robot can observe within its sensor’s field of view at a given point has been widely studied in various areas, such as data gathering [Popović et al., 2017], reconstruction [Maboudi et al., 2023], inspection [Bircher et al., 2015], and exploration [Bircher et al., 2016, Charrow et al., 2015, González-Banos and Latombe, 2002, Zhang et al., 2024]. Beyond viewpoint-based reasoning, some works have explored more densely sampled points along planned trajectories in the context of path planning. Binney et al. [2013] proposed variants of informative path planning algorithms that leverage edge-based samples, taking dense points along edges into account. In contrast with sampling points along a path, our work in IA-TIGRIS approximates the information gain along the path using an efficient lower bound. The closest work to ours is Liu et al. [2022], which aims to estimate path information gain for exploration by measuring sensor coverage along the trajectory from the current frontier to the next. However, their approach approximates cumulative sensor coverage using a parallelogram-based estimation, which doesn’t capture accurate information gain within complex layouts like indoor environments. Our work differs from previous approaches by providing a precise estimation of the information gain a robot can obtain along its path using raycast modeling. Additionally, we introduce a technique for efficiently computing the sensor coverage area, enabling more accurate and computationally feasible path information gain estimates for exploration planning.

Our work also utilizes map prediction as a crucial component for exploration,

helping to avoid overestimation when computing pathwise information gain. Without predicted maps, a robot planning a path beyond its observed area risks overestimating the amount of information its sensors could acquire. By assuming a plausible predicted map, we establish a better estimate of the environment and provide a reasonable bound on the expected information gain along the path.

8.6 Approach

The approach for **PIPE** planner is summarized in Fig. 8.3. The global map predictor and the uncertainty calculation is outlined in Section 8.6.1. We introducing the concept of sensor coverage and visibility masks in Section 8.6.2. In Section 8.6.3, we propose a path-integrated sensor coverage approach for computing information gain, followed by our methods to accelerate computation and optimize performance in Section 8.6.4. Finally, in Section 8.6.5, we explain how the pathwise information gain metric is integrated into the exploration planner and propose the **PIPE** planner.

8.6.1 Global Map Prediction and Uncertainty Estimation

While LiDAR provides an accurate occupancy map O_t in observed areas, large areas in the environment often remain unobserved due to sensor range and occlusions, especially in indoor settings. Therefore, we use a global map prediction module F , which predicts unknown regions and produces a complete predicted map P_t as

$$P_t = F(O_t). \quad (8.2)$$

Given the prevalence of large unknown areas during indoor exploration, we use LaMa network architecture [Suvorov et al., 2022], which has large receptive field and the ability to understand global context, making it suitable for inpainting maps with large unknown areas. We treat a top-down view occupancy grid as a 2D image (where 1 pixel = 0.1m) and regard the unknown areas of the environment similarly to missing parts of an image. LaMa is used to predict what lies beyond the robot’s observation. To increase map prediction performance, we fine-tuned LaMa’s out-of-the-box Places2 weights using the KTH floor plan dataset [Aydemir et al., 2012]. Our training set consists of (*observed map*, *ground truth map*) data pairs collected through nearest-frontier exploration. Unlike prior works that predicts local maps [Georgakis et al., 2022, Shrestha et al., 2019], we generate global predicted maps to enable more comprehensive global planning in exploration. Fig. 8.4 shows an example of an observed map, associated predicted map and ground-truth map.

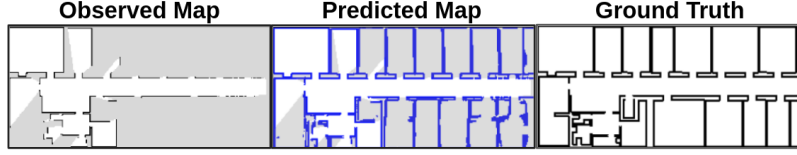


Figure 8.4: Given the observed map, our model predictor can generate a reasonable predicted map that is close to the ground-truth map. These predictions are used for viewpoint scoring.

While the robot can reasonably predict the map P_t given the observed map O_t , there is inherent ambiguity in the unobserved areas. Therefore, the robot should estimate the *uncertainty* of the predictions. To achieve this, we maintain an ensemble of separate LaMa networks F_i , and let each F_i generate a prediction:

$$P_{i,t} = F_i(O_t). \quad (8.3)$$

We used the entire training set to fine-tune F , and split the training set into n_p subsets, each used to fine-tune the parameters of F_i . This allows n_p independent predictions $P_{i,t}$ given the same observed map O_t . In this work, we set prediction ensemble size $n_p = 3$. Using these predictions, we can generate a pixel-wise variance map V_t :

$$V_t = \text{Variance}(P_{i,t}), \quad i \in \{1, 2, \dots, n_p\}. \quad (8.4)$$

8.6.2 Sensor Coverage and Visibility Masks

To estimate the LiDAR sensor coverage that a robot can obtain at a state $\mathbf{x}$, it must perform a raycast by emitting virtual rays from $\mathbf{x}$. For example, when calculating the sensor coverage at a frontier f in the 2D NBV-based methods [González-Banos and Latombe, 2002, Visser et al., 2008], we perform a raycast from f with a range of λ . The ray stops when it hits an occupied cell in O_t , while in free space, it extends until it reaches the range limit λ .

$$\{v_{1:l}\} \leftarrow \text{RAYCAST}(f, \lambda, O_t) \quad (8.5)$$

Here, $\{v_{1:l}\}$ are the vertices that each ray hits at the map, and l is the number of laser samples.

As we leverage map predictions, we can perform raycast on predicted map P_t instead of O_t ,

$$\{v_{1:l}\} \leftarrow \text{RAYCAST}(f, \lambda, P_t) \quad (8.6)$$

While we can apply the same raycasting method to the predicted map P_t , where the ray stops upon hitting a predicted occupied cell, in the work with MapEx we have shown that this approach can underestimate sensor coverage, particularly when

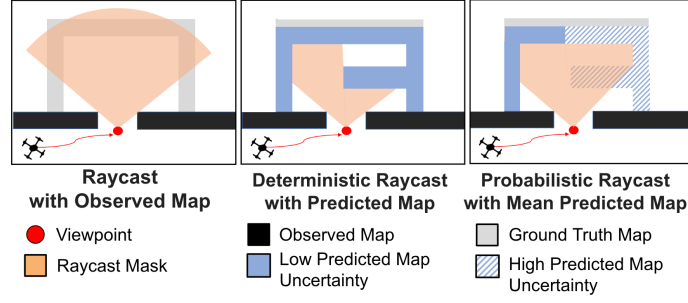


Figure 8.5: Illustration of different raycast methods. By raycasting only with an observed map, the robot may overestimate the potential information gain. Using a deterministic raycast may lead to incorrect gain estimates when map predictions are not correct. In contrast, our method uses a probabilistic raycast to reason on potential sensor coverage, resulting in improved exploration.

erroneous predictions falsely classify free space as occupied. To address this, we adopt the probabilistic raycasting method proposed in MapEx. Fig. 8.5 illustrates the difference between raycast methods.

Given a viewpoint $c \in \mathcal{E}$, we generate l hypothetical rays R_j (where $j = 1, 2, \dots, l$) with a range of λ , emitted in a 360° pattern originating from c . In a deterministic raycast approach, each ray R_j stops when it hits an occupied cell, and if there are no obstructing cells along its path, R_j reach the end at a distance of λ . In this work, we use a *probabilistic raycast* approach instead: first, we calculate the mean predicted map $\bar{P}_t$, which is the average of predicted maps $\{P_{i,t}\}$. Each ray R_j starts with an initial accumulated occupancy value $h = 0$, and as R_j passes through the pixels of $\bar{P}_t$, it adds the occupancy value of each pixel in the mean map $\bar{P}_t$ to h . When h reaches a threshold ϵ , the raycast terminates. Through empirical testing, we found that $\epsilon = 0.8$ yields the best performance.

From the set of l vertices generated from the raycast, we can draw a polygon $\mathcal{G}$, and use a flood fill algorithm to generate a visibility mask ν , which is an estimate sensor coverage at the frontier point f .

$$\begin{aligned} \mathcal{G} &\leftarrow \text{DRAWPOLYGON}(\{v_{1:l}\}) \\ \nu &\leftarrow \text{FLOODFILL}(\mathcal{G}) \end{aligned} \tag{8.7}$$

8.6.3 Path-Integrated Sensor Coverage for Information Gain

The core component of our **PIPE** planner is to evaluate the information gain of each frontier point $f \in \mathcal{F}$ (where $\mathcal{F}$ is a set of frontiers) by integrating the cumulative information gain along the path from the robot's current pose $\mathbf{x}_t$ to f_i , rather than estimating sensor coverage solely at f_i .

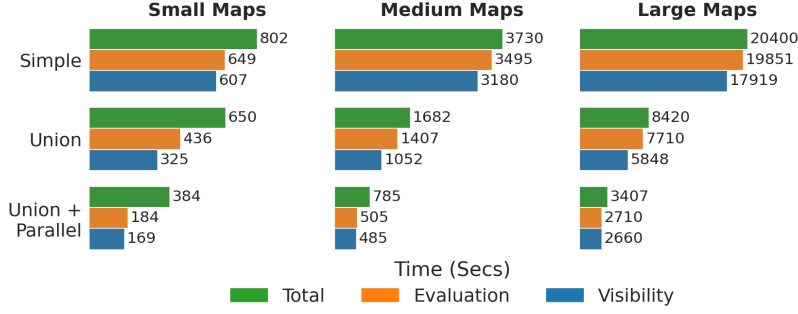


Figure 8.6: Computation time comparison of our optimized approach (union + parallel) across map sizes, with the greatest reduction for large maps.

For each frontier $f_i \in \mathcal{F}$, we generate an A* path ξ_i connecting the robot’s current pose $\mathbf{x}_t$ to f_i . Along ξ_i , we sample each point p_j to perform a raycast with a range of λ on the predicted maps P_t . These sampled points along ξ_i are every Δ , which is the same distance for each robot time step and update to the observed occupancy grid map O_t .

A simple approach to calculating the total visibility mask along the path is to first generate the visibility masks ν_j for each p_j :

$$\begin{aligned}
 \{v_{1:l}\}_j &\leftarrow \text{RAYCAST}(p_j, \lambda, P_t) \\
 \mathcal{G}_j &\leftarrow \text{DRAWPOLYGON}(\{v_{1:l}\}_j) \\
 \nu_j &\leftarrow \text{FLOODFILL}(\mathcal{G}_j)
 \end{aligned} \tag{8.8}$$

The mask ν_j represents the sensor coverage the robot can obtain at each point p_j along the path to the frontier f_i . The path-integrated sensor coverage, ν , is then obtained by computing the union of each ν_j along the path. The combined sensor coverage ν represents the pathwise information gain the robot can acquire while moving toward the frontier f . However, calculating the pathwise information gain in this simple manner is extremely computationally inefficient. In the next subsection, we propose an efficient technique to reduce the computational overhead.

8.6.4 Optimization of Calculating Information Gain

To identify the most computationally demanding components of our naive planner implementation, we conducted a simple experiment. We ran our planner on two small, two medium, and two large maps from the KTH dataset [Aydemir et al., 2012] (which will be further detailed in Section 8.7.1), each with two different starting positions (one from a corner and one from the center), measuring the computation time. All experiments were conducted on a desktop computer equipped with an *Intel Core i5-10400F* CPU (2.90GHz), an *NVIDIA GeForce RTX 3060* GPU, and 16GB of RAM. The

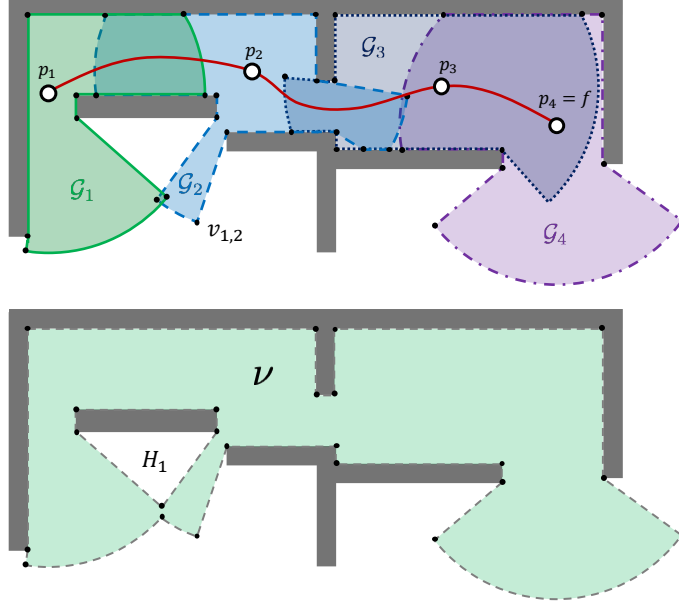


Figure 8.7: A visualization of the path visibility mask computation. Individual polygons are first generated using raycasting. These polygons are then merged into a single polygon, including any holes, and a flood-fill algorithm is applied to determine the final visibility mask.

results are shown in Fig. 8.6. As expected, we found that in the simple approach, frontier evaluations accounted for 80.9%, 93.7%, and 97.3% of the total computation time in small, medium, and large maps, respectively. The increasing proportion in larger maps is likely due to the greater number of frontiers and longer path lengths. Among frontier evaluations, visibility mask generation was the most time-consuming step, with repeated flood fill operations being the primary bottleneck.

To mitigate the computational overhead, we developed a more efficient implementation. Rather than performing a flood fill separately for each raycast-generated polygon $\mathcal{G}_j$ and then taking the union of the masks, we first merge the raycast-generated polygons into $\mathcal{G}_{union}$ and apply a single flood fill operation to the merged polygon. This greatly reduced computation by only performing the more computationally expensive operation once.

Since most paths contain multiple, often nonconvex raycasted polygons, their union typically creates holes H , or trapped regions. To identify these holes and account for them in the flood fill operation, we follow these steps: first, compute the symmetric difference S_j by performing symmetric difference operation Δ between each polygon $\mathcal{G}_j$ and the polygon union ($S \leftarrow \bigcup_j (\mathcal{G}_j \Delta \mathcal{G}_{union})$). Then extract the parts of S that are not contained in the polygon union ($H \leftarrow S \setminus \mathcal{G}_{union}$).

We then compute the final visibility mask ν using the polygon union and holes ($\nu \leftarrow$

Algorithm 8: PathVisibilityMask

Input: Path ξ_i (from the robot’s current pose $\mathbf{x}_t$ to the candidate frontier f_i),
map M , LiDAR range λ

Output: Visibility Mask ν

```
1  $\mathbb{G} \leftarrow \{\}$  // Initialize
2 foreach point  $p_j \in \xi_i$  do
    // Probabilistic Raycast and Polygon Construction
3      $\{v_{1:l}\}_j \leftarrow \text{RAYCAST}(p_j, \lambda, M)$  // Vertices
4      $\mathcal{G}_j \leftarrow \text{DRAWPOLYGON}(\{v_{1:l}\}_j)$  // Polygon
5      $\mathbb{G} \leftarrow \mathbb{G} \cup \{\mathcal{G}_j\}$ 
    // Merge Polygons and Generate Visibility Mask
6  $\mathcal{G}_{\text{union}} \leftarrow \bigcup_{\forall j} \mathcal{G}_j$ 
7  $S \leftarrow \bigcup_{\forall j} (\mathcal{G}_j \triangle \mathcal{G}_{\text{union}})$ 
8  $H \leftarrow S \setminus \mathcal{G}_{\text{union}}$ 
9  $\nu \leftarrow \text{FLOODFILL}(\mathcal{G}_{\text{union}}, H)$ 
```

$\text{FLOODFILL}(\mathcal{G}_{\text{union}}, H)$). The full pseudocode for this process is shown in Algorithm 8, with a visualization in Fig. 8.7.

Additionally, to take advantage of computational parallelism, we used Python’s multiprocessing package to distribute the computation across multiple CPU cores, with each process handling a separate frontier. In an ideal scenario, this approach can reduce computation time by a factor approximately equal to the number of available CPU cores, assuming efficient implementation and minimal overhead.

The impact of these optimizations, shown in Fig. 8.6, is most pronounced in the large maps, with a 83.3% reduction in computation time.

8.6.5 *PIPE* Planner

The overall procedure of *PIPE* planner uses classical frontier-based exploration [Yamauchi, 1997] as our planning framework. In frontier-based approaches, the robot identifies boundaries between known and unknown regions and selects the nearest frontier to explore. Building on this, we incorporate an information gain metric to select the next frontier to travel that maximizes expected information gain, following the “next-best-view” paradigm [Bircher et al., 2016, Delmerico et al., 2018, González-Banos and Latombe, 2002, Vasquez-Gomez et al., 2017, Visser et al., 2008].

When the robot needs to select a new waypoint, it first extracts a set of candidate frontier points, $\mathcal{F}$, from the current occupancy grid O_t , and generates predictions P_t and variance map V_t . For each frontier $f \in \mathcal{F}$, it generates an A* path, ξ , from current

Algorithm 9: *PIPE* Planner

Input: Robot's start pose $\mathbf{x}_0$, Time budget T
Output: Final observed map O_T , final predicted map P_t

```
1  $t \leftarrow 0$  // Initialize time
2 while  $t \leq T$  do
    // Sense and Update Map
3    $O_t \leftarrow \text{RAYCAST}(\mathbf{x}_t, \lambda, O_t)$  if robot needs to select a new waypoint then
        // Extract Frontiers and Predictions
4      $\mathcal{F} \leftarrow \text{EXTRACTFRONTIERS}(O_t)$ 
5      $P_t \leftarrow \text{GENERATEPREDICTIONS}(O_t)$  // Ensemble
6      $V_t \leftarrow \text{COMPUTEVARIENCE}(P_t)$  // Uncertainty Map
        // Evaluate Frontiers
7     foreach frontier  $f \in \mathcal{F}$  do
8          $P \leftarrow \text{ASTARPATH}(\mathbf{x}_t, f)$ 
9          $\nu \leftarrow \text{PATHVISIBILITYMASK}(P, P_t, \lambda)$  // Algorithm 8
10         $I \leftarrow \sum_{(x_k, y_k) \in \nu} V_t(x_k, y_k)$ 
11         $f.\text{score} \leftarrow I/|P|$  // Normalize by path length
12     $f_{\max} \leftarrow \arg \max_{f \in \mathcal{F}} f.\text{score}$ 
        // Plan Local Actions
13     $a_t \leftarrow \text{ASTARPATH}(\mathbf{x}_t, f_{\max})$ 
14     $\mathbf{x}_{t+1} \leftarrow \text{TRANSITION}(\mathbf{x}_t, a_t)$ 
15     $t \leftarrow t + 1$ 
```

pose $\mathbf{x}_t$ to the frontier f , and then applies Algorithm 8 to compute the pathwise sensor coverage, yielding the visibility mask ν . The information gain, I , is then calculated by summing the variance within the visibility mask and normalizing by the length of the path ξ . The frontier with the highest score is selected as the next waypoint until it has been reached, at which a new waypoint is selected through the same process described as above. The complete procedure of *PIPE* planner is presented in Algorithm 9. In contrast with previous works, our work is selecting not the next best frontier observation but rather the holistic next best frontier by including the path to the frontier.

8.7 Experiments

In this section, we describe the experimental setup, baselines, and evaluation metrics, followed by the experimental results. We provide both qualitative and quantitative analyses comparing our *PIPE* planner against the baselines.

8.7.1 Experimental Setup

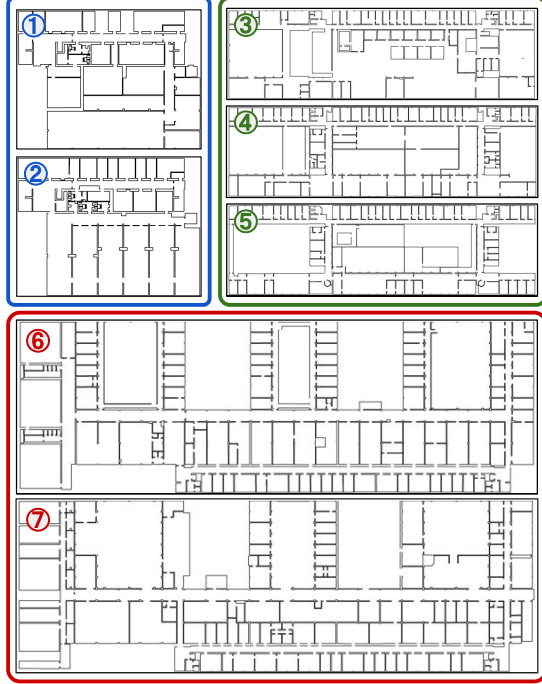


Figure 8.8: The seven indoor maps used to evaluate our method. Small, medium, and large maps are shown in blue, green, and red, respectively. Not shown to scale relative to each other.

We use our custom-built simulator based on the KTH floorplan dataset [Aydemir et al., 2012], which includes over 100 campus floorplans represented as XML files containing wall and door locations for each room. To ensure accurate raycasting, we corrected any inaccuracies in the maps that could lead to false projections into invalid spaces. The maps were downsampled to a resolution of 10 pixels per meter.

From the dataset, we selected seven floorplans, categorized into three sizes: two small ($65\text{m} \times 85\text{m}$), three medium ($60\text{m} \times 205\text{m}$), and two large maps ($88\text{m} \times 265\text{m}$). Fig. 8.8 visualizes these environments. For each floorplan, the robot starts from 25 distinct initial locations. In total, we conducted 175 experiments to evaluate performance.

8.7.2 Baselines

We select five baselines for comparison, from which one is a conventional nearest frontier-based method, two for observed map based method, and other two for map prediction based exploration methods.

- Nearest [206]: A classical exploration method that selects frontiers based on the shortest euclidean distance.
- NBV-2D [76, 198]: A ‘Next-best-view’ strategy where, for each frontier, the robot performs virtual raycasting on the observed map O_t , computes sensor coverage, and normalize it by the Euclidean distance. (*Pointwise, No Prediction, Sensor Coverage-Based*)
- PW-NBV-2D: An extension of NBV-2D from a pointwise to a pathwise approach. The robot performs raycasting from multiple points along the path on O_t , aggregates the sensor coverage as information gain, and normalizes it by the A* path distance. (*Pathwise, No Prediction, Sensor Coverage-Based*)

- UPEN [74]: A map-prediction based approach that leverages an ensemble of predicted maps to estimate uncertainty. Information gain is computed as the cumulative sum of variances (uncertainty) along the path, normalized by path distance. (*Pathwise, Prediction-Based, No Sensor Coverage, but Uncertainty-Only*)
- MapEx: A map-prediction-driven exploration frontiers are scored using pointwise probabilistic raycasting on predicted maps, combined with uncertainty map, and normalized by Euclidean distance. (*Pointwise, Prediction-Based, Sensor Coverage and Uncertainty*)

8.7.3 Metrics

We assess the performance of the planners using the IoU (Intersection over Union) of the occupied cells in the predicted and ground truth maps. IoU measures the quality of the predicted 2D occupancy map, by comparing the map’s predicted occupied cells with the ground truth occupancy, calculated as

$$\text{IoU} = \frac{\text{TP}}{\text{FP} + \text{TN} + \text{TP}}$$

where TP, FP, TN denote true positives, false positives, and true negatives, respectively. To avoid penalizing minor map discrepancies—like wall thickness variations, gaps, dataset artifacts, or small unobservable spaces—we use binary dilations as a buffer in IoU calculation.

