Autonomous Mission-Driven Robots in Extreme Environments

Thesis by

Amanda Bouman

In Partial Fulfillment of the Requirements for the
Degree of
Doctor of Philosophy

CALIFORNIA INSTITUTE OF TECHNOLOGY
Pasadena, California

2022
Defended April 11th, 2022

ACKNOWLEDGEMENTS

I want to extend my deepest gratitude to my advisor Prof. Joel Burdick, for his steadfast support, kindness, and dedication to the pursuit of *salt of the earth* robotics. I hope to carry this philosophy not only professionally, but through all my future endeavors.

To past and present members of my research group, particularly Nikola Georgiev, Matt Burkhardt, Daniel Pastor, Joe Bowkett, and Anushri Dixit: it was a rare pleasure working alongside a group of uniquely motivated and skilled individuals.

To Lynn Seymour and Sonya Lincoln: thank you for all your efforts behind the scenes. Field robotics is often a logistical nightmare, and you were always willing to work through the mess.

I want to thank Jacob Izraelevitz and Brett Kennedy for their effective leadership on *SQUID*, as well as Prof. Mory Gharib for his strong support of this work in CAST.

I am thankful to my SubT teammates who found a way to make even the bleakest settings a lively and productive work environment. Some of my fondest memories of graduate school will have happened in the mines, caves, tunnels, and random conference rooms we found ourselves in during those three crazy years. I want to especially thank Ali Agha for his unyielding determination in creating opportunities for the team, as well as my teammates: Thomas Touma, Nikhilesh Alatur, Sung Kim, Brett Lopez, David Fan, John Mayo, Kenny Chen, Matt Anderson, Kyon Otsu, Ben Morrell, Angel Santamaria-Navarro, Oriana Peltzer, Sammi Lei, Jeffrey Edlund, Torkom Pailevanian, Mike Wolf, Josh Ott, and Shehryar Khattak.

I want to particularly thank two people from this list for their technical expertise and guidance: Sung Kim and Brett Lopez. Whether academic or personal, our discussions over the years almost always had a way of getting to the core of an issue and shedding light on a productive path forward.

To my friends Marcel Veismann, Abbas Tutcuoglu, Akshita Ganesh, Danilo Kusanovic, Drew Singletary, Eric Ambrose, Gautam Venugopalan, Jorge Castillo, Magnus Hoffmann, Raj Katti, Rachel Gehlhar, Shushman Choudhury, Tori Lee, and Vin Narasimhan: thank you for your friendship and support through thick and thin.

And finally to Mom, Dad, Katie, Joe, and Alexander: without you, none of this work would have been possible.

ABSTRACT

Robotic autonomy systems that can negotiate harsh environments under time and communication constraints are critical to accomplishing many real-world missions. Such systems require an integrated software-hardware solution capable of robustly reasoning about a time-limited mission across a complex environment and negotiating extreme physical conditions during mission execution. To this end, I will discus the development of two field-tested systems designed for operation in GPS-denied areas: (*i*) a coverage planning framework that enables efficient exploration of large, unknown environments, and (*ii*) a ballistically-launched aircraft that converts to an autonomous, free-flying multirotor in order to provide rapid aerial surveillance.

The first system addresses the time-limited exploration problem by providing a planning strategy that seeks to maximize the area covered by a robot's sensor footprint along a planned trajectory. In order to find solutions over large spatial extents (>1 km) and long temporal horizons (>1 hour), this coverage problem is decomposed into tractable subproblems by introducing spatial and temporal abstractions. Spatially, the robot-world belief is approximated by a task-dependent structure, enriched with environment map estimates. Temporally, the belief is approximated by the aggregation of multiple structures, each spanning a different spatial range. Cascaded uncertainty-aware solvers return a coverage plan over the stratified belief in real time. Coverage policies are constructed in a receding horizon fashion to ensure motion smoothness and resiliency to real-world stochasticity in perception and control. This coverage planning framework was extensively tested on physical robots in various real-world environments (caves, mines, subway systems, etc.) and served as the exploration strategy for a competing entry in the DARPA Subterranean Challenge.

The second system addresses rapid multirotor deployment for aerial data collection during emergencies. While multirotors are advantageous over fixed-winged systems due to their high maneuverability, their rotating blades are hazardous and require stable, uncluttered takeoff sites. To overcome this issue, a ballistically-launched, autonomously-stabilizing multirotor (*SQUID – Streamlined Quick Unfolding Investigation Drone*) was designed, fabricated, and tested. *SQUID* follows a deterministic trajectory, transitioning from a folded launch configuration to an autonomous, fully-controllable hexacopter. The entire process from launch to position stabilization requires no user- or GPS-input and demonstrates the viability of using ballistically-launched multirotors to achieve safe and rapid deployment from moving vehicles.

PUBLISHED CONTENT AND CONTRIBUTIONS

[1] **A. Bouman**, J. Ott, S. Kim, K. Chen, M. Kochenderfer, B. Lopez, A. Agha-Mohammadi, and J. Burdick. "Adaptive Coverage Path Planning for Efficient Exploration of Unknown Environments". In: *Submitted to IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022.
Contribution: planner software development, implementation, and simulation/hardware testing.

[2] O. Peltzer∗, **A. Bouman**∗, S. Kim, R. Senanayake, J. Ott, H. Delecki, M. Sobu, M. Schwager, M. Kochenderfer, J. Burdick, and A. Agha-Mohammadi. "FIG-OP: Exploring Large-Scale Unknown Environments on a Fixed Time Budget". In: *Submitted to IEEE Robotics and Automation Letters (RA-L)*. IEEE, 2022.
Contribution: planner software development, implementation, and simulation/hardware testing.

[3] A. Agha-Mohammadi, K. Otsu, B. Morrell, D. Fan, R. Thakker, A. Santamaria-Navarro, S. Kim, **A. Bouman**, and [63 others]. "NeBula: Quest for Robotic Autonomy in Challenging Environments; TEAM CoSTAR at the DARPA Subterranean Challenge". In: *Journal of Field Robotics (JFR)* (2021).
Contribution: global planner software development, implementation, and simulation/hardware testing.

[4] S. Kim∗, **A. Bouman**∗, G. Salhotra, D. Fan, K. Otsu, J. Burdick, and A. Agha-Mohammadi. "PLGRIM: Hierarchical Value Learning for Large-scale Exploration in Unknown Environments". In: *International Conference on Automated Planning and Scheduling (ICAPS)*. Vol. 31. 2021, pp. 652–662.
Contribution: planner software development, implementation, and simulation/hardware testing.

[5] **A. Bouman**, P. Nadan, M. Anderson, D. Pastor, J. Izraelevitz, J. Burdick, and B. Kennedy. "Design and Autonomous Stabilization of a Ballistically-Launched Multirotor". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 8511–8517. DOI: `10.1109/ICRA40945.2020.9197542`.
Contribution: vehicle, software, and experimental setup design.

[6] **A. Bouman**∗, M. Ginting∗, N. Alatur∗, M. Palieri, D. Fan, T. Touma, T. Pailevanian, S. Kim, K. Otsu, J. Burdick, and A. Agha-Mohammadi. "Autonomous Spot: Long-Range Autonomous Exploration of Extreme Environments with Legged Locomotion". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020. DOI: `10.1109/IROS45743.2020.9341361`.
Contribution: coverage planner software development, implementation, and simulation/hardware testing.

[7] D. Pastor, J. Izraelevitz, P. Nadan, **A. Bouman**, J. Burdick, and B. Kennedy. "Design of a Ballistically-Launched Foldable Multirotor". In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2019. DOI: `10.1109/IROS40897.2019.8968549`.
Contribution: design of vehicle-based field testing setup.

*These authors contributed equally to the listed work.

TABLE OF CONTENTS

Chapter 1

INTRODUCTION

Robotic autonomy systems that can operate in harsh environments under time and communication constraints are critical to accomplishing many real-world missions, such as search-and-rescue after a natural disaster, repair of leaking oil wells, or exploration of planetary bodies. Such systems require an integrated hardware-software solution capable of: (*i*) robustly reasoning about a time-limited mission across a complex environment, and (*ii*) negotiating extreme physical conditions during mission execution.

Consider the Tham Luang cave rescue in 2018. Thirteen members of a soccer team became trapped deep inside a Thai cave after rainfall flooded portions of the cave. What ensued was a massive, time-sensitive effort to rescue the team before the start of Monsoon season (Fig. 1.1). Piloted drones were deployed above the jungle-covered mountainside in order to locate vents in the cave roof for water drainage and personnel access. Meanwhile, expert divers searched the complex cave network while contending with rushing water and near darkness [1]. During one of the dives, a Thai Navy SEAL died of asphyxiation as he tried navigating one of the many twisting cave chambers. After two weeks and the involvement of nearly 10,000 people, all thirteen team members were rescued alive – an outcome many consider to be a miracle.

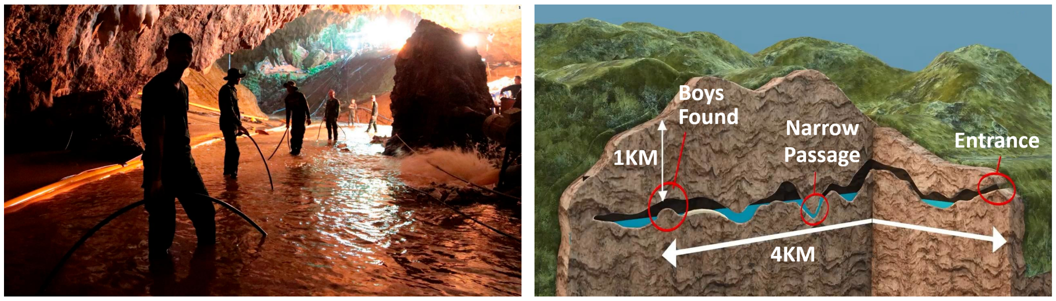


Figure 1.1: Tham Luang cave rescue: members of the rescue team navigating a flooded cave chamber [2], and a schematic of the mission site. The survivors were located 2.5 miles (4 km) from the cave entrance [3].

No commercial technology exists today that could have autonomously executed the Tham Luang rescue mission. Despite robotic autonomy gaining widespread usage in manufacturing, severe challenges still remain for time-limited operations in unknown and extreme environments. These technical challenges arise when a system must navigate mobility-stressing elements (uneven terrain, obstacles, stairs, slopes, narrow passages) while simultaneously coping with perceptually-degraded conditions (darkness, obscurants, topological self-similarity), without the aid of a prior map or GPS localization.

The Tham Luang mission calls for an integrated team of air and ground robots that can rapidly deploy with little to no reliance on skilled human operators. These systems must simultaneously be nimble enough to negotiate complex topologies and obstructions, yet large enough to carry a payload capable of providing sensing, communication, and computing capabilities over a long mission duration. As the robots navigate the surface and subsurface environments, they must: *(i)* maintain and synchronize an internal representation of the world that encodes traversability and mission-specific information, and *(ii)* plan risk-mitigating paths that maximize the area observed, or *covered*, by a task-specific sensor within a mission-dictated time budget. This sensor may be a thermal camera for detecting cave vents, an optical camera for identifying visual clues of survivors, or an omnidirectional range finder for constructing 3D cave maps. As a robot moves, its sensor footprint sweeps the environment, expanding the covered area, or more generally, the task-relevant information about the world.

To provide initial reconnaissance, highly-maneuverable aircraft must quickly launch through a potentially dense forest canopy to vantage points along the jagged mountain range. Then, out of sight of ground operations, the deployed aircraft must maintain localization without relying on intermittent GPS signals, which are likely to cut-out during flight through canyons and other sky-obstructing terrain. Meanwhile, the cave-bound ground robots must repeatedly face mission-critical decisions while equipped with only a partial understanding of their situation – both in terms of the world and their place in it. As a robot pushes through the cave's dark twisting chambers, consumed by water, dust, and fog, it must carefully evaluate each action, weighing expectations about the risk of mission-failure against the value of gathering new information in service of accomplishing mission objectives.

Motivated by real-world missions like the Tham Luang rescue, this thesis develops two field-tested systems designed for operation in GPS-denied areas: (i) a coverage planning framework that enables efficient exploration of large, unknown environments, and (ii) a ballistically-launched aircraft that converts to an autonomous, free-flying multirotor for rapid aerial surveillance.

1.1 Coverage Planning in Unknown Environments

The coverage planning objective is to find the optimal sequence of sensing actions that maximizes some task-specific information about the environment. In the robotics field, this problem is commonly motivated by tasks such as surveillance, object inspection, and exploration. While a variety of coverage planning algorithms have been proposed, this thesis is interested in those used to solve the exploration problem where policies are constructed in a receding horizon fashion as the robot gathers sensory information about its environment. The challenges associated with the exploration-driven coverage problem are broadly identified as follows. Note that these challenges are discussed in greater detail in Chaps. 3-5.

Long-Horizon Planning under Uncertainty: In exploration-drive coverage problems, the robot's knowledge about its environment is incomplete and often inaccurate at runtime. The robot must accumulate data to incrementally build a model of its environment. To do so efficiently, it needs to reason about the long-term effects of its actions on the quality and quantity of data it collects. All the while, the robot must account for uncertainty in hazard assessment, localization, and motion execution in order to make decisions for maximal information gain in a real-world stochastic setting.

World Representation: The computational complexity of the coverage planning solution is highly dependent on how the robot maintains and updates a representation of the world, encoding coverage and traversability information. One of the primary challenges associated with finding online coverage trajectories arises from the fact that the representation must simultaneously: (*i*) span up to several kilometers to facilitate global coverage mission objectives, and (*ii*) capture high-fidelity information about traversability to enable safe navigation through hazardous terrain. The problem complexity is further exacerbated by the need to reason about a non-additive coverage reward across a trajectory. Therefore, special attention must be paid to the manner by which the world representation is updated with coverage information when simulating a trajectory; i.e., naively replicating the properties of a real-world coverage sensor can have diminishing returns in an uncertain world.

Resiliency to Unexpected Risks: The robot must replan frequently so that it can account for new traversability risk information in a coverage trajectory. As the robot explores its environment, it receives new information, updates its understanding of the world, and constructs a new coverage plan in a receding horizon fashion. This approach introduces yet another complication – how to transition from one policy to the next. Policies generated during consecutive planning episodes should: (*i*) respect the dynamic constraints of the robot to achieve maximal coverage efficiency, and (*ii*) adapt to unexpected hazards in the environment to ensure the robot's safety. It is imperative to find a balance between these two distinct, and often opposed, objectives during real-world execution.

1.2 Rapid Aerial Surveillance

Rapid deployment of an aircraft is critical to gathering aerial data during emergency response. In these situations, a multirotor aircraft is advantageous over a fixed-wing system, since it can both hover in place and aggressively maneuver in cluttered environments. However, the rotating blades of a multirotor are sensitive to disturbances and can be hazardous to nearby assets and personnel. Thus, a precise, highly-deterministic, and fully autonomous deployment pipeline is necessary in order to rapidly achieve a safe operating altitude. This autonomous deployment pipeline should comprise two major phases: post-ballistic launch *passive* stabilization, and rotor-engaged *active* stabilization. The requirements of each phase are thoroughly discussed in Chap. 6, and only briefly mentioned here.

Passive Stabilization: A ballistic launch is necessary to achieve a deterministic operating altitude, particularly in precarious takeoff conditions, such as deployment through cluttered areas or from a moving vehicle. The aircraft must thus boast: (*i*) an airframe strong enough to carry and transmit launch loads without damaging an onboard autonomy package, and (*ii*) an aerodynamic design that ensures attitude stability as the aircraft travels along the ballistic trajectory. The ultimate objective of the passive phase is to set the necessary preconditions for autonomous flight.

Active Stabilization: After the aircraft has reached a desired altitude, the rotors must spool up and transition the platform from ballistic motion to full 6-degree of freedom control using only onboard sensing, without the aid of GPS. After successful completion of this phase, the vehicle is ready to initiate a coverage path planning mission.

1.3 Thesis Structure

Chapter 2 introduces the underlying autonomy components required for the development of the coverage planning strategy proposed in Chaps. 3-5. Next, the DARPA Subterranean Challenge is described, as it served as the concrete mission for which much of this thesis work was based upon. This chapter also gives a brief tour of classical coverage path planning approaches, as well as a POMDP overview.

Chapter 3 develops a hierarchical framework for exploration of large-scale unknown environments, called *PLGRIM* (*Probabilistic Local and Global Reasoning on Information roadMaps*). A hierarchical belief space representation is proposed, which effectively encodes a large-scale world state, while simultaneously capturing high-fidelity information local to the robot. Global policies (developed in Chap. 5) guide local policies (developed in Chap. 4), resulting in a cascaded decision process. This chapter also introduces a policy reconciliation method, which respects the robot's dynamic constraints while ensuring resiliency to unexpected risks.

Chapter 4 presents an online method for finding local coverage polices over a grid-based representation of the environment. Here, the robot is tasked with planning a path over a local horizon such that the accumulated area swept out by its sensor footprint is maximized. To quickly find near-optimal solutions, this chapter proposes an effective approximation to the coverage sensor model which adapts to the local environment. As a result, the robot is able to reason about a submodular, non-additive coverage reward function across a trajectory, while simultaneously accounting for traversal distance and risk.

Chapter 5 presents an online method for finding global coverage polices over a graph-based representation of the environment. This chapter proposes and solves the *Frontloaded Information Gain Orienteering Problem* (*FIG-OP*) – a generalization of the traditional orienteering problem where the assumption of a reliable environmental model no longer holds. This OP variant compensates for model uncertainty by incorporating a greedy incentive that shifts information gain earlier in time. By biasing towards short term gain, the robot visits frontiers earlier with the expectation that they will significantly alter its world understanding by exposing new opportunities for information gathering.

Chapter 6 presents a ballistically-launched, autonomously-stabilizing multirotor prototype called *SQUID* (*Streamlined Quick Unfolding Investigation Drone*). *SQUID* follows a deterministic trajectory, transitioning from a folded launch configuration to an autonomous, fully-controllable hexacopter. The entire process from launch to position stabilization requires no user- or GPS-input and demonstrates the viability of using ballistically-launched multirotors to achieve safe and rapid deployment from moving vehicles.

Chapter 7 lists the specific contributions of this thesis, and discusses future directions based off real-world performance of the proposed coverage planning strategy.