Let P represent the predicted binary map of occupied cells, G represent the ground truth occupied binary map, and O represent the observed map. We define the masked prediction P_m such that $P_m = P$ where $O = 0.5$ (unknown space) and $P_m = O$ otherwise, where only unknown space in the observed map is replaced by the global map prediction. We then compute a corrected prediction P_c from P_m by excluding all predictions that fall outside the building. We dilate P_m and G by a fixed kernel size of 5 pixels (50 cm) to create P_d and G_d , respectively. By evaluating all pixels i , we can then calculate

$$\begin{aligned} \text{TP} &= \sum_i (P_d(i) \wedge G(i)) \\ \text{FP} &= \sum_i (P_m(i) \wedge P_c(i) \wedge \neg G_d(i)) \\ \text{FN} &= \sum_i (\neg P_d(i) \wedge G(i)) \end{aligned}$$

where $\wedge$ and $\neg$ denote the logical AND and NOT operators, respectively. This process allows the IoU to focus on overall map topology and wall placement. Without this

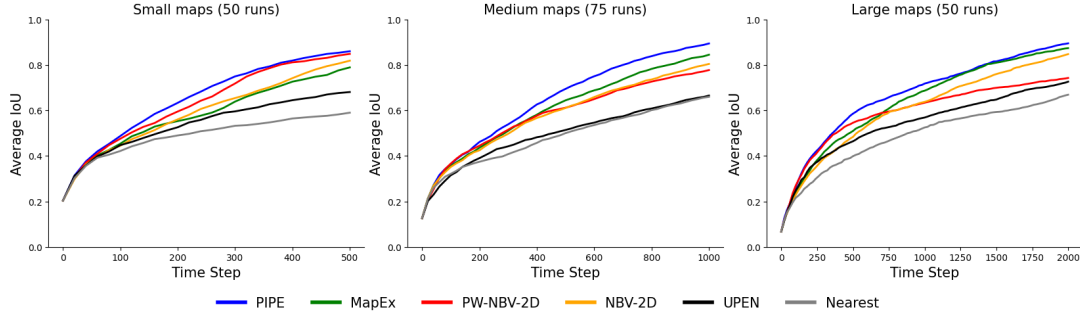


Figure 8.9: Quantitative comparison of methods for each map size shows **PIPE** outperforms all others in most time steps, with the second-best method varying by map size. Prediction-based methods excel on larger maps, while pathwise NBV-2D performance declines as map size increases.

correction, IoU plateaus at 75-85%, even with full area coverage.

Higher IoU means the algorithm can create more accurate predicted map. Methods that leverage prediction (**PIPE**, MapEx, UPEN) have a clear definition of predicted maps, whereas methods without prediction (NBV-2D, PW-NBV-2D, Nearest) lack them. For these non-prediction-based methods, we collect the observed maps at every time step and generate predicted maps afterward to define predicted IoU.

8.7.4 Experimental Results

In this section, we first present the quantitative evaluation of our experiments. We analyze how much each method explores within a fixed time budget in Section 8.7.4 and examine the time required to complete map exploration, along with their later-stage behaviors, in Section 8.7.4.

IoU Comparison with time budgets

Fig. 8.9 compares exploration progress over time across three map scales for each method. Overall, we observe that **PIPE** consistently outperforms all other methods, demonstrating stable and high performance across all map sizes. In small maps, **PIPE** achieves the highest final IoU, surpassing PW-NBV-2D by 1.15%, MapEx by 7.08%, NBV-2D by 4.13%, UPEN by 17.91%, and Nearest by 26.99%.

In the medium-sized maps, **PIPE** maintains its lead, though the ranking among the baselines shifts. Here, MapEx emerges as the second-best method, outperforming the rest. At $T = 1000$, **PIPE** exceeds PW-NBV-2D by 11.69%, MapEx by 4.93%, NBV-2D by 8.99%, UPEN by 22.89%, and Nearest by 23.45%.

In large maps, **PIPE**, MapEx, and NBV-2D perform at comparable levels, but PW-

NBV-2D experiences a significant drop in performance. At $T = 2000$, **PIPE** outperforms MapEx by 2.1%, NBV-2D by 4.73%, UPEN by 16.85%, and Nearest by 22.57%. Although the final IoU differences between the top three methods appear moderate, the trajectory suggests that MapEx is reaching saturation, while **PIPE**'s IoU continues to increase. This trend is further analyzed in Section 8.7.4. Additionally, **PIPE** surpasses PW-NBV-2D by 15.20% in large maps—a notably larger gap than in small and medium maps—indicating that pathwise integration of information gain alone is insufficient and can even degrade performance when combined with certain estimation methods.

Method	Small Maps	Medium Maps	Large Maps
PIPE	327.00	648.35	1338.16
MapEx	290.63	607.05	1278.39
PW-NBV-2D	316.81	587.59	1184.69
NBV-2D [198]	295.68	582.70	1209.37
UPEN [74]	270.04	494.08	1084.57
Nearest [206]	244.80	482.79	971.30

Table 8.1: Area Under the Curve (AUC) of IoU for each method. Higher is better. **PIPE** outperforms other methods in all maps.

The performance of each method can also be measured with Area Under the Curve (AUC) of each method's plot. According to Table 8.1, **PIPE** surpassed all other methods for every map size, verifying its efficient map exploration.

Complete Map Exploration

In addition to evaluating each method's performance under fixed time budgets, we also analyzed their ability to achieve complete map exploration. Here, we define this as reaching at least 95% IoU with the ground truth map. To better understand the long-tail behavior of the methods, we also report results for 90% IoU, highlighting the difficulty of achieving the additional 5% increase. For both IoU thresholds, we measure the number of time steps required by each method.

The results are shown in the Table 8.2. Overall, **PIPE** outperformed all other methods in both efficiency and stability. It required the fewest time steps to reach both 90% and 95% IoU across all maps and was the only method to achieve a zero failure rate for complete exploration at both thresholds.

	Small Maps (50 runs)				Medium Maps (75 runs)				Large Maps (50 runs)			
Method	90% IoU	FR (%)	95% IoU	FR (%)	90% IoU	FR (%)	95% IoU	FR (%)	90% IoU	FR (%)	95% IoU	FR (%)
PIPE	505.2	0	718.4	0	1039.3	0	1496.9	0	1979.0	0	2657.5	0
MapEx	609.7	0	751.5	0	1164.9	0	1602.3	3	2226.0	0	3398.6	2
PW-NBV-2D	519.0	0	784.0	12	1535.2	15	1959.5	51	3732.0	4	4825.3	32
NBV-2D [198]	577.2	0	851.9	0	1321.0	0	1693.6	4	2335.5	0	3037.1	0
UPEN [74]	833.4	34	1008.9	64	2170.7	40	2337.4	63	4085.2	18	4874.8	62
Nearest [206]	1049.6	8	1160.1	12	2272.8	11	2524.3	33	4109.0	2	4634.2	4

Table 8.2: Comparison of required time steps to reach 90% and 95% IoU with ground truth maps. The failure rate (FR) represents the percentage of runs that did not achieve the target IoU within the maximum allowed time steps. Lower values indicate better performance. The maximum time steps for small, medium, and large maps are 1,500, 3,000, and 6,000, respectively.

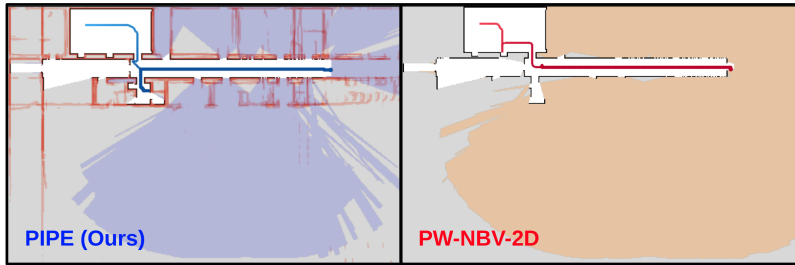


Figure 8.10: Qualitative comparison between sensor coverages estimated by **PIPE** (left) and PW-NBV-2D (right). PW-NBV-2D overestimates cumulative sensor coverage, while **PIPE** uses predictions to mitigate it.

Qualitative Discussion

As shown in Fig. 8.9 and Table 8.1, pathwise methods (**PIPE**, PW-NBV-2D) outperform pointwise methods (MapEx, NBV-2D) in small maps. However, in medium and large maps, **PIPE** continues to outperform MapEx, while PW-NBV-2D performs notably worse. This suggests that simply integrating cumulative sensor coverage can degrade performance as map size increases. The qualitative comparison in Fig. 8.10 provides insights into why PW-NBV-2D struggles in larger maps. **PIPE** computes sensor coverage using probabilistic raycasting over predicted maps, providing a more conservative and accurate estimation of unobserved areas. In contrast, PW-NBV-2D significantly overestimates sensor coverage in unobserved regions. While the accumulation of errors along the pathwise raycasts may have little noticeable impact in small maps, it becomes increasingly significant as map size and complexity increases.

Lastly, Fig. 8.11 presents an example of the exploration progress of **PIPE**, MapEx, PW-NBV-2D, and NBV-2D. All methods start from the same location. The two pointwise methods, NBV-2D and MapEx, exhibit similar behaviors, primarily focusing on the left half of the map. This highlights a key limitation of pointwise methods—their tendency to fixate on short-sighted local information gain. Meanwhile, PW-NBV-2D initially explores effectively but later gravitates toward traversing long corridors,

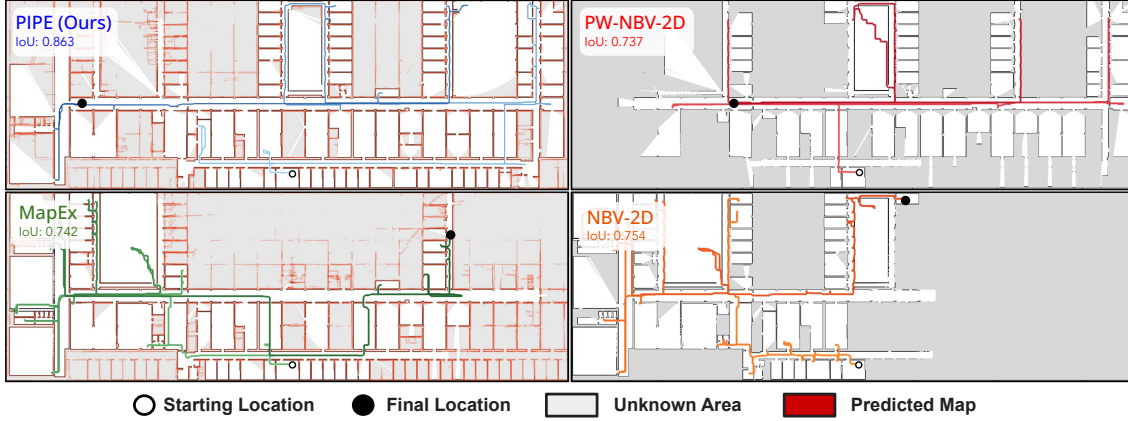


Figure 8.11: Qualitative comparison of methods at time step $T = 2000$ on Map 6, with the path transitioning from light to dark to indicate the passage of time. **PIPE** performs the best, with more even exploration across the space and less backtracking compared to other methods.

skipping rooms. In contrast, **PIPE** leverages both predictive uncertainties and pathwise coverage, balancing exploration between rooms and corridors, ultimately achieving the best performance.

8.8 Conclusion

In this chapter, we presented **PIPE**, a planner based on Pathwise Information Gain with Map Prediction for Exploration. Our approach addresses the need to compute information gain cumulatively along a path rather than merely at a point. To achieve this efficiently, we proposed techniques to reduce computational overhead and incorporated map predictions to mitigate the overestimation of cumulative sensor coverage. Experiments on a real-world floorplan dataset simulator demonstrated that **PIPE** outperforms baselines, including map-prediction-based exploration method relying solely on pointwise gain and pathwise exploration methods without map prediction.

Chapter 9

Human-Inspired Indoor Exploration using Reinforcement Learning

In the previous chapter, we presented our state-of-the-art exploration planner and evaluated its performance against baseline methods. However, this raised a natural question: how effective is our planner, really? How does its performance compare to that of a human, and how large is the gap? In this chapter, we investigate these questions by studying human exploration behavior and then leveraging those insights to design a reinforcement learning approach for indoor robotic exploration.¹

9.1 Introduction

In exploration, a robot navigates in an unknown space to build map representations and understand the environment. At its core, it is the problem of deciding where to go to gain the most information [Best et al., 2024, Oriolo et al., 1998, Yamauchi, 1997]. In the previous chapter we followed a conventional paradigm of deciding where to go using frontier-based exploration [Yamauchi, 1997], in which a robot incrementally updates the boundary between known and unknown spaces, or *frontier*, and systematically visits the frontiers. While this approach has proven to be robust in large-scale environments, such algorithms are often short-sighted, reasoning only on the next step. In contrast, humans excel at crafting long-term strategies even with limited observations, that balance efficiency, uncertainty, and resource constraints. *How can we achieve human-level long-horizon exploration?*

One advantage humans have in exploration is being able to reason over the unobserved space and predict what they will see. To extend the perception horizon

¹Associated paper: [Harutyunyan et al. \[2025\]](#)

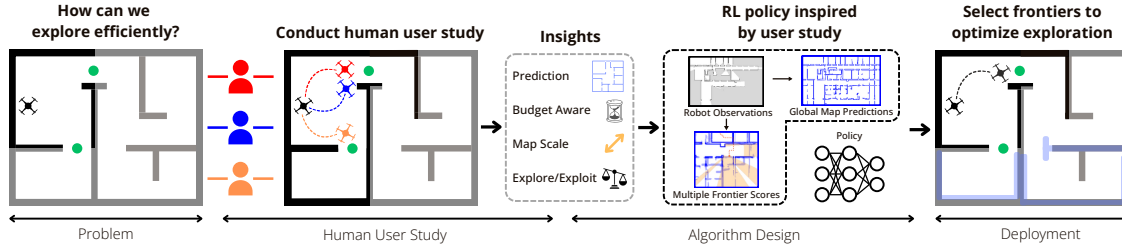


Figure 9.1: An overview of the development and research process for MapExRL. This work examines the question: How can a robot explore efficiently? We conduct a human user study to gain insights into effective exploration strategies. These insights, in turn, inform the design of our reinforcement learning-based exploration policy, leveraging global map predictions and other environmental contexts and enabling state-of-the-art performance.

in robot exploration, some studies have proposed methods that go beyond reasoning about only the observed areas. Like in our previous chapter, these approaches use observed data to predict regions beyond what has been seen [Georgakis et al., 2022, Shrestha et al., 2019]. By leveraging deep learning models, these works generate broader map predictions from partial observations of the environment, improving exploration strategies. However, while these works augment perception by employing predictions, their decision-making remains focused on greedy and myopic action selection.

Reinforcement learning (RL) has been used to train robots to learn navigation policies through trial and error [Li et al., 2019, Niroui et al., 2019]. Previous works have shown that data-driven RL can outperform classical planning approaches [Cao et al., 2023, Lee et al., 2021]. However, these RL-based methods often rely on motion-primitives for action spaces, which could present challenges for long-horizon planning in complex, large-scale environments.

Existing approaches remain limited in achieving long-term human-level decision making for exploration. Toward human-level exploration, we identify two key gaps: (1) an insufficient understanding of *what* signals humans use to make long-horizon decisions and *how* they leverage them, and (2) the lack of a planning framework that integrates these human-inspired insights into robotic exploration.

To bridge these gaps, we conduct a user study comparing human performance to our exploration algorithm MapEx and derive insights to guide our approach. Based on these insights, we introduce MapExRL, a *budget-aware* RL-based exploration framework that, like humans, leverages predictive context about the environment. We validate MapExRL on real-world floorplan data, evaluated against baseline methods and human users. An overview of this work is shown in Fig. 9.1.

Our contributions are as follows:

- We present insights on human performance for indoor exploration through a user

study that illustrates the gap in performance to existing methods and extract actionable insights on how humans explore.

- We develop a new RL-based exploration planner, MapExRL, that leverages the insights from our user study on observation space features and action spaces.
- Finally, we conduct thorough comparisons of SOTA frontier-based and learning-based methods, as well as human performance, to show the benefits of MapExRL.

9.2 Background

9.2.1 Robot Exploration

Autonomous robot exploration has been studied from multiple perspectives. Frontier-based exploration [Best et al., 2024, Yamauchi, 1997] guides robots toward the boundary between known and unknown space but often suffers from slow progress and a lack of global planning. Information-theoretic approaches seek to maximize the information gain of consecutive actions [Bai et al., 2016, Bourgault et al., 2002], yet they are often constrained by their greedy and myopic nature. Sampling-based methods [Bircher et al., 2016] construct a randomly generated tree or graph structure, such as [LaValle, 1998], to expand into a traversable space, typically selecting the most informative nodes or viewpoints for exploration.

9.2.2 Map Prediction for Exploration

A common limitation of the previously mentioned approaches is their reliance on sensor data with a limited perceptual range, often failing to reason beyond the observed areas. Recently, a growing body of work has explored leveraging deep learning models to predict maps and use these predictions to enhance exploration [Ericson and Jensfelt, 2024, Georgakis et al., 2022, Luperto and Amigoni, 2022, Luperto et al., 2021, Ramakrishnan et al., 2020, Saroya et al., 2020, Shrestha et al., 2019, Zwecher et al., 2022]. These approaches utilize accumulated observations in a 2D top-down view as input and infer unexplored areas using predictive models trained on offline datasets. Using these predictions, robots can prioritize areas with high uncertainty [Georgakis et al., 2022], areas with high expected coverage gain [Ericson and Jensfelt, 2024, Luperto and Amigoni, 2022, Luperto et al., 2021, Shrestha et al., 2019], or areas with both, such as in MapEx.

These approaches allow robots to reason beyond observed areas and focus on exploring areas guided by predictions; however, they still struggle due to their their

action selection strategy, particularly when handling environments of varying sizes and topological complexities or operating under budget constraints. In this work, we aim to address these limitations.

9.2.3 Learning-based Exploration

Studies have addressed exploration problems using deep reinforcement learning [Cao et al., 2023, Chen et al., 2019, 2020, Lee et al., 2021, Li et al., 2019, Niroui et al., 2019, Zhu et al., 2018]. Some have focused on training policies that generate simple, primitive actions, such as moving to adjacent cells or taking a few steps in a diagonally connected grid [Chen et al., 2019, Li et al., 2019]. However, this approach often results in myopic decision-making, prioritizing immediate actions over long-term strategy.

The work in Niroui et al. [2019] used an RL method that outputs a single weight to balance frontier scoring based on distance and estimated information gain, selecting the highest-scoring frontier. While this approach enables exploration of large, complex topologies, it does not directly score frontiers or exploit the predictability of indoor structures. Some studies have employed graph structures to encode the states, defining actions as movements toward nearby nodes within the graphs [Cao et al., 2023, Lee et al., 2021, Zhu et al., 2018]. In the work in this chapter, we use frontier points as the action space and score them directly, allowing us to consider the full frontier context and avoid myopic decision-making.

Lastly, some studies have integrated the map prediction paradigm into RL-based exploration to expand the observation space [Tao et al., 2024, Zwecher et al., 2022]. However, these studies commonly have limitations in that they set fixed primitive motions as the action space or validate their methods only on small, simplified maps. In our work, we expand the observation space using map prediction while simultaneously training a policy that outputs longer-horizon decision-making.

Our method, MapExRL, differs from previous approaches in three key ways: (1) it expands the observation space beyond direct observations using map predictions, (2) it leverages RL-based methods to enhance adaptability to various human-inspired temporal and spatial contexts, and (3) it sets frontiers as the action space, enabling longer-horizon action selection compared to primitive motions.

9.3 Problem Statement

We address the problem of efficient indoor mapping and exploration. Our simulation environment is a 2D space $\mathcal{E} \subset \mathcal{R}^2$ where the robot’s state is represented as a discrete

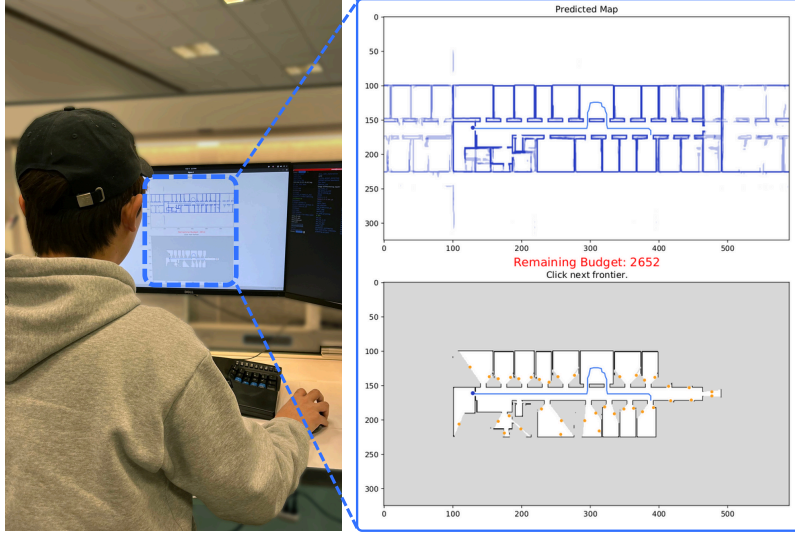


Figure 9.2: User interface for the study: Participants are shown the observed occupancy grid (bottom) and predicted maps (top), and must select the next frontier to explore among the frontiers marked as orange dots. The goal is to reach 95% IoU as efficiently as possible within the distance budget B .

2D position $\mathbf{x} = [x, y]$. The robot is equipped with a 2D LiDAR that provides 360° coverage to construct an occupancy grid map O_t (observed map), categorizing the space as free, unknown, or occupied. At each timestep t , the robot moves one step Δx . We assume that the range measurements are noise-free and the robot pose is known. We aim to plan paths that maximize environment understanding as quickly as possible within a distance budget of B .

We define the predicted map’s Intersection over Union (IoU) as a measure of exploration quality. The exploration continues until the distance budget B is reached or a sufficient understanding of the environment is achieved, which we define as 95% IoU for the predicted map.

9.4 User Study on Indoor Exploration

We conducted a user study to uncover key human exploration strategies and compare the performance of MapEx with human decision-making in exploring large indoor buildings. This section covers the design and results of the user study and how we leveraged its insights to inform the design of our approach. The study aims to answer the following questions:

- Q1:** What is the performance gap between human decision-making and MapEx?
- Q2:** How much does prior exposure to the map improve human exploration efficiency?
- Q3:** What key context do humans use during exploration?
- Q4:** What long-term strategies do humans use to explore?



Figure 9.3: The set of test maps used in this work. These maps were selected to ensure a diverse range of common floorplan layouts and topologies. Note that the maps are not shown to scale relative to one another, and Map 5 is the largest in actual size, followed by Map 4.

9.4.1 User Study Design

To answer the above questions, we conducted a user study with 13 participants of varying experience levels in robotic exploration. To ensure a fair evaluation, participants were not allowed to see the maps beforehand, preventing any advantage from prior knowledge or mental priors. This ensured that their performance was based solely on their ability to explore the environment without prior familiarity. Users are given the same problem formulation as defined in Section 9.3—either achieve 95% IoU on the predicted map as quickly as possible or maximize IoU within the distance budget. The interface (Fig. 9.2) displays the average output of the ensemble global map prediction network (same network as in Chapter 8) and the accumulated observed occupancy grid map O_t . Frontiers are displayed on the observed map, and the user chooses which frontier to explore next. The robot then moves to the selected frontier, updating the maps after each robot step. This process repeats until the episode ends.

Each participant first completes a training round to familiarize themselves with the interface and task. They then complete nine test episodes across three different maps, completing three episodes per map before moving to the next. We selected maps 1, 4, and 5 (Fig. 9.3) to ensure a range of map sizes and shapes. Successive episodes on the same map are referred to as rounds 1, 2, and 3. In rounds 1 and 2, participants start from the same position, while round 3 uses a different starting location. Round 1 assesses performance without prior knowledge of the map scale or features. Round 2 evaluates improvements based on prior knowledge and the opportunity to adapt their previous strategy. Round 3 examines how strategies adapt to new starting conditions while retaining prior map knowledge. Multiple runs per map also help capture participants’ best performances. To introduce variety while

	Map 1	Map 2	Map 4	Combined
Human Round 1	1156 $\pm$ 32	902 $\pm$ 78	768 $\pm$ 37	911 $\pm$ 60
Human Round 2	1160 $\pm$ 39	885 $\pm$ 67	795 $\pm$ 32	915 $\pm$ 56
MapEx Round 1&2	1071 $\pm$ 770	888 $\pm$ 446	731 $\pm$ 103	875 $\pm$ 146
Human Round 3	1189 $\pm$ 35	941 $\pm$ 61	814 $\pm$ 50	951 $\pm$ 57
MapEx Round 3	1037 $\pm$ 1036	899 $\pm$ 467	875 $\pm$ 215	924 $\pm$ 116

Table 9.1: Results of the user study compared to MapEx. Values reflect the average final reward (higher is better) and the 95% confidence interval.

maintaining comparability, we changed the set of starting positions for every three participants. Testing took approximately 30 minutes, and participants could withdraw at any point to prevent fatigue-related bias.

9.4.2 Quantitative Results

The user study results are summarized in Table 9.1, showing the average reward and 95% confidence interval. The reward was calculated as $R = \text{IoU} \times 1000 + B_r$, where R is the reward and B_r is the remaining budget at the end of the episode. MapEx results are divided based on the changes in starting positions between rounds 2 and 3. Notably, MapEx has large confidence intervals due to its deterministic nature; unlike human participants, it could not repeat runs from the same starting location.

The combined scores in the rightmost column of Table 9.1 show that in round 1, even without prior knowledge of the map, participants already outperform MapEx. Examining the scores more closely, participants surpassed the baseline across all maps and rounds, except for Map 2 in round 2 and Map 4 in round 3. Notably, user performance on Map 2 did not improve between rounds 1 and 2. Additionally, the confidence interval for this map was significantly larger than for the others. This discrepancy may stem from the difficulty of discovering the bottom left region, which was easily overlooked. Many participants missed these rooms and continued searching elsewhere, while those who found the region achieved much higher rewards. This led to greater variance in the results, highlighting the need for reasoning beyond directly observed areas.

Generally, we observe a performance gap between humans and MapEx. While this gap is small in round 1, it generally increases across rounds, indicating that exposure to the maps aids in exploration. Notably, performance varied among participants. Most were graduate students in robotics or related fields, with some conducting planning research and field tests using frontier-based planning methods. These participants consistently outperformed others across all maps and rounds. One participant with

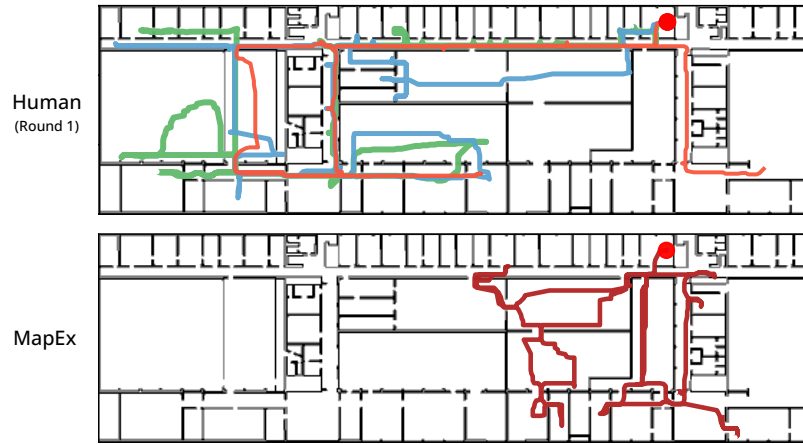


Figure 9.4: Three user runs and MapEx on Map 4, with colors differentiating the separate users and the red dot marking the starting position. Users explored more evenly across the space, leading to higher final reward.

extensive hands-on systems experience with robot exploration scored 907, 1220, and 1089 in round 1, and 873, 1260, and 1056 in round 3—significantly higher than peers or the baseline. This suggests that certain strategies or heuristics are more effective and that general planning experience may be beneficial. These higher scores further highlight the disparity between SOTA performance and expert human behavior.

9.4.3 Qualitative Results and Insights

Observing the participants’ strategies, those who performed best focused on maximizing prediction IoU rather than defaulting to maximizing map coverage. Some took unnecessary actions, such as searching rooms that already had clear predictions. Users often adjusted their strategies based on the map’s scale—exploring rooms more in smaller maps and focusing on main hallways in larger ones. Participants also aimed to minimize backtracking, describing different approaches of thinking a few steps ahead or considering map topology for efficient traversal. The benefit of an action was often estimated by imagining the observations at the frontier. Most ignored the remaining budget until later in the episodes, when they began estimating the cost of reaching frontiers with limited steps left. Many were surprised at the task’s difficulty and reported feeling fatigued by the end.

Fig. 9.4 shows the paths chosen by three users in round 1 compared to MapEx. MapEx concentrated its exploration budget in a limited region, resulting in insufficient exploration of the entire space. In contrast, the human users distributed their exploration more evenly, despite having no prior knowledge of the environment. We observed that between rounds 1 and 2, where the start position remained the same, users realized

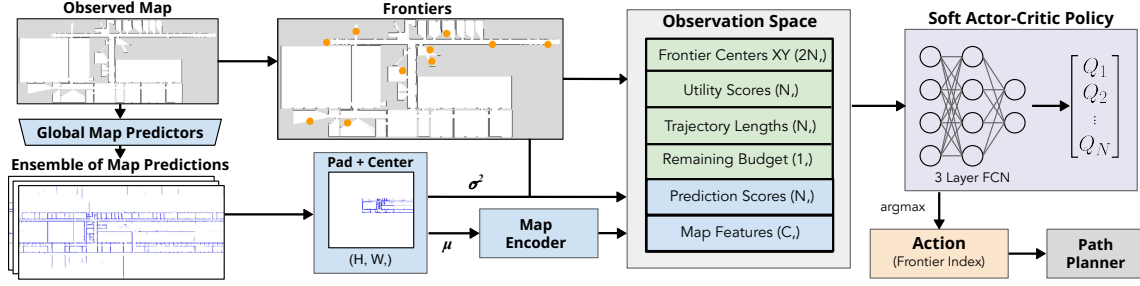


Figure 9.5: The MapExRL pipeline, where observed maps are processed through three independent global map prediction models, generating prediction maps. These maps are averaged and passed through a convolutional encoder to extract a 256-dimensional feature vector. This vector is concatenated with frontier centers, prediction and utility scores, distances from the agent, and remaining budget. The resulting vector is fed into a fully connected network that outputs N values, and the argmax is selected as the index of the frontier action.

the overall size and shape of the space and adjusted their strategies to explore more evenly.

From the quantitative and qualitative results from the user experiments, we found the following actionable insights:

1. Leveraging the global map predictions can likely benefit exploration as it allows for maximization of prediction accuracy through prioritizing observing uncertain areas in the predictions instead of exploring exhaustively.
2. The value of an action relates to the balance of exploitation and exploration while also considering both area coverage and prediction uncertainty.
3. Strategies must adapt to the map scale relative to the budget, as a single heuristic cannot perform effectively across all map sizes.
4. An ideal policy has budget awareness throughout the entire planning process to minimize backtracking. Thinking more than one step ahead helps avoid backtracking and creates more efficient paths.

9.5 Approach

Based on our user study, we propose a Reinforcement Learning (RL) pipeline using the Soft Actor-Critic (SAC) algorithm [Haarnoja et al., 2018], leveraging frontiers as an efficient action space. Our policy input incorporates global map predictions for global context and calculates predicted frontier information gain. This approach enables the model to integrate multiple scoring methods (user insights 1 and 2), global context

(user insight 3), and remaining budget (user insight 4) to select actions more effectively than heuristic-based methods. Fig. 9.5 provides an overview. For implementation, we used Stable Baselines 3 [Raffin et al., 2021], with SAC consisting of two fully connected layers for both actor and critic.

9.5.1 Map Encoder

Building on user insight 1, we use a map prediction model [Suvorov et al., 2022], which predicts the global map based on the accumulated occupancy map from LiDAR observations. Our work in the last chapter has shown its effectiveness in estimating global topology beyond the robot’s current observations. Following our previous work, we model the prediction uncertainty by generating multiple independent predictions from an ensemble of networks and calculating the variance of their outputs. This encourages a robot to prioritize uncertain areas.

Using the average of the ensemble of predicted maps, we generate a top-down map centered on the robot’s position, applying padding and cropping to produce a 1600x1600 pixel map. This map is then resized to 256x256 pixels using interpolation, followed by max pooling to obtain a 128x128 pixel input for the encoder. These operations ensure consistency across maps of varying shapes and sizes, following user insight 3—no single heuristic works optimally in all scenarios, and strategies must adjust to different map scales.

The 128x128 pixel map is processed by an encoder with three convolutional layers and a fully connected layer, producing a $C=256$ dimensional latent vector. Separate map encoders are used for the actor and critic.

9.5.2 Observation and Action Spaces

When a new action needs to be chosen, frontiers are calculated from the current occupancy map. In order to have a fixed action space size, this set of frontiers must be reduced to a fixed number N . We choose the top N frontiers based on their prediction scores, excluding those outside of the robot-centered map. In cases when there are fewer than N frontier centers, we add zero padding for the remaining. In our implementation, we select $N = 10$ frontiers based on experimental results, as they provide sufficient exploration options in all directions across training and testing maps.

To ensure the selected frontiers are unique and well-distributed across the map, we compare the prediction and utility scores of frontier points within a 5-meter distance threshold. If the score differences are below an experimentally determined threshold (0.01) and the points are spatially close, we remove duplicates to maintain diversity in

the observation space.

The observation space includes the 256 encoded map features, $[x, y]$ coordinates of the N frontiers, utility and prediction scores for these frontiers, trajectory length from the robot’s current position to each frontier, and the remaining distance budget. Each frontier center is defined relative to the robot’s position to align with the map transform.

The frontier features are calculated as follows:

- **Frontier Centers:** convolve an edge detection kernel across the observed map, group and filter to be above a minimum frontier edge size (following [Yamauchi, 1997]), find the center, translate into a robot-centered convention, and normalize by half the robot-centered map width.
- **Utility Scores:** raycast and create visibility mask in the occupancy map, sum the number of unknown cells in the visibility mask, normalize by the A* distance to the frontier, and apply min-max normalization.
- **Prediction Scores:** probabilistic raycast in the mean predicted map, sum the variance of the cells in the visibility mask, normalize by the A* distance (rather than Euclidean as in MapEx), and apply min-max normalization.
- **Trajectory Lengths:** A* distances to the frontiers from the robot’s current position, normalized by B
- **Remaining Budget B_r :** the remaining mission budget, normalized by the total budget B .

The policy network outputs a N -dimensional vector (action space), where each value represents one of the top N frontier options. The frontier corresponding to the highest value in this vector is selected as the next action.

9.5.3 Reward Function

To achieve 95% IoU as efficiently as possible, our reward function considers both the final episode IoU and the remaining budget. We designed the reward function to provide feedback only at the end of the episode, based on the accuracy of the predicted map compared to the ground truth. This fully sparse reward is feasible given our efficient action space, which inherently limits the number of steps per episode. The reward R is defined as

$$R = \begin{cases} \max(0, \text{IoU} - \delta_{\text{IoU}}) \times 1000 + B_r, & B_r = 0 \text{ or } \text{IoU} \geq 0.95 \\ 0, & \text{otherwise} \end{cases}$$

where B_r represents the remaining mission time or budget, while $\delta_{\text{IoU}} = 0.4$ clips the IoU to create a more evenly distributed reward signal and strengthen the feedback to the network. With nearly all training episodes achieving an IoU above 0.4, this clipping significantly improved training performance. This reward function encourages efficient exploration by rewarding the model for reaching 95% IoU while minimizing total distance usage.

We use the Intersection over Union (IoU) of the occupied cells for the global map prediction as our metric of performance. IoU is computed the same as in Section 8.7.3, where

$$\text{IoU} = \frac{\text{TP}}{\text{FP} + \text{TN} + \text{TP}}$$

and TP, FP, TN denote true positives, false positives, and true negatives, respectively. We again add a binary dilation to have IoU not penalize small pixel-wise errors, but rather focus on overall topology correctness.

9.5.4 Overall Exploration Cycle

At the start of each episode, the robot continuously builds a 2D occupancy map with the incoming LiDAR observations. The observed map is then fed into 3 independent global map prediction models, each trained on different subsets of the training maps, to produce an average map and variance map for frontier’s prediction score calculations. After selecting the top N frontiers, the frontier features and the mean map are computed and subsequently provided to the policy model, which determines the frontier for the robot to explore. An A* path from the current location to the frontier center is found, and the robot advances until the goal is reached or is deemed unreachable. This iterative process persists until the mission budget B or the robot successfully predicts the global occupancy map with 95% IoU.

9.5.5 Simulation Details

We evaluate MapExRL on the real-world KTH floorplan dataset [Aydemir et al., 2012]. The dataset consists of 149 campus floorplan blueprints from the KTH Royal Institute of Technology, which detail wall and door positions. We use 7 maps for training and 5 maps (shown in Fig. 9.3) for testing MapExRL, selecting maps that are medium to large in size, have complex topologies, and were not in the training set for the global map predictor model. We simulate a robot operating in the building with a LiDAR sensor that has 2500 scanning beams at a 20-meter range. Our maps are scaled such that one pixel is equal to 10 cm.

9.5.6 Baselines

Our first baseline is the conventional frontier-based exploration method [Yamauchi, 1997], where the agent chooses the nearest frontier based on Euclidean distance, without utilizing any global map predictions. Our second baseline, MapEx, selects the frontier with the highest information gain, calculated by jointly considering sensor coverage and variance from predicted maps, normalized by the Euclidean distance. We also include a random exploration baseline, where the agent randomly selects a frontier from the top N frontiers in the RL observation space. This baseline validates that our model learns to navigate efficiently using the top N frontiers rather than randomly picking from the list.

Additionally, we compare against a baseline inspired by Zwecher et al. [2022], which uses RL with motion primitives as actions. We adopt their action spaces and reward functions to train the policy while employing our global map prediction module. The action space consists of 8 primitive motions ($\mathcal{A} = \nwarrow, \uparrow, \nearrow, \leftarrow, \rightarrow, \swarrow, \downarrow, \searrow$), each with a constant step size Δx , uniform across all methods to ensure consistency with our approach. The reward function heavily penalizes actions that result in collisions with occupied cells and rewards actions that increase the number of newly observed cells. For the details of the reward function, refer to Equation 4 of Zwecher et al. [2022]. This baseline helps validate the impact of using frontier points as action spaces instead of primitive motions. We used the PPO algorithm to train the policy, following the approach in Zwecher et al. [2022], and trained the model with the same number of steps to ensure a fair comparison with our method.

Lastly, we compared our model with an ablation model, referred to as MapExRL No Pred, which excludes global map predictions from its observation space. If our model outperforms this ablation, it validates user insight 1, confirming the importance of incorporating global map predictions.

9.5.7 Training Details

During training, we set a maximum mission budget of $B = 500$, which is equivalent to 300 meters. The episode ends when the predicted map reaches 95% IoU or the robot exhausts the mission time. For comparison across all methods, we calculate IoU for each step using the predicted map, including for baselines that do not incorporate global predictions for frontier scoring.

The discount factor for SAC is $\gamma = 0.99$. The batch size is set to 256, the replay buffer capacity is 10,000, and training begins once the replay buffer exceeds 1,000 data points. Training steps is set to 4, τ is set to 0.02, and training frequency is set to 1. The Adam optimizer is used with a learning rate of 0.00073 for both policy and critic

Method	Map 1		Map 2		Map 3		Map 4		Map 5		Combined	
	Reward	IoU (%)	Reward	IoU (%)	Reward	IoU (%)	Reward	IoU (%)	Reward	IoU (%)	Reward	IoU (%)
MapExRL	726 ± 50	96 ± 1	465 ± 91	84 ± 7	722 ± 41	96 ± 0	370 ± 40	77 ± 4	329 ± 27	73 ± 3	522 ± 46	85 ± 3
MapEx	715 ± 40	96 ± 0	404 ± 90	79 ± 7	695 ± 52	95 ± 0	397 ± 36	80 ± 4	277 ± 40	68 ± 4	498 ± 48	83 ± 3
MapExRL No Pred	723 ± 69	95 ± 1	386 ± 94	77 ± 7	677 ± 49	95 ± 0	368 ± 31	76 ± 3	314 ± 38	71 ± 3	494 ± 47	83 ± 2
Random Policy	562 ± 88	90 ± 4	384 ± 57	78 ± 6	577 ± 52	94 ± 1	266 ± 23	67 ± 2	232 ± 32	63 ± 3	401 ± 41	78 ± 3
Nearest Frontier	482 ± 15	88 ± 2	310 ± 33	65 ± 8	561 ± 67	90 ± 4	232 ± 27	63 ± 3	94 ± 47	49 ± 5	336 ± 44	71 ± 4
RL Motion Primitive	57 ± 36	45 ± 4	16 ± 23	33 ± 6	77 ± 40	45 ± 6	0 ± 0	29 ± 5	0 ± 0	20 ± 7	30 ± 13	34 ± 3

Table 9.2: Average and 95% confidence interval for reward and IoU on the test maps. Each map includes 15 experiments from different starting positions, totaling 75 experiments across 5 maps.

networks. Our model is trained on an NVIDIA A100 GPU, which typically takes less than a day without parallelization to achieve 10,000 steps and train the optimal model.

9.6 Results

The performance of our model was evaluated on the 5 test maps with the same 15 randomly assigned start positions per map, resulting in a total of 75 experiments per method. The detailed results showing the average and 95% confidence interval for reward and IoU are presented in Table 9.2.

Our policy outperforms all baseline methods in four of five test maps, ranking second only on Map 4. It also surpasses the ablation model across all maps, validating user insight 1, which motivated this ablation. These results demonstrate our model’s ability to reason about and effectively utilize predicted maps, exploration budget, and robot observations, reinforcing the correctness of the users’ insights. Notably, our policy performed exceptionally on Map 2 and Map 5, with Map 5 being the largest and most complex in our dataset. For comparison, the nearest frontier-based exploration method achieves an average IoU of only 49% on Map 5, whereas our method reaches 73%. This map also underscores the inherent challenges associated with large maps and emphasizes the advantages of predictive global maps. The nearest-frontier method exhausts its budget, exploring only a small portion of the map, due to many close-proximity frontiers.

Map 2 is an irregular L-shaped building that is challenging to explore, creating numerous opportunities for backtracking, which is a common issue with the baseline methods. In contrast, our policy often effectively selected paths that minimized backtracking and efficiently explored hard-to-predict areas using the global map prediction models. A sample trajectory across all methods on Map 2 is illustrated in Fig. 9.6, demonstrating the efficiency of our policy in topologically challenging environments. As observed, the global map prediction models struggled to predict the bottom-left part of the building accurately, and our RL model was the only method to



Figure 9.6: Comparison of the six evaluated methods on Map 2 from the same start position. Notably, MapExRL is the only one that successfully explored the entire building and navigated the narrow passage leading to the bottom-left section that was not reached by the baseline methods.

successfully explore that area.

Overall, we observe a 4.8% improvement in exploration efficiency in all 75 combined evaluations compared to MapEx, the strongest baseline, with improvements reaching up to 18.8% on the largest and most topologically challenging Map 5. Additionally, we achieved a 30% improvement in performance compared to the random policy and a 55% improvement over the traditional nearest-frontier exploration method. Notably, our model outperformed all baselines on the largest and most topologically complex map (Map 5) and demonstrated strong performance on the irregular L-shaped map (Map 2), highlighting its advantages in challenging exploration tasks and its efficient use of Global Map Predictions and budget awareness. While performance gains are less pronounced on smaller maps due to their simplicity and limited time available to differentiate in performance, our model still maintains a consistent advantage.

Despite our careful implementation of the RL motion primitive method and earnest efforts to optimize the various parameters, the planner showed limited effectiveness. As shown in Fig. 9.6, this approach tended to resemble a random walk, with prolonged training primarily leading to a reduction in wall collisions. We verified our motion-primitive RL baseline on a simpler map, where it converged and explored well. However, when trained and tested on larger, complex maps, performance degraded significantly, likely due to many factors, including narrow doorways that are difficult to navigate and a weak training signal, as most actions yield no reward given the maps’ scale. While the model learned to avoid collisions, efficient exploration remained a challenge. This outcome highlights one of the advantages of our action space, which employs classical methods to effectively extract a set of actions from the map that will facilitate new observations.

To evaluate our method against the baseline established by our user study, we tested the MapExRL planner using the two sets of starting positions utilized in the study. Using the reward function from the user study, the combined reward for the start positions for rounds 1 and 2 was 860 ± 176 , while for round 3 it was 956 ± 164 . Notably, our model outperformed the human baseline in round 3. However, it did not

surpass the users’ performance in rounds 1 and 2, primarily due to a few low-reward runs that significantly impacted the average score. While a larger sample size would allow a more definitive comparison with human operators, these results underscore the continued high performance of skilled human operators and highlight the necessity for robustness and consistency in an exploration planner. Furthermore, the findings indicate that there remains room for improvement in our approach to further bridge the performance gap between robot and human exploration strategies.

9.7 Conclusion

In the work in this chapter, we present `MapExRL`, a novel human-inspired method combining reinforcement learning with frontier-based exploration and global map predictions to improve indoor mapping efficiency. Our user study informs key design choices and provides a benchmark for evaluating exploration methods. By optimizing frontier selection through a learned policy, `MapExRL` improves exploration strategies, achieving higher predicted map IoU with fewer resources. Experimental results on a real-world floorplan dataset shows its advantage over traditional SOTA methods and ablation studies, particularly in large, complex environments.

Chapter 10

Unified-Objective Belief Model for Search & Track

In our final chapter, we shift our focus to improving our belief model to generate more effective plans for maximizing information gain, with a particular emphasis on search and tracking tasks. Path planning for autonomous search and tracking of multiple objects is a critical problem in applications such as reconnaissance, surveillance, and data gathering. Due to the inherent competing objectives of searching for new objects while maintaining tracks for found objects, most current approaches rely on multi-objective planning methods, leaving it up to the user to tune parameters to balance between the two objectives, usually based on heuristics or trial and error.

In the work in this chapter, we introduce *UniSaT* (*Unified Search and Track*), a novel unified-objective formulation for the search and track problem based on Random Finite Sets (RFS). Our approach models unknown and known objects using a combined generalized labeled multi-Bernoulli (GLMB) filter. For unseen objects, *UniSaT* leverages both cardinality and spatial prior distributions, allowing it to operate without prior knowledge of the exact number of objects in the search space. The planner maximizes the mutual information of this unified belief model, creating balanced search and tracking behaviors. We demonstrate our work in a simulated environment, presenting both qualitative results and quantitative improvements over a multi-objective method.

10.1 Introduction

There are numerous applications in which autonomous robots are tasked with searching for or tracking multiple objects. They include scenarios such as disaster relief [Al-Kaff et al., 2019, Baeck et al., 2019, Karaca et al., 2018, Lyu et al., 2023],

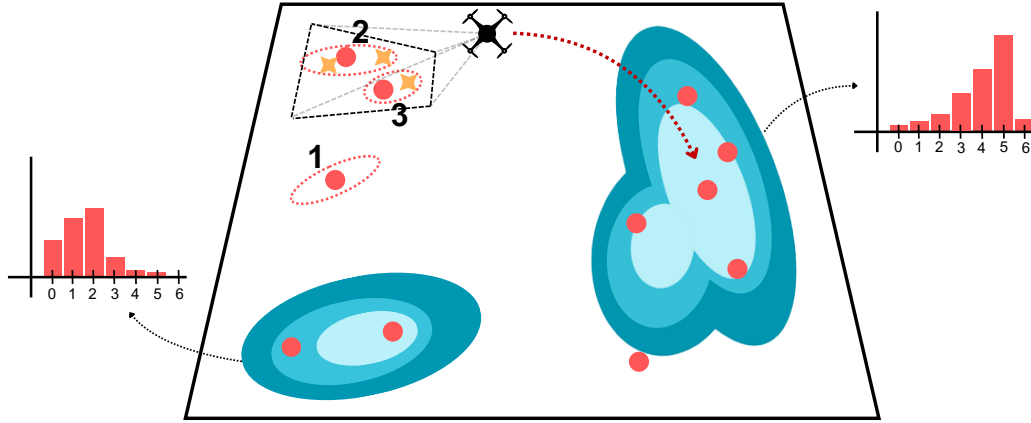


Figure 10.1: An example scenario of a UAV searching for and tracking objects in a space. The objects are represented by red dots. The UAV is tracking three objects, with two of them currently in the field of view. The Gaussian priors are represented by blue ellipses, and each cluster has an associated cardinality distribution, with the one on the right having a higher cardinality distribution. Using our unified objective function, the UAV planner chooses a path that maximizes both tracking and search performance, in this case leaving the tracked targets and searching the area on the right.

reconnaissance [Day and Salmon, 2021, Moon and Peterson, 2018, Moore et al., 2021], wildfire management [Akhroufi et al., 2021], and wildlife sciences [Kabir and Lee, 2021, Pfeifer et al., 2019, Shah et al., 2020]. In all of these cases, the primary objective is to maximize the knowledge about the objects within the environment, either by increasing the number of discovered objects or by maintaining low uncertainty regarding their current locations.