Chapter 2

BACKGROUND AND PRELIMINARIES

This chapter begins by providing a description of the DARPA Subterranean Challenge (Sec. 2.1) and the autonomy components underlying the proposed coverage planning structures and algorithms (Sec. 2.2). Then, a brief review of the coverage path planning literature is provided (Sec. 2.3). Finally, the partially observable Markov decisioin process (POMDP) – a framework for sequential decision-making when the state is not fully observable – is discussed (Sec. 2.4).

This chapter was adapted from:

> A. Agha, K. Otsu, B. Morrell, D. Fan, R. Thakker, A. Santamaria-Navarro, S. Kim, **A. Bouman**, and [63 others]. “NeBula: Quest for Robotic Autonomy in Challenging Environments; Team CoSTAR at the DARPA Subterranean Challenge”. In: *Journal of Field Robotics (JFR)* (2021).
>
> **A. Bouman***, M. Ginting*, N. Alatur*, M. Palieri, D. Fan, T. Touma, T. Pailevanian, S. Kim, K. Otsu, J. Burdick, and A. Agha-Mohammadi. “Autonomous Spot: Long-Range Autonomous Exploration of Extreme Environments with Legged Locomotion”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020.

2.1 DARPA Subterranean Challenge

The DARPA Subterranean, or “SubT,” Challenge is a robotic competition that seeks novel approaches to rapidly navigate, map, and search underground environments. The teams participating in the systems track develop and implement physical systems for autonomous traversal of various extreme subterranean environments, including mines, industrial complexes, and natural caves. The competition spans a period of three years and includes four major events: Tunnel, Urban, Cave, and Final. During a “run” in a course, a competitor team has a fixed time window (1 hour) to deploy their robots in order to locate artifacts in the course. See Fig. 2.1. Each team receives 1 point per artifact if their robots: (*i*) reach, detect, and recognize the artifact, (*ii*) localize the artifact in global coordinates with less than 5 meters error, and (*iii*) report the artifact during the 1-hour mission period. The team with the most points wins the run.

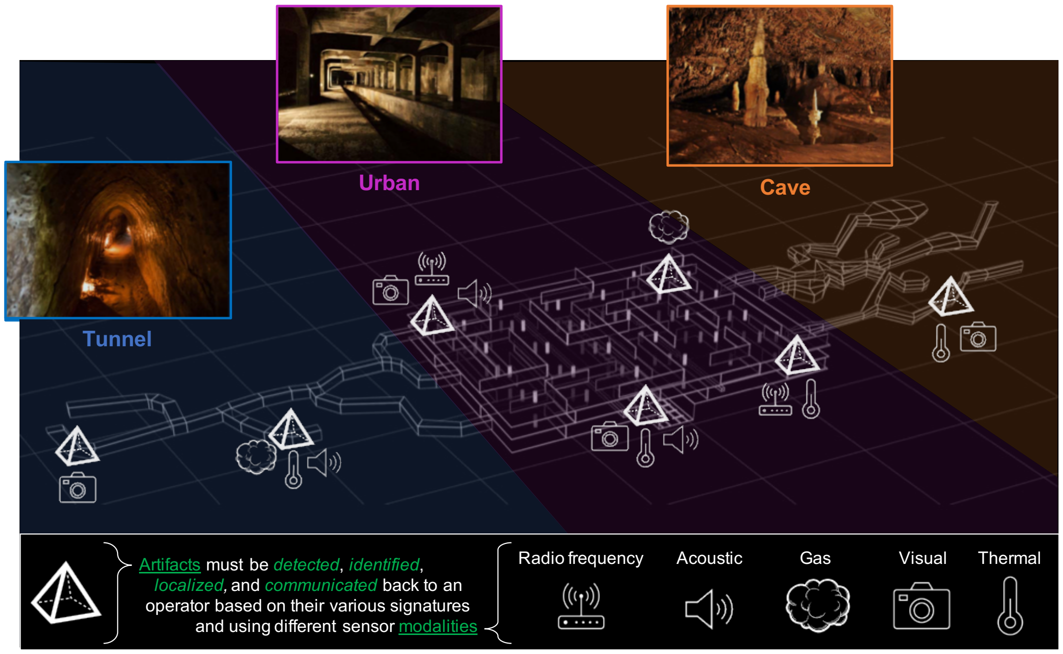
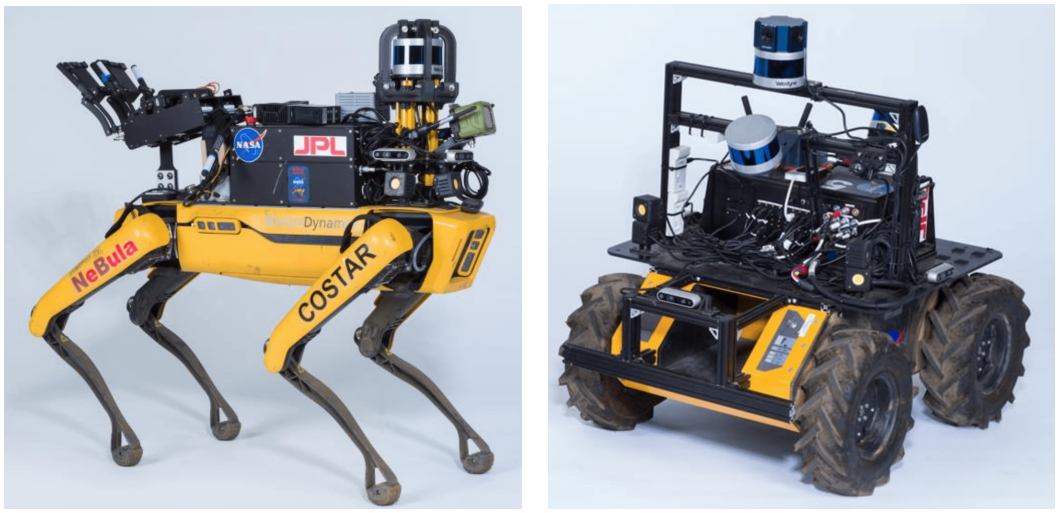

Urban
Cave
Tunnel
Artifacts must be detected, identified, localized, and communicated back to an operator based on their various signatures and using different sensor modalities
Radio frequency
Acoustic
Gas
Visual
Thermal

Figure 2.1: The Final Circuit of the DARPA SubT Challenge consists of elements from all subdomains (i.e., tunnel, urban, cave). To demonstrate efficient exploration of the environment, competitors score points by accurately reporting the type and location of artifacts distributed along the course. [4]

2.2 Underlying Autonomy Modules

This section presents the software architecture developed by the Team CoSTAR (Collaborative SubTerranean Autonomous Robots), while competing in the DARPA SubT Challenge. The autonomy framework is called NeBula: Networked Belief-aware Perceptual Autonomy [4]. NeBula is fully integrated on two robot platforms: a four-wheeled vehicle (Clearpath Husky robot) and a quadraped (Boston Dynamics Spot robot), as shown in Fig. 2.2. Throughout this text, the NeBula-integrated Spot robot is referred to as *Au-Spot*. Au-Spot demonstrated one of the first efforts to enable large-scale and long-duration autonomy using the Boston Dynamics Spot robot [5].

NeBula provides the underlying autonomy modules required for the coverage planning structures and algorithms discussed in Chaps. 3-5. This section will focus on the NeBula modules: state estimation, traversability analysis, and motion planning.



(a) NeBula-integrated Spot (or *Au-Spot*)   (b) NeBula-integrated Husky

Figure 2.2: Both robots are equipped with custom sensing and computing systems, enabling high levels of autonomy and communication capabilities. The sensing system consists of 3D LiDARs, Intel RealSense cameras, high-intensity LEDs, an IMU, and a thermal camera. The entire autonomy stack runs in real-time on an Intel Core i7 processor with 32 GB of RAM.

State Estimation

The most fundamental component of the NeBula architecture is reliable state estimation under perceptually-degraded conditions. This includes environments with large variations in lighting, obscurants (e.g., dust, fog, and smoke), self-similar scenes, reflective surfaces, and featureless/feature-poor surfaces.

Multi-Sensor Fusion: To overcome these challenges, NeBula relies on a LiDAR-centric uncertainty-aware, multi-sensor fusion framework where a *selected* odometry source is fused as a prior with LiDAR information to enable accurate ego-motion estimation under challenging perceptual conditions. The main components of the proposed approach are: *(i)* an anomaly-aware odometry multiplexer, *(ii)* a multi-sensor LiDAR-centric SLAM front-end, and *(iii)* a SLAM back-end [6]. Fig. 2.3 provides a high-level overview of the approach.

Odometry Multiplexer: To select the best odometry prior to be fused with LiDAR information, multiple heterogeneous odometry sources (e.g., visual-inertial, thermal-inertial, kinematic-inertial, etc.) are fed into an anomaly-aware odometry multiplexer, referred to as HeRO (Heterogeneous Redundant Odometry) [7]. At every time step, HeRO runs a confidence test on each odometry stream (prior) to detect potential anomalies (e.g., gaps, jumps, divergences) and identifies the most reliable input odometry $\mathbf{Y} \in SE(3)$ to be used as a prior in the LiDAR-based front-end.

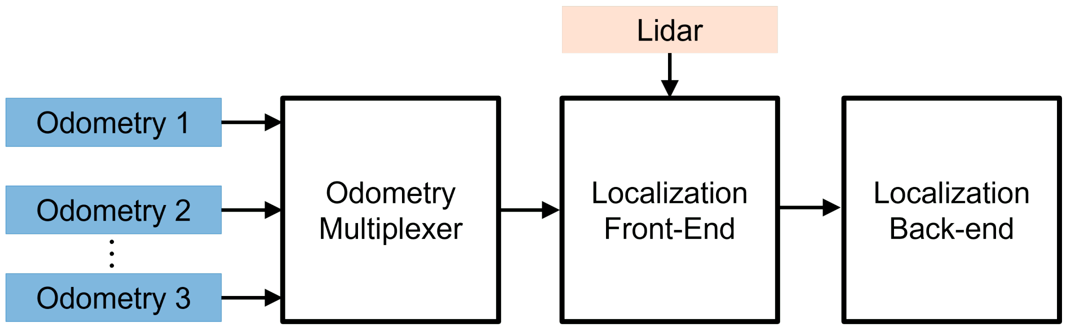


Figure 2.3: State Estimation Architecture Overview [5]

.

Localization Front-End: The output of the odometry multiplexer is sent to a multi-sensor LiDAR-centric SLAM front-end module, referred to as LOCUS (Lidar Odometry for Consistent operations in Uncertain Settings) [8]. LOCUS performs a cascaded GICP-based scan-to-scan and scan-to-submap matching operation to estimate the relative motion of the robot between consecutive LiDAR acquisitions.

In the scan-to-scan matching stage, Generalized Iterative Closest Point (GICP) computes the optimal transformation $\hat{\mathbf{T}}_k^{k-1}$ that minimizes the residual error $\mathcal{E}$ between corresponding points in L_{k-1} and L_k:

$$\hat{\mathbf{T}}_k^{k-1} = \arg\min_{\mathbf{T}_k^{k-1}} \mathcal{E}(\mathbf{T}_k^{k-1} L_k, L_{k-1}), \tag{2.1}$$

where L_k denotes the LiDAR scan acquired at the k-th time step. The optimization is intialized with an initial transform estimated by HeRO. In the case where no input is received by HeRO, the identity transformation is used as the prior, and the system reverts to pure LiDAR odometry.

To enable global consistency across the history of scans, the motion estimated in the scan-to-scan matching stage is further refined by a scan-to-submap matching step. Here, L_k is matched to a local robot-centered subset S_k of the global map. Note that the global map is an accumulation of past point clouds aligned to the robot pose in the world frame:

$$\tilde{\mathbf{T}}_k^{k-1} = \arg\min_{\mathbf{T}_k^{k-1}} \mathcal{E}(\mathbf{T}_k^{k-1} L_k, S_k). \tag{2.2}$$

The initial guess of this optimization is $\hat{\mathbf{T}}_k^{k-1}$, which comes from Eq. (2.1). After scan-to-scan and scan-to-submap matching, the final estimated motion $\tilde{\mathbf{T}}_k^{k-1}$ between consecutive LiDAR acquisitions is the estimated robot odometry.

Localization Back-End: The odometry produced by the front-end is fed into the back-end of the SLAM system, referred to as LAMP (Large-scale Autonomous Mapping and Positioning) [6]. LAMP receives pose-to-pose constraints, based on the accumulation of odometry measurements, and solves a Pose Graph Optimization (PGO) [9] and Incremental Consistency Measurement (ICM) [10] problem for global localization when loop closures are detected. The output of this step is an estimate of the robot's past trajectory, called the *pose graph*. The use of this structure in the proposed coverage planning framework is discussed in Chap. 3.

Traversability Risk Analysis and Motion Planning

Another fundamental component of NeBula is traversability analysis and motion planning, referred to as STEP (Stochastic Traversability Evaluation and Planning) [11]. STEP allows the robots to safely traverse challenging terrains by (*i*) quantifying uncertainty and risk, and (*ii*) performing risk- and constraint-aware planning using model predicative control (MPC). Fig. 2.4 provides an overview of the approach.

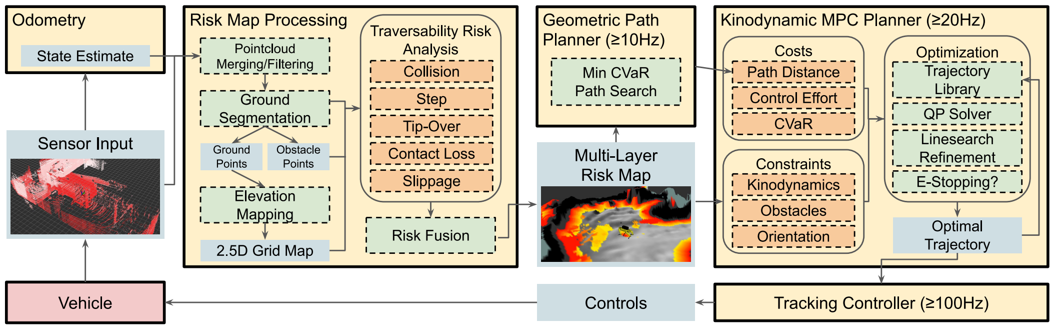
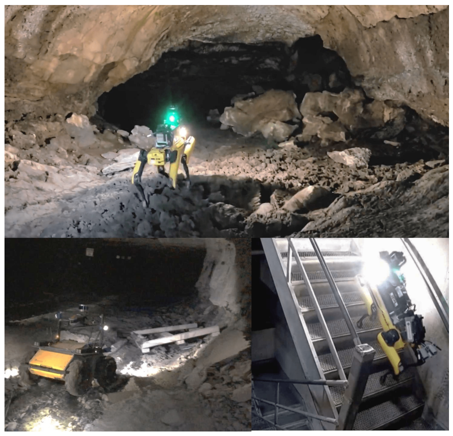


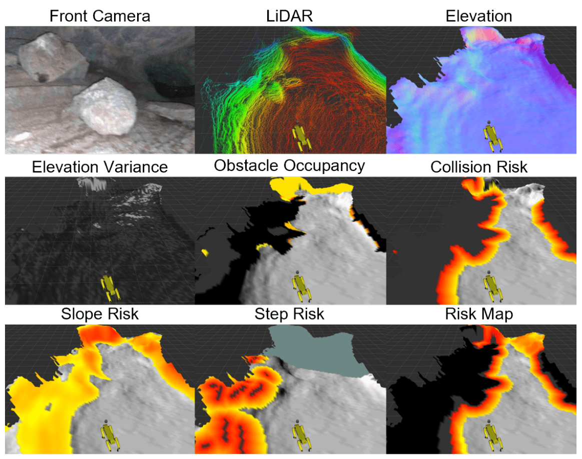
Figure 2.4: Traversability and Motion Planning Architecture Overview [11].

Risk Sources: Assessment of traversability risk involves the following key challenges: (i) localization error severely affects how sensor measurements are aggregated to construct environment maps, (ii) Sensor noise, sparsity, and occlusion induce significant biases and uncertainty in mapping, and (iii) the presence of various mobility-stressing elements (e.g., rubble, slopes, rocks, etc.) in the environment create highly complex constraints on the motion of the robot. STEP addresses these challenges by modeling traversability cost as a function of the following risk factors: collision, elevation changes, tip-over, contact loss, slippage, and sensor uncertainty.

Risk Analysis: Analysis relevant to the each risk source is performed independently. For instance, to compute the traversability cost associated with mobility-stressing elements (slopes, changes in elevation, obstacles, etc.), data from multiple sensors is combined, as shown in Fig. 2.5. Depth cameras are used for short-range sensing,



(a) Extreme Environments



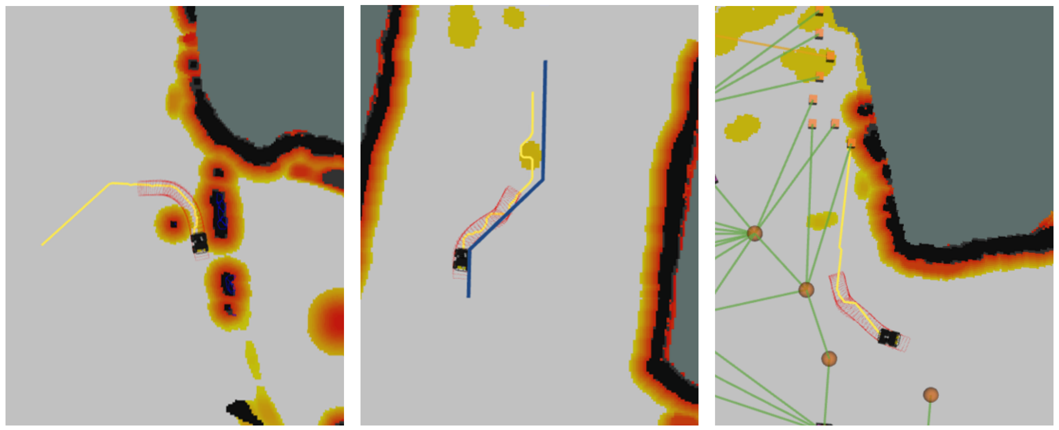
(b) Traversability Analysis

Figure 2.5: Risk-aware planning: (a) shows Nebula-integrated Spot and Husky exploring different mobility stressing environments: Valentine Cave at Lava Beds National Monument, CA (top), Arch Mine in Beckley, WV (bottom right), and Satsop power plant in Elma, WA (bottom left). (b) shows the traversability risk analysis based on multiple risk factors during Au-Spot's exploration of the cave. Risk map colors indicate traversability risk (white: safe, yellow to red: moderate risk, black: risky). [11]

instantaneous LiDAR point clouds for medium-range sensing, and spatially fused point clouds [12] for long-range sensing. Positive and negative obstacles, as well as steep slopes, are detected within a short-range by applying a step filter relative to the local ground plane. At medium- and long-range sensing: *(i)* positive obstacles are detected by performing ground segmentation [13] and settling-based collision checks [14], *(ii)* negative obstacles are identified by searching for surface discontinuities (holes) in the LiDAR point cloud, and *(iii)* steep slopes are detected using settling methods [14].

Risk Aggregation: Individual risk analysis is fused into a single risk value estimate with a confidence value. The risk value is defined in terms of the CVaR (Conditional Value-at-Risk) metric. Given a local grid-based discretization of the world, the CVaR metric is computed for each cell in order to generate a *risk map*. See Fig. 2.5 (b) for the risk map generated on board Au-Spot during its exploration of a cave. The use of the risk map in the proposed coverage planning framework is discussed in Chap. 3.

Motion Planning: A path generated by the coverage planners, discussed in Chaps. 3-5, is executed using the STEP motion planner. This involves a two-step process. First, given a series of coverage waypoints, an A* algorithm over the Risk Map generates a geometric plan that seeks to minimize path risk. The geometric planner outputs a sequence of poses. Given this sequence, a kinodynamically feasible trajectory is constructed using a model predicative control (MPC) planner. The planner aims to find a path near the sequence that satisfies all robot constraints and minimizes traversability risk to the robot. See Fig. 2.6 for a visualization of the STEP paths. The geometric and kinodynamic planners replan at a high rate so as to react to the sudden changes in the risk map. The planned trajectory is executed with a tracking controller, which sends control inputs to the robot platform.



(a) Two-layered STEP motion plan (b) Waypoint Sequence (blue path) (c) Single Waypoint (orange node)

Figure 2.6: Planned geometric path (yellow) and kinodynamic path (red squares). Given a sequence of waypoints (i.e. local coverage plan) in (a), and a single waypoint (i.e. global coverage plan) in (b), the geometric and kinodynamic planners find a feasible trajectory for the robot.

2.3 Coverage Path Planning Review

Traditional coverage methods decompose the environment into topologically distinct regions. Provably complete coverage of an environment is achieved by exhaustively traversing each free region. According to Choset's coverage taxonomy [15], classical coverage path planning can be broken down into two categories based on mapping techniques: exact cellular decomposition and approximate cellular decomposition [16], [17]. For ease of discussion, we expand these categories to include widely-adopted approaches that seek complete coverage using topological- and metric-based maps. Note that the coverage path planning algorithms most relevant to our proposed schemes are discussed in the *Related Work* sections in Chaps. 3-5.

Topological-based Approaches

Exact cellular decomposition involves separating a space into a set of non-intersecting regions, called cells, whose union completely fills a target area. The shape and dimension of each cell is based on geometric landmarks (obstacles, doors, etc.) or other sensed features in the environment. From this decomposition, a graph representation of the environment can be constructed where nodes correspond to cells and edges connect adjacent nodes [15]. Since the resolution is dependent upon the complexity of the environment, these topological graphs are typically compact and scale well to large environments [18].

A prominent example of exact cellular decomposition is Boustrophedon Cellular Decomposition (BCD), developed by Choset and Pignon [19]. In this method based on Morse decomposition, a slice is swept through an environment resulting in spatial connectivity changes occurring at critical points. The critical points decompose the target space into polygons that can be covered using simple sweeping motions.

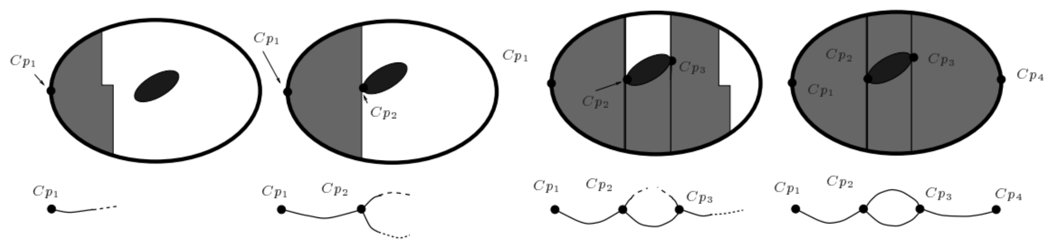


Figure 2.7: Sensor-based Boustrophen Cellular Decomposition [20]. The topological graph is incrementally built as the robot senses critical points C_{p_i}.

For coverage in *a priori* unknown environments, Cao et al. [21] introduced sensor-based Boustrophedon Cellular Decomposition. Later, Acar and Choset [20] built upon this method by proposing an improved critical point detection scheme based on the work of Rimon and Canny [22], which guarantees complete coverage of an unknown environment (Fig. 2.7). In these methods, critical points are obstacle vertices detected by on-board sensors [23]. Critical points divide the target space into polygons, which can be further fragmented based on a robot's sensing range [24]. Each cell is covered using back-and-forth sweeping motions, and traversal between cells is guided by simple graph search algorithms [15].

The Voronoi coverage technique is an online method for finding a topological graph of the environment based on a Voronoi skeleton – the set of points equidistant to two obstacles [25]. Local minima on the skeleton are designated as critical points, which are then used to divide the space into different, often semantic-based, regions which can be sequentially tackled [26]. An important note is that trajectories that follow the Voronoi skeleton have maximum clearance [27], thereby reducing risk to the robot as it traverses between regions.

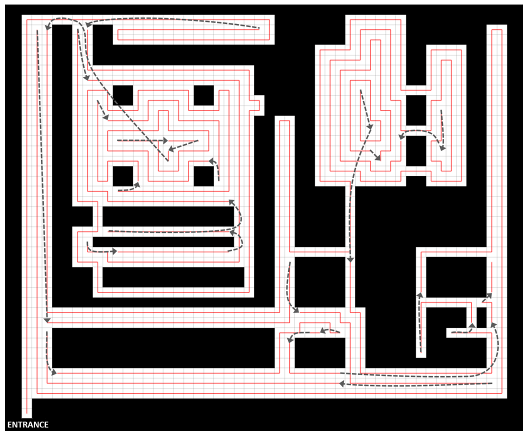
Similar to online Voronoi methods, frontier-based method rely on a continuous recalibration of the environment representation based on updated sensor information. These schemes identify the boundary between uncovered and covered space, regions termed *frontiers* [28], in order to construct a map of the environment. Frontiers are used extensively throughout the exploration literature [29–32], which in part is due to computational efficiency improvements in frontier detection algorithms [33]. As a robot visits frontiers and uncovers new areas, the graph-based representation grows and changes dynamically.

Metric-based Approaches

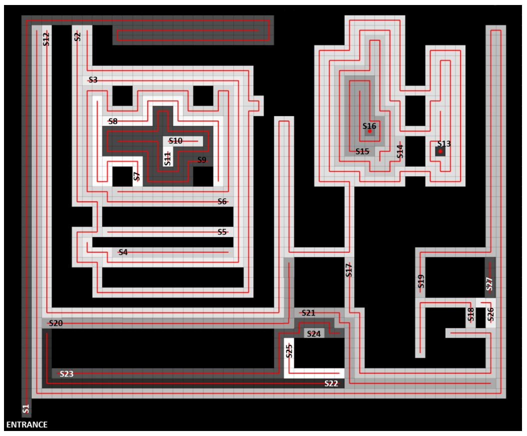
Approximate cellular decomposition, first introduced by Moravec and Elfes [34] [35], is a grid-based approach. Approximate methods are agnostic to environment complexity, and thus divide a space into cells with identical shape and size, based roughly on the robot's sensor footprint. Since the union of cells only approximates the target area [15], the completeness of these methods is determined by the resolution of the grid. When compared to graph-based methods, grid maps are easily constructed and maintained. However, since the grid needs to resolve detailed features in the environment, grid-based approaches can suffer from large space and time complexities [18].

Zelinsky et al. [36] pioneered use of grid representations to find offline coverage paths. The method, employing the conventional distance transform algorithm, propagates a wavefront through an environment represented by rectangular cells [36]. Adopting a gradient ascent-like strategy, traversal through the cells is based on a wavefront potential function, which can account for robot safety, as well as start and end locations. Later work by Oh et al. [37] applied triangular decomposition methods in order to increase the number of robot navigational directions.

Gabriely and Rimon [38] provided an offline and online coverage solution based on Minimum Spanning Trees. First the space is decomposed into rectangular cells. After a spanning tree is constructed, each cell is divided into four smaller cells, which are visited by traversing the tree. Another online algorithm based on spiral filling paths is the Backtracking Spiral Algorithm (BSA) [39], as illustrated in Fig. 2.8. Here, the robot navigates along the boundary of the environment. When a full loop is completed, the robot continues along the inner loop. This process continues until the spiral ends, at which point the robot backtracks to an uncovered pocket in the environment.



(a)



(b)

Figure 2.8: Coverage path based on the Backtracking Spiral Algorithm (BSA). The agent's spiral path (red lines) is shown overlaid on a grid-based world. Each spiral is indicated in (b) with a gray color-coding and label. The backtracking path (dashed gray lines) between spirals is shown in (a).

2.4 Partially Observable Markov Decision Process

The Partially Observable Markov Decision Process (POMDP) is a general framework for sequential decision-making when the current state is not perfectly observable by the robot. A POMDP is described as a tuple $\langle \mathbb{S}, \mathbb{A}, \mathbb{Z}, T, O, R \rangle$, where $\mathbb{S}$ is the set of joint robot-and-world states, $\mathbb{A}$ and $\mathbb{Z}$ are the set of robot actions and observations. At every time step, the agent performs an action $a \in \mathbb{A}$ and receives an observation $z \in \mathbb{Z}$ resulting from the robot's perceptual interaction with the environment. The motion model $T(s, a, s') = p(s' \mid s, a)$ defines the probability of being at state s' after taking action a at state s. The observation model $O(s, a, z) = p(z \mid s, a)$ is the probability of receiving observation z after taking action a at state s. The reward function $R(s, a)$ returns the expected utility for executing action a at state s [40, 41].

In a POMDP, it is assumed that the state is not perfectly observable. Rather, the robot only has access to an observation $z \in \mathbb{Z}$ that provides partial information about the current state. With only partial state knowledge, the robot must consider a complete history of past actions and observations:

$$h_t = \{a_0, z_1, ..., z_{t-1}, a_{t-1}, z_t\}. \tag{2.3}$$

The belief state, b_t, is an effective way of capturing h_t. At time t, the belief b_t is a posterior distribution over states conditioned on the initial belief b_0 and the past action-observation sequence:

$$b_t = p(s \mid b_0, a_{0:t-1}, z_{1:t}) \ . \tag{2.4}$$

A POMDP can be formulated as a *belief MDP*, or a Markov decision process where every state is a belief. A belief MDP is described as a tuple $\langle \mathbb{B}, \mathbb{A}, \tau, r \rangle$, where $\mathbb{B}$ is the set of belief states over the POMDP states, $\mathbb{A}$ is the set of robot actions, $\tau(b, a, b') = p(b' \mid b, a)$ is the belief state transition function, and the reward function $r(b, a)$ returns the utility for executing action a in belief b.

A solution to a POMDP is a policy that maps beliefs to actions, and the optimal policy is one that maximizes the expected cumulative reward. The optimal policy of a POMDP for all time $t \in [0, \infty)$, $\pi^*_{0:\infty} : \mathbb{B} \longrightarrow \mathbb{A}$, is defined as:

$$\pi^*_{0:\infty}(b) = \operatorname*{argmax}_{\pi \in \Pi_{0:\infty}} \mathbb{E} \sum_{t=0}^{\infty} \gamma^t r(b_t, \pi_t(b_t)), \tag{2.5}$$

where $\gamma \in (0, 1]$ is a discount factor for the future rewards, $\Pi_{0:\infty}$ is the space of possible policies, and $r(b, a) = \int_s R(s, a) b(s) \mathrm{d}s$ denotes a belief reward which is the expected reward of taking action a at belief b.

Due the computational complexity of POMDPs, they cannot be solved exactly in real-time. Online POMDP algorithms approximate the optimal policy by reasoning over the states reachable from the current belief state over a finite horizon. Common online methods use a Monte Carlo tree search (MCTS) approach that simulates, or *rolls out*, action-observation sequences according to a provided rollout policy. Given a generative model (or a black box simulator) for discrete action and observation spaces, MCTS learns the value function of the reachable belief subspace with an adequate exploration-exploitation trade-off [41, 42].

Chapter 3

COVERAGE PLANNING FRAMEWORK

In order for an autonomous robot to efficiently explore an *a priori* unknown environment, it must account for uncertainty in sensor measurements, hazard assessment, localization, and motion execution. Making decisions for maximal reward in a stochastic setting requires value learning and policy construction over a belief space, i.e., probability distribution over all possible robot-world states. However, belief space planning in a large spatial environment over long temporal horizons suffers from severe computational challenges. Moreover, constructed policies must safely adapt to unexpected changes in the belief at runtime. This chapter proposes a scalable value learning framework, PLGRIM (Probabilistic Local and Global Reasoning on Information roadMaps), that bridges the gap between *(i)* local, risk-aware resiliency and *(ii)* global, reward-seeking mission objectives. Leveraging hierarchical belief space planners with information-rich graph structures (Fig. 3.1), PLGRIM addresses large-scale exploration problems while providing locally near-optimal coverage plans.

This chapter was adapted from:

S. Kim*, **A. Bouman***, G. Salhotra, D. Fan, K. Otsu, J. Burdick, and A. Agha-Mohammadi. “PLGRIM: Hierarchical Value Learning for Large-scale Exploration in Unknown Environments”. In: *International Conference on Automated Planning and Scheduling (ICAPS)*. Vol. 31. 2021, pp. 652– 662.

3.1 Introduction

Consider a large-scale coverage mission in an unknown environment, in which a robot is tasked with exploring and searching a GPS-denied unknown area, under given time constraints. This problem has a wide range of applications, such as interplanetary exploration and search-and-rescue operations [43, 44]. Essential elements of an autonomy architecture needed to realize such a mission include creating a map of the environment, accurately predicting risks, and planning motions that can meet the coverage and time requirements while minimizing risks. In such an architecture, quantifying and planning over uncertainty is essential for creating robust, intelligent, and optimal behaviors.

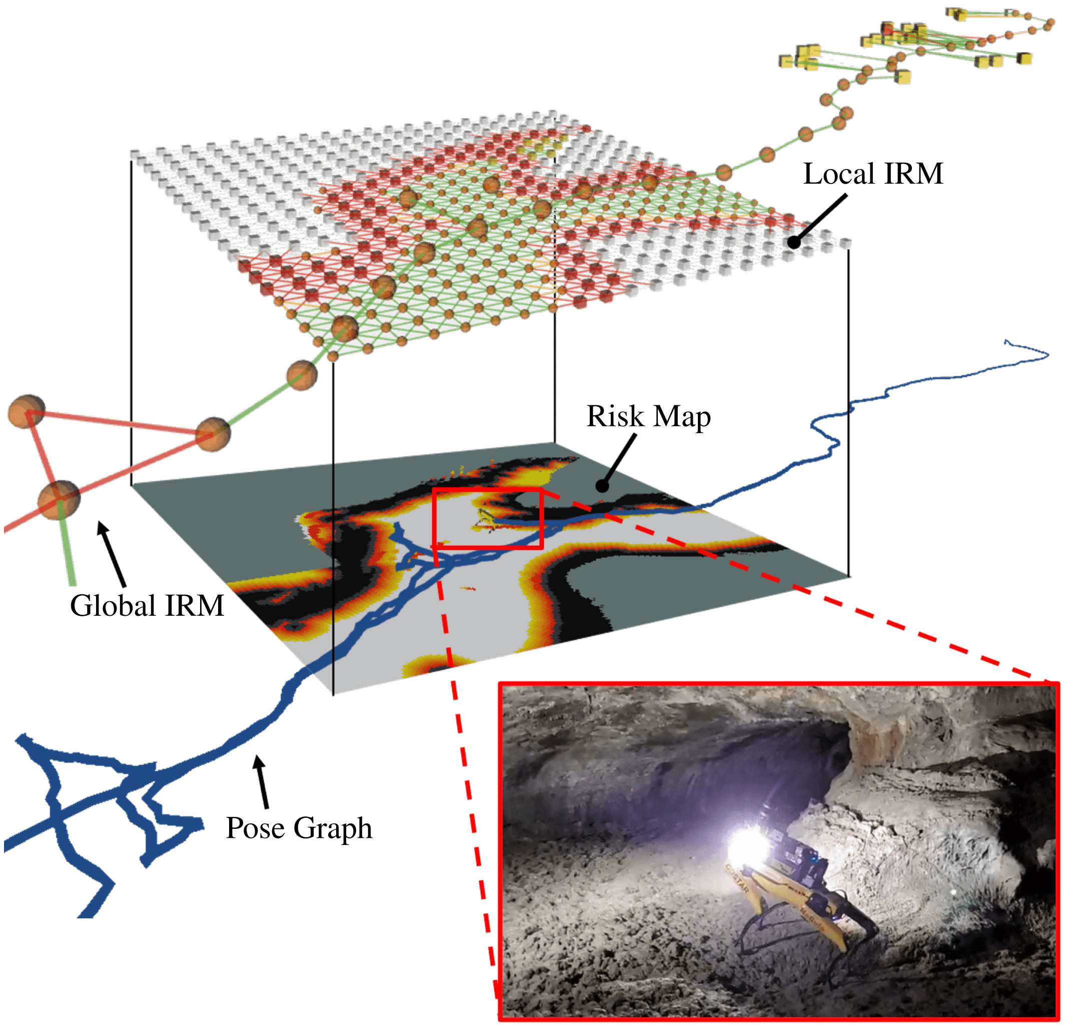


Figure 3.1: Hierarchical Information RoadMaps (IRMs) generated during Au-Spot's autonomous exploration of Martian-analog caves at Lava Beds National Monument, Tulelake, CA.

From a value learning perspective, a coverage planning problem in an unknown space can be considered an active learning problem over the robot's belief, where belief is defined as the probability distributions over all possible joint robot-world states. The objective is to find the best action sequence that maximizes the accumulated reward over time. The agent must accumulate data to incrementally build a model of its environment, and needs to understand the effects of its actions on the quality and quantity of data it collects.