However, when combining the objectives of discovering undetected objects and tracking detected ones, typical approaches rely on multi-objective formulations that require heuristics or hand-tuned weights to balance competing goals, or they rely on simplifying assumptions. A natural trade-off exists between observing known objects to increase the probability of their existence and reduce state uncertainty, and exploring unknown areas to discover new objects. In cases where the exact number of objects is known—or in even simpler scenarios, such as single-object searches like in Furukawa et al. [2012]—the problem can be simplified to a single objective, such as minimizing entropy in a particle filter for each object. However, in real-world applications with multiple objects, knowing the exact count is rare and is more realistically represented by a probability distribution over the number of objects in the environment.

We propose leveraging an object population count distribution in order to create a unified objective function that takes into account the trade-off between gaining observations for known and tracked objects or searching the space for undiscovered objects. This is not merely a prior population density distribution across the space, but also a probability distribution of objects we expect within the search space. This

probability distribution gives more than just an expectation on the total number of objects and is not merely a prior density function or occupancy map over the space. Having a distribution allows for encoding the uncertainty and flexibility based on the confidence of the prior. In most search problems, there is a guess on the number and spatial distribution of objects, but the exact count and whereabouts of the objects may not be certain. Examples include finding cattle grazing in a pasture, finding missing hikers, animal censuses, or reconnaissance.

The contributions of this work are as follows:

- A novel belief model formulation for tracked and undiscovered objects through combining them into a single generalized multi-Bernoulli filter (GLMB).
- An approach that can incorporate both prior cardinality distributions as well as spatial distributions.
- A unified objective function to search for and track objects using the metric of differential entropy.
- Demonstration of our belief model in a partially observable Markov decision process (POMDP) formulation and presenting planner results with both qualitative and quantitative improvements.

Our method, termed *UniSaT* (*Unified Search and Track*), is shown to be effective in leveraging the prior beliefs and the cardinality and spatial distributions while still being flexible to inaccurate priors. The combined GLMB filter in our method is effective at handling measurement association, processing the updates for undiscovered and track targets, and outputting object tracks. *UniSaT* does not require choosing parameters to weight search and track objective functions and create intuitive paths that utilize the given priors.

This chapter is organized as follows: Section 10.2 gives an overview of related work for searching and tracking objects. Section 10.3 defines our problem statement. We present the relevant background for random finite set (RFS) filtering and our POMDP formulation in Section 10.4. Section 10.5 details *UniSaT* and implementation. The results are shown and analyzed in Section 10.6, and Section 10.7 is our conclusion and future work.

10.2 Related Works

The problem of Multi-Object Tracking (MOT) seeks to provide tracks for an unknown number of objects. When measurements are labeled and data association can be ignored,

object states are commonly estimated using the Kalman filter [Zhou and Kumar, 2023]. Otherwise, as in most multi-object tracking applications, the Kalman filter cannot be directly applied due to the necessity of computing data association. Alternative filtering approaches that handle the data association problem have been proposed, such as the Joint Probability Data Association (JPDA) [Fortmann et al., 1980], Multiple Hypothesis Tracker (MHT) [Blackman, 2004], and, more recently, RFS-based frameworks, such as the Probability Hypothesis Density (PHD) filter [Mahler, 2003] and the Generalized Labeled Multi-Bernoulli (GLMB) filter [Vo and Vo, 2013].

Random finite sets frameworks model both the number of objects and the object's states as random variables. This makes them suitable for applications where the number of objects is unknown. Previous works have used the RFS-based filters to guide the search to the point of maximum likelihood of finding objects in the local neighborhood [Sung and Tokekar, 2021] or use greedy search strategies [Dames et al., 2017, Nguyen et al., 2023], but these methods lack the ability to simultaneously optimize search and track.

When planning agent paths to maximize the results for MOT, many of the approaches handle simplified cases such as searching for one target [Coffin et al., 2022], assuming the number of objects is known and starting with initial tracks for each object in the space [Jeong et al., 2021], or searching until an object is found and then switching to purely tracking [Papaioannou et al., 2021]. This leads to a simpler problem where there does not have to be a tradeoff between exploring unobserved areas or exploiting the current object tracks. However, these methods would not work well when these assumptions are not met and the number of targets is unknown.

Other methods work with a fully unknown number of targets. The work by Dames et al. [2017] initializes the RFS filter with a prior belief over the space to guide an initial search, which then usually switches to tracking behavior once a target is found. The work of Yousuf et al. [2022] uses an SMC-PHD filter to estimate object locations and has a tunable parameter that weighs between refining object tracks or searching the space, where search rewards changes based on time and how well the space was seen. There are similar approaches that use occupancy grids or other models for the search space and then try to maximize the objective function for search as well as the one for track [Nguyen et al., 2023]. There are others that use combinations of objective functions weighted by scalar values [Yan et al., 2021]. One way to handle multiple objectives is to use the Pareto set and the global criterion method such as in Nguyen et al. [2023]. However, these methods still necessitate formulating two sometimes disparate objective functions that may not necessarily directly correlate to better performance on the final metrics.

The question that remains unclear is how one can solve the more general search and track problem, where the exact number of objects may be unknown, and generalize

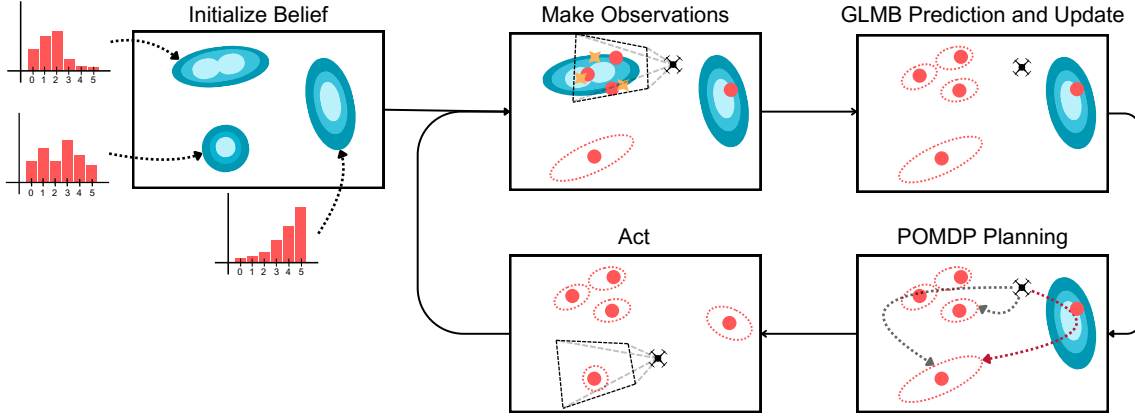


Figure 10.2: An overview of the *UniSaT* pipeline. The first step is the belief initialization from the population count, shown on the left. The algorithm then enters the execution loop: observe, update belief, plan, and execute plan. The measurements are represented by purple crosses and the tracks are represented by red dots. With our unified approach, the highest reward plan, highlighted in red, visits the blue density to try to find new objects before visiting one of the tracks to update its state estimation.

this solution to different scenarios. Although multi-objective search and track has been shown to be able to handle this more general case, it requires hand-picking of the scalar weights [Yan et al., 2021] or utilizing two conflicting objectives [Nguyen et al., 2023].

In this work, we propose a *unified objective* approach that unifies the two objectives (i.e., searching and tracking) and can be applied to different scenarios without modifications. *UniSaT* leverages prior information, often available to the user in real-world applications, to construct a single belief model that contains objects that were found and objects that are expected to be found based on spatial and cardinality distributions. With the searching and tracking being unified, object measurements can be directly associated with the prior rather than updating the search and track beliefs in a decoupled manner. We then plan on this unified belief model, calculating the value of an action for both search and track and simultaneously optimizing for the two objectives.

10.3 Problem Statement

We address the problem of a single robot tasked with searching and tracking multiple objects (also referred to as targets) with a limited field of view of the workspace. The robot is able to localize itself in the environment, *e.g.* using GPS. The number of objects in the workspace is not known exactly beforehand, but we assume we have access to a population count probability distribution and spatial distribution

that we can use to draw information on the expected number of objects and their probable locations. Our goal is to find all objects in the workspace while minimizing the uncertainty of their states. An overview of *UniSaT* is seen in Fig. 10.2.

Robot: The robot (also referred to as agent) follows a discrete time motion model

$$u_{k+1} = \tau(u_k, a_k),$$

where u is the robot state and $a \in \mathcal{A}$ is the control input (i.e., robot/agent action). $\mathcal{A}$ is a discrete set of control inputs.

Tracks: Let X be the multi-object state, unknown to the robot. Each object $x \in X$ follows a discrete-time stochastic motion model

$$x_{k+1} = f(x_k) + v_k,$$

where v is zero-mean additive Gaussian noise $v \sim \mathcal{N}(\cdot; 0, P)$.

Measurements: We assume that we receive a set $Z_k = \{z_{1,k}, \dots, z_{n,k}\}$ of position measurements at each time step. The measurement set Z_k is a disjoint union of clutters (i.e., false positives) and detected objects. Each object $x \in X$ has a probability of $P_D(x, u)$ of being detected by the robot with state u and generates a measurement $z \in Z$, given by

$$z_k = q(x_k, u_k) + w_k,$$

with additive zero-mean Gaussian noise $w \sim \mathcal{N}(\cdot; 0, \Sigma)$.

Remark 1: The measurement set Z always depends on the robot state u and the multi-object state X . For notational simplicity, we denote $P_D(x) \equiv P_D(x, u)$, $q(x) \equiv q(x, u)$.

10.4 Background

10.4.1 POMDP for Searching and Tracking

The RFS framework is an effective method to incorporate the stochasticity in target measurements and the multiple track hypotheses as a result. We use a POMDP formulation to plan over this partially-observable problem and define our POMDP tuple as

- $\mathbb{X}$ is the space of multi-object states,
- $\mathcal{A}$ is the set of all agent actions,

- $f(\mathbf{X}|\mathbf{X}', a')$ is the set of conditional transition probabilities of the agent for $\mathbf{X}, \mathbf{X}' \in \mathbb{X}$ and action $a' \in \mathcal{A}$ is the previous action,
- $\mathbb{Z}$ is the observation space,
- $g(Z|\mathbf{X}, a')$ is the conditional observation probability, or likelihood, of the observation $Z \in \mathbb{Z}$ from $\mathbf{X} \in \mathbb{X}$ after the action $a' \in \mathcal{A}$,
- $\mathcal{R}(Z, \mathbf{X}, a')$ is the reward for the observation $Z \in \mathbb{Z}$ from $\mathbf{X} \in \mathbb{X}$ after the action $a' \in \mathcal{A}$, and
- H is the time horizon for the look-ahead.

The system evolves according to the transition density $f(\mathbf{X}|\mathbf{X}', a')$, that coordinates how the multi-object state $\mathbf{X} \in \mathbb{X}$ is updated given the previous state $\mathbf{X}' \in \mathbb{X}$ and the current agent's action $a' \in \mathcal{A}$. After taking action $a \in \mathcal{A}$, the agent obtains an observation $Z \in \mathbb{Z}$ that has likelihood $g(Z|\mathbf{X}, a)$. The agent does not have access to the true multi-object state, instead, it uses these observations to update a belief of the true state.

The goal is to find a sequence of actions $(a_k, \dots, a_{k+H-1})$ that will lead to the highest cumulative reward over the time horizon. For that, a value function $V(a_k : a_{k+H-1})$ is constructed from the belief state and cumulative reward over H and assessed to pick the best action sequence. In general, however, this value function does not have an analytical form and numerical approximations are shown to be intractable [Hauskrecht, 2000]. A tractable approximation can be computed based on the idea of a Predicted Ideal Measurement Set (PIMS) [Mahler, 2007]. Using the multi-object density as the information state $\pi_{a_{k-1}, k}(\cdot|Z_k)$ at time k , we first compute the predicted multi-object density in the next time step

$$\hat{\pi}_{a,j}(\mathbf{X}_j) = \int \prod_{i=k+1}^j f_i(\mathbf{X}_i|\mathbf{X}_{i-1}, a) \pi_{a_{k-1}, k}(\mathbf{X}_k) \delta \mathbf{X}_{k:j-1}$$

from which the multi-object state $\hat{\mathbf{X}}_{a,j}$ can be computed via an estimator. To make the computation of the value function tractable, we then assume that *i*) we will apply the same action a over the horizon $k : k + j - 1$, and *ii*) we will obtain an ideal set of measurements $Z_{a,j}$ (without noise, clutters) at each time step after applying the action a . The approximated value function $V(a)$ is then computed by

$$V(a) = \sum_{j=k+1}^{k+H} \mathcal{R}(Z_{a,j}, \mathbf{X}_j, a)$$

where $Z_{a,j}$ is the ideal measurement calculated using the measurement model and the current belief of multi-object states.

10.4.2 Labelled Multi Bernoulli RFS

The RFS framework models multi-object states and measurements as RFS-valued random variables. The realizations of such random variables are sets in which both the number of elements (in our case, the number of objects) and the elements themselves (for us, the objects' states) are random. In this work, we adopt the labeled extension of RFS which assigns a unique label to each object.

A Bernoulli RFS X on $\mathbb{X}$ follows a Bernoulli distribution: either has one element with probability r , or is empty with probability $1 - r$. Its distribution is given by

$$\pi(X) = \begin{cases} 1 - r, & X = \emptyset \\ r p(x), & X = \{x\}. \end{cases} \quad (10.1)$$

In the case of multi-object tracking, it means that an object either exists with probability r , in which case its state distribution is $p(x)$, or it does not exist with probability $1 - r$. A Labeled Bernoulli RFS is a Bernoulli RFS in which the state $x \in \mathbb{X}$, where $\mathbb{X}$ is the kinematic space, is extended by a label $l \in \mathbb{L}$ to construct a *labeled* state $\mathbf{x} = (x, l)$. A Labeled Multi-Bernoulli RFS (LMB RFS) is the union of multiple independent Labeled Bernoulli RFSs.

10.4.3 Multi-Object Dynamic Model

The evolution of the RFS of objects being tracked is described by the transition density f_+ . In each time step, an object with state $\mathbf{x} = (x, l)$ either survives to the next time step with probability $P_S(x)$ or dies with probability $1 - P_S(x)$. In case of survival, the object's new state $\mathbf{x}_+$ is given by the transition density $f_s(x_+|x)\delta_l[l_+]$, where $\delta_l[l_+]$ makes sure the label stays the same. New objects can also be born with probability $P_B(l_+)$. In this case, the object's state is $\mathbf{x}_+ = (x_+, l_+)$ and its unlabeled state x_+ is distributed according to $f_B(\cdot, l_+)$. It is often assumed that objects survive/die or are born in the next step independently. The transitions of objects to the next time step are independent of the agent's actions.

10.4.4 Multi-Object Observation Model

The measurements and associated noise (misdetections, false alarms and data association uncertainty) are captured by the multi-object observation likelihood g . An object with state $\mathbf{x} = (x, l)$ that is in the sensor's field of view is either detected with probability $P_D(x)$, in which case it generates an observation z with likelihood $g(z|x)$, or missed with probability $1 - P_D(x)$. An observation z can also be generated by false

alarms, modeled as a Poisson distribution with rate $\kappa(z)$. It is often assumed that detections are independent of each other and clutter. The observations obtained by the agent, including ones generated by objects and false alarms, make the observation set Z .

In each time step of the filtering process, all the combinations of measurement-to-object associations are considered. This is needed because data association is unknown. An association map γ is a tuple of 1-1 maps

$$\gamma : \mathbb{L} \rightarrow \{-1 : |Z|\},$$

where $\gamma(l) = -1$ if object l is not in the RFS (dead or unborn), $\gamma(l) = 0$ if the object was misdetected, and $\gamma(l) > 0$ if the object was detected. It is assumed that each object generates a single measurement, which is guaranteed by the 1-1 property.

10.4.5 GLMB Filter

The GLMB filter is a rigorous Bayesian filter for multi-object tracking that, under the multi-object system model described above, recursively updates a density of tracked objects in the form of a GLMB. The GLMB density used in this chapter has the form:

$$\pi(\mathbf{X}) = \sum_{I, \xi} w^{(I, \xi)} \delta_I(\mathcal{L}(\mathbf{X})) \left[p^{(\xi)} \right]^{\mathbf{X}} \quad (10.2)$$

where $p^{\mathbf{X}} = \prod_{\mathbf{x} \in \mathbf{X}} p(\mathbf{x})$ is the joint multi-object distribution, and $p(\cdot, l)$ is a distribution in $\mathbb{X}$; $\mathcal{L}(\mathbf{X})$ gives the labels in $\mathbf{X}$; each I is a finite subset of the label space $\mathbb{L}$; each ξ is a history $\gamma_{1:k}$ of association maps up to the current time; and each $w^{(I, \xi)}$ is a non-negative weight such that $\sum_{I, \xi} w^{(I, \xi)} = 1$. For convenience, we refer to the GLMB density as the set of components $\pi \triangleq \{(w^{(I, \xi)}, p^{(\xi)})\}$.

Before updating the density with new measurements, a prediction step is applied using the Chapman-Kolmogorov equation to account for the evolution of the targets [Mahler, 2007]

$$\pi(\mathbf{X}_+) = \int f(\mathbf{X}_+ | \mathbf{X}) \pi(\mathbf{X}_+) \delta(\mathbf{X}). \quad (10.3)$$

The multi-target posterior is then updated with new measurements via Bayes formula [Mahler, 2007]

$$\pi(\mathbf{X} | Z) = \frac{g(Z | \mathbf{X}) \pi(\mathbf{X})}{\int g(Z | \mathbf{X}) \pi(\mathbf{X}) \delta \mathbf{X}} \quad (10.4)$$

where the integral in the denominator is a set integral defined as

$$\int f(\mathbf{X})\delta\mathbf{X} = \sum_{n=0}^{\infty} \int_{\mathbb{X}^n} f(\{\mathbf{x}_1, \dots, \mathbf{x}_n\})d(\mathbf{x}_1, \dots, \mathbf{x}_n).$$

These two steps are recursively repeated over time. The GLMB is closed under the prediction and the update, therefore, the result of these operations is also a GLMB. In practice, because the number of GLMB components grows quickly, truncation is done by removing components with a very low probability of existence [Vo et al., 2019].

10.5 Approach

We present a unified belief model to jointly plan for searching and tracking, leveraging information from a population count distribution. This information is used to initialize an RFS belief of target states and target count. This belief is a joint representation of known and unknown targets that we expect to find. This joint representation is used to compute a differential entropy reward that simultaneously reasons over search and track.

10.5.1 Belief Space Setup

The belief initialization is done at the beginning of the mission to model unknown targets that we expect to find based on the information provided in the population count distribution. The population count distribution has two parts: one spatial and one discrete. The spatial part $p^X = \prod_{x \in X} p(x)$ is a probability density over the workspace that informs the probability of finding a target at a certain position. This distribution allows the user to guide the search to areas with a higher likelihood of finding targets. If no prior information is available, a uniform distribution can be used. The discrete part $\rho(n) = \Pr(|X| = n)$ is a cardinality distribution (a discrete probability distribution with $n = 0, 1, 2, \dots$ is an integer number) that informs us of the probability of the number of targets in the workspace.

On initialization, we set up the belief space leveraging the information in the population count distribution. We first extract the highest possible cardinality in the population count distribution, which we call N . In other words, N is the only integer that satisfies $\Pr(|X| > N) = 0$. It then allows us to initialize the GLMB density to match the population count distribution. The first step is to initialize the hypotheses. Each hypothesis is a set containing the labels of the tracks in this hypothesis. Let $i_k \in \mathcal{I}$ be the set of all hypotheses with k tracks and $\mathcal{I}$ is the set of all GLMB hypotheses.

Formally,

$$i_k = C(k, N),$$

where $C(a, b)$ is a function that outputs the set of all a choose b combinations. Considering all combinations is important because the population count does not inform us on which tracks exist—it gives us a probability distribution over the number of tracks, but no information about the exact labels l_i . For instance, if we have a total of 4 tracks in the population count distribution, possible hypotheses with 3 tracks are $\{l_1, l_2, l_3\}$, or $\{l_1, l_2, l_4\}$, and so on. At the end, the set of all hypothesis $\mathcal{I}$ is the union of the sets i_k , $\forall k = 1, \dots, N$ tracks,

$$\mathcal{I} = i_0 \cup i_1 \cup \dots \cup i_j \dots \cup i_N.$$

The weight $w(i_k)$ of a hypothesis i_k with k elements is equal to the probability of k elements existing,

$$w(i_k) = \rho(k) = \Pr(|X| = k).$$

Once we initialize all hypotheses, we must initialize the states of the tracks in the hypothesis. Since all hypotheses are combinations of up to N elements, we must initialize N tracks. This is done using the spatial portion of the population count distribution X . For each, track $l_i, i = 1, \dots, N$, we sample N_p realizations of X to create a particle cloud that represents the distribution over this track's state $p^{(l_i)}(x)$. We use a Sequential Monte Carlo (SMC), or particle cloud, representation of the track state distribution to allow for multi-modal priors.

At the end of this stage, the GLMB density is a representation of the unknown targets that we expect to find during the search. Multi-object tracking is then performed over this same density, which means that we have a single RFS containing the information used to optimize both search and track.

10.5.2 Multi-Object State Estimation

Multi-object state estimation is done using a combination of the Gaussian Mixture GLMB (GM-GLMB) and the Sequential Monte Carlo GLMB (SMC-GLMB) filters. They only differ in that the joint multi-object distribution in (10.2) is represented by a mixture of Gaussians in the GM-GLMB and by a particle cloud in the SMC-GLMB. The GM-GLMB is used to track objects that have already been found, *i.e.*, it is used for tracking. The SMC-GLMB is used to maintain the belief of objects expected to be found, as described above, *i.e.*, it is used for search. We use a combination of both methods because, while the GM-GLMB is much more efficient, the SMC-GLMB allow us to handle multi-modal priors for search.

After we initialize the belief with the SMC-GLMB, the GLMB density is updated as usual using the recursion described in (10.3) and (10.4), with the addition of negative observations. Negative observations are used to update the belief when we *don't* receive a measurement for an object we predicted could be there: if we don't see an object there, we might not know where it is, but we know we either misdetected it or it is not there. The observation likelihood is thus given by

$$g(z_j|\mathbf{x}) = \begin{cases} \frac{P_D(x)\mathcal{N}(z_j; q(x), \Sigma)}{\kappa(z_j)} & , \text{ if } \gamma(\mathcal{L}(\mathbf{x})) = j > 0 \\ 1 - P_D(x) & , \text{ if } \gamma(\mathcal{L}(\mathbf{x})) = 0 \end{cases} \quad (10.5)$$

When an SMC track's uncertainty is below a threshold set by the user, meaning that an object expected to be found was indeed found, it is converted into a GM track. Potential mismatches (underestimates/overestimates) between the prior population count distribution and the actual number of objects in the workspace are automatically handled by the filter. In the case of underestimates, objects that were not expected to be found are added to the belief by the birth model. Because our agent has a limited field of view of the workspace, our birth model is distributed only across the agent's current field of view. In the case of overestimates, the filter's truncation operation removes SMC tracks when their probabilities of existence gets too low, meaning they were expected but never found.