Since the agent's future actions affect its belief of the world and robot state, this coverage problem is fundamentally a Partially Observable Markov Decision Process (POMDP) problem [45]. Belief value learning in the POMDP setting intrinsically suffers from the curse of dimensionality [40] and curse of history [46]. Many powerful methods have been proposed to extend the spatial and temporal horizons of POMDPs with varying degrees of efficiency and accuracy, such as [47–50]. This chapter focuses on challenging exploration problems with very large spatial extents (>1 km), long temporal horizons (>1 hour), and high dimensional belief states (including beliefs on the state of the environment) that exacerbate the curses of dimensionality and history for POMDPs.

The main contribution of this work is three-fold:

1) Scalable belief representation of local traversability and global coverage states of large environments.
2) Hierarchical value learning for efficient coverage policy search over a long horizon under uncertainty.
3) Policy reconciliation between planning episodes for adaptive and resilient execution in the real world.

More precisely, this chapter introduces spatial and temporal approximations of the coverage policy space to enable computational tractability for real-time online solvers. Spatially, the belief space is decomposed into task-relevant partitions of space, enriched with environment map estimates. The partitioning structure is called an Information Roadmap (IRM), as shown in Fig. 3.1 [51]. Temporally, the problem is decomposed into local and global hierarchical levels, and then we solve for belief space policies that provide locally near-optimal coverage plans with global completeness. A Receding Horizon Planning (RHP)-based technique is proposed to address real-world stochasticity in state estimation and control at runtime.

The remainder of this chapter is as follows. Following the related work discussion, Sec. 3.3 discusses the unknown environment coverage problem. In Sec. 3.5, we propose a hierarchical belief representation and value learning framework. Experimental results in simulation and on a physical robot are presented in Sec. 3.10, and Sec. 3.11 summarizes the chapter.

3.2 Related Work

Frontier-based exploration is a widely used approach for autonomous exploration (e.g., [28, 33, 52–55]). By continuing exploration until exhausting all remaining frontiers, frontier-based approaches can guarantee completeness of the coverage of reachable spaces. These methods typically rely on myopic (e.g., one-step) look-ahead greedy policies, selecting the best frontier upfront. Hence, they can be subject to local minima and provide suboptimal solutions in time.

Model-free reinforcement learning (RL) has been applied to coverage and exploration problems (e.g., [56–59]). In this setting, the typical approach is to find a policy which maps sensor data to actions, with the objective of maximizing the reward. When it comes to long-range, large-scale missions on physical robots, collecting necessary data can be a significant challenge for this class of methods.

POMDP-based approaches generate a non-myopic policy by considering long-horizon action sequences (e.g., [60], [61]), interactively learning the value function, and returning the best action sequence that maximizes the accumulated rewards. Different methods have reduced the complexity of the POMDP problem in coverage and exploration problems. Indelman, Carlone, and Dellaert [62] and Martinez-Cantin et al. [63] employed a direct policy search scheme with a Gaussian belief assumption. Lauri and Ritala [64] extended this to non-Gaussian beliefs using the POMCP (Partially Observable Monte-Carlo Planning) solver. However, when it comes to the large-scale coverage missions, the prior approaches do not scale well due to the curse of history and dimensionality [46].

Hierarchical planning structures [65] aim to tackle larger problems by employing multiple solvers running at different resolutions, and are often found to be effective. In the coverage and exploration context, Umari and Mukhopadhyay [32] applied hierarchical planning to frontier-based exploration, while [66] extended the lower-level module to a more sophisticated frontier selection algorithm which considers the information gain along each path. Lauri and Ritala [64] replaced the lower-level module with a POMDP-based planner to improve local coverage performance with non-myopic planning. Kim, Thakker, and Agha-Mohammadi [67] proposed a hierarchical online-offline solver for risk-aware navigation. Vien and Toussaint [68] suggested a hierarchical POMCP framework which outperformed Bayesian model-based hierarchical RL approaches in some benchmarks.

3.3 Problem Formulation

Autonomous exploration in unknown environments under motion and sensing uncertainty can be formulated as a Partially Observable Markov Decision Process (POMDP), which is one of the most general models for sequential decision making.

In this section, we present a POMDP formulation for coverage problems and address its intrinsic challenges.

For our coverage planning problem, we define the state as $s = (q, W)$, where q is the robot state and W is the world state. We maintain two representations of the world, i.e., $W = (W_r, W_c)$, where W_r denotes the world traversal risk state and W_c is the world coverage state.

W_r encodes the traversability risk of the world with respect to a robot's dynamic constraints. This state is critical in capturing traversability-stressing elements of the environment (slopes, rough terrain, and narrow passages, etc.) and is typically constructed by aggregating long-range sensor measurements. The cost function $C(W_r, q, a)$ returns a value which abstracts the actuation effort and risk associated with executing action a at robot state q on W_r.

W_c provides an estimation of what parts of the world have been observed, or *covered*, by a particular sensor. The coverage state is generated by specific sensor measurements, which may not necessarily be useful as navigational feedback, but instead are based on a task at hand. For instance, the coverage sensor may be a thermal camera for detecting thermal signatures, or a vision-based camera for identifying visual clues in the environment. As a robot moves, the sensor footprint sweeps the environment, expanding the covered area, or more generally, the task-relevant information about the world.

The coverage planning objective is to determine a trajectory through an environment that maximizes information gain I while simultaneously minimizing action cost C. As such, the traversal risk and coverage states form the basis of the coverage reward function:

$$R(s, a) = f(I(W_c, a),\ C(W_r, q, a)), \tag{3.1}$$

where $I(W_c, a) = H(W_c) - H(W_c \mid a)$ is quantified as reduction of the entropy H in W_c after taking action a. Note that when covering an unknown space, we do not have strong priors about the parts of the world that have not yet been observed. Hence, knowledge about W_c and W_r in Eq. (3.1) at runtime is incomplete and often inaccurate. Thus, in such domains, a Receding Horizon Planning (RHP) scheme has been widely adopted as the state-of-the-art [69].

In POMDP formulation with RHP, the objective function in Eq. (2.5) is modified:

$$\pi^*_{t:t+T}(b) = \underset{\pi \in \Pi_{t:t+T}}{\operatorname{argmax}} \ \mathbb{E} \sum_{t'=t}^{t+T} \gamma^{t'-t} r(b_{t'}, \pi_{t'}(b_{t'})), \tag{3.2}$$

where T is a finite planning horizon for a planning episode at time t. Given the policy from the last planning episode, only a part of the optimal policy, $\pi^*_{t:t+\Delta t}$ for $\Delta t \in (0, T]$, will be executed at runtime. A new planning episode will start at time $t + \Delta t$ with updated belief about q, W_c, and W_r.

3.4 Challenges

This section broadly identifies the challenges associated with solving the unknown coverage planning problem, Eq. (3.2), as computational complexity—in both time and space—and conflicting policy objectives over consecutive planning episodes, arising from unexpected updates in the belief at runtime.

Time Complexity

POMDP planning suffers from *the curse of dimensionality* [40] and *the curse of history* [46]. The former difficulty refers to fact that size of the belief grows exponentially with the size of the underlying state space. In a grid world where n and k denote the grid dimension and number of discretization levels, respectively, the space complexity for a single belief state is $O(|k|^n)$. Refer to Fig. 3.2. The latter difficulty refers to the fact that the policy space (i.e., number of action-observation sequences) grows exponentially with the planning depth d, i.e., $O(|\mathbb{A}|^d |\mathbb{Z}|^d)$. As an example, for large-scale exploration of a 1 km-long environment with an action resolution of 1 m, the planning depth d must be at least 10^3 in order to reason about the coverage plan across the environment.

Space Complexity

In addition to the classic time complexity of POMDPs, space complexity also poses a considerable challenge when handling the unknown environment coverage problem. As referenced above, in a grid world, the memory complexity is $O(|k|^n)$, with n and k denoting the grid dimension and number of discretization levels, respectively. For a 1 km^2 environment at a 0.1 m resolution, with floating-point risk and coverage values associated with every cell, the required memory is 800 MB. This amount of memory should be allocated for every search node, and thus the full space complexity of planning is $O(|\mathbb{A}|^d |\mathbb{Z}|^d |k|^n)$ during each planning episode.

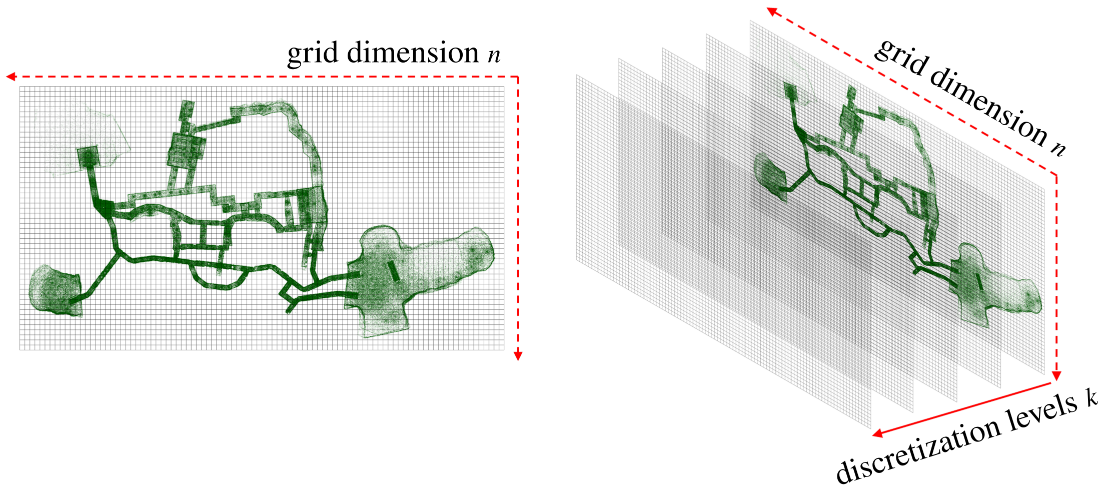


Figure 3.2: An illustrative example of the space complexity associated with a grid world, shown overlaid on the pointcloud map of the DARPA SubT final circuit. The grid dimension is n, and the number of discretization levels is k. Here there are "quint" ($k = 5$) coverage levels. For a single particle state, each cell is assigned one of five coverage probabilities: $p_{cov} = 1.0, 0.75, 0.5, 0.25$, or 0.0. The belief state is the sum of all particles [47], hence the single belief state complexity is $O(|k|^n)$.

Unexpected Belief Updates

As the robot explores its environment, it receives new sensory information, updates its belief, and constructs a new coverage policy in a receding horizon fashion. Policies generated during consecutive planning episodes must respect the kinodynamic constraints of the robot, while simultaneously adapting to unexpected hazards in the environment. We refer to these two distinct, and often opposed, objectives as *consistency* and *resiliency* of the receding-horizon policy, respectively. Path consistency ensures smooth trajectories and continuous velocities during transitions from one policy to the next, while path resiliency ensures the path adapts to unexpected changes in the world risk state. Thus, it is imperative to find a balance between policy consistency and resiliency, particularly for safety-critical systems.

3.5 PLGRIM: Hierarchical Coverage Planning on Information Roadmaps

In this section, we present a novel and field-tested coverage planning autonomy framework, *PLGRIM (Probabilistic Local and Global Reasoning on Information roadMaps)*, for exploration of large-scale unknown environments with complex terrain. Our proposed methods to tackle the challenges described in Sec. 3.4 are:

1) Space Complexity: We introduce a hierarchical belief space representation that is compact, versatile, and scalable. We refer to this representation as

an Information RoadMap (IRM). Hierarchical IRMs can effectively encode a large-scale world state, while simultaneously capturing high-fidelity information locally.

2) Time Complexity: We propose hierarchical POMDP solvers that reason over long horizons within a suitable replanning time with locally near-optimal performance. Higher-level policies guide lower-level policies, resulting in a cascaded decision process.

3) Unexpected Belief Updates: We introduce a receding-horizon policy reconciliation method that respects the robot's dynamic constraints while ensuring resiliency to unexpected belief updates.

In the following subsections, we provide the technical details about the proposed framework, illustrated in Fig. 3.3.

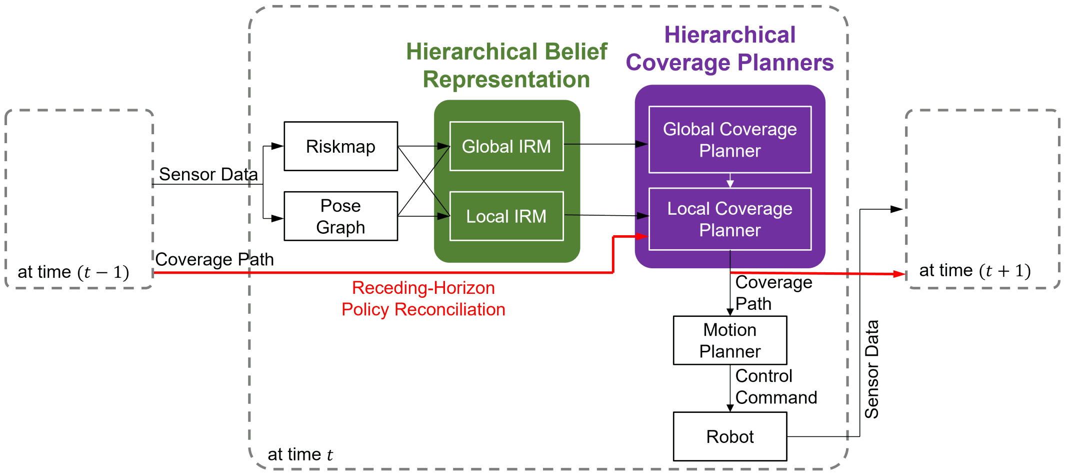


Figure 3.3: Illustration of PLGRIM framework for large-scale exploration in unknown environments. It *i)* maintains hierarchical beliefs about the traversal risks and coverage states, *ii)* performs hierarchical value learning to construct an exploration policy, and *iii)* reconciles policies over receding-horizon planning episodes.

3.6 Overview

To enable efficient and reactive robot behaviors on very large scales, we decompose the problem into tractable subproblems by introducing spatial and temporal abstractions. Spatially, the belief space is approximated by a task-dependent structure, enriched with environment map estimates. Temporally, the belief space is approximated by the aggregation of multiple structures, each spanning a different spatial range. Finally, we introduce a cascaded optimization problem that returns a policy over the stratified belief space in real time.

Belief Decomposition

Let us denote the global world state as W^g and the local world state as W^ℓ, which is a subset of the global state, i.e., $W^\ell \subset W^g$, around the robot. Recall that each world state can be decomposed into their traversal risk and coverage components, i.e., $W^g = (W_r^g, W_c^g)$ and $W^\ell = (W_r^\ell, W_c^\ell)$, as discussed in Section 3.3. We define local and global belief states as $b^\ell = p(q, W^\ell)$ and $b^g = p(q, W^g)$, respectively, where $p(W^\ell)$ is a local, robot-centric, rolling-window world belief representation with high-fidelity information, and $p(W^g)$ is a global, unbounded world belief representation with approximate information.

Policy Decomposition

We decompose the policy into local and global policies: π^ℓ and π^g, respectively. The overall policy $\pi \in \Pi$ is constructed by combining the local and global policies:

$$\pi(b) = \pi^\ell(b^\ell; \pi^g(b^g)). \tag{3.3}$$

We approximate the original RHP optimization problem in Eq. (3.2) using the following cascaded hierarchical optimization problem:

$$\begin{aligned}
\pi_{t:t+T}(b) &= \underset{\pi \in \Pi_{t:t+T}}{\operatorname{argmax}} \, \mathbb{E} \sum_{t'=t}^{t+T} \gamma^{t'-t} r(b_{t'}, \pi(b_{t'})) \\
&\approx \underset{\pi^\ell \in \Pi^\ell_{t:t+T}}{\operatorname{argmax}} \, \mathbb{E} \sum_{t'=t}^{t+T} \gamma^{t'-t} r^\ell(b^\ell_{t'}, \pi^\ell(b^\ell_{t'}; \pi^g_{t:t+T}(b^g_t))), && (3.4) \\
&\text{where } \pi^g_{t:t+T}(b^g) = \underset{\pi^g \in \Pi^g_{t:t+T}}{\operatorname{argmax}} \, \mathbb{E} \sum_{t'=t}^{t+T} \gamma^{t'-t} r^g(b^g_{t'}, \pi^g(b^g_{t'})). && (3.5)
\end{aligned}$$

where $r^\ell(b^\ell, \pi^\ell(b^\ell))$ and $r^g(b^g, \pi^g(b^g))$ are approximate belief reward functions for the local and global belief spaces, respectively. Note that the codomain of the global policy $\pi^g(b^g)$ is a parameter space Θ^ℓ of the local policy $\pi^\ell(b^\ell; \theta^\ell)$, $\theta^\ell \in \Theta^\ell$.

According to this formulation, we maintain the hierarchical belief representations (Sec. 3.7) and solve for hierarchical POMDP policies (Sec. 3.8). For local planning consistency and resiliency, we extend Eq. (3.4) to a joint optimization problem given the previous planning episode policy (Sec. 3.9).

Algorithm 1 Hierarchical IRM Construction

```
input: Risk Map, Pose Graph

# Local IRM
Local IRM $G^\ell = (N^\ell, E^\ell) \leftarrow (\emptyset, \emptyset)$
Add uniformly sampled nodes $\{n_i^\ell\}_i$ around the robot to $N^\ell$
for each $n_i^\ell \in N^\ell$ do
    Compute risk probability $p(n_{i,r}^\ell)$ and coverage probability $p(n_{i,c}^\ell)$ from Risk Map and
    Pose Graph for $n_i^\ell$
    Add $p(n_{i,r}^\ell)$ and $p(n_{i,c}^\ell)$ to the properties of $n_i^\ell$
end for
Add edges for 8-connected neighbors, $\{e_{ij}\}_{i,j}^\ell$, to $E^\ell$
for each $e_{ij}^\ell \in E^\ell$ do
    Compute traversal risk $\rho_{ij}$ and distance $d_{ij}$ for $e_{ij}^\ell$
    Add $\rho_{ij}$ and $d_{ij}$ to the properties of $e_{ij}^\ell$
end for

# Global IRM
if not initialized then
    Global IRM $G^g = (N_b^g \cup N_f^g, E^g) \leftarrow (\emptyset, \emptyset)$
end if
Get the current robot pose $q$ from Pose Graph
if $q$ is farther from any breadcrumb node $\forall n_i^g \in N_b^g$ than $\bar{d}_b$ then
    Add a new breadcrumb node $n^g = q$ to $N_b^g$
end if
Run FrontierManager to add new frontiers $\{n_{f^+}^g\}$ with coverage probabilities
$\{p(n_{f^+,c}^g)\}$, and prune invalidated frontiers, $\{n_{f^-}^g\}$, based on the current Risk Map
and Pose Graph ▷ [33]
for each node $n_i^g \in \mathcal{N}_{G^g}(q)$ do
    for each nearby node $n_j^g \in \mathcal{N}_{G^g}(n_i^g)$ do
        Compute the traversal distance $d_{ij}$ and risk $\rho_{ij}$
        if $d_{ij} < \bar{d}_e$ and $\rho_{ij} < \bar{\rho}_e$ then
            Add an edge $e_{ij}^g$ to $E^g$ with properties $d_{ij}$ and $\rho_{ij}$
        else
            Remove the edge $e_{ij}^g$ from $E^g$
        end if
    end for
end for
return $G^\ell$ and $G^g$
```

3.7 Hierarchical Belief Representation

We introduce a hierarchical approximation of the belief space by decomposing the environment representation into multiple information-rich structures, each referred to as an Information Roadmap (IRM). We construct and maintain IRMs at two hierarchical levels: the *Local IRM* and *Global IRM*, as illustrated in Fig. 3.1.