10.5.3 Planning

This section gives an overview of our planning approach to jointly search for and track objects in an environment. We explain our use of differential entropy for both the SMC-GLMB and GM-GLMB, and outline our POMDP planning framework.

Differential Entropy

Differential entropy provides a way of measuring the uncertainty of a continuous random variable. In multi-object tracking, it can be used as a measure of the uncertainty of multi-object states. While differential entropy is intractable to labelled RFS in general, it has an analytical solution to the LMB RFS, given by the set of components $\boldsymbol{\pi} \triangleq \{(r^{(l)}, p^{(l)})\}_{l \in \mathbb{L}}$, where $r^{(l)}$ is the existence probability of track l , and $p^{(l)}$ is the state of track l . As a result, we propose approximating the GLMB density by an LMB density, which preserves the first moment of the original GLMB density. As defined in

[Nguyen et al., 2023], the differential entropy for an LMB $\mathbf{X}$ is

$$h(\mathbf{X}) = - \sum_{l \in \mathbb{L}} \left[r^{(l)} \ln r^{(l)} + \tilde{r}^{(l)} \ln \tilde{r}^{(l)} + r^{(l)} \langle p^{(l)}, \ln(Kp^{(l)}) \rangle \right] \quad (10.6)$$

where $\tilde{r}^{(l)} = 1 - r^{(l)}$, $\langle a, b \rangle = \int a(x)b(x)dx$ is the inner product, and K is the hyper-volume unit on $\mathbb{X}$. For the GM-GLMB, this inner product is defined as $\langle p^{(l)}, \ln(Kp^{(l)}) \rangle = 0.5 \log[\det(2\pi e \Sigma^{(l)})]$, where $\Sigma^{(l)}$ is the covariance of $p^{(l)}$, and for the SMC-GLMB it is given by the dot product $\langle p^{(l)}, \ln(Kp^{(l)}) \rangle = [\omega^{(l)}]^\dagger \cdot \log(r^{(l)} \omega^{(l)})$, where $\omega^{(l)}$ is the column vector of particle's weights of track l and $\dagger$ denotes a vector/matrix transpose operation. Using this definition of differential entropy, we can define the mutual information between the multi-object RFS $\mathbf{X}$ and measurement RFS Z as

$$I(\mathbf{X}; Z) = h(\mathbf{X}) - h(\mathbf{X}|Z). \quad (10.7)$$

Mutual information gives a measure of how much information is gained about a random variable by observing another. Therefore, in the context of state estimation where our goal is to choose actions that result in measurements that maximize the mutual information $I(\mathbf{X}; Z)$, which is equivalent to minimizing the differential entropy $h(\mathbf{X}|Z)$ since $h(\mathbf{X})$ does not depend on the control action a .

Planner

Planning is done in a typical POMDP planning framework, where the goal is to maximize the cumulative reward over a finite horizon H . The reward $\mathcal{R}(Z, \mathbf{X}, a)$ is the mutual information between the current belief and observed measurements. We define our value function as

$$V(a) = \sum_{j=k+1}^{k+H} I(\mathbf{X}_j; Z_j) \quad (10.8)$$

so as to maximize the mutual information between the current belief $\mathbf{X}$ and Z , where Z is the PIMS measurement set, which is equivalent to maximizing:

$$V(a) = - \sum_{j=k+1}^{k+H} h(\mathbf{X}_j | Z_j). \quad (10.9)$$

While we use a GLMB density as the multi-object state density, the differential entropy in (10.6) is defined for an LMB density. The first step in the planning step is therefore to convert a GLMB into an LMB [Reuter et al., 2014]. Since our belief $\mathbf{X}$ already contains the expected targets and their states from initialization, and new

tracks are always added to this same belief, we only need reason over one value function. When minimizing the differential entropy $h(\mathbf{X}|Z)$, the planner factors in the differential entropy of the SMC-GLMB in the sense of discovering a new target or reducing uncertainty through negative observations, and the differential entropy of the GM-GLMB tracks that have already been found. The plan thus minimizes the joint uncertainty of search and track.

10.6 Results

To demonstrate the efficacy of *UniSaT*, we compared its performance against the multi-objective approach in [Nguyen et al., 2023]. This baseline uses the global criterion method to handle the objective of minimizing differential entropy for the tracks and maximizing the probability of discovering a target in the occupancy grid.

We tested both methods in simulation in a handful of different scenarios with different prior distributions to demonstrate that *UniSaT* is able to consistently leverage the prior information despite its form. All tests were performed with a single agent operating in an environment of $1216 \text{ m} \times 1230 \text{ m}$ for a total time of $T = 500 \text{ s}$. We provide comparisons of the average optimal sub-pattern assignment (OSPA²) distance [Beard et al., 2017] achieved by *UniSaT* and the baseline at the end of the mission across 100 Monte Carlo (MC) runs.

10.6.1 Experiments Settings

The initial position of the agent in each MC run was randomly sampled from a uniform distribution over the workspace. Similarly, the initial positions of the targets were randomly sampled from the prior distribution in each run. The targets' states comprise position and velocity $\mathbf{x} = [p_x, p_y, v_x, v_y]^\top$, but for the scope of these tests, we use a random walk motion model for state estimation. The state distributions are represented by particle clouds with a maximum of 1000 particles per target and 1000 particles per hypothesis. Our actions space $\mathcal{A}$ comprises 12 even spaced trajectories emanating outward from the current agent state, with one additional action of staying in the same state. The planning horizon is 40 steps and each action belief sampling interval is 10 steps.

We assume the sensor is subject to noisy observations. We model false alarms as a Poisson distribution with a clutter rate of $\lambda = 0.01$ and account for misdetections assuming that a target within the sensor's field of view is detected with a detection probability of $P_D = 0.8825$. We measure performance using the OSPA² distance with a cut-off $c = 50 \text{ m}$. This metric accounts for errors in both the estimated number of

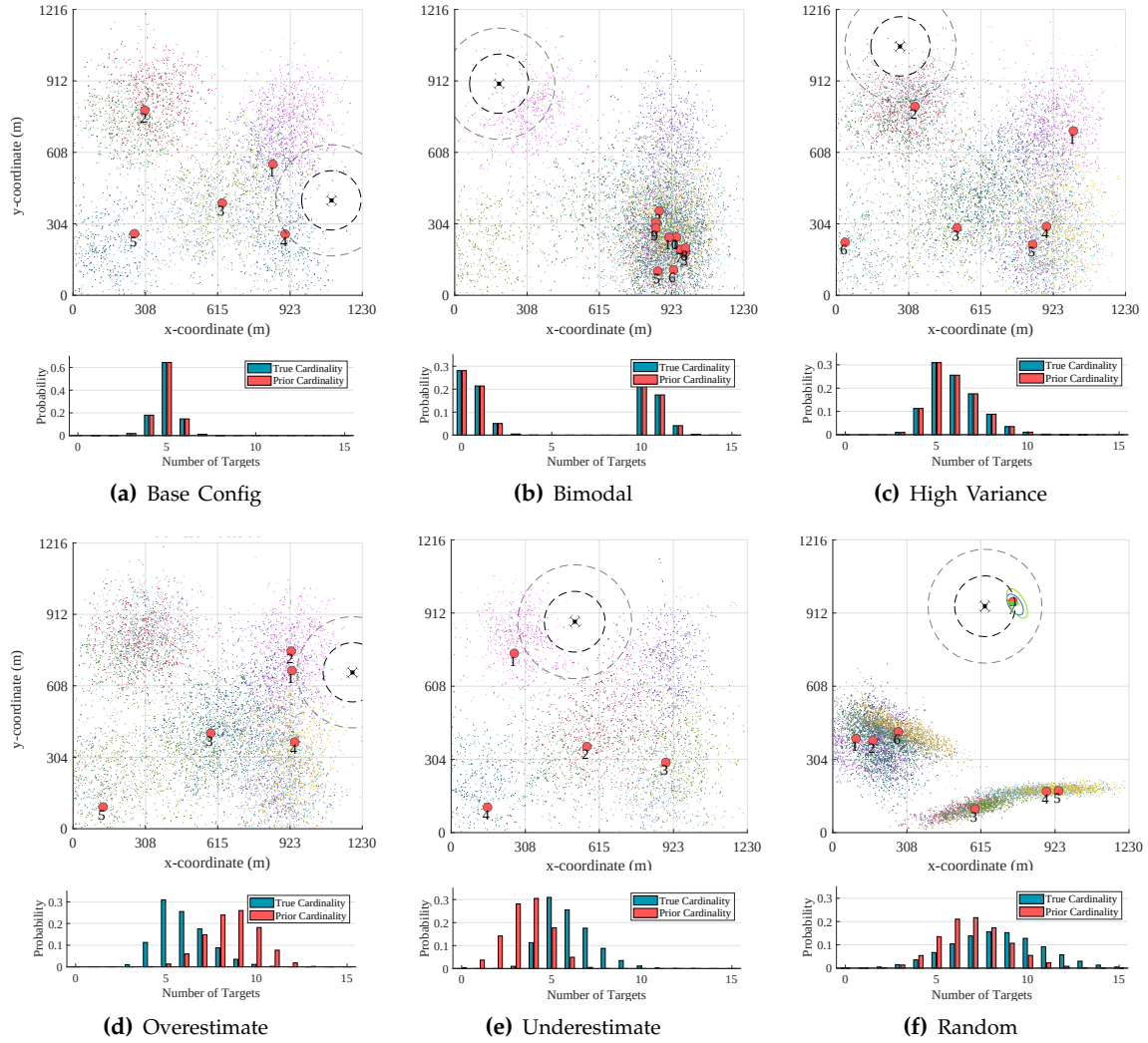


Figure 10.3: A randomly sampled environment from the five characteristic scenarios and a random environment setup. The true target states are shown in red and the prior SMC particles are plotted in distinct colors. Below each environment plot is the true environment cardinality distribution, which is what we sample the ground truth from, and the prior cardinality, which is used for belief initialization.

objects and their states. Therefore, lower $OSPA^2$ means better performance.

10.6.2 Results and Discussion

We first create a set of characteristic scenarios to demonstrate various environmental conditions and the effect of the prior on the behavior and performance of *UniSaT*. *Base Config*: Our base configuration is to compare against the subsequent variations. There are five normal distributions of targets placed throughout the environment. In the

Scenario	Ours (m)	Multi-Objective [135] (m)
Base Config	25.01 $\pm$ 1.14	32.78 $\pm$ 0.96
Bimodal	20.56 $\pm$ 2.36	29.04 $\pm$ 2.35
High Variance	28.15 $\pm$ 1.36	33.73 $\pm$ 1.02
Overestimate	29.98 $\pm$ 1.39	33.93 $\pm$ 1.02
Underestimate	22.35 $\pm$ 1.03	33.79 $\pm$ 1.03

Table 10.1: Results from 100 MC tests for each characteristic scenario. Table values are the mean OSPA² error and margin of error for a 95% confidence level.

base configuration, the prior distributions match the true distributions. *Bimodal*: In this scenario, the spatial distributions stay the same, but the cardinality distributions are altered to create a population cardinality distribution that is bimodal. *High Variance*: This scenario has a higher spread in the cardinality distributions than the base configuration. *Overestimate*: We keep the same cardinality distribution as the high variance case, but we test *UniSaT*'s performance when the prior cardinality distribution is an overestimate. The true expected target count is 5.95, but the prior has an expected target count of 8.63. *Underestimate*: This case tests when the prior is an underestimate. The true expected target count again is 5.95, but the prior has an expected target count of 3.60.

A random environment sampled for each of these scenarios is shown in Fig. 10.3. We also visualize the cardinality distributions for each of the priors. We note that each distribution has a unique cardinality distribution rather than one combined distribution for the total population. Each test randomly samples the true target cardinality and location from the true distributions.

The results for the five scenarios are summarized in Table 10.1. We can see that *UniSaT* outperforms the baseline in all of the scenarios. *UniSaT* performs well in both the base configuration and bimodal case, where there is a strong incentive to first visit clusters with high expected cardinality rather than searching uniformly in the space or the other priors. If a low number of targets are discovered in a prior cluster, the agent may move on to the other priors. Conversely, if the agent finds a high number of targets, it may choose to revisit the targets many times or even keep them within the field of view to maintain their tracks.

In the cases of high cardinality variance, the performance gap between the methods decreases. This is likely because the baseline search strategy performs well by searching the entire space. With the targets spread out fairly evenly, almost any action would lead to good performance. As expected, when we alter the prior to be an overestimate, the performance gap between the methods decreases. This is due to *UniSaT* being

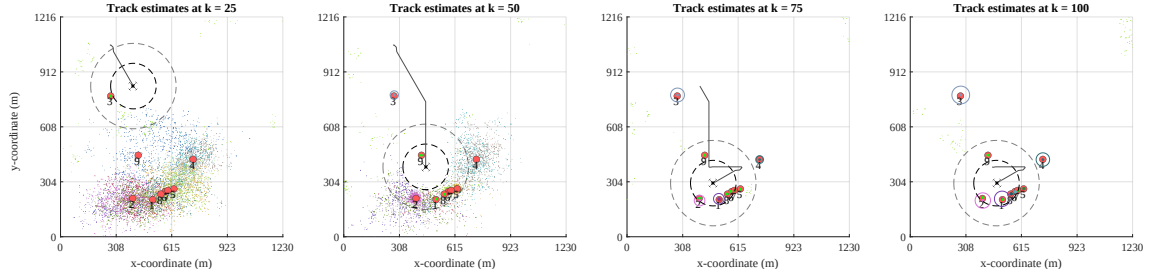


Figure 10.4: An example run of *UniSaT* at different time steps in our simulation environment. The range of detection is shown in dotted lines around the UAV, where the probability of detection drops off between black to grey circles. The targets are shown as red dots and the EKF track covariance can be seen around the targets being tracked. SMC belief particles are visualized with each hypothesis having a different color.

imbalanced in the searching versus tracking behavior and being biased to search more, as it expects to discover more targets in the space. This shows that *UniSaT* doesn't fail when the prior is an overestimate, but rather its performance decreases toward being closer to the baseline performance. In the underestimate case, our proposed method performs much better than the baseline. This demonstrates *UniSaT*'s flexibility in discovering unexpected targets without affecting the remaining expected number of targets. Thus, the cardinality distributions do not have to be completely accurate for *UniSaT* to outperform the baseline and benefit the planner.

We present an example execution of *UniSaT* in Fig. 10.4. This example demonstrates the intuitive behavior of the UAV (our considered agent), driven by our unified objective function that leverages the prior spatial and cardinality distributions. The UAV quickly discovers two targets, but rather than staying to track them, it moves on to search the other priors due to the potential to discover a high number of new targets. The UAV effectively discovers all of the targets in the cluster in the bottom left and then chooses to stay in a position where it can receive measurements for a large number of the targets.

We then test *UniSaT* on an environment with a random number of prior Gaussian clusters, locations, covariance, and cardinality distributions. The number of clusters is between 1 – 6, and they are placed randomly within the search space with a 100 m buffer from the edge. For the covariance of each Gaussian, we sample the standard deviation of the diagonal elements between 0 – 64 and for the off-diagonal elements, we sample a correlation coefficient between -1 and 1 , which we then multiply by the sampled standard deviations. For the cardinality distributions, we allocate a maximum of 3 targets per cluster, and we sample probabilities between 0 and 1 for each element of the cardinality distribution and then normalize the vector of probabilities. An example of a sampled environment is shown in Fig. 10.3(f).

The results for 100 MC tests are shown in Fig. 10.5. We can see that *UniSaT* quickly

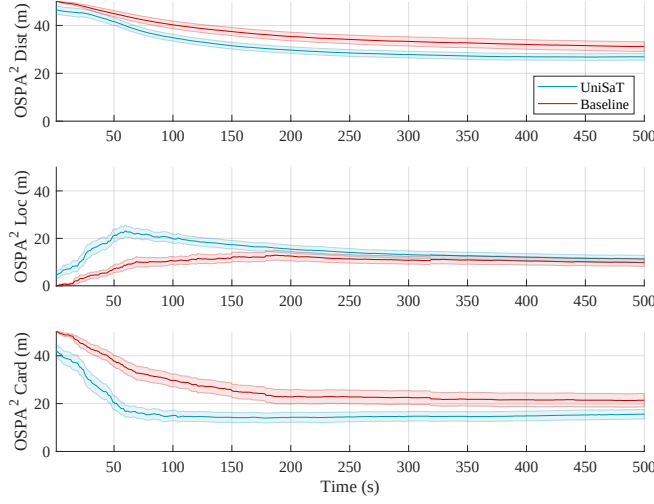


Figure 10.5: The mean OSPA² results of *UniSaT* and the baseline over 100 MC runs on the environment with random configurations, plotted with the 95% confidence interval.

reduces the cardinality error, as it is able to exploit its understanding of the prior and the value of identifying the undiscovered targets left in the search space. As a result of this, the localization error is higher than the baseline in the beginning but decreases to closely match the baseline over the course of the tests. The overall OSPA² for *UniSaT* is lower than that of the baseline. The final mean OSPA² error is 26.89 ± 1.44 m for *UniSaT* and 31.14 ± 2.05 m for the baseline, with the $\pm$ values representing the margin of error for a 95% confidence level.

10.7 Conclusion and Future Work

This work presents *UniSaT*, a planner and novel unified belief model for searching and tracking with an autonomous agent. The proposed belief model is constructed using prior knowledge of the objects in the environment and allows the planner to simultaneously optimize for searching and tracking, which normally are treated as two competitive objectives in hand-tuned multi-objective approaches. *UniSaT* can incorporate both the prior spatial and cardinality distributions and even uses them when creating new tracks from measurements. We compare *UniSaT* to a multi-objective approach through Monte Carlo simulations and show we outperform this baseline in a variety of scenarios with different distributions of objects in the environment.

Future work could choose to strictly adhere to the prior population cardinality distribution and prune all hypotheses that do not fall within a threshold of the

distribution. This approach would allow for all measurements to affect all hypotheses in the environment. When an object that was expected to be found in a region would not be found, the prior would be redistributed to other regions to keep the expectations on the number of targets constant. Another avenue could include exploring other methods of encoding the search space that is efficient while still taking into account measurement association and correlations between priors.

Chapter 11

Conclusion

This thesis advances the field of autonomous robotic information gathering by addressing key challenges in solving informative path planning (IPP) problems in high-dimensional spaces, incorporating real-world disturbances into planning, and improving world belief models. The proposed approaches enhance IPP by accounting for motion constraints and realistic sensor models, develop planning methods that incorporate environmental disturbances such as wind. Additionally, this work introduces techniques for leveraging prior information to improve exploration efficiency and information gain prediction, enabling more effective decision-making in complex environments. The effectiveness of these methods is demonstrated through extensive simulations and field deployments on multiple UAV platforms, validating their applicability across diverse real-world scenarios.

We begin by summarizing the key findings and contributions of this work in Section 11.1, structured around the three main challenges addressed in this thesis: high-dimensional informative path planning, disturbance-aware navigation, and world belief modeling. In Section 11.2, we outline future research directions, including generalizable and adaptive IPP frameworks, dynamic and causal world belief models, and human-on-the-loop information gathering. These directions emphasize opportunities for integrating learning-based adaptations, multi-agent coordination, and real-world deployment strategies to further advance autonomous information-gathering systems.

11.1 Summary and Contributions

11.1.1 Addressing Challenge 1: Solving IPP Problems in High-Dimensional Spaces with Non-Trivial Sensor Constraints

Challenge 1 considers the difficulty of solving IPP problems in high-dimensional spaces while accounting for real-world sensor constraints. Many approaches simplify planning by discretizing the configuration space and assuming simplistic sensor models (Chapter 2). These assumptions, while computationally efficient, introduce gaps between planned and executed paths, limiting real-world effectiveness.

In Chapter 3, we introduced **TIGRIS**, a sampling-based IPP planner that uses informed sampling to prioritize high-reward areas. We then extended this work in Chapter 4 by developing **IA-TIGRIS**, an adaptive and incremental planner that builds upon the informed sampling approach while incorporating real-time updates to belief maps. Unlike previous methods, IA-TIGRIS efficiently handles high-dimensional state spaces by balancing global trajectory planning with online local refinement, ensuring paths remain feasible and information-rich even in dynamic environments.

Additionally, this work introduces edge reward approximations to estimate information gain along a trajectory more accurately while maintaining computational efficiency. This allows IA-TIGRIS to plan in large spaces with complex sensor footprints, moving beyond traditional circular field-of-view models. We demonstrate that our method leads to higher-quality paths in both simulated and real-world deployments. By addressing these limitations, this thesis improves the applicability of IPP methods to real-world robotic systems operating under motion and sensing constraints.

11.1.2 Addressing Challenge 2: Incorporating Real-World Disturbances and Risk into the Path-Planning Framework

Challenge 2 considers the problem of incorporating real-world disturbances into IPP, particularly in the presence of wind. Many existing approaches assume an idealized environment where disturbances are ignored or approximated at a high level. However, as demonstrated in field tests, ignoring wind can lead to significant deviations from planned paths, resulting in suboptimal or even infeasible trajectories.

To address this, Chapter 5 introduces a **wind field estimation framework** that leverages in-situ UAV measurements to infer wind conditions in urban environments. Unlike static weather models, this approach allows UAVs to estimate local wind conditions in real time, leading to more accurate planning. Chapter 6 then builds on this by developing a **time-optimal trajectory planner** that incorporates wind effects.

Using a geometric reasoning approach, this method reduces computation time by 37.4% compared to previous methods, making it feasible for onboard deployment.

Finally, in Chapter 7, we introduce a **deep learning-based risk assessment framework** for evaluating UAV energy consumption in wind. Using Monte Carlo forward simulations and Conditional Value-at-Risk (CVaR) metrics, this approach quantifies flight risk under environmental uncertainty. This provides a principled way to balance mission success against operational risk, allowing for safer and more robust deployments. Together, these contributions improve the ability of UAVs to plan, navigate, and adapt in real-world environments affected by disturbances.

11.1.3 Addressing Challenge 3: Improving and Leveraging World Belief Models

Challenge 3 focuses on improving world belief models for IPP. The effectiveness of information gathering depends on how the environment is modeled, updated, and leveraged to evaluate the value of a path. In Chapter 8, we introduce a framework that leverages prior information maps to improve exploration strategies. This method, **PIPE**, enhances information gain prediction by incorporating learned priors, allowing for more efficient allocation of sensing resources. By testing this approach on an indoor exploration dataset, we show that leveraging prior knowledge can significantly reduce uncertainty while maintaining computational efficiency.

Building on this, Chapter 9 presents a reinforcement learning-based approach for human-inspired exploration called **MapExRL**. Motivated by insights from user studies on human navigation strategies, this method enables robots to explore leverage environmental context and multiple frontier scoring metrics. Our results demonstrate that integrating human-inspired heuristics into IPP leads to improved exploration efficiency, particularly in structured environments where prior knowledge is beneficial.

These contributions establish new connections between belief-space modeling, predictive learning, and reinforcement-based planning, broadening the scope of IPP beyond traditional formulations. By integrating learning-based approaches, this thesis bridges the gap between perception and planning, making IPP systems more adaptable to real-world deployment scenarios.

11.1.4 Lessons Learned

Over the course of this thesis, there are some general high-level lessons I have learned for developing data gathering algorithms.

User intuition is a strong baseline, but not always optimal.

Our user study for indoor exploration revealed that human-in-the-loop planning can be effective, and humans leverage many contextual cues and long-horizon reasoning. However, not all user strategies were equally effective, regardless of confidence levels. Human tactics can become overconfident heuristics that don't consistently yield good performance. Moreover, human performance tends to degrade as the number of competing factors increases and the cognitive load grows. For example, when deploying IA-TIGRIS on a fixed-wing UAV with a forward-facing sensor, users often believed the generated paths were ineffective, simply because it was hard to intuit what regions of space were actually being observed from the flight path alone. These impressions were difficult to correct, even when the paths were objectively effective. Additionally, humans tend to be better at identifying optimal paths in cost-minimizing problems than in cost-maximizing ones, especially when additional constraints complicate reasoning.

The objective function is just as important as the planner.

Even an optimal planner cannot produce useful behavior if it optimizes the wrong objective. A key lesson is that understanding what constitutes a "successful" mission takes time and often requires multiple iterations with stakeholders. It's tempting to formalize an objective function quickly, but initial consensus may mask subtle, subjective, or evolving expectations. A well-aligned objective function is critical to achieving the behavior stakeholders truly care about, and it often requires sustained dialogue and refinement.

Sensor modeling matters more than one might expect.

Changes in sensor footprint, altitude, and orientation can dramatically affect planning outcomes. Explicitly incorporating the sensing model into the planner, rather than treating sensing as a passive process, improves alignment between expected and actual robot performance. Additionally, the true positive and true negative rates (TPR and TNR) of the sensor model have significant influence on planner behavior. A conservative sensor model leads to more repeated observations, while an overconfident one can result in under-exploration. In practice, stakeholders may prefer to choose the model, deviating from what is the most accurate, because it produces behavior that better aligns with their expectations of how the robot should operate.

11.2 Future Directions

While this thesis has made significant contributions to informative path planning, real-world disturbances, and world belief modeling, there remain several promising avenues for future research. Expanding on the methods and insights presented in this work, the following directions aim to further improve the adaptability, robustness, and real-world deployment of autonomous information-gathering systems.

11.2.1 Generalizable and Adaptive Informative Path Planning

A key challenge in informative path planning (IPP) is ensuring that planners remain generalizable across different environments, sensing modalities, and task requirements. While IA-TIGRIS demonstrated adaptive online planning, future work should focus on meta-planning frameworks that dynamically adjust search strategies, exploration-exploitation tradeoffs, and path optimization techniques based on the problem instance. One promising direction is to learn structured priors on planning distributions, enabling planners to refine their decision-making based on previous deployments and online observations. This would allow IPP systems to be more adaptive and efficient in unseen environments.

Additionally, increasing the scalability of IPP planners is critical for deploying multi-robot and large-scale search missions. Future research could explore decentralized IPP frameworks that leverage distributed computing and communication-efficient planning strategies to coordinate multi-agent teams without excessive computational overhead. Leveraging VLMs or VLAs could also be an approach to mission and task allocation in multi-robot IPP applications.

11.2.2 Dynamic and Causal World Belief Models

Current world belief models typically assume static environments or rely on relatively simple representations of uncertainty, such as fixed Gaussian Process (GP) regressions over a static map. However, many real-world scenarios are inherently dynamic, with environmental states evolving over time due to external processes or human activity. A promising direction for future research is the development of causal and dynamic information models that can integrate sequential observations gathered across multiple deployments, capturing both the evolution of the environment and the causal relationships that drive these changes. Advancing beyond traditional GP-based methods, future models should incorporate spatiotemporal learning frameworks, structured priors informed by domain knowledge, and uncertainty propagation techniques that can maintain coherent beliefs over long timescales. Such models would provide richer, more accurate representations of the world, enabling planners to make decisions that are robust not only to current uncertainties but also to anticipated environmental changes.

By leveraging historical data and contextual cues from previous missions, these models could significantly enhance information gain prediction, allowing autonomous systems to anticipate where valuable information is likely to emerge in the future. Rather than assuming that information content is static, planners could reason about expected dynamics, adapting paths and sensing strategies in ways that proactively

account for evolving conditions. This capability would be particularly critical for long-duration deployments in complex and changing environments, such as environmental monitoring (e.g., tracking the spread of pollution or ecological changes), infrastructure inspection (e.g., monitoring wear or damage that progresses over time), and disaster response (e.g., adapting to shifting conditions during recovery operations). Developing these capabilities would enable autonomous systems to not only be reactive but strategically predictive, leading to more efficient, resilient, and impactful missions.

11.2.3 Human-On-The-Loop Information Gathering

While autonomous IPP systems are designed to operate independently, human-in-the-loop decision-making remains crucial in many real-world applications. Future research should explore how humans interact with planners and how robotic systems can infer user intent, adapt online, and provide interpretable decision-making.

One important research direction is to develop adaptive user interfaces and interaction models that allow non-expert users to intuitively guide autonomous data collection missions. This could include real-time replanning interfaces, user-defined priority regions, and mixed-initiative planning frameworks that enable seamless collaboration between humans and robots.

Furthermore, improving interpretability and trust in autonomous decision-making is critical for deployment in real-world, high-stakes applications. By incorporating explainable AI techniques, interactive feedback loops, and predictive models of user intent, planners can become more transparent and aligned with human operators' goals.

Appendices

Appendix A

IPP Target Tracking Adaptation

Our first analysis of IA-TIGRIS was based on an information reward function seeking to reduce entropy over a grid belief space. This works well for situations such as searching for an object or mapping a static belief. However, we wanted to expand upon this and be able to plan informative paths with dynamic and moving beliefs for such cases like target tracking.

We chose to represent the object state beliefs using particles filters. One reason for using particle filters is the state of the objects can be multi-modal. This is useful for cases such as if a UAV flew through the middle of a group of particles without having a positive observation. This would leave two distinct group of particles, with each having roughly the same probability of being correct.

Estimating the information gain over a flown path becomes much more challenging than in the case of grid cells. As which particles will be observed over a flow path depends on where both the particle and robot are at a given time, our edge value approximation would not work here. We instead discretize the path into viewpoints and update the beliefs at each observation. Fig. [A.1](#) shows a simulation view of the discretized footprints and the particle filters.

The objective function for tracking could be defined in many different ways depending on what behavior we would like to see. We could seek to minimize covariance of the particle filters at the end of the planning time, minimize the average covariance, maximize the number of observed particles, etc. In our case we chose to minimize the average covariance over the budget. Rather than computing the integral over this function, we approximate it using linear interpolation between two of our discretized points.

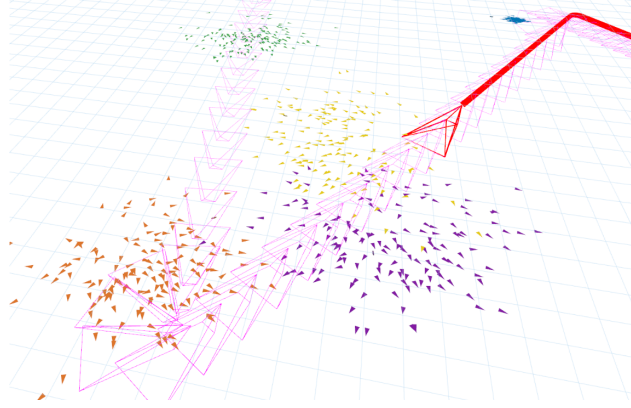


Figure A.1: Agent path and sensor footprint is shown in red and the expected footprint observations are shown in pink. The particle filters for each tracked object are shown with separate colors with each particle being shown as a triangle on the ground. The particles are propagated using a motion model. The object in the top right of the frame was recently viewed, giving a positive observation and collapsing the covariance of the state belief.

$$I(\xi) = \sum_{f=0}^F \frac{r_t(\xi_{f-1}) + r_t(\xi_f)}{2T} \Delta t_f$$

where Δt_f is time elapsed between footprints, and f is one of those footprints. This objective function has seemed to work well and give satisfactory paths.

In order to incorporate target tracking into the IA-TIGRIS planner, there are a couple other modifications that had to be made. The first is rather than performing an informed sample across a search space, we sample random particles from the particle filters. This ensures that every action is leading toward minimizing the uncertainty of target states rather than randomly propagating the tree out into empty space.

The next modification is when building the tree, there are a couple times where a distance metric is used to calculate how far away one node is from another. This happens when a new sample is made as well as when all near neighbors are extended to an added point. The mathematically correct solution would be to calculate the distance to an intersection point for every node on the tree, but this would be computationally expensive. Another option would be to augment the state space with adding time. What we decided to do was to rather propagate forward the target state using a heuristic and then calculating a typical nearest neighbors lookup using our existing efficient structure. This allowed us to have minimal changes to the stack. The heuristic we used was calculating the intersection time from the start state to the particle. This would be the shortest time to get to the particle. This means that the heuristic is fairly accurate for nodes that belong to fairly linear paths between the start and sampled node. The more complex the paths get, the less accurate the heuristic is.

The overall performance for using IA-TIGRIS in a tracking application is wanting. The largest bottleneck comes from inefficiencies in the code. More than half of the computation time is checking if particles are within a given footprint. Implementing this in a parallel architecture would greatly increase the performance. The tree size for tracking only is around 5% of the size as that for search with the same planning time. The second main performance hit comes from our heuristic. On long plans, the end portions sometimes do not intersect with the particle filters. The third is that tree recycling has not yet been implemented, so each replan has to be created from scratch.

Another challenge in general is balancing the two objectives of search and track. The objective functions do not have the same units or scales and can be challenging balance given a users underlying expectations of what they want to see.

Adding tracking to IA-TIGRIS gives us a baseline and helped identify areas of improvement for our future works. It has led us to exploring using MCTS with learned rollout evaluation and other ongoing works in multi-target tracking.

Appendix B

Dubins Set Classification Correction

In [Shkel and Lumelsky \[2001\]](#), a decision table is proposed to compute the shortest feasible path among the Dubins set $D = \{LSL, RSR, RSL, LSR, RLR, LRL\}$, given the quadrants of α and β . After implementing and validating each block within this table, 4 cases were found, $a_{lm} \in \{a_{12}, a_{21}, a_{34}, a_{43}\}$, where the previously proposed table did not give the actual shortest path option. Symmetrically, these 4 cases are equivalent. As such, the specific case of block a_{12} , which starts in quadrant 1 and finishes in quadrant 2, is addressed below.

According to the proposition 9 in [Shkel and Lumelsky \[2001\]](#), the shortest path for a_{12} is either RSR or RSL based on the results of the switching function S_{12} . However, after comprehensive testing, there are cases where the shortest path is LSR. This is particularly evident when the departing angle α is close to 0 and the arriving angle β is close to π . This was overlooked in the proof of proposition 9 based on the differential analysis in the neighborhood of the edge case.

To visualize the solution space, we sampled across α and β for a_{12} with $d = 4.01R_0$ and Fig. B.1 shows the results of the shortest path distribution. The shortest path for all the quadrant pairs in the blue region are LSR, contradicting the proposition 9 in [Shkel and Lumelsky \[2001\]](#).

This proves that the switching function presented in [Shkel and Lumelsky \[2001\]](#)

$$S_{12} = p_{rsr} - p_{rsl} - 2(q_{rsl} - \pi)$$

sometimes doesn't select the actual shorter path between RSR and RSL. The p_{rsr} , p_{rsl} and q_{rsl} lengths of different sections of the path and are defined in [Shkel and Lumelsky \[2001\]](#) and shown in Fig. B.2, with the subscript referring to the trajectory type. For an example case shown in Fig. B.2, the above switch function provided in [Shkel and Lumelsky \[2001\]](#) would select RSR as the shortest path, but with nearly a whole circle

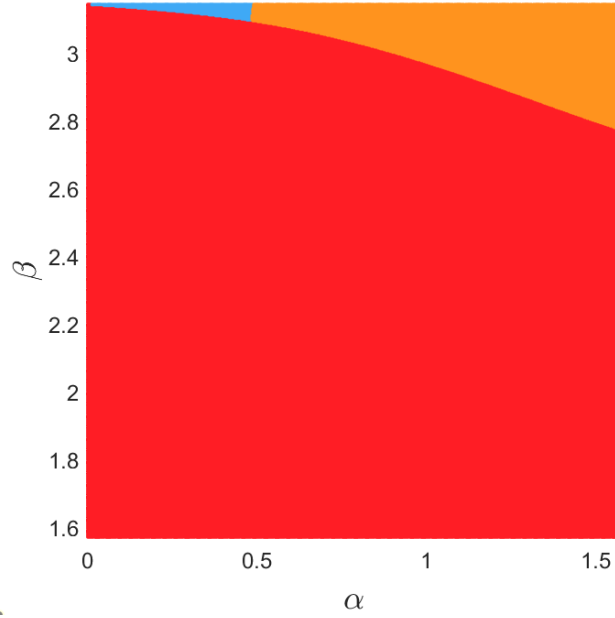


Figure B.1: The shortest path distribution for case a_{12} with $\alpha \in [0, \pi/2]$ and $\beta \in [\pi/2, \pi]$ with $d = 4.01R_0$. The red region marks the quadrant pairs for which the shortest path is RSL. The orange region marks the quadrant pairs for which the shortest path is RSR and the blue region marks the quadrant pairs for which the shortest path is LSR.

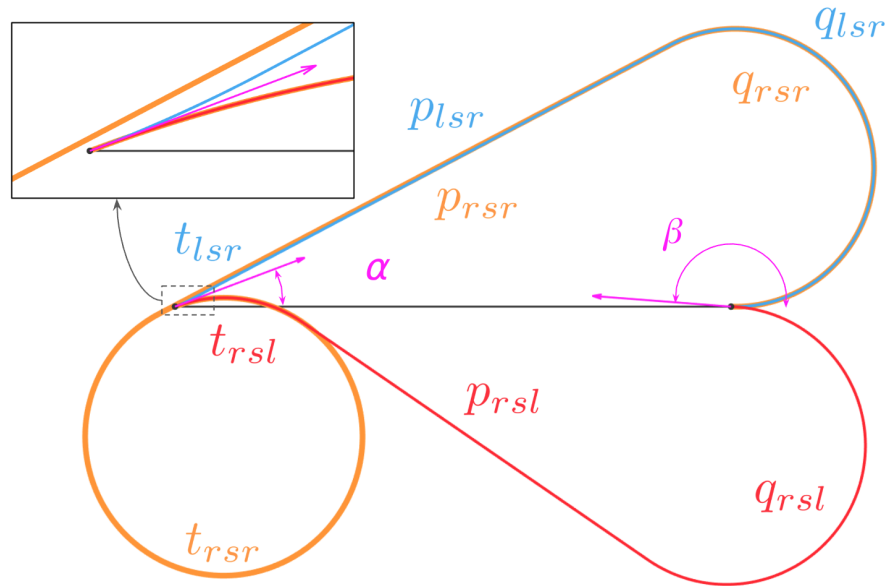


Figure B.2: Plot of RSL, LSR and RSR path for a point $\alpha = 0.36$ and $\beta = 3.111$ in the blue region of B.1. Though the length of LSR (blue) and RSL (red) are very close, the LSR path is the shortest among all options after calculation. Area near the starting point is zoomed in for better illustration of the path difference.

as the first section of its path, RSR is obviously longer than the RSL path.

For our methodology, the decision table is modified so that it can still provide the shortest path for the cases shown above. The modified decision table is provided in Table 6.1 with updated entries in $a_{lm} \in \{a_{12}, a_{21}, a_{34}, a_{43}\}$.

Bibliography

- [1] Hasini Viranga Abeywickrama, Beeshanga Abewardana Jayawickrama, Ying He, and Eryk Dutkiewicz. Empirical power consumption model for uavs. In *2018 IEEE 88th Vehicular Technology Conference (VTC-Fall)*, pages 1–5. IEEE, 2018.
- [2] Pramod Abichandani, Deepan Lobo, Gabriel Ford, Donald Bucci, and Moshe Kam. Wind measurement and simulation techniques in multi-rotor small unmanned aerial vehicles. *IEEE Access*, 8:54910–54927, 2020.
- [3] Florian Achermann, Nicholas RJ Lawrance, René Ranftl, Alexey Dosovitskiy, Jen Jen Chung, and Roland Siegwart. Learning to predict the wind for safe aerial vehicle planning. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 2311–2317. IEEE, 2019.
- [4] Moulay A. Akhloufi, Andy Couturier, and Nicolás A. Castro. Unmanned aerial vehicles for wildland fires: Sensing, perception, cooperation and assistance. *Drones*, 5(1):15, February 2021. ISSN 2504-446X. doi: 10.3390/drones5010015. URL <http://dx.doi.org/10.3390/drones5010015>.
- [5] Abdulla Al-Kaff, María Gómez-Silva, Francisco Moreno, Arturo de la Escalera, and José Armingol. An appearance-based tracking algorithm for aerial search and rescue purposes. *Sensors*, 19(3):652, February 2019. ISSN 1424-8220. doi: 10.3390/s19030652. URL <http://dx.doi.org/10.3390/s19030652>.
- [6] Soroush Alamdari, Elaheh Fata, and Stephen L Smith. Persistent monitoring in discrete environments: Minimizing the maximum weighted latency between observations. *The International Journal of Robotics Research*, 33(1):138–154, 2014. doi: 10.1177/0278364913504011. URL <https://doi.org/10.1177/0278364913504011>.
- [7] Sam Allison, he Bai, and Balaji Jayaraman. Modeling trajectory performance of quadrotors under wind disturbances, 01 2018.
- [8] Saeed Hamood Alsamhi, Alexey V. Shvetsov, Santosh Kumar, Svetlana V. Shvetsova, Mohammed A. Alhartomi, Ammar Hawbani, Navin Singh Rajput, Sumit Srivastava, Abdu Saif, and Vincent Omollo Nyangaresi. Uav computing-assisted search and rescue mission framework for disaster and harsh environment mitigation. *Drones*, 6(7):154, June 2022. ISSN 2504-446X. doi: 10.3390/drones6070154. URL <http://dx.doi.org/10.3390/drones6070154>.

- [9] Akash Arora, P. Michael Furlong, Robert Fitch, Salah Sukkarieh, and Terrence Fong. Multi-modal active perception for information gathering in science missions. *Autonomous Robots*, 10 2019. doi: 10.1007/s10514-019-09836-5.
- [10] Sankalp Arora and Sebastian Scherer. Randomized algorithm for informative path planning with budget constraints. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4997–5004, Singapore, Singapore, May 2017. IEEE. ISBN 978-1-5090-4633-1. doi: 10.1109/ICRA.2017.7989582. URL <http://ieeexplore.ieee.org/document/7989582/>.
- [11] Alper Aydemir, Patric Jensfelt, and John Folkesson. What can we learn from 38,000 rooms? reasoning about unexplored space in indoor environments. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4675–4682. IEEE, 2012.
- [12] Alper Aydemir, Patric Jensfelt, and John Folkesson. What can we learn from 38,000 rooms? Reasoning about unexplored space in indoor environments. In *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4675–4682, 2012. doi: 10.1109/IROS.2012.6386110.
- [13] PJ Baeck, N Lewyckij, B Beusen, W Horsten, and K Pauly. Drone based near real-time human detection with geographic localization. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 42:49–53, 2019.
- [14] Seungjae Baek, Brady Moon, Seungchan Kim, Muqing Cao, Cherie Ho, Sebastian Scherer, and Jeong hwan Jeon. Pipe planner: Pathwise information gain with map predictions for indoor robot exploration, 2025. URL <https://arxiv.org/abs/2503.07504>.
- [15] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.
- [16] Shi Bai, Jinkun Wang, Fanfei Chen, and Brendan Englot. Information-theoretic exploration with Bayesian optimization. In *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1816–1822, 2016.
- [17] Rafael Bailon-Ruiz, Arthur Bit-Monnot, and Simon Lacroix. Real-time wildfire monitoring with a fleet of uavs. *Robotics and Autonomous Systems*, 152:104071, 2022. ISSN 0921-8890. doi: <https://doi.org/10.1016/j.robot.2022.104071>. URL <https://www.sciencedirect.com/science/article/pii/S0921889022000355>.
- [18] Efstathios Bakolas and Panagiotis Tsiotras. Time-optimal synthesis for the zermelo-markov-dubins problem: The constant wind case. In *Proceedings of the 2010 American Control Conference*, pages 6163–6168, 2010. doi: 10.1109/ACC.2010.5531602.
- [19] Jayme Garcia Arnal Barbedo. A review on the use of unmanned aerial vehicles and imaging sensors for monitoring and assessing plant stresses. *Drones*, 3(2):40, 2019.
- [20] Ashvin Bashyam and Jacob W. Guggenheim. Uavs for wilderness search and rescue: Real-world considerations and technology roadmap for fixed wing uavs. *The Journal of Search and Rescue*, 2019. URL <https://api.semanticscholar.org/CorpusID:198948502>.

- [21] Michael Beard, Ba Tuong Vo, and Ba-Ngu Vo. Ospa (2): Using the ospa metric to evaluate multi-target tracking performance. In *2017 International Conference on Control, Automation and Information Sciences (ICCAIS)*, pages 86–91. IEEE, 2017.
- [22] Randal W. Beard and Timothy W. McLain. *Small Unmanned Aircraft: Theory and Practice*. Princeton University Press, 2012. ISBN 9780691149219. URL <http://www.jstor.org/stable/j.ctt7sbc4>.
- [23] Mesay Belete Bejiga, Abdallah Zeggada, Abdelhamid Nouffidj, and Farid Melgani. A convolutional neural network approach for assisting avalanche search and rescue operations with uav imagery. *Remote Sensing*, 9(2):100, 2017.
- [24] Asher Bender, Stefan B. Williams, and Oscar Pizarro. Autonomous exploration of large-scale benthic environments. In *2013 IEEE International Conference on Robotics and Automation*, pages 390–396, 2013. doi: 10.1109/ICRA.2013.6630605.
- [25] Graeme Best and Robert Fitch. Probabilistic maximum set cover with path constraints for informative path planning. In *Australasian Conference on Robotics and Automation*. ARAA, 12 2016.
- [26] Graeme Best, Rohit Garg, John Keller, Geoffrey A Hollinger, and Sebastian Scherer. Resilient multi-sensor exploration of multifarious environments with a team of aerial robots. In *Proceedings of Robotics: Science and Systems*, 2022.
- [27] Graeme Best, Rohit Garg, John Keller, Geoffrey A Hollinger, and Sebastian Scherer. Multi-robot, multi-sensor exploration of multifarious environments with full mission aerial autonomy. *The International Journal of Robotics Research*, 43(4):485–512, 2024.
- [28] Jonathan Binney and Gaurav S. Sukhatme. Branch and bound for informative path planning. In *2012 IEEE International Conference on Robotics and Automation*, pages 2147–2154, 2012. doi: 10.1109/ICRA.2012.6224902.
- [29] Jonathan Binney, Andreas Krause, and Gaurav S. Sukhatme. Informative path planning for an autonomous underwater vehicle. In *2010 IEEE International Conference on Robotics and Automation*, pages 4791–4796, 2010. doi: 10.1109/ROBOT.2010.5509714.
- [30] Jonathan Binney, Andreas Krause, and Gaurav S Sukhatme. Optimizing waypoints for monitoring spatiotemporal phenomena. *The International Journal of Robotics Research*, 32(8): 873–888, 2013.
- [31] Jonathan Binney, Andreas Krause, and Gaurav S. Sukhatme. Optimizing waypoints for monitoring spatiotemporal phenomena. *The International Journal of Robotics Research*, 32(8): 873–888, July 2013. ISSN 0278-3649, 1741-3176. doi: 10.1177/0278364913488427.
- [32] Andreas Bircher, Kostas Alexis, Michael Burri, Philipp Oettershagen, Sammy Omari, Thomas Mantel, and Roland Siegwart. Structural inspection path planning via iterative viewpoint resampling with application to aerial robotics. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6423–6430. IEEE, 2015.