World Belief Information Sources

During its exploration of an unknown environment, at any given time, the robot's understanding of the world is limited to noisy estimates of an observed subset of the world. IRMs are constructed from these estimates – namely, the *Risk Map* and *Pose Graph*. A Risk Map, constructed through the aggregation of point cloud sensor measurements, is a local rolling-window map that provides risk assessment, effectively encoding the risk belief over the local world state W^{ℓ} [11]. A Pose Graph estimates the past trajectory of the robot from relative pose measurements and informs the coverage belief over the global world state W^{g} [6].

World Belief Construction

We choose to represent the world as a generic graph structure, $G = (N, E)$ with nodes N and edges E, as the data structure to represent the belief about the world state. Nodes represent discrete areas in space, and edges represent actions. More precisely, we define an action as a motion control from the current node $n_i \in N$ to a neighboring node $n_j \in N$, connected by an edge $e_{ij} \in E$.

For a detailed description of the Local and Global IRM construction processes, see Algorithm 1. We now describe the distinguishing features of each IRM:

1) *Local IRM*: As an instantiation of the local world belief $p(W^{\ell})$, we employ a rolling, fixed-sized grid structure $G^{\ell} = (N^{\ell}, E^{\ell})$, which is centered at the robot's current position. We uniformly sample nodes $n_i^{\ell} \in N^{\ell}$ from W^{ℓ}, and compute the risk and coverage probability distribution over a discrete patch centered at each node, i.e., $p(n_{i,r}^{\ell})$ and $p(n_{i,c}^{\ell})$, which are stored as node properties. For an edge e_{ij}^{ℓ}, we compute and store the traversal distance d_{ij} and risk ρ_{ij}, which effectively encodes $p(W_r^{\ell})$ between two connected nodes. In summary, the Local IRM contains relatively high-fidelity information at a high resolution, but locally.

2) *Global IRM*: As an instantiation of the global world belief $p(W^{g})$, we employ a sparse bidirectional graph structure $G^{g} = (N^{g}, E^{g})$, which is fixed in the global reference frame. Due to the space complexity concerns detailed in Sec. 3.4, a densely-sampled grid structure, like G^{ℓ}, is not a viable option for G^{g}, as it should span up to several kilometers. Instead, we sparsely and non-uniformly sample nodes $n_i^{g} \in N^{g}$ from W^{g} based on certain node-classifying conditions. Specifically, N^{g} contains two mutually exclusive subsets of nodes: *breadcrumbs* and *frontiers*, as shown in Fig. 3.4. Breadcrumb

nodes are sampled directly from the Pose Graph, and thus capture the *covered traversable* space of W^g. Alternatively, frontier nodes are sampled from the border between covered and uncovered areas, and thus capture the *uncovered traversable* space of W^g. Finally, in order for such a candidate node n_i^g to be added to G^g, there must exist a traversable path to at least one nearby node $n_j^g \in N^g$. If such a path exists, an edge e_{ij}^g, storing traversal distance d_{ij} and risk ρ_{ij}, is added to G^g. In summary, the Global IRM captures the free-space connectivity of W^g with a notion of coverage, and does not explicitly encode highly-likely untraversable or uncertain areas in W^g in order to achieve compact representation of the large-scale environment.

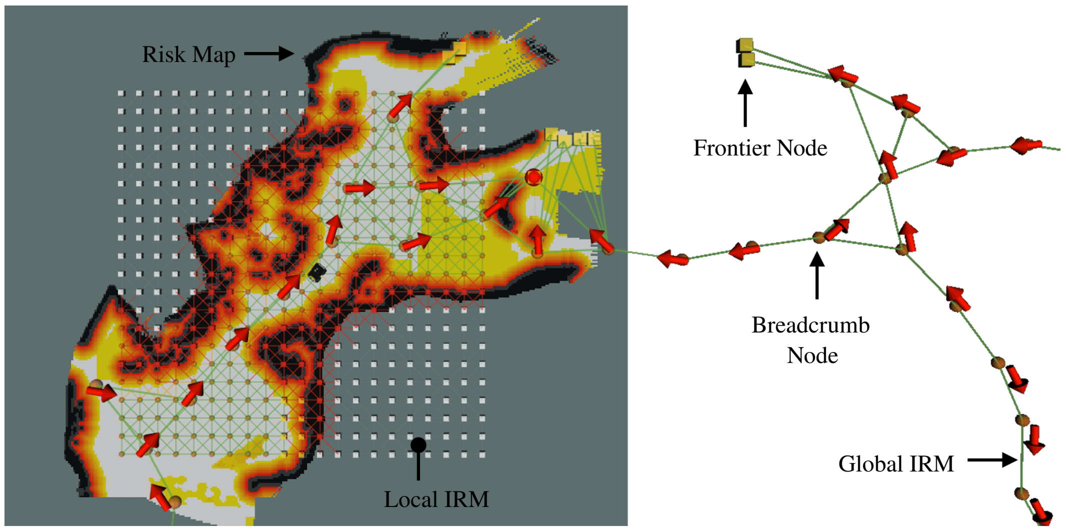


Figure 3.4: QMDP policy (red arrows displayed above *breadcrumb* nodes) for Global Coverage Planning (GCP). A red sphere indicates the QMDP frontier goal.

3.8 Hierarchical Value Learning

Given Local and Global IRMs as the hierarchical belief representation, we solve the cascaded hierarchical POMDP problems, Eq. (3.4) and Eq. (3.5), for coverage in an unknown environment.

Solver Formulation

We start by introducing some notations. The *value function* $V(b;\pi)$ is the expected reward of following policy π, starting from belief b:

$$V(b;\pi) = \mathbb{E}\Big[\sum_t \gamma^t r(b_t, \pi(b_t))\Big]. \tag{3.6}$$

The value of taking action a in belief b under a policy π is the *action-value function*:

$$Q(b, a; \pi) = r(b, a) + \sum_{b' \in \mathbb{B}} \gamma \mathcal{T}(b, a, b') V(b'; \pi), \tag{3.7}$$

where $\mathcal{T}(b, a, b')$ is the transition probability from b to b' under action a, as follows:

$$\mathcal{T}(b, a, b') = \sum_{z \in \mathbb{Z}} p(b' | b, a, z)\ p(z | b, a). \tag{3.8}$$

A POMDP solver tries to learn $Q(b, a)$ and $V(b) = \max_{a \in \mathbb{A}} Q(b, a)$, and returns the policy π that specifies the best action for a given belief b, i.e., $\pi(b) = \operatorname{argmax}_{a \in \mathbb{A}} Q(b, a)$.

Generalized Coverage Reward

Entropy provides a measure of uncertainty of a random variable's belief. Given an IRM $G = (N, E)$ containing a $p(n_{i,c})$ value for each node $n_i \in N$, the entropy of the world coverage state is:

$$H(p(W_c)) = -\sum_{i}^{|N|} \Big[p(n_{i,c}) \log p(n_{i,c}) + p(n_{i,\neg c}) \log p(n_{i,\neg c}) \Big]. \tag{3.9}$$

If $a \in \mathbb{A}$ is a motion from node $n_i \in N$ to node $n_j \in N$ along edge $e_{ij} \in E$, then the coverage information gain (i.e., coverage uncertainty reduction) in *coverage belief* $p(W_c)$ induced by a is defined as:

$$I(W_c \,|\, a) = \underbrace{H(p(W_c))}_{\text{current entropy}} - \underbrace{H(p(W_c \,|\, a))}_{\text{future entropy}}, \tag{3.10}$$

where the second term represents the expected future entropy of the world coverage state after execution of action a. In practice, Eq. (3.10) is difficult to compute for a large world state. For a practical computation of the information gain associated with an action on the local and global IRMs, see Chaps. 4.6 and 5.5.

Although the action cost function at each hierarchical level is dependent upon the IRM's particular action set E (i.e., Local and Global IRMs have different action sets), it can be generically formulated as:

$$C(W_r, q, a) = k_d d_{ij} + k_\rho \rho_{ij} + k_\mu \mu_{ij}(q, a), \tag{3.11}$$

where d_{ij} and ρ_{ij} are the traversal distance and risk along edge e_{ij}, respectively. The cost $\mu_{ij}(q, a)$ is associated with the current motion primitive, and is a consequence of the robot's non-holonomic constraints, such as the heading direction. Constants k_d, k_ρ, and k_μ weigh the importance of traversal distance, risk, and motion primitive history on the total action cost.

Then, finally the coverage reward function is defined as a weighted sum of the information gain and action cost:

$$R(s, a) = k_I I(W_c, z) - k_C\, C(W_r, q, a)), \tag{3.12}$$

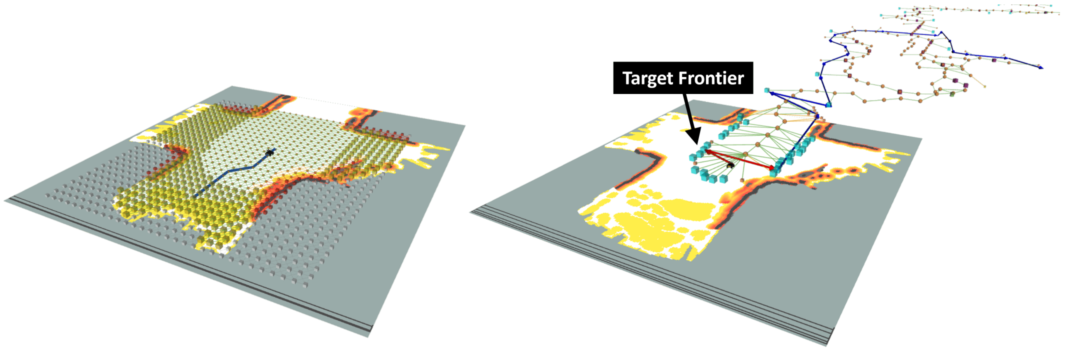
where k_I and k_C are constant weights.

Local-Global Coverage Planner Coordination

In our cascaded hierarchical optimization framework, we first solve for the global policy in Eq. (3.5). The global policy solution then serves as an input parameter to the local policy in Eq. (3.4). This means that Global Coverage Planner (GCP) provides *global guidance* to the Local Coverage Planner (LCP).

The role of GCP is to construct a low-fidelity policy that provides global guidance to uncovered areas, at which point, LCP instructs a local coverage behavior. More concretely, a target frontier node in the Global IRM, $n_f^g \in N_f^g$, can be extracted from the global-level control $a^g \in \mathbb{A}^g$ provided by GCP. Since the environment can be very large (>1 km), GCP must be capable of reasoning over hundreds of nodes on the Global IRM. To alleviate this scalability challenge, we assume that GCP's policy terminates at frontier nodes. By classifying frontier nodes as terminal in the belief space, we can assume no changes occur to the world coverage state before termination. Therefore, we omit W from the state space for GCP.

The role of LCP is to construct a high-fidelity policy that provides local guidance based on information gathering, traversal risk (e.g., proximity to obstacles, terrain roughness, and slopes), and the robot's mobility constraints (e.g., acceleration limits and non-holonomic constraints of wheeled robots). LCP has two phases: i) reach the target area based on GCP's guidance, and ii) construct a local coverage path after reaching the target area. If the target frontier is outside the Local IRM range, i.e., $n_f^g \notin W^\ell$, LCP simply instantiates high-fidelity control based on the global-level control a^g, i.e., the robot backtracks along global IRM breadcrumbs in order to reach the target frontier. If the target frontier n_f^g is within the Local IRM range, i.e., $n_f^g \in W^\ell$, then LCP performs the nominal information-gathering coverage optimization, as described in Eq. (3.4). Refer to Fig. 3.5 as an illustration of this local-global coverage planner coordination.



(a) Local Coverage Plan

(b) Global Coverage Plan

Figure 3.5: Illustration of local-global coverage planner coordination. Simultaneously constructed local (a) and global (b) coverage plans during Husky's exploration of a limestone mine are shown. Since the target frontier associated with the GCP policy is inside the bounds of the Local IRM, LCP maintains control guides the robot. The local and global policies shown are developed in Chaps. 4 and 5, respectively.

Global Coverage Planner (GCP) Algorithm

In this work, we adopt the QMDP approach for the global coverage planning problem [70]. The key idea of QMDP is to assume the state becomes fully observable after one action, so that the value function for further actions can be evaluated efficiently in an MDP (Markov Decision Process) setting. In our global coverage planning domain, we define the first action to be the robot's relocation to a nearby node on the Global IRM. At this point, the robot pose is assumed to be fully observable, while the world risk and coverage states remain unchanged.

More formally, we solve for $Q^g_{\text{MDP}}(q^g, a^g)$ by ignoring uncertainty in the robot pose q^g and changes in the world coverage state W^g_c. In this MDP setting, $Q^g_{\text{MDP}}(q^g, a^g)$ can be learned by Value Iteration very efficiently, even for long discount horizons. Then, we evaluate the *action-value function* in Eq. (3.7) in a POMDP setting for the current belief and the feasible one-step actions:

$$Q(b^g, a^g) = \int_{q^g} b(q^g) Q^g_{\text{MDP}}(q^g, a^g) \mathrm{d}q^g. \tag{3.13}$$

Finally, a POMDP policy can be obtained as follows:

$$\pi^g(b^g) = \underset{a^g \in \mathbb{A}^g}{\operatorname{argmax}}\, Q(b^g, a^g). \tag{3.14}$$

An example of the GCP policy is depicted in Fig. 3.4.

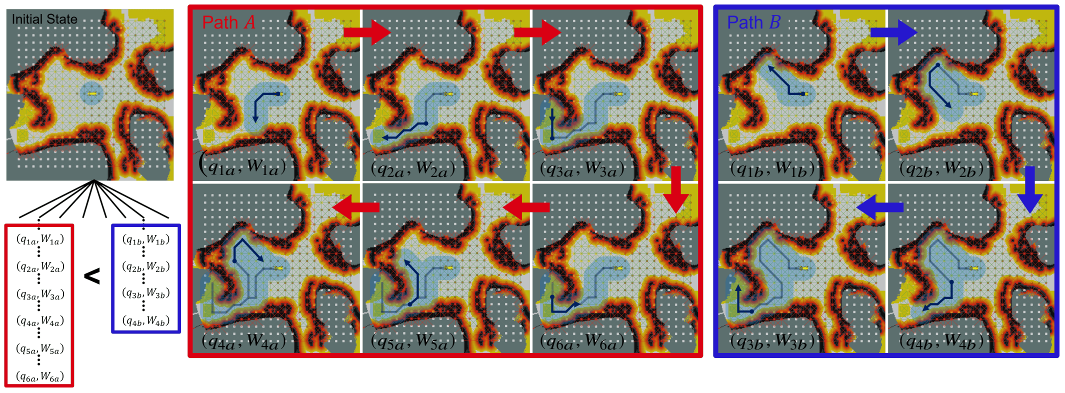


Figure 3.6: Illustrative example of coverage path planning on the Local IRM with Monte-Carlo Tree Search. The field-of-view of the robot's coverage sensor is represented by a blue circle. Macro actions (6 steps on Local IRM in this example) associated with the two tree branches, paths A and B, are shown. Note that the final world coverage states in both branches are identical. Path A is evaluated to be more rewarding than B since fewer actions were required to cover the same area.

Local Coverage Planner (LCP)

In order to solve Eq. (3.4), we employ POMCP (Partially Observable Monte Carlo Planning) algorithm [47]. POMCP is a widely-adopted POMDP solver that leverages the Monte Carlo sampling technique to alleviate both of the *curse of dimensionality* and *history*. Given a generative model (or a black box simulator) for discrete action and observation spaces, POMCP can learn the value function of the reachable belief subspace with an adequate exploration-exploitation trade-off.

More concretely, POMCP evaluates $Q^{\ell}(b^{\ell}, a^{\ell})$ in Eq. (3.7) by unrolling recursive value backpropagation through sampled action-observation sequences. The UCT algorithm for action selection helps to balance between exploration and exploitation in order to learn the action-value function [41]. Initially, it explores the search space (possible action-observation sequences) with a random or a heuristically guided rollout policy. While incrementally building the belief tree, it gradually exploits the learned values for more focused exploration. See the illustration of local coverage planning in Fig. 3.6.

3.9 Receding-Horizon Policy Reconciliation

We extend the receding-horizon *local* coverage planning problem to address the trade-off between policy consistency and resiliency, as described in Sec. 3.4.

We define a policy reconciliation optimization problem by introducing the previous planning episode policy into Eq. (3.2) for the current planning episode. For notational brevity, let us denote the time when the previous policy was generated as t_0 and the current time as $t_1 = t_0 + \Delta t$. In order to reconcile consecutive policies over receding horizons, we extend Eq. (3.2) as follows, given the previous policy $\pi^-_{t_0:t_0+T}$ constructed at time t_0 for a finite horizon of T:

$$\pi^*_{t_1:t_1+T}(b; \pi^-_{t_0:t_0+T}) = \underset{\pi \in \Pi_{t_1:t_1+T}}{\operatorname{argmax}} \left[\mathbb{E} \sum_{t'=t_1}^{t_1+T} \gamma^{t'-t_1} r(b_{t'}, \pi(b_{t'})) - \lambda \mathcal{R}(\pi^-_{t_0:t_0+T}, \pi_{t_1:t_1+T}) \right], \tag{3.15}$$

where $\mathcal{R}(\pi^-_{t_0:t_0+T}, \pi_{t_1:t_1+T})$ is a regularizing cost function that penalizes inconsistency between the previous and current policies in terms of kinodynamic constraints, and λ is a regularization weight parameter. The first term in Eq. (3.15) pursues policy resiliency based on the up-to-date world belief, which may encode unexpected hazards, while the second term promotes policy consistency.

Since the conflict between policy consistency and resiliency is most severe at the junction between two consecutive policies, we decompose Eq. (3.15) into two time frames, $(t_1 : t_1 + \tau)$ and $(t_1 + \tau : t_1 + T)$ for $\tau \in [0, T - \Delta t]$, and formulate it as a simplified joint optimization problem for τ^* and $\pi^*_{t_1+\tau^*:t_1+T}$:

$$\tau^* = \underset{\tau \in [0, T-\Delta t]}{\operatorname{argmax}} \mathbb{E} \sum_{t'=t_1}^{t_1+\tau} \gamma^{t'-t_1} r(b_{t'}, \pi^-_{t_0:t_0+T}(b_{t'})), \tag{3.16}$$

$$\pi^*_{t_1+\tau^*:t_1+T} = \underset{\pi \in \Pi_{t_1+\tau^*:t_1+T}}{\operatorname{argmax}} \mathbb{E} \sum_{t'=t_1+\tau^*}^{t_1+T} \gamma^{t'-t_1} r(b_{t'}, \pi(b_{t'})), \tag{3.17}$$

$$\pi^*_{t_1:t_1+T} = [\pi^-_{t_1:t_1+\tau^*}; \pi^*_{t_1+\tau^*:t+T}]. \tag{3.18}$$

Policy reconciliation is performed in Eq. (3.16) over a single optimization variable τ. By re-evaluating the previous policy $\pi^-_{t_0:t_0+T}$ with updated robot-world belief $b_{t'}$, τ dictates how much of the new $\pi^*_{t_1:t_1+T}$ should be in agreement with the previous policy. Effectively, a larger τ promotes policy consistency, while a smaller τ promotes policy resiliency. See Fig. 3.12.

Given τ^* from Eq. (3.16), the optimization problem in Eq. (3.17) becomes identical to Eq. (3.2), except the change of start time, and can be solved by LCP, as described in Sec. 3.8. The final receding-horizon policy $\pi^*_{t_1:t_1+T}$ is then constructed by concatenating the previous policy and a new partial policy, as in Eq. (3.17).

3.10 Experimental Results

In order to evaluate our proposed framework, we perform high-fidelity simulation studies with a four-wheeled vehicle (Husky robot) and real-world experiments with a quadruped (Boston Dynamics Spot robot). Both robots are equipped with custom sensing and computing systems, enabling high levels of autonomy and communication capabilities [4, 71]. The entire autonomy stack runs in real-time on an Intel Core i7 processor with 32 GB of RAM. The stack relies on a multi-sensor fusion framework. The core of this framework is 3D point cloud data provided by LiDAR range sensors mounted on the robots [6]. We refer to our autonomy stack-integrated Spot as Au-Spot [5].

Baseline Algorithms

We compare our PLGRIM framework against a local coverage planner baseline (next-best-view method) and a global coverage planner baseline (frontier-based method).

1) *Next-Best-View (NBV):* NBV first samples viewpoints in a neighborhood of the robot, and then plans a deterministic path over a high-fidelity local world representation to each viewpoint [69]. The set of viewpoint paths serves as the policy search space. Each policy in the space is evaluated, and NBV selects the policy with the maximum reward, computed using action cost and information gain from the world representation. While NBV is able to leverage local high-fidelity information, it suffers due to its spatially limited world belief and sparse policy space.

2) *Hierarchical Frontier-based Exploration (HFE)*: Frontier based exploration methods construct a global, but low-fidelity, representation of the world, where frontiers encode approximate local information gain. The set of frontiers serves as the policy search space. Exploration interleaves a one-step look-ahead frontier selection and the creation of new frontiers, until all frontiers have been explored. Hierarchical approaches can enhance the performance of frontier-based methods by modulating the spatial scope of frontier selection [32]. However, while HFE is able to reason across the global world belief, it suffers from downsampling artifacts and a sparse policy space composed of large action steps, i.e., edges on the global IRM.

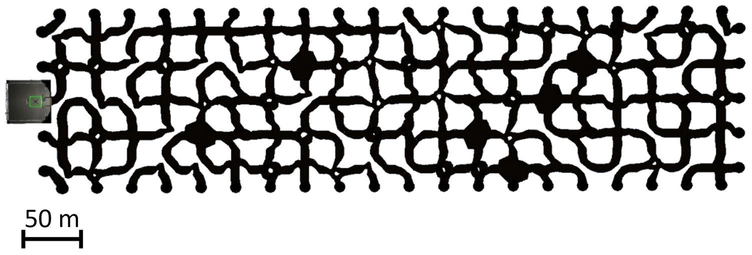
Note that in order to achieve reasonable performance in the complex simulated environments, we allow each baseline to leverage our Local and Global IRM structures as the underlying search space.

Simulation Evaluation

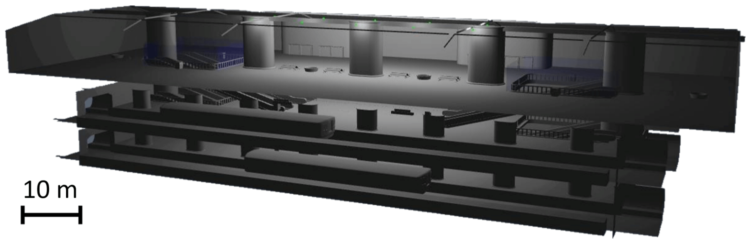
We demonstrate PLGRIM's performance, as well as that of the baseline algorithms, in a simulated subway, maze, and cave environment. Fig. 3.7 visualizes these environments.

Simulated Subway Station: The subway station consists of large interconnected, polygonal rooms with smooth floors, devoid of obstacles. There are three varying sized subway environments, whose scales are denoted by 1x, 2x, and 3x. Fig. 3.11(a)-(c) shows the scalable performance of PLGRIM against the baselines. In a relatively small environment without complex features (Subway 1x), NBV performance is competitive as it evaluates high-resolution paths based on information gain. However, as the environment scale grows, its myopic planning easily gets *stuck* and the robot's coverage rate drops significantly. HFE shows inconsistent performance in the subway environments. The accumulation of locally suboptimal decisions, due to its sparse environment representation, leads to the construction of a globally inefficient IRM structure. As a result, the robot must perform time-consuming detours in order to *pick up* leftover frontiers.

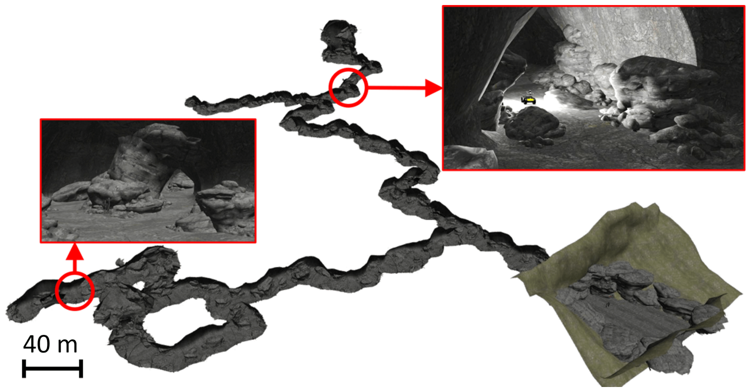
Simulated Maze and Cave: The maze and cave are both unstructured environments with complex terrain (rocks, steep slopes, etc.) and topology (narrow passages, sharp bends, dead-ends, open-spaces, etc.). The coverage rates for each algorithm are displayed in Fig. 3.11(d)-(e). PLGRIM outperforms the baseline methods in these environments. By constructing long-horizon coverage paths over a high-resolution world belief representation, PLGRIM enables the robot to safely explore through hazardous terrain. Simultaneously, it maintains an understanding of the global world, which is leveraged when deciding where to explore next after exhausting all local information. In the cave, NBV's reliance on a deterministic path, without consideration of probabilistic risk, causes the robot to drive into a pile of rocks and become inoperable. NBV exhibits similarly poor performance in the maze. However, in this case, NBV's myopic planning is particularly ineffectual when faced with navigating a topologically-complex space, and the robot ultimately gets *stuck*. As was the case in the subway, HFE suffers in the topologically-complex maze due to an accumulation of suboptimal local decisions. In particular, frontiers are sometimes not detected in the sharp bends of the maze, leaving the robot with an empty local policy space. As a result, the robot cannot progress and spends considerable time backtracking along the IRM to distant frontiers.



(a) **Simulated Maze** (top-down view): large irregular network of narrow passages with sharp bends and dead-ends.



(b) **Simulated Subway** (1x scale): large interconnected, polygonal rooms with smooth floors, devoid of obstacles.



(c) **Simulated Cave**: large irregular structure with complex terrain, consisting of rocks, steep slopes, and narrow passages.

Figure 3.7: The performances of PLGRIM and the baseline algorithms were evaluated in various simulated environments.

Real-World Evaluation

We extensively validated PLGRIM on physical robots in real-world environments. In particular, PLGRIM was run on Au-Spot in a lava tube, located in Lava Beds National Monument, Tulelake, CA. The cave consists of a main tube, which branches into smaller, auxiliary tubes. The floor is characterized by ropy masses of cooled lava. Large boulders, from ceiling breakdown, are scattered throughout the tube. Fig. 3.10 shows the robot's trajectory overlaid on the aggregated LiDAR point cloud. Fig. 3.8 and 3.9 discus how PLGRIM is able to overcome the challenges posed by large-scale environments with complex terrain and efficiently guide the robot's exploration. Fig. 3.11(f) shows the area covered over time.

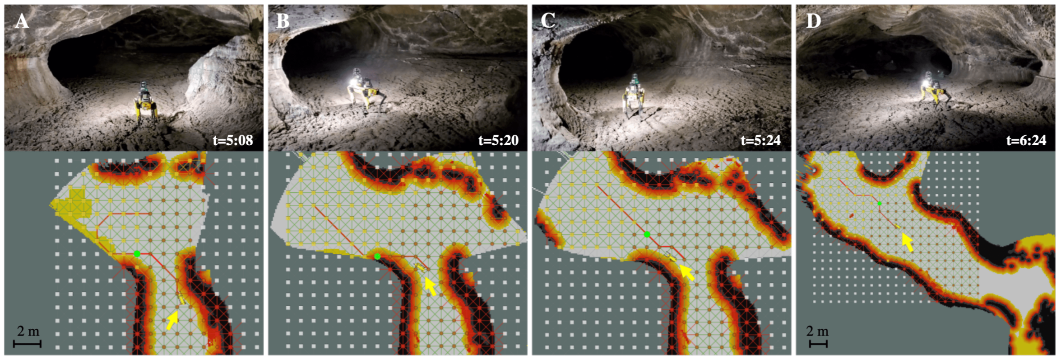


Figure 3.8: The Local IRM (yellow, brown, and white nodes represent uncovered, covered and unknown areas, respectively) is shown overlaid on the Risk Map. A yellow arrow indicates the robot's location. LCP plans a path (red) that fully covers the local area (snapshot A). When $p(W^{\ell})$ updates, the path is adjusted to extend towards the large uncovered swath while maintaining smoothness with the previous path. Another $p(W^{\ell})$ update reveals that the path has entered a hazardous area—wall of lava tube (snapshot B). As a demonstration of LCP's resiliency, the path shifts away from the hazardous area, and the robot is re-directed towards the center of the tube (snapshot C). One minute later, the robot encounters a fork in the cave. The LCP path curves slightly toward fork apex (for maximal information gain) before entering the wider, less-risky channel (snapshot D).

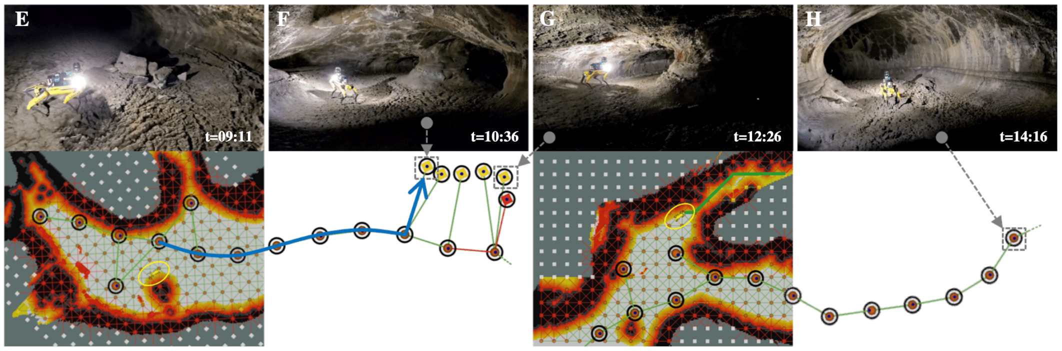


Figure 3.9: Portions of the Global IRM constructed in the lava tube are visualized–yellow nodes represent frontiers, brown nodes represent breadcrumbs. Gray arrows associate a frontier with a snapshot of the robot exploring that frontier. GCP plans a path (blue) along the Global IRM to a target frontier after the local area is fully covered (snapshot E). The robot explores the area around the frontier (snapshot F), and then explores a neighboring frontier at the opening of a narrow channel to its right. LCP plans a path (green) into the channel (snapshot G). Later, after all local areas have been explored, the robot is guided back towards the mouth of cave along the breadcrumb nodes (snapshot H).

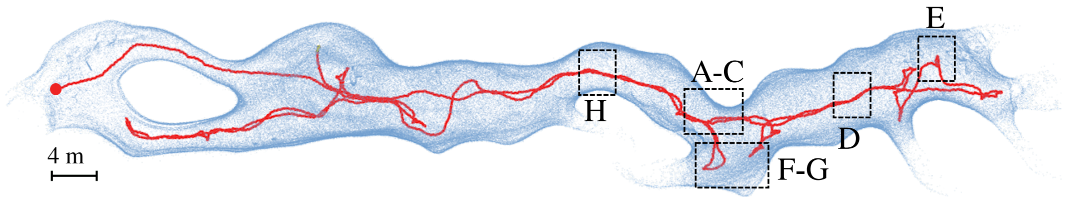


Figure 3.10: PLGRIM's exploration trajectory in Valentine Cave, Lava Beds National Monument, Tulelake, CA. Exploration started at the mouth of the cave (red circle), reached the end of the cave on the right, and returned back to visit uncovered areas. Boxes indicate the portions of the trajectory associated with the alphabetized snapshots in Fig. 3.8 and Fig. 3.9.

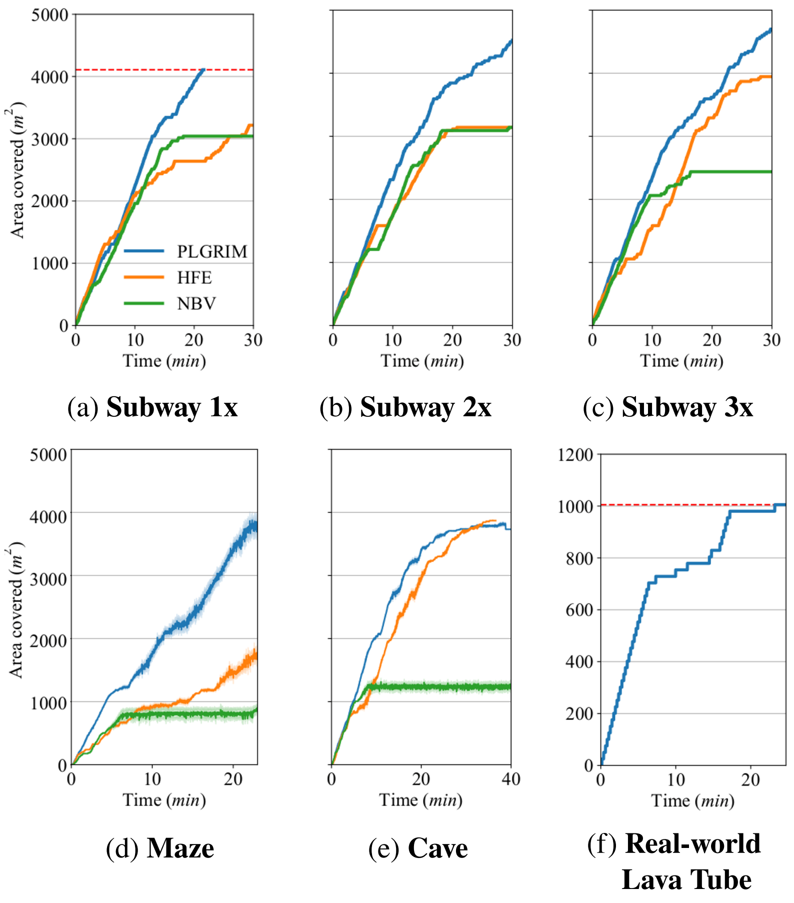


Figure 3.11: Exploration by PLGRIM and baseline methods in simulated subway environments of increasing size (a)-(c), simulated maze (d), simulated cave (e), and real-world lava tube (f). For (d) and (e), the covered area is the average of two runs. Red dashed lines indicate 100% coverage of the environments, where applicable. In (f), due to the extreme terrain conditions of the lava tube, we restricted Au-Spot's maximum speed to be 0.5 m/s, or half of the manufacturer-specified maximum speed, due to the extremely rough terrain conditions in the lava tube.