- [33] Andreas Bircher, Mina Kamel, Kostas Alexis, Helen Oleynikova, and Roland Siegwart. Receding horizon" next-best-view" planner for 3d exploration. In *2016 IEEE international conference on robotics and automation (ICRA)*, pages 1462–1468. IEEE, 2016.
- [34] Andreas Bircher, Mina Kamel, Kostas Alexis, Helen Oleynikova, and Roland Siegwart. Receding horizon" next-best-view" planner for 3d exploration. In *2016 IEEE international conference on robotics and automation (ICRA)*, pages 1462–1468. IEEE, 2016.
- [35] Samuel S Blackman. Multiple hypothesis tracking for multiple target tracking. *IEEE Aerospace and Electronic Systems Magazine*, 19(1):5–18, 2004.
- [36] Lorenzo Bottarelli, Manuele Bicego, Jason Blum, and Alessandro Farinelli. Orienteering-based informative path planning for environmental monitoring. In *Engineering Applications of Artificial Intelligence*, volume 77, pages 46–58, 2019. doi: <https://doi.org/10.1016/j.engappai.2018.09.015>. URL <https://www.sciencedirect.com/science/article/pii/S095219761830201X>.
- [37] Mohamed Nadir Boukoberine, Zhibin Zhou, and Mohamed Benbouzid. A critical review on unmanned aerial vehicles power supply and energy management: Solutions, strategies, and prospects. *Applied Energy*, 255:113823, 2019.
- [38] Frederic Bourgault, Alexei A Makarenko, Stefan B Williams, Ben Grocholsky, and Hugh F Durrant-Whyte. Information based adaptive robotic exploration. In *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 1, pages 540–545, 2002.
- [39] Paolo Bruschi, Massimo Piotto, F Dell’Agnello, J Ware, and N Roy. Wind speed and direction detection by means of solid-state anemometers embedded on small quadcopters. *Procedia Engineering*, 168:802–805, 2016.
- [40] H Bruyninckx, E Scioni, H Nico, J Philips, F Reniers, D Monari, M Frigerio, SP Diez, SV Driessche, N Tsiogkas, et al. Building blocks for the design of complicated systems featuring situational awareness. *Composable components for compositional, adaptive, and explainable systems-of-systems*, 2021.
- [41] Tauã Cabreira, Lisane Brisolara, and Paulo R. Ferreira Jr. Survey on Coverage Path Planning with Unmanned Aerial Vehicles. *Drones*, 3(1), January 2019. ISSN 2504-446X. doi: 10.3390/drones3010004.
- [42] Chao Cao, Hongbiao Zhu, Howie Choset, and Ji Zhang. TARE: A hierarchical framework for efficiently exploring complex 3D environments. In *Proc. of Robotics: Science and Systems*, volume 5, 2021.
- [43] Yuhong Cao, Tianxiang Hou, Yizhuo Wang, Xian Yi, and Guillaume Sartoretti. Ariadne: A reinforcement learning approach using attention-based deep networks for exploration. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10219–10225. IEEE, 2023.
- [44] H. Jacky Chang, C. S. George Lee, Yung-Hsiang Lu, and Y. Charlie Hu. P-slam: Simultaneous localization and mapping with environmental-structure prediction. *IEEE Transactions on Robotics*, 23(2):281–293, 2007. doi: 10.1109/TRO.2007.892230.

- [45] Devendra Singh Chaplot, Dhiraj Prakashchand Gandhi, Abhinav Gupta, and Russ R Salakhutdinov. Object goal navigation using goal-oriented semantic exploration. *Advances in Neural Information Processing Systems*, 33:4247–4258, 2020.
- [46] Benjamin Charrow, Gregory Kahn, Sachin Patil, Sikang Liu, Ken Goldberg, Pieter Abbeel, Nathan Michael, and Vijay Kumar. Information-theoretic planning with trajectory optimization for dense 3d mapping. In *Robotics: Science and Systems*, volume 11, pages 3–12. Rome, 2015.
- [47] Chandra Chekuri and Martin Pal. A recursive greedy algorithm for walks in directed graphs. In *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '05, page 245–253, USA, 2005. IEEE Computer Society. ISBN 0769524680. doi: 10.1109/SFCS.2005.9. URL <https://doi.org/10.1109/SFCS.2005.9>.
- [48] Fanfei Chen, Shi Bai, Tixiao Shan, and Brendan Englot. Self-learning exploration and mapping for mobile robots via deep reinforcement learning. In *Aiaa scitech 2019 forum*, page 0396, 2019.
- [49] Fanfei Chen, John D Martin, Yewei Huang, Jinkun Wang, and Brendan Englot. Autonomous exploration under uncertainty via deep reinforcement learning on graphs. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6140–6147. IEEE, 2020.
- [50] Hong Cheng, Zhifeng Yang, and Christine Chan. An expert system for decision support of municipal water pollution control. *Engineering Applications of Artificial Intelligence*, 16: 159–166, 03 2003. doi: 10.1016/S0952-1976(03)00055-1.
- [51] Arnav Choudhry, Brady Moon, Jay Patrikar, Constantine Samaras, and Sebastian Scherer. Cvar-based flight energy risk assessment for multirotor uavs using a deep energy model. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 262–268, 2021. doi: 10.1109/ICRA48506.2021.9561658.
- [52] B. R. Christensen. Use of uav or remotely piloted aircraft and forward-looking infrared in forest, rural and wildland fire management: evaluation using simple economic analysis. *New Zealand Journal of Forestry Science*, 45(1), September 2015. ISSN 1179-5395. doi: 10.1186/s40490-015-0044-9. URL <http://dx.doi.org/10.1186/s40490-015-0044-9>.
- [53] Howard Coffin, Ian Abraham, Guillaume Sartoretti, Tyler Dillstrom, and Howie Choset. Multi-agent dynamic ergodic search with low-information sensors. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 11480–11486, 2022. doi: 10.1109/ICRA46639.2022.9812037.
- [54] Micah Corah and Nathan Michael. Volumetric objectives for multi-robot exploration of three-dimensional environments. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9043–9050. IEEE, 2021.
- [55] Bohdan Z Cybyk, Brian E McGrath, Timothy M Frey, David G Drewry, John F Keane, and Gopal Patnaik. Unsteady airflows and their impact on small unmanned air systems in urban environments. *Journal of Aerospace Information Systems*, 11(4):178–194, 2014.

- [56] Tuan Dam, Georgia Chalvatzaki, Jan Peters, and Joni Pajarinen. Monte-carlo robot path planning. *IEEE Robotics and Automation Letters*, 7(4):11213–11220, 2022. doi: 10.1109/LRA.2022.3199674.
- [57] Philip Dames, Pratap Tokekar, and Vijay Kumar. Detecting, localizing, and tracking an unknown number of moving targets using a team of mobile robots. *The International Journal of Robotics Research*, 36(13-14):1540–1553, 2017.
- [58] Jnaneshwar Das, Julio Harvey, Frederic Py, Harshvardhan Vathsangam, Rishi Graham, Kanna Rajan, and Gaurav S. Sukhatme. Hierarchical probabilistic regression for auv-based adaptive sampling of marine phenomena. In *2013 IEEE International Conference on Robotics and Automation*, page 5571–5578. IEEE, May 2013. doi: 10.1109/icra.2013.6631377. URL <http://dx.doi.org/10.1109/ICRA.2013.6631377>.
- [59] Ryan Day and John Salmon. A framework for multi-uav persistent search and retrieval with stochastic target appearance in a continuous space. *Journal of Intelligent & Robotic Systems*, 103(4), November 2021. ISSN 1573-0409. doi: 10.1007/s10846-021-01484-1. URL <http://dx.doi.org/10.1007/s10846-021-01484-1>.
- [60] Danilo Roberti Alves De Almeida, Eben North Broadbent, Matheus Pinheiro Ferreira, Paula Meli, Angelica Maria Almeyda Zambrano, Eric Bastos Gorgens, Angelica Faria Resende, Catherine Torres de Almeida, Cibele Hummel Do Amaral, Ana Paula Dalla Corte, et al. Monitoring restored tropical forest diversity and structure through uav-borne hyperspectral and lidar fusion. *Remote Sensing of Environment*, 264:112582, 2021.
- [61] Jeffrey Delmerico, Stefan Isler, Reza Sabzevari, and Davide Scaramuzza. A comparison of volumetric information gain metrics for active 3d object reconstruction. *Autonomous Robots*, 42(2):197–208, 2018.
- [62] Di Deng, Runlin Duan, Jiahong Liu, Kuangjie Sheng, and Kenji Shimada. Robotic exploration of unknown 2d environment using a frontier-based automatic-differentiable information gain measure. In *2020 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 1497–1503. IEEE, 2020.
- [63] DoD. Flying qualities of piloted aircraft. In *Department of Defense Handbook. MIL-HDBK-1797*. Washington, DC: U.S. Department of Defense, 1997.
- [64] L. E. Dubins. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of Mathematics*, 79(3):497–516, 1957. ISSN 00029327, 10806377. URL <http://www.jstor.org/stable/2372560>.
- [65] Matthew Dunbabin and Lino Marques. Robots for environmental monitoring: Significant advancements and applications. *IEEE Robotics & Automation Magazine*, 19(1):24–39, 2012. doi: 10.1109/MRA.2011.2181683.
- [66] Ludvig Ericson and Patric Jensfelt. Beyond the frontier: Predicting unseen walls from occupancy grids by learning from floor plans. *IEEE Robotics and Automation Letters*, 9(8): 6832–6839, 2024. doi: 10.1109/LRA.2024.3410164.

- [67] Thomas E Fortmann, Yaakov Bar-Shalom, and Molly Scheffe. Multi-target tracking using joint probabilistic data association. In *1980 19th IEEE Conference on Decision and Control including the Symposium on Adaptive Processes*, pages 807–812. IEEE, 1980.
- [68] Sergey Frolov, Bartolame Garau, and James Bellingham. Can we do better than the grid survey: Optimal synoptic surveys in presence of variable uncertainty and decorrelation scales. *Journal of Geophysical Research: Oceans*, 119(8):5071–5090, 2014. doi: <https://doi.org/10.1002/2013JC009521>. URL <https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1002/2013JC009521>.
- [69] Tomonari Furukawa, Lin Chi Mak, Hugh Durrant-Whyte, and Rajmohan Madhavan. Autonomous bayesian search and tracking, and its experimental validation. *Advanced Robotics*, 26(5–6):461–485, January 2012. ISSN 1568-5535. doi: 10.1163/156855311x617461. URL <http://dx.doi.org/10.1163/156855311x617461>.
- [70] David Galway, J Etele, and Giovanni Fusina. Modeling of urban wind field effects on unmanned rotorcraft flight. *Journal of aircraft*, 48(5):1613–1620, 2011.
- [71] Jonathan D. Gammell, Siddhartha S. Srinivasa, and Timothy D. Barfoot. Batch informed trees (bit*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs. *2015 IEEE International Conference on Robotics and Automation (ICRA)*, May 2015.
- [72] C García-Sánchez, DA Philips, and C Gorré. Quantifying inflow uncertainties for cfd simulations of the flow in downtown oklahoma city. *Building and environment*, 78:118–129, 2014.
- [73] Oscar Garibaldi Castillo and Edward Londner. Uav flight testing with an airborne sonic anemometer, imu and gps. In *28th AIAA Applied Aerodynamics Conference*, page 4572, 2010.
- [74] Georgios Georgakis, Bernadette Bucher, Anton Arapin, Karl Schmeckpeper, Nikolai Matni, and Kostas Daniilidis. Uncertainty-driven planner for exploration and navigation. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 11295–11302. IEEE, 2022.
- [75] Georgios Georgakis, Bernadette Bucher, Anton Arapin, Karl Schmeckpeper, Nikolai Matni, and Kostas Daniilidis. Uncertainty-driven planner for exploration and navigation. In *IEEE Conference on Robotics and Automation (ICRA)*, 2022.
- [76] Héctor H González-Banos and Jean-Claude Latombe. Navigation strategies for exploring indoor environments. *The International Journal of Robotics Research*, 21(10-11):829–848, 2002.
- [77] Google. Benchmark. <https://github.com/google/benchmark>, 2024.
- [78] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.
- [79] Astghik Hakobyan, Gyeong Chan Kim, and Insoon Yang. Risk-aware motion planning and control using cvar-constrained optimization. *IEEE Robotics and Automation Letters*, 4(4):3924–3931, 2019.

- [80] Narek Harutyunyan, Brady Moon, Seungchan Kim, Cherie Ho, Adam Hung, and Sebastian Scherer. Mapexrl: Human-inspired indoor exploration with predicted environment context and reinforcement learning, 2025. URL <https://arxiv.org/abs/2503.01548>.
- [81] Milos Hauskrecht. Value-function approximations for partially observable markov decision processes. *Journal of artificial intelligence research*, 13:33–94, 2000.
- [82] Gregory Hitz, Alkis Gotovos, Marie-Éve Garneau, Cédric Pradalier, Andreas Krause, Roland Y Siegwart, et al. Fully autonomous focused exploration for robotic environmental monitoring. In *Proc. of IEEE International Conference on Robotics and Automation (ICRA)*, pages 2658–2664, 2014.
- [83] Cherie Ho, Seungchan Kim, Brady Moon, Aditya Parandekar, Narek Harutyunyan, Chen Wang, Katia Sycara, Graeme Best, and Sebastian Scherer. Mapex: Indoor structure exploration with probabilistic information gain from global map predictions. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, 2025. URL <https://arxiv.org/abs/2409.15590>.
- [84] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
- [85] Geoffrey Hollinger. Long-horizon Robotic Search and Classification using Sampling-based Motion Planning. In *Robotics: Science and Systems XI*. Robotics: Science and Systems Foundation, July 2015. ISBN 978-0-9923747-1-6. doi: 10.15607/RSS.2015.XI.010. URL <http://www.roboticsproceedings.org/rss11/p10.pdf>.
- [86] Geoffrey A. Hollinger and Gaurav S. Sukhatme. Sampling-based robotic information gathering algorithms. *The International Journal of Robotics Research*, 33(9):1271–1287, August 2014. ISSN 0278-3649, 1741-3176. doi: 10.1177/0278364914533443. URL <http://journals.sagepub.com/doi/10.1177/0278364914533443>.
- [87] Heejin Jeong, Hamed Hassani, Manfred Morari, Daniel D. Lee, and George J. Pappas. Deep reinforcement learning for active target tracking. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1825–1831, 2021. doi: 10.1109/ICRA48506.2021.9561258.
- [88] Harold H. Johnson. An application of the maximum principle to the geometry of plane curves. *Proceedings of the American Mathematical Society*, 44(2):432–435, 1974. doi: 10.1090/s0002-9939-1974-0348631-6. URL <https://doi.org/10.1090/s0002-9939-1974-0348631-6>.
- [89] Austin Jones, Mac Schwager, and Calin Belta. A receding horizon algorithm for informative path planning with temporal logic constraints. In *2013 IEEE International Conference on Robotics and Automation*, pages 5019–5024, 2013. doi: 10.1109/ICRA.2013.6631294.
- [90] Rabiul Hasan Kabir and Kooktae Lee. Wildlife monitoring using a multi-uav system with optimal transport theory. *Applied Sciences*, 11(9):4070, April 2021. ISSN 2076-3417. doi: 10.3390/app11094070. URL <http://dx.doi.org/10.3390/app11094070>.
- [91] Siva Kailas. Multi-robot information gathering for spatiotemporal environment modelling. Master’s thesis, Carnegie Mellon University, Pittsburgh, PA, August 2023.

- [92] Yunus Karaca, Mustafa Cicek, Ozgur Tatli, Aynur Sahin, Sinan Pasli, Muhammed Fatih Beser, and Suleyman Turedi. The potential use of unmanned aircraft systems (drones) in mountain search and rescue operations. *The American Journal of Emergency Medicine*, 36(4):583–588, 2018. ISSN 0735-6757. doi: <https://doi.org/10.1016/j.ajem.2017.09.025>. URL <https://www.sciencedirect.com/science/article/pii/S0735675717307507>.
- [93] Sertac Karaman and Emilio Frazzoli. Incremental sampling-based algorithms for optimal motion planning. *CoRR*, abs/1005.0416, 2010. URL <http://arxiv.org/abs/1005.0416>.
- [94] Kapil Katyal, Katie Popek, Chris Paxton, Phil Burlina, and Gregory D. Hager. Uncertainty-aware occupancy map prediction using generative networks for robot navigation. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 5453–5459, 2019. doi: 10.1109/ICRA.2019.8793500.
- [95] Viktor Kaufmann, Andreas Kellerer-Pirklbauer, and Gernot Seier. Conventional and uav-based aerial surveys for long-term monitoring (1954–2020) of a highly active rock glacier in austria. *Frontiers in Remote Sensing*, 2:732744, 2021.
- [96] Kian Kerman, Alan Luntz, Venkatasubramanian Viswanathan, Yet-Ming Chiang, and Zhebo Chen. practical challenges hindering the development of solid state li ion batteries. *Journal of The Electrochemical Society*, 164(7):A1731, 2017.
- [97] Soonkyum Kim, Subhrajit Bhattacharya, Robert Ghrist, and Vijay Kumar. Topological exploration of unknown and partially known environments. In *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3851–3858, 2013.
- [98] Thomas Kirschstein. Comparison of energy demands of drone-based and ground-based parcel delivery services. *Transportation Research Part D: Transport and Environment*, 78: 102209, 2020.
- [99] Mangal Kothari, Ian Postlethwaite, and Da-Wei Gu. Uav path following in windy urban environments. *Journal of Intelligent & Robotic Systems*, 74(3-4):1013–1028, 2014.
- [100] Andreas Krause and Carlos Guestrin. Submodularity and its applications in optimized information gathering. *ACM TIST*, 2:32, 07 2011. doi: 10.1145/1989734.1989736.
- [101] Mihir Kulkarni, Brady Moon, Kostas Alexis, and Sebastian Scherer. *Aerial Field Robotics*, pages 1–15. Springer Berlin Heidelberg, Berlin, Heidelberg, 2020. ISBN 978-3-642-41610-1. doi: 10.1007/978-3-642-41610-1_221-1. URL https://doi.org/10.1007/978-3-642-41610-1_221-1.
- [102] Jack W Langelaan, John Spletzer, Corey Montella, and Joachim Grenestedt. Wind field estimation for autonomous dynamic soaring. In *2012 IEEE International conference on robotics and automation*, pages 16–22. IEEE, 2012.
- [103] Steven LaValle. Rapidly-exploring random trees: A new tool for path planning. *Research Report 9811*, 1998.

- [104] Nicholas Lawrance and Salah Sukkarieh. Simultaneous exploration and exploitation of a wind field for a small gliding uav. In *AIAA Guidance, Navigation, and Control Conference*, page 8032, 2010.
- [105] Woo-Cheol Lee, Ming Chong Lim, and Han-Lim Choi. Extendable navigation network based reinforcement learning for indoor robot exploration. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11508–11514. IEEE, 2021.
- [106] Haoran Li, Qichao Zhang, and Dongbin Zhao. Deep reinforcement learning-based automatic exploration for navigation in unknown environment. *IEEE transactions on neural networks and learning systems*, 31(6):2064–2076, 2019.
- [107] Hong Li. Practical evaluation of li-ion batteries. *Joule*, 3(4):911–914, 2019.
- [108] Jaeyoung Lim, Nicholas Lawrance, Florian Achermann, Thomas Stastny, Rik Bähneemann, and Roland Siegwart. Fisher information based active planning for aerial photogrammetry. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1249–1255, 2023. doi: 10.1109/ICRA48891.2023.10161136.
- [109] Jie Liu, Chaoqun Wang, Wenzheng Chi, Guodong Chen, and Lining Sun. Estimated path information gain-based robot exploration under perceptual uncertainty. *Robotica*, 40(8): 2748–2764, 2022.
- [110] Zhilong Liu, Raja Sengupta, and Alex Kurzhanskiy. A power consumption model for multi-rotor small unmanned aircraft systems. In *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 310–315. IEEE, 2017.
- [111] Matteo Luperto and Francesco Amigoni. Reconstruction and prediction of the layout of indoor environments from two-dimensional metric maps. *Engineering Applications of Artificial Intelligence*, 113:104910, 2022. ISSN 0952-1976. doi: <https://doi.org/10.1016/j.engappai.2022.104910>. URL <https://www.sciencedirect.com/science/article/pii/S0952197622001397>.
- [112] Matteo Luperto, Luca Fochetta, and Francesco Amigoni. Exploration of indoor environments through predicting the layout of partially observed rooms. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS '21*, page 836–843, Richland, SC, 2021. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 9781450383073.
- [113] Mingyang Lyu, Yibo Zhao, Chao Huang, and Hailong Huang. Unmanned aerial vehicles for search and rescue: A survey. *Remote Sensing*, 15(13):3266, June 2023. ISSN 2072-4292. doi: 10.3390/rs15133266. URL <http://dx.doi.org/10.3390/rs15133266>.
- [114] Kai-Chieh Ma, Lantao Liu, Hordur Heidarsson, and Gaurav Sukhatme. Data-driven learning and planning for environmental sampling. *Journal of Field Robotics*, 35, 02 2017. doi: 10.1002/rob.21767.
- [115] Mehdi Maboudi, MohammadReza Homaei, Soohwan Song, Shirin Malihi, Mohammad Saadatseresht, and Markus Gerke. A review on viewpoints and path planning for uav-based 3-d reconstruction. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 16:5026–5048, 2023.

- [116] Kotaro Maekawa, Shunsuke Negoro, Ittetsu Taniguchi, and Hiroyuki Tomiyama. Power measurement and modeling of quadcopters on horizontal flight. In *2017 Fifth International Symposium on Computing and Networking (CANDAR)*, pages 326–329. IEEE, 2017.
- [117] Ronald Mahler. *Statistical multisource-multitarget information fusion*. Artech, 2007.
- [118] Ronald PS Mahler. Multitarget bayes filtering via first-order multitarget moments. *IEEE Transactions on Aerospace and Electronic systems*, 39(4):1152–1178, 2003.
- [119] Anirudha Majumdar and Marco Pavone. How should a robot assess risk? towards an axiomatic theory of risk in robotics. In *Robotics Research*, pages 75–84. Springer, 2020.
- [120] Roman Marchant and Fabio Ramos. Bayesian optimisation for informative continuous path planning. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6136–6143, 2014. doi: 10.1109/ICRA.2014.6907763.
- [121] Matthew Marino, Alex Fisher, Reece Clothier, Simon Watkins, Samuel Prudden, and Chung Sing Leung. An evaluation of multi-rotor unmanned aircraft as flying wind sensors. *International Journal of Micro Air Vehicles*, 7(3):285–299, 2015.
- [122] Seth McCammon and Geoffrey A. Hollinger. Topological path planning for autonomous information gathering. *Autonomous Robots*, 45(6):821–842, September 2021. ISSN 1573-7527. doi: 10.1007/s10514-021-10012-x. URL <http://dx.doi.org/10.1007/s10514-021-10012-x>.
- [123] Timothy Mcgee, Stephen Spry, and J Hedrick. Optimal path planning in a constant wind with a bounded turning rate. *Collection of Technical Papers - AIAA Guidance, Navigation, and Control Conference*, 5, 08 2005. doi: 10.2514/6.2005-6186.
- [124] Timothy G. McGee and J. Karl Hedrick. Optimal path planning with a kinematic airplane model. *Journal of Guidance, Control, and Dynamics*, 30(2):629–633, 2007. doi: 10.2514/1.25042. URL <https://doi.org/10.2514/1.25042>.
- [125] Alexandra Meliou, Andreas Krause, Carlos Guestrin, and Joseph M. Hellerstein. Non-myopic informative path planning in spatio-temporal models. In *Proceedings of the 22nd National Conference on Artificial Intelligence - Volume 1, AAAI’07*, page 602–607. AAAI Press, 2007. ISBN 9781577353232.
- [126] Lauren M. Miller, Yonatan Silverman, Malcolm A. MacIver, and Todd D. Murphey. Ergodic exploration of distributed information. *IEEE Transactions on Robotics*, 32(1): 36–52, February 2016. ISSN 1941-0468. doi: 10.1109/tro.2015.2500441. URL <http://dx.doi.org/10.1109/TRO.2015.2500441>.
- [127] Syed Agha Hassnain Mohsan, Nawaf Qasem Hamood Othman, Yanlong Li, Mohammed H. Alsharif, and Muhammad Asghar Khan. Unmanned aerial vehicles (uavs): practical aspects, applications, open challenges, security issues, and future trends. *Intelligent Service Robotics*, January 2023. ISSN 1861-2784. doi: 10.1007/s11370-022-00452-4. URL <http://dx.doi.org/10.1007/s11370-022-00452-4>.