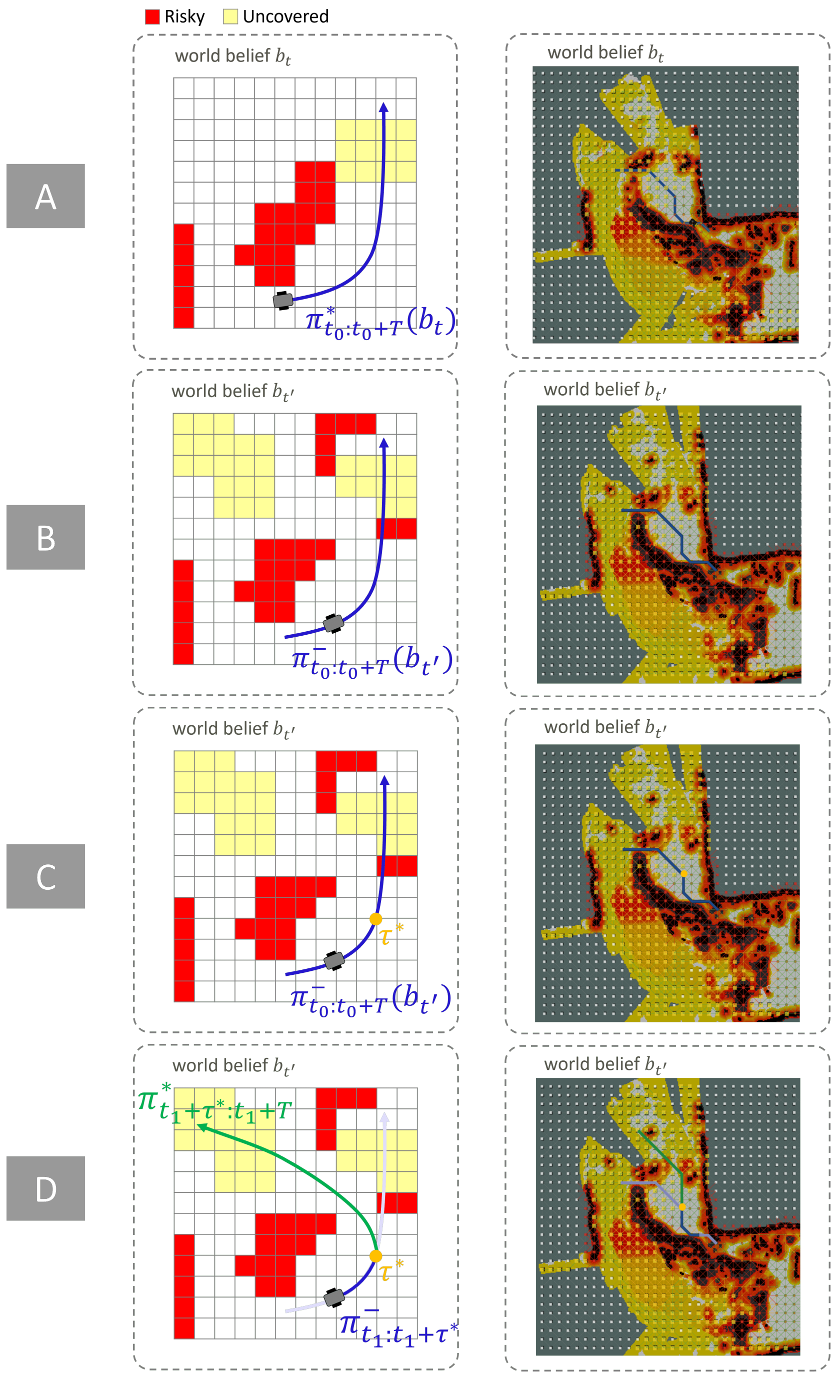


Figure 3.12: Receding-Horizon Policy Reconciliation process for optimal root node $\tau*$ determination (Sec. 3.9). The right column is the corresponding step from Husky's exploration of a real-world limestone mine. Snapshot A is in world belief state b_t, and snapshots B – D are in the next belief state $b_{t'}$. By re-evaluating the previous policy $\pi^-_{t_0:t_0+T}$ with updated robot-world belief $b_{t'}$ (snapshot B), τ dictates how much of the new $\pi^*_{t_1:t_1+T}$ should be in agreement with the previous policy (snapshot C). The final receding-horizon policy $\pi^*_{t_1:t_1+T}$ is then constructed by concatenating the previous policy and a new partial policy (snapshot D).

3.11 Summary

This chapter presented a hierarchical framework for exploring large-scale, unknown environments with complex terrain in a POMDP setting. To obtain a tractable solution, we introduced a hierarchical belief space representation that effectively encodes a large-scale world state, while simultaneously capturing high-fidelity information local to the robot. Then we proposed cascaded POMDP solvers that reason over long horizons within a suitable replanning time. We demonstrated our framework in high-fidelity dynamic simulation environments and in real-world environments. Chaps. 4 and 5 discus alternative local and global coverage planners, which rely on the same underlying Information Roadmap structures and interact according to the PLGRIM framework.

Chapter 4

LOCAL COVERAGE PLANNER

This chapter presents a method for solving the time-limited coverage problem over a local grid-based representation of the environment, with the objective being autonomous exploration of an unknown environment. Here, the robot is tasked with planning a path over a local horizon such that the accumulated area swept out by its sensor footprint is maximized. Because this problem exhibits a *diminishing returns* property known as submodularity, we choose to formulate it as a tree-based sequential decision making process. This formulation allows us to evaluate how the robot's observation of the world affects the utility of future observations, while simultaneously accounting for traversability risk and the robot's dynamic constraints. In order to adequately investigate the search space, we reduce computation using an effective approximation to the coverage sensor model which adapts the coverage range to the local environment, as shown in Fig. 4.1. As a result, we can solve the submodular coverage problem in a unified manner, which we contend is more robust to real-world uncertainty than widely-adopted decoupled methods.

This chapter was adapted from:

> **A. Bouman**, J. Ott, S. Kim, K. Chen, M. Kochenderfer, B. Lopez, A. Agha-Mohammadi, and J. Burdick. "Adaptive Coverage Path Planning for Efficient Exploration of Unknown Environments". In: *Submitted to IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022.

4.1 Introduction

Consider a time-limited mission wherein a ground robot must autonomously explore an unknown environment with complex terrain. The robot explores by maximizing the area observed, or *covered*, by a task-specific *coverage sensor*. This sensor may be a thermal camera for detecting thermal signatures, an optical camera for identifying visual clues, or in our case, an omnidirectional range finder for constructing 3D environment maps. As the robot moves, the sensor footprint sweeps the environment, expanding the covered area, or more generally, the task-relevant information about the world. The problem of finding efficient and safe coverage trajectories is computationally complex [29, 53] – one must consider the fact that a robot's observation of the world affects the utility of future observations, while concurrently minimizing traversability risk.

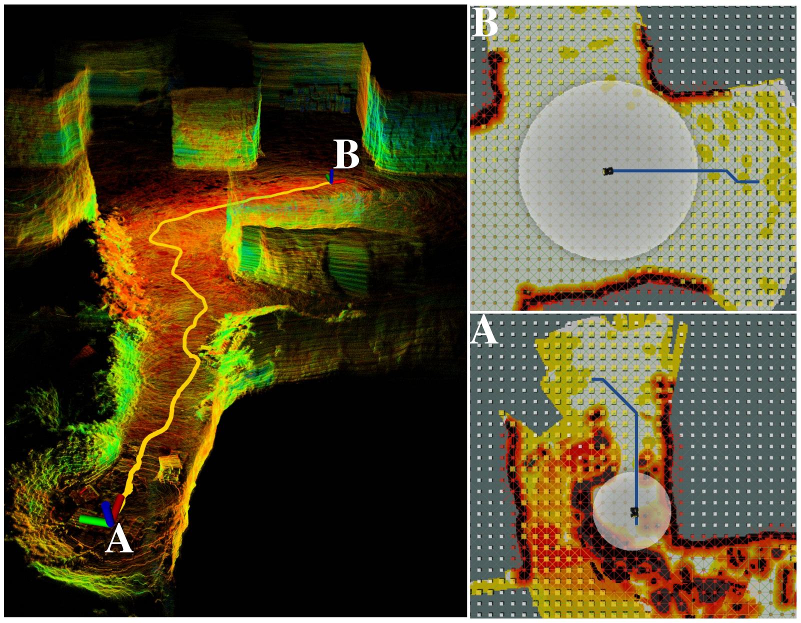


Figure 4.1: Adaptive coverage range (translucent circle) and coverage path from single planning episode (blue) in a locally confined (A) and spacious area (B) during Husky's autonomous exploration of a limestone mine in Nicholasville, KY. The robot's trajectory (yellow) is overlaid on pointcloud (left figure).

Our proposed method quickly finds non-myopic coverage paths by rolling out future coverage observations using an effective sensor model. Our model is carefully designed to replicate critical features of a range finder in a computationally efficient manner. First, the model is probabilistic – coverage probability decreases with increasing ray sparsity along the radial direction. As a result, the density of coverage is dictated by the local environment geometry, and large topological features in the environment are quickly exposed and mapped. Second, to account for ray-surface interactions that regulate surface visibility, the coverage range, or distance at which a sensor measurement is performed, adapts to the scale of the local environment (Fig. 4.1). This approach obviates the need for expensive ray-tracing operations that make forward rollout algorithms prohibitively slow for a real-time system.

We begin by noting that the time-limited coverage task is a submodular orienteering problem. Since the robot must understand the effects of its actions on the quality of future coverage measurements, we choose to formulate this problem as a sequential decision process. To find near-optimal trajectories at high replanning rates, we use an online forward rollout search algorithm that plans from the current world-robot state to a travel budget-defined horizon. Our method was evaluated on hardware in various environments, and served as the local planner for team CoSTAR's entry in the Final Circuit of the DARPA SubT Challenge [4].

4.2 Related Work

The problem of finding the optimal sequence of sensing actions, or viewpoints, in order to maximize some task-specific information has been extensively studied, both in computer vision and robotics. In the robotics field, the problem of viewpoint selection is commonly motivated by tasks such as surveillance, object inspection, and exploration. While a variety of viewpoint selection algorithms have been proposed, we address those used to solve the exploration problem where policies are constructed in a receding horizon fashion as the robot gathers more sensory information about its environment.

Viewpoint selection algorithms employ a sensor model to determine future sensing locations that maximize scene information. In the context of exploration, these schemes often rely on identification of the boundary between unmapped and mapped space, regions termed *frontiers*, and seek new robot poses that extend the boundary of mapped space [28]. Traditional frontier-based approaches construct one-step lookahead policies that find the next most favorable sensing action, the quality of which is determined by the amount of unmapped area that can be visualized [28], [54]. Underpinning many approaches is the next-best-view planner (NBV) [69], where a rapidly exploring random tree is constructed. Each vertex represents a viewpoint, and the vertex that maximizes a utility function, weighing volumetric gain against path distance, is greedily selected as the next goal [72], [73]. While computationally efficient, NBV-based planners are greedy and therefore susceptible to local minima, leading to suboptimal decision making. An accumulation of suboptimal local decisions can significantly reduce the amount of sensor information gathered over time.

In order to optimize viewpoint selection over a multi-step horizon, the exploration problem has been framed as a variant of the *art gallery* problem [74]. Here the objective is to find a minimal set of viewpoints that maximizes coverage of an area. A critical feature of this problem is the fact that the marginal benefit of selecting a new viewpoint decreases as the set of already selected viewpoints increases – a property known as *submodularity*. A greedy algorithm has been shown to provide a good approximation of the optimal solution to the submodular function maximization problem [75].

Leveraging the effectiveness of greedy methods for submodular maximization, many have adopted a decoupled approach to the exploration problem [76], [53], [24]. First, sensing locations are selected using a greedy algorithm. Then a path through the sensing locations is determined. For instance, in the work of Cao et al. [76], a set of viewpoints is first sampled from a grid-based environment representation. Then viewpoints are selected in the order of marginal coverage reward. To account for submodularity, the coverage rewards of the remaining viewpoints in the set are recomputed after each selection. The final ordering of viewpoints is determined by solving the standard *traveling salesman problem* [77]. While a decoupled approach provides a non-myopic solution in a computationally efficient manner, we contend that it can be sensitive to model uncertainty, which we discuss in Section 4.7.

The main contribution of this chapter is a unified exploration planning algorithm for finding an optimal sequence of sensing actions. We propose a reliable, yet computationally efficient, sensor model that allows us to leverage a rollout-based search algorithm in order to evaluate the submodular coverage problem without needing to approximate it as a modular problem. As a result, we can find near-optimal coverage paths that are robust to model uncertainty.

4.3 Problem Definition

The coverage problem considered here has been abstractly formulated as the *shortest covering path problem* (SCPP) where the objective is to identify the shortest path that covers all nodes in a priori known graph geometry [78]. A node is considered covered if it is within a pre-defined distance from another node in the path. By adding a time constraint and removing the pre-specification of a terminal node, we generalize the SCPP to a time-limited coverage problem using an orienteering problem (OP) framework.

Given a known environment represented by an abstract graph structure $G = (N, E)$, with free and occupied nodes $N_{free} \cup N_{occ} = N$, our OP objective is to find an unbounded sequence of nodes $p = \{n_0, ..., n_K\} \subseteq N_{free}$ such that the number of free nodes within an accumulated coverage sensor footprint F is maximized, subject to a budget constraint:

$$\begin{aligned} p^* = \operatorname*{argmax}_{p} \sum_{n_i \in p} F(n_i), \\ \text{subject to} \quad a(p) \leq a_{\max}, \end{aligned} \tag{4.1}$$

where $a(p)$ is the path action cost, $a_{\max}$ is a user-defined action cost budget, and the footprint F maps each node to a set of "covered" nodes: $F(n_i) = (n_{i_1}, n_{i_2}, .., n_{i_j})$.

Recall that the coverage problem exhibits submodularity; that is, the marginal benefit of appending the path with a node n_2 "close" to $n_1 \in p$ is less than that if $n_1 \notin p$. To account for this *diminishing returns* property, we re-define the OP reward function to be the marginal coverage given all the previously traversed nodes:

$$\tilde{F}(n_i) = F(n_i \mid n_0, .., n_{i-1}). \tag{4.2}$$

Then we can define the coverage OP as:

$$\begin{aligned} p^* = \underset{p}{\operatorname{argmax}} & \sum_{n_i \in p} \tilde{F}(n_i), \\ \text{subject to} \quad & a(p) \le a_{\max}. \end{aligned} \tag{4.3}$$

For the remainder of this chapter, we refer to Eq. 4.3 as our coverage problem.

4.4 Methodology Overview

We model the time-limited coverage problem as a discrete-time sequential decision making process where the optimal policy is a sequence of actions chosen to maximize a cumulative coverage reward. To find near-optimal policies in real-time, we employ a rollout-based search algorithm that estimates the value of an action sequence by simulating interactions between the robot and world. During a simulated episode, or rollout, the robot and world states evolve *together* – the robot executes an action and makes a coverage observation of its environment, Eq. (4.2), which yields a subsequent robot-world state and reward. Thus, rollouts provide a method of solving the submodular coverage problem in a unified manner, i.e. a policy is evaluated on both the accumulated marginal coverage reward and the path cost.

We introduce our world representation (Sec. 4.5), and then model the coverage problem as a Markov decision process with the goal of formulating an online solvable submodular optimization problem (Sec. 4.6). To solve this problem in real-time on a computationally-constrained robot, we propose an effective approximation to the coverage sensor model, which significantly reduces rollout computation. As a result, we are able to construct high-quality coverage paths at a high planning rate (Sec. 4.7).

4.5 World Representation

We represent the local environment around the robot by the Local Information Roadmap (IRM), introduced in Chap. 3.7. An annotated view of the Local IRM is shown in Fig. 4.2. The Local IRM is a fixed-size lattice graph $G = (N, E)$ with nodes N and edges E. Nodes represent discrete areas in space, and edges represent actions. We store two type of information in the IRM: *(i)* the traversability risk of the world with respect to the robot's dynamic constraints, and *(ii)* what parts of the environment have been observed, or *covered*, by a task-specific coverage sensor. The robot-centered, rolling window Local IRM is continuously updated with traversability and coverage information based on incoming sensor data.

To construct G, we uniformly sample nodes $n_i \in N$ in a neighborhood of the robot, and compute the traversability risk and coverage probability distribution over a discrete patch centered at each node, i.e., $p_r(n_i)$ and $p_c(n_i)$, which are stored as node properties. For scalability, we bin node traversability risk probabilities into three groups: *occupied* $p_r(n_i) = 1$, *unknown* $p_r(n_i) = 0.5$, and *free* $p_r(n_i) = 0$. For an edge $e_{ij} \in E$, we compute and store the traversal distance d_{ij} and traversal risk ρ_{ij} between two connected nodes.

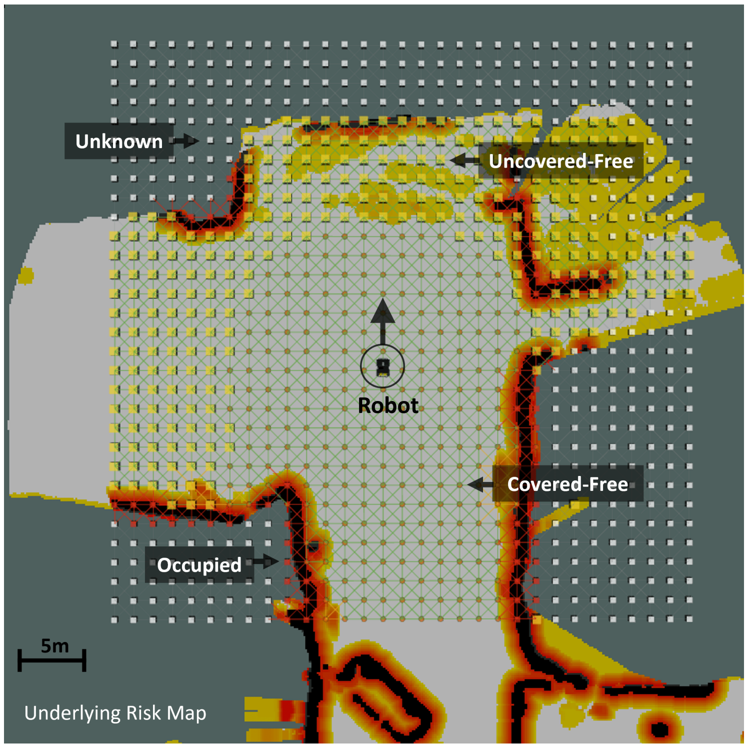


Figure 4.2: Local Information Roadmap (IRM) shown overlaid on the Risk Map. The IRM contains world coverage and traversability risk information. The goal of the coverage planner is to construct paths on the IRM that convert nodes from uncovered traversable (yellow) to covered traversable (brown). By constructing coverage paths in a receding-horizon fashion, the robot extends the boundaries of explored space.

4.6 Markov Decision Process

A Markov decision process (MDP) is described as a tuple $\langle \mathbb{S}, \mathbb{A}, T, R \rangle$, where $\mathbb{S}$ is the set of joint robot-and-world states, and $\mathbb{A}$ is the set of robot actions. The motion model $T(s, a, s') = p(s' \mid s, a)$ defines the probability of being in state s' after taking action a in state s, and the reward function $R(s, a)$ returns the utility for executing action a in state s. The objective is to find a mapping from states to actions, i.e. the policy π, that maximizes the expected sum of future reward.

State

The robot-world state is defined as $s = (q, W)$, where q is the robot state and W is the world state. We define q and W in terms of the Local IRM. The robot state $q = (n_q, \mu)$, where n_q is the node closest to the robot's current location, and μ is the robot's heading direction, defined with respect to the lattice geometry. The world state is $W = G$, where G is the IRM containing traversability risk and coverage world state estimates.

We define an action a as the controlled robot traversal from node $n_i \in N$ to neighboring node $n_j \in N$, along an edge $e_{ij} \in E$. A node is directly connected to its eight neighbors, discretizing the valid action space for a single state into movement along the four cardinal/non-diagonal (N, E, S, W) and four intercardinal/diagonal (NE, SE, SW, NW) directions. We denote actions along the cardinal and intercardinal directions by $a_{\sqrt{2}}$ and a_1, respectively.

Robot Dynamics

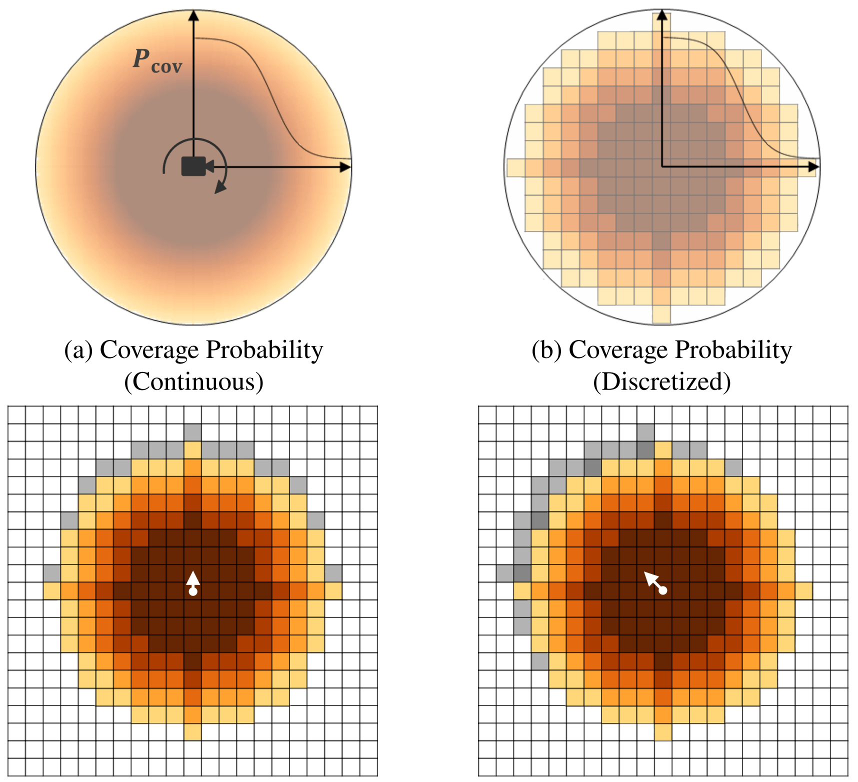
We approximate the robot motion model $T(q, a, q')$ as deterministic. Given an action a directing traversal of edge e_{ij}, the robot will reach node n_j with probability 1. Actions that cause the robot to leave the bounds of G or enter nodes that are unknown or occupied, $p_r(n_i) = 0.5$ or 1, have no effect. Note that while we do not explicitly model motion stochasticity, we manage the effects of such uncertainty by planning at a high-rate in a receding-horizon fashion.

Probabilistic Coverage Sensor Model

We model our coverage sensor as an omnidirectional range finder. The robot covers nodes within its line-of-sight, computed using ray-tracing techniques on the traversal risk map $\{p_r(n_i)\}$ in combination with sensor range constraints. To account for increasing ray sparsity in the radial direction, we compute the coverage probability for a node as a function of the robot-to-node distance. Given the robot node n_q, a node n_i is covered with probability $P_{\text{cov}}(n_i|\ n_q)$. We heuristically model the coverage probability P_{cov} as an S-shaped logistic function:

$$P_{\text{cov}}(n_i|\ n_q) = \frac{1}{1 + e^{k\,(r_i - r_0)}}, \tag{4.4}$$

where r_i is the euclidean distance between the robot node n_q and node n_i, and constants r_0 and k are the sigmoid's midpoint and steepness, respectively. The coverage probability distribution over the radial distance from the center of the sensor in shown in Fig. 4.3. The probabilistic sensor model plays a crucial role in efficiently exploring an environment when the fundamental objective is to construct a map of the environment. By modulating coverage density based on the local geometry, the probabilistic model can quickly expose large topological features in the environment, as detailed in Fig. 4.5.



(a) Coverage Probability (Continuous)

(b) Coverage Probability (Discretized)

(c) Non-Diagonal Action a_1

(d) Diagonal Action $a_{\sqrt{2}}$

Figure 4.3: Our coverage sensor model, based on Eq. (4.4), displayed over continuous space (a), and over the discretized lattice graph world representation (b). The diffused color map mimics the coverage probability curve– darker shades indicate higher coverage probabilities. The marginal coverage after a non-diagonal action (c) and diagonal action (d) is represented by the shaded gray cells. Note that the ratio of marginal coverage to distance traveled over the lattice is not equivalent for non-diagonal and diagonal actions: $I(s^o, a_{\sqrt{2}})/d_{ij} \neq I(s^o, a_1)/d_{ij}$, where s^o indicates a risk-free world. We address this discrepancy with Eq. (4.7).

Algorithm 2 World Coverage Update

```
Function CoverageUpdate
Input: robot node $n_q$
       world state $G$
       maximum sensor range $r_{\max}$
 1: for all angles $\theta_k$ of the sensor's angular displacement do
 2:     for all nodes $n_i$ along ray from $n_q$ in direction $\theta_k$ do
 3:         Compute robot-to-node euclidean distance $r_i$
 4:         if $p_r(n_i) < \rho_{\max}$ and $r_i < r_{\max}$ then
 5:             $p_c(n_i)' \leftarrow \max\big[p_c(n_i), P_{\text{cov}}(n_i|\ n_q)\big]$  ▷ Eq. (4.4)
 6:         else
 7:             break
 8:         end if
 9:     end for
10: end for
11: return $\{p_c(n_i)'\}$
```

World Transition Model

We approximate the world transition function $T(W, a, W')$ as deterministic. Function CoverageUpdate in Alg. 2 presents the process for updating the world coverage state based upon the the probabilistic coverage sensor model in Eq. (4.4). When integrating new sensor measurements, we assume independence and compute the maximum of the old and new coverage probability (Alg. 2-line 5). This yields an optimistic estimate of coverage.

Reward Function

We now redefine our marginal coverage from Eq. (4.2) to be the uncertainty reduction in the world coverage state induced by an action a:

$$I(s, a) = \sum_{n_i \in N} \beta\Big(p_c(n_i \,|\, a) - p_c(n_i)\Big), \tag{4.5}$$

where β controls the reward received from covering a node based on its occupancy status. Due to its sparsity, the Local IRM sometimes fails to identify nodes as occupied in high risk regions. For instance in Fig. 4.2, the environment boundary, indicated in the underlying Risk Map, is not fully represented by occupied nodes in the Local IRM. To stay robust to this unreliable world model, we define the value of β to be larger for nodes of known occupancy (occupied, uncovered-free, and covered-free), when compared to the value of β for unknown nodes. As a result, the constructed coverage paths are more likely to stay within the traversable space of the environment.

The reward function is defined as a weighted sum of marginal coverage and action penalties:

$$R(s,a) = k_I\, I(s,a) - \left[k_d\, d_{ij} + k_\rho\, \rho_{ij} + k_\mu\, \Delta_\mu\right], \tag{4.6}$$

where d_{ij} is the traversal distance, ρ_{ij} traversal risk, and Δ_μ is the cost of rotation due to the robot's non-holonomic constraints. Constants k_I, k_d, k_ρ, and k_μ weigh the importance of coverage, traversal distance, risk, and motion primitive history on the total reward.

Given a coverage sensor with a circular field-of-view, the uncovered area after a diagonal and non-diagonal action should scale equivalently with distance traveled. However, since Eq. (4.5) is evaluated over a discretized space G, the ratio of marginal coverage to distance traveled is not equivalent for all actions on the lattice, as illustrated in Fig. 4.3. Given this marginal coverage discrepancy between actions, we define k_d as a function of coverage parameters in order to ensure non-diagonal (a_1) and diagonal actions ($a_{\sqrt{2}}$) are equally rewarding; that is, $R(s,a_1) = R(s,a_{\sqrt{2}})$ for the same ρ_{ij} and Δ_μ. If w is the width of a grid cell in G, then we define k_d as:

$$k_d = \frac{k_I}{w} \cdot \frac{I(s^o, a_{\sqrt{2}}) - I(s^o, a_1)}{(1-\sqrt{2})} \tag{4.7}$$

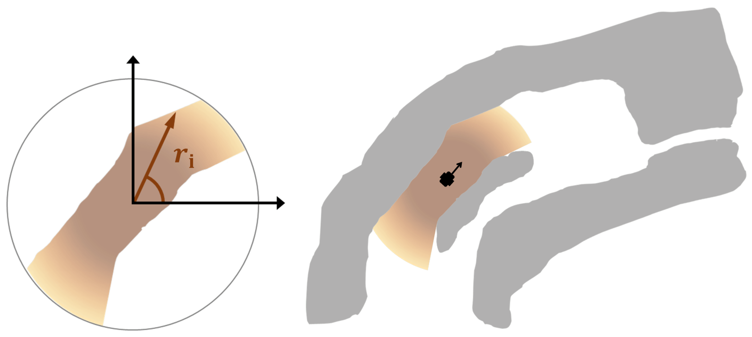
where state s^o denotes a risk-free world where the only covered region is aligned with the robot's current sensor footprint.

Optimal Policy

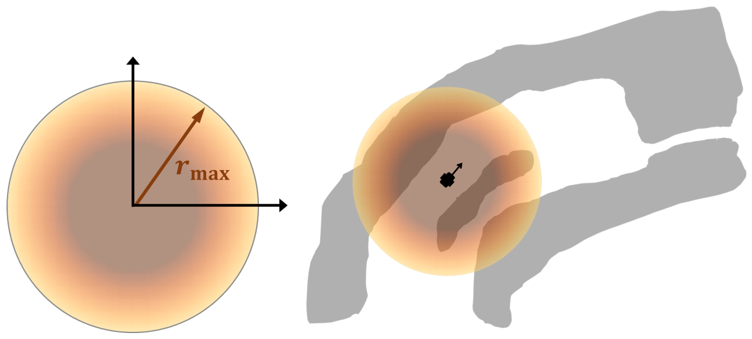
It is fundamentally infeasible to solve an unknown environment coverage problem over an infinite horizon since information about the world is incomplete, and often inaccurate, at runtime. Instead, in such domains, a Receding Horizon Planning (RHP) scheme has been widely adopted as the state-of-the-art [69]. The optimal RHP policy over a horizon T is:

$$\pi^*_{t:t+T}(s) = \underset{\pi \in \Pi_{t:t+T}}{\operatorname{argmax}} \sum_{t'=t}^{t+T} \gamma^{t'-t} R(s_{t'}, \pi(s_{t'})), \tag{4.8}$$

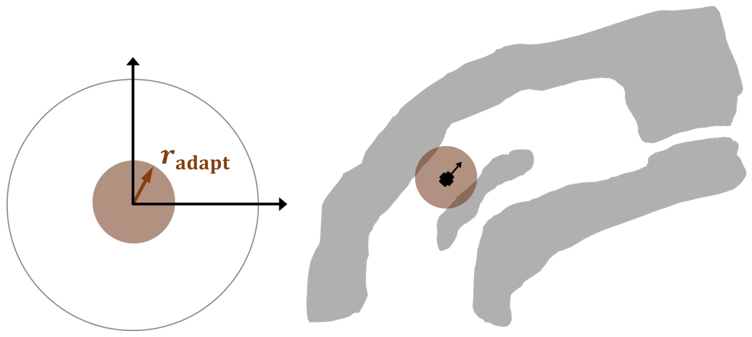
where T is a finite planning horizon for a planning episode at time t. Given the policy from the last planning episode, only a part of the optimal policy, $\pi^*_{t:t+\Delta t}$ for $\Delta t \in (0, T]$, will be executed at runtime. A new planning episode will start at time $t + \Delta t$ using an updated robot-world state.



(a) Exact Coverage Range

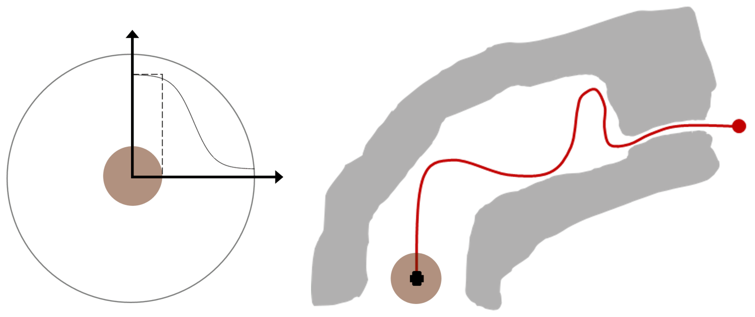


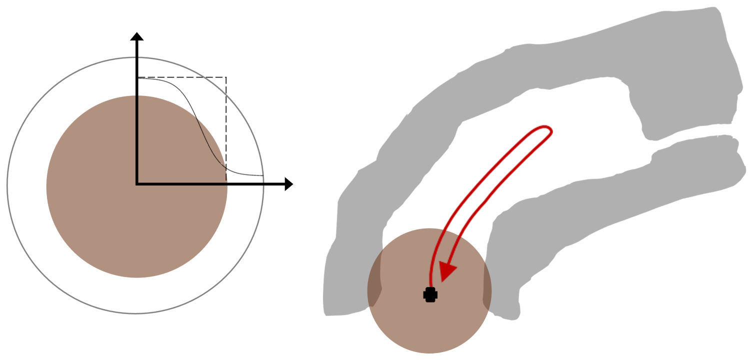
(b) Static Coverage Range



(c) Adaptive Coverage Range (Proposed)

Figure 4.4: Illustrative example of the effect of different coverage sensor models on exploration completeness: "exact" observation where the coverage range is based on ray-tracing (a), approximate observation where the coverage range is static (b), and the proposed approximate coverage sensor model where the range adapts to the local environment (c). While the exact model provides the best estimate of future coverage, it is computationally expensive and prevents proper investigation of the policy space during MCTS. Alternatively, while the static model is inexpensive, it overestimates the covered area. As a consequence, the passageway below the robot may not be explored since it provides inaccurately low coverage reward.


(a) Pessimistic Coverage Model


(b) Optimistic Coverage Model

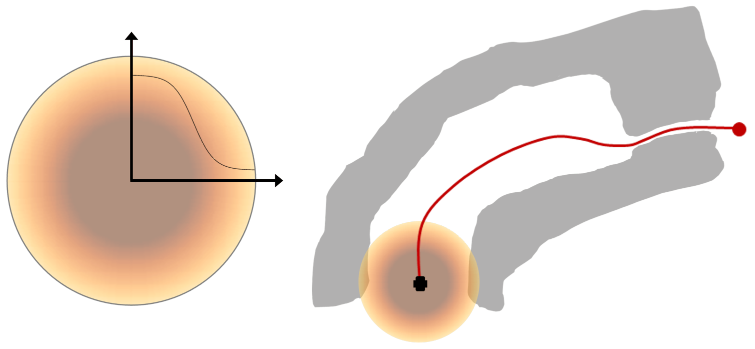

(c) Probabilistic Coverage Model (Proposed)

Figure 4.5: Illustrative example of the effect of different coverage sensor models on the constructed path's efficiency and completeness: pessimistic model where the coverage range underestimates the true range (a), optimistic model where the coverage range overestimates the true range (b), and the proposed probabilistic model (c). With the pessimistic model, the robot uncovers the narrow channel after executing an inefficient ping-pong trajectory, which poses undue risk given its occasional proximity to obstacles. Alternatively, the optimistic model generates an efficient path down the centerline of the cavern, but the robot fails to detect the narrow channel and terminates exploration early. Our proposed probabilistic model combines the benefits of pessimistic and optimistic coverage. The robot travels down the centerline, sparsely covering the main cavern until it encounters the end; at which point, it densely covers the space and ultimately exposes the narrow channel.

4.7 Online Planning

We now discus our proposed online coverage planner algorithm, which runs real-time on hardware. Alg. 3 presents the major components of the planner.

Search Algorithm

In order to solve Eq. (4.8), we use Monte Carlo tree search (MCTS) [79]. Refer to Function MCTS in Alg. 3. During every planning episode, a lookahead tree, rooted in an initial robot-world state, is iteratively constructed by simulating action sequences using a random rollout policy $\pi_{rollout}$, as illustrated in Fig. 3.6. During a single iteration, rollouts and tree expansion stop when a predefined depth, or our path budget, is reached. Given a state s and action a, a generative model $\mathcal{G}$ (i.e. the black box simulator of the MDP) provides a sample successor state s' and reward r. Since we do not have access to the ground truth state of the environment, our generative model is an estimate based on the most recent robot sensor measurements used to construct the world representation G. MCTS terminates after reaching a user-defined maximum number of simulations.

Action Sequence Extraction

The action sequence with the highest estimated value is extracted from the lookahead tree (Alg. 3 – #3). Then the first N actions from that sequence, $a^*_{1:N}$, is sent to the STEP motion planner for execution, as described in Chap. 2.2. The number of actions N is defined such that $R(s_i, a_i) > \gamma \; \forall \; i \in \{1 : N\}$, where γ is an empirically selected one-step reward lower bound. This cropping of the action sequence is critical to global exploration performance; it ensures the local coverage path uncovers “enough” area to justify the path travel cost. If $a^*_{1:N}$ is empty, then a global planner takes control and guides the robot to areas with high expected information gain.

Planning Root Update

At the end of every planning episode, $a^*_{1:N}$ is stored and then used to update the root of the lookahead tree during the subsequent episode (Alg. 3 – #2). Our root update approach is based on a receding-horizon policy reconciliation method detailed in Chap. 3.9. Fig. 