- [128] Brady Moon, Satrajit Chatterjee, and Sebastian Scherer. Tigris: An informed sampling-based algorithm for informative path planning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022. doi: 10.1109/IROS47612.2022.9981992. URL <https://arxiv.org/abs/2203.12830.pdf>.
- [129] Brady Moon, Sagar Sachdev, Junbin Yuan, and Sebastian Scherer. Time-optimal path planning in a constant wind for uncrewed aerial vehicles using dubins set classification. *IEEE Robotics and Automation Letters*, 2023. doi: 10.1109/LRA.2023.3333167. URL <https://arxiv.org/pdf/2306.11845.pdf>.
- [130] Brady Moon, Nayana Suvarna, Andrew Jong, Satrajit Chatterjee, Junbin Yuan, and Sebastian Scherer. Ia-tigris: An incremental and adaptive sampling-based planner for online informative path planning, 2025. URL <https://arxiv.org/abs/2502.15961>.
- [131] Brady G. Moon and Cameron K. Peterson. Learned search parameters for cooperating vehicles using gaussian process regressions. In *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 493–502, 2018. doi: 10.1109/ICUAS.2018.8453323.
- [132] Brady G. Moon and Cameron K. Peterson. Learned search parameters for cooperating vehicles using gaussian process regressions. In *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 493–502, 2018. doi: 10.1109/ICUAS.2018.8453323.
- [133] Jared J. Moore, Craig C. Bidstrup, Cameron K. Peterson, and Randal W. Beard. Tracking multiple vehicles constrained to a road network from a uav with sparse visual measurements. *Frontiers in Robotics and AI*, 8, October 2021. ISSN 2296-9144. doi: 10.3389/frobt.2021.744185. URL <http://dx.doi.org/10.3389/frobt.2021.744185>.
- [134] Chase C Murray and Amanda G Chu. The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery. *Transportation Research Part C: Emerging Technologies*, 54:86–109, 2015.
- [135] Hoa Van Nguyen, Ba-Ngu Vo, Ba-Tuong Vo, Hamid RezaTofighi, and Damith C. Ranasinghe. Multi-objective multi-agent planning for discovering and tracking multiple mobile objects, 2023.
- [136] Joseph L. Nguyen. *Long-term Informative Path Planning with Autonomous Soaring*. PhD thesis, The University of Sydney, 2015. URL <http://hdl.handle.net/2123/15364>.
- [137] Tomasz Niedzielski, Carsten Skjøth, Małgorzata Werner, Waldemar Spallek, Matylda Witek, Tymoteusz Sawiński, Anetta Drzeniecka-Osiadacz, Magdalena Korzystka-Muskała, Piotr Muskała, Piotr Modzel, et al. Are estimates of wind characteristics based on measurements with pitot tubes and gnss receivers mounted on consumer-grade unmanned aerial vehicles applicable in meteorological studies? *Environmental monitoring and assessment*, 189(9):431, 2017.
- [138] Farzad Niroui, Kaicheng Zhang, Zendai Kashino, and Goldie Nejat. Deep reinforcement learning robot for search and rescue applications: Exploration in unknown cluttered environments. *IEEE Robotics and Automation Letters*, 4(2):610–617, 2019.

- [139] Aaron Van Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1747–1756, New York, New York, USA, 20–22 Jun 2016. PMLR. URL <http://proceedings.mlr.press/v48/oord16.html>.
- [140] Giuseppe Oriolo, Giovanni Ulivi, and Marilena Vendittelli. Real-time map building and navigation for autonomous robots in unknown environments. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 28(3):316–333, 1998.
- [141] Matthew W Orr, Steven J Rasmussen, Etan D Karni, and William B Blake. Framework for developing and evaluating mav control algorithms in a realistic urban setting. In *Proceedings of the 2005, American Control Conference, 2005.*, pages 4096–4101. IEEE, 2005.
- [142] Mark Owen, Randal W. Beard, and Timothy W. McLain. *Implementing Dubins Airplane Paths on Fixed-Wing UAVs**, pages 1677–1701. Springer Netherlands, Dordrecht, 2015. ISBN 978-90-481-9707-1. doi: 10.1007/978-90-481-9707-1_120. URL https://doi.org/10.1007/978-90-481-9707-1_120.
- [143] S. Pang and J.A. Farrell. Chemical plume source localization. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 36(5):1068–1080, 2006. doi: 10.1109/TSMCB.2006.874689.
- [144] Savvas Papaioannou, Panayiotis Kolios, Theodoris Theodorides, Christos G. Panayiotou, and Marios M. Polycarpou. A cooperative multiagent probabilistic framework for search and track missions. *IEEE Transactions on Control of Network Systems*, 8(2):847–858, 2021. doi: 10.1109/TCNS.2020.3038843.
- [145] Jay Patrikar. Wind-field estimation and curvature continuous path planning for low altitude urban aerial mobility. Master’s thesis, Carnegie Mellon University, Pittsburgh, PA, August 2020.
- [146] Jay Patrikar, Brady Moon, and Sebastian Scherer. Wind and the city: Utilizing UAV-based in-situ measurements for estimating urban wind fields. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1254 – 1260, October 2020. URL https://www.ri.cmu.edu/app/uploads/2020/10/IROS2020__Camera_ready.pdf.
- [147] Liam Paull, Carl Thibault, Amr Nagaty, Mae Seto, and Howard Li. Sensor-driven area coverage for an autonomous fixed-wing unmanned aerial vehicle. *IEEE Transactions on Cybernetics*, 44(9):1605–1618, 2014. doi: 10.1109/TCYB.2013.2290975.
- [148] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
- [149] Christian Pfeifer, Andres Barbosa, Osama Mustafa, Hans-Ulrich Peter, Marie-Charlott Rümmler, and Alexander Brenning. Using fixed-wing uav for detecting and mapping the distribution and abundance of penguins on the south shetlands islands, antarctica. *Drones*, 3(2):39, April 2019. ISSN 2504-446X. doi: 10.3390/drones3020039. URL <http://dx.doi.org/10.3390/drones3020039>.

- [150] Marija Popović, Gregory Hitz, Juan Nieto, Inkyu Sa, Roland Siegwart, and Enric Galceran. Online informative path planning for active classification using uavs. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 5753–5758. IEEE, 2017.
- [151] Marija Popović, Teresa Vidal-Calleja, Gregory Hitz, Jen Jen Chung, Inkyu Sa, Roland Siegwart, and Juan Nieto. An informative path planning framework for UAV-based terrain monitoring. *Autonomous Robots*, 44(6):889–911, July 2020. ISSN 1573-7527. doi: 10.1007/s10514-020-09903-2. URL <https://doi.org/10.1007/s10514-020-09903-2>.
- [152] Alex S Prasetya, Rong-Jong Wai, Yi-Lun Wen, and Yu-Kai Wang. Mission-based energy consumption prediction of multirotor uav. *IEEE Access*, 7:33055–33063, 2019.
- [153] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021. URL <http://jmlr.org/papers/v22/20-1364.html>.
- [154] Santhosh K Ramakrishnan, Ziad Al-Halah, and Kristen Grauman. Occupancy anticipation for efficient exploration and navigation. In *Proc. of European Conference on Computer Vision (ECCV)*, pages 400–418, 2020.
- [155] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.
- [156] John H. Reif. Complexity of the mover’s problem and generalizations. In *20th Annual Symposium on Foundations of Computer Science (sfcs 1979)*, pages 421–427, 1979. doi: 10.1109/SFCS.1979.10.
- [157] Stephan Reuter, Ba-Tuong Vo, Ba-Ngu Vo, and Klaus Dietmayer. The labeled multi-bernoulli filter. *IEEE Transactions on Signal Processing*, 62(12):3246–3260, 2014.
- [158] A Ricci, I Kalkman, B Blocken, M Burlando, and MP Repetto. Impact of turbulence models and roughness height in 3d steady rans simulations of wind flow in an urban environment. *Building and Environment*, 171:106617, 2020.
- [159] R Tyrrell Rockafellar, Stanislav Uryasev, et al. Optimization of conditional value-at-risk. *Journal of risk*, 2:21–42, 2000.
- [160] Thiago A. Rodrigues, Jay Patrikar, Arnav Choudhry, Jacob Feldgoise, Vaibhav Arcot, Aradhana Gahlaut, Sophia Lau, Brady Moon, Bastian Wagner, H Scott Matthews, and et al. Data collected with package delivery quadcopter drone, Jul 2020. URL <https://doi.org/10.1184/R1/12683453.v3>.
- [161] Thiago A. Rodrigues, Jay Patrikar, Arnav Choudhry, Jacob Feldgoise, Vaibhav Arcot, Aradhana Gahlaut, Sophia Lau, Brady Moon, Bastian Wagner, H. Scott Matthews, Sebastian Scherer, and Constantine Samaras. In-flight positional and energy use data set of a dji matrice 100 quadcopter for small package delivery, 2021.
- [162] Leopoldo Rodriguez, Jose A Cobano, and Anibal Ollero. Wind field estimation and identification having shear wind and discrete gusts features with a small uas. In *2016*

- IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5638–5644. IEEE, 2016.
- [163] Ryo Sakagami, Florian S. Lay, Andreas Dömel, Martin J. Schuster, Alin Albu-Schäffer, and Freek Stulp. Robotic world models—conceptualization, review, and engineering best practices. *Frontiers in Robotics and AI*, 10, 2023. ISSN 2296-9144. doi: 10.3389/frobt.2023.1253049. URL <https://www.frontiersin.org/journals/robotics-and-ai/articles/10.3389/frobt.2023.1253049>.
 - [164] Carlos Sampedro, Alejandro Rodriguez-Ramos, Hriday Bavle, Adrian Carrio, Paloma de la Puente, and Pascual Campoy. A fully-autonomous aerial robot for search and rescue applications in indoor environments using learning-based techniques. *Journal of Intelligent & Robotic Systems*, 95:601–627, 2019.
 - [165] Manish Saroya, Graeme Best, and Geoffrey A. Hollinger. Online exploration of tunnel networks leveraging topological CNN-based world predictions. In *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6038–6045, 2020.
 - [166] Klaus Schilling and Christoph Jungius. Mobile robots for planetary exploration. *Control Engineering Practice*, 4(4):513–524, 1996.
 - [167] Lukas Schmid, Michael Pantic, Raghav Khanna, Lionel Ott, Roland Siegwart, and Juan Nieto. An Efficient Sampling-Based Method for Online Informative Path Planning in Unknown Environments. *IEEE Robotics and Automation Letters*, 5(2):1500–1507, April 2020. ISSN 2377-3766, 2377-3774. doi: 10.1109/LRA.2020.2969191. URL <https://ieeexplore.ieee.org/document/8968434/>.
 - [168] Kunal Shah, Grant Ballard, Annie Schmidt, and Mac Schwager. Multidrone aerial surveys of penguin colonies in antarctica. *Science Robotics*, 5(47):eabc3000, 2020. doi: 10.1126/scirobotics.abc3000. URL <https://www.science.org/doi/abs/10.1126/scirobotics.abc3000>.
 - [169] Alexander Shapiro, Darinka Dentcheva, and Andrzej Ruszczyński. *Lectures on stochastic programming: modeling and theory*. SIAM, 2014.
 - [170] Vishnu D Sharma, Maymoonah Toubeh, Lifeng Zhou, and Pratap Tokekar. Risk-aware planning and assignment for ground vehicles using uncertain perception from aerial vehicles. *arXiv preprint arXiv:2003.11675*, 2020.
 - [171] Vishnu Dutt Sharma, Anukriti Singh, and Pratap Tokekar. Pre-trained masked image model for mobile robot navigation. *arXiv preprint arXiv:2310.07021*, 2023.
 - [172] Andrei M. Shkel and Vladimir Lumelsky. Classification of the dubins set. *Robotics and Autonomous Systems*, 34(4):179–202, 2001. ISSN 0921-8890. doi: [https://doi.org/10.1016/S0921-8890\(00\)00127-5](https://doi.org/10.1016/S0921-8890(00)00127-5). URL <https://www.sciencedirect.com/science/article/pii/S0921889000001275>.
 - [173] Rakesh Shrestha, Fei-Peng Tian, Wei Feng, Ping Tan, and Richard Vaughan. Learned map prediction for enhanced mobile robot exploration. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 1197–1204. IEEE, 2019.

- [174] Rakesh Shrestha, Fei-Peng Tian, Wei Feng, Ping Tan, and Richard Vaughan. Learned map prediction for enhanced mobile robot exploration. In *Proc. of International Conference on Robotics and Automation (ICRA)*, pages 1197–1204, 2019.
- [175] A. Singh, A. Krause, C. Guestrin, and W. J. Kaiser. Efficient informative sensing using multiple robots. *Journal of Artificial Intelligence Research*, 34:707–755, Apr 2009. ISSN 1076-9757. doi: 10.1613/jair.2674.
- [176] Amarjeet Singh, Andreas Krause, Carlos Guestrin, and William Kaiser. Efficient informative sensing using multiple robots. *J. Artif. Intell. Res. (JAIR)*, 34:707–755, 01 2009. doi: 10.1613/jair.2674.
- [177] Amarjeet Singh, Andreas Krause, Carlos Guestrin, and William J. Kaiser. Efficient informative sensing using multiple robots. In *CoRR*, volume abs/1401.3462, 2014. URL <http://arxiv.org/abs/1401.3462>.
- [178] Jorge Sousa and Catherine Gorlé. Computational urban flow predictions with bayesian inference: Validation with field data. *Building and Environment*, 154:13–22, 2019.
- [179] Jorge Sousa, Clara García-Sánchez, and Catherine Gorlé. Improving urban flow predictions through data assimilation. *Building and Environment*, 132:282–290, 2018.
- [180] L.D. Stone. *Theory of Optimal Search*. Mathematics in science and engineering. Academic Press, 1975. ISBN 9780126724509. URL <https://books.google.com/books?id=Cvb-L0WHLJ8C>.
- [181] Andreas S Stordal, Hans A Karlsen, Geir Nævdal, Hans J Skaug, and Brice Vallès. Bridging the ensemble kalman filter and particle filters: the adaptive gaussian mixture filter. *Computational Geosciences*, 15(2):293–305, 2011.
- [182] Ioan A. Şucan, Mark Moll, and Lydia E. Kavraki. The Open Motion Planning Library. *IEEE Robotics & Automation Magazine*, 19(4):72–82, December 2012. doi: 10.1109/MRA.2012.2205651.
- [183] Wen Sun, Niteesh Sood, Debadeepta Dey, Gireeja Ranade, Siddharth Prakash, and Ashish Kapoor. No-regret replanning under uncertainty. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6420–6427, 2017. doi: 10.1109/ICRA.2017.7989758.
- [184] Yoonchang Sung and Pratap Tokekar. Gm-phd filter for searching and tracking an unknown number of targets with a mobile sensor with limited fov. *IEEE Transactions on Automation Science and Engineering*, 19(3):2122–2134, 2021.
- [185] Roman Suvorov, Elizaveta Logacheva, Anton Mashikhin, Anastasia Remizova, Arsenii Ashukha, Aleksei Silvestrov, Naejin Kong, Harshith Goka, Kiwoong Park, and Victor Lempitsky. Resolution-robust large mask inpainting with Fourier convolutions. In *Proc. of IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2149–2159, 2022.
- [186] Aaron Hao Tan and Goldie Nejat. Enhancing robot task completion through environment and task inference: A survey from the mobile robot perspective. *Journal of Intelligent & Robotic Systems*, 106(4):73, 2022.

- [187] Aaron Hao Tan, Siddarth Narasimhan, and Goldie Nejat. 4cnet: A diffusion approach to map prediction for decentralized multi-robot exploration. *arXiv preprint arXiv:2402.17904*, 2024.
- [188] Yuezhan Tao, Eran Iceland, Beiming Li, Elchanan Zwecher, Uri Heinemann, Avraham Cohen, Amir Avni, Oren Gal, Ariel Barel, and Vijay Kumar. Learning to explore indoor environments using autonomous micro aerial vehicles. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 15758–15764, 2024. doi: 10.1109/ICRA57147.2024.10610464.
- [189] Alare Technologies. What we do. <https://alaretech.com/capabilities.html>, 2022. URL <https://alaretech.com/capabilities.html>. [Accessed: 15 December 2024].
- [190] Laszlo Techy and Craig A. Woolsey. Minimum-time path planning for unmanned aerial vehicles in steady uniform winds. *Journal of Guidance, Control, and Dynamics*, 32(6): 1736–1746, 2009. doi: 10.2514/1.44580. URL <https://doi.org/10.2514/1.44580>.
- [191] Amila Thibbotuwawa, Peter Nielsen, Banaszak Zbigniew, and Grzegorz Bocewicz. Factors affecting energy consumption of unmanned aerial vehicles: an analysis of how energy consumption changes in relation to uav routing. In *International Conference on Information Systems Architecture and Technology*, pages 228–238. Springer, 2018.
- [192] Ryan L Thorpe, Matthew McCrink, and James W Gregory. Measurement of unsteady gusts in an urban wind field using a uav-based anemometer. In *2018 Applied Aerodynamics Conference*, page 4218, 2018.
- [193] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*, volume 1. MIT press Cambridge, 2000.
- [194] S M Towhidul Islam and Xiaolin Hu. Real-time on-board path planning for uas-based wildfire monitoring. In *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 527–535, 2021. doi: 10.1109/ICUAS51884.2021.9476725.
- [195] Joan Vallvé and Juan Andrade-Cetto. Dense entropy decrease estimation for mobile robot exploration. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6083–6089. IEEE, 2014.
- [196] Pieter Vansteenwegen, Wouter Souffriau, and Dirk Van Oudheusden. The orienteering problem: A survey. *European Journal of Operational Research*, 209(1):1–10, February 2011. ISSN 0377-2217. doi: 10.1016/j.ejor.2010.03.045. URL <http://dx.doi.org/10.1016/j.ejor.2010.03.045>.
- [197] J Irving Vasquez-Gomez, L Enrique Sucar, and Rafael Murrieta-Cid. View/state planning for three-dimensional object reconstruction under uncertainty. *Autonomous Robots*, 41: 89–109, 2017.
- [198] Arnoud Visser, Merlijn Van Ittersum, Luis A González Jaime, and Laurențiu A Stancu. Beyond frontier exploration. In *RoboCup 2007: Robot Soccer World Cup XI 11*, pages 113–123. Springer, 2008.

- [199] Ba-Ngu Vo, Ba-Tuong Vo, and Michael Beard. Multi-sensor multi-object tracking with the generalized labeled multi-bernoulli filter. *IEEE Transactions on Signal Processing*, 67(23): 5952–5967, 2019.
- [200] Ba-Tuong Vo and Ba-Ngu Vo. Labeled random finite sets and multi-object conjugate priors. *IEEE Transactions on Signal Processing*, 61(13):3460–3475, 2013. doi: 10.1109/TSP.2013.2259822.
- [201] Yiheng Wang, Weiyu Cai, and Yahong Rosa Zheng. Dubins curves for 3d multi-vehicle path planning using spline interpolation. In *OCEANS 2017 - Anchorage*, pages 1–5, 2017.
- [202] John Ware and Nicholas Roy. An analysis of wind field estimation and exploitation for quadrotor flight in the urban canopy layer. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1507–1514. IEEE, 2016.
- [203] Steven Waslander and Carlos Wang. Wind disturbance estimation and rejection for quadrotor position control, Apr 2009. URL <http://dx.doi.org/10.2514/6.2009-1983>.
- [204] H. G. Weller, G. Tabor, H. Jasak, and C. Fureby. A tensorial approach to computational continuum mechanics using object-oriented techniques, Nov 1998. URL <http://dx.doi.org/10.1063/1.168744>.
- [205] DJ Wise, VBL Boppana, KW Li, and HJ Poh. Effects of minor changes in the mean inlet wind direction on urban flow simulations. *Sustainable cities and society*, 37:492–500, 2018.
- [206] Brian Yamauchi. A frontier-based approach for autonomous exploration. In *Proc. of IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA)*, pages 146–151, 1997.
- [207] Brian Yamauchi, Alan Schultz, and William Adams. Mobile robot exploration and map-building with continuous localization. In *Proc. of IEEE International Conference on Robotics and Automation (ICRA)*, volume 4, pages 3715–3720, 1998.
- [208] Peng Yan, Tao Jia, and Chengchao Bai. Searching and tracking an unknown number of targets: A learning-based method enhanced with maps merging. *Sensors*, 21(4), 2021. ISSN 1424-8220. doi: 10.3390/s21041076. URL <https://www.mdpi.com/1424-8220/21/4/1076>.
- [209] Soo-Hyun Yoo, Andrew Stuntz, Yawei Zhang, Robert Rothschild, Geoffrey A. Hollinger, and Ryan N. Smith. *Experimental Analysis of Receding Horizon Planning Algorithms for Marine Monitoring*, page 31–44. Springer International Publishing, 2016. ISBN 9783319277028. doi: 10.1007/978-3-319-27702-8_3. URL http://dx.doi.org/10.1007/978-3-319-27702-8_3.
- [210] Bilal Yousuf, Zsófa Lendek, and Lucian Buşoniu. Exploration-based search for an unknown number of targets using a uav. *IFAC-PapersOnLine*, 55(15):93–98, 2022. ISSN 2405-8963. doi: <https://doi.org/10.1016/j.ifacol.2022.07.614>. URL <https://www.sciencedirect.com/science/article/pii/S2405896322010254>. 6th IFAC Conference on Intelligent Control and Automation Sciences ICONS 2022.

- [211] Jingjin Yu, Mac Schwager, and Daniela Rus. Correlated orienteering problem and its application to informative path planning for persistent monitoring tasks. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 342–349, 2014. doi: 10.1109/IROS.2014.6942582.
- [212] Juan Zhang, James F. Campbell, Donald C. Sweeney II, and Andrea C. Hupman. Energy consumption models for delivery drones: A comparison and assessment. *Transportation Research Part D: Transport and Environment*, 90:102668, 2021. ISSN 1361-9209. doi: <https://doi.org/10.1016/j.trd.2020.102668>. URL <https://www.sciencedirect.com/science/article/pii/S1361920920308531>.
- [213] Shiyong Zhang, Xuebo Zhang, Tianyi Li, Jing Yuan, and Yongchun Fang. Fast active aerial exploration for traversable path finding of ground robots in unknown environments. *IEEE Transactions on Instrumentation and Measurement*, 71:1–13, 2022.
- [214] Yichen Zhang, Xinyi Chen, Chen Feng, Boyu Zhou, and Shaojie Shen. Falcon: Fast autonomous aerial exploration using coverage path guidance. *IEEE Transactions on Robotics*, 2024.
- [215] Lifeng Zhou and Vijay Kumar. Robust multi-robot active target tracking against sensing and communication attacks. *IEEE Transactions on Robotics*, 39(3):1768–1780, 2023. doi: 10.1109/TRO.2022.3233341.
- [216] Delong Zhu, Tingguang Li, Danny Ho, Chaoqun Wang, and Max Q-H Meng. Deep reinforcement learning supervised autonomous exploration in office environments. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 7548–7555. IEEE, 2018.
- [217] Hai Zhu, Jen Jen Chung, Nicholas R. J. Lawrance, Roland Siegwart, and Javier Alonso-Mora. Online informative path planning for active information gathering of a 3d surface. *CoRR*, abs/2103.09556, 2021. URL <https://arxiv.org/abs/2103.09556>.
- [218] Elchanan Zwecher, Eran Iceland, Sean R Levy, Shmuel Y Hayoun, Oren Gal, and Ariel Barel. Integrating deep reinforcement and supervised learning to expedite indoor mapping. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 10542–10548. IEEE, 2022.
- [219] Elchanan Zwecher, Eran Iceland, Sean R. Levy, Shmuel Y. Hayoun, Oren Gal, and Ariel Barel. Integrating deep reinforcement and supervised learning to expedite indoor mapping. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 10542–10548, 2022. doi: 10.1109/ICRA46639.2022.9811861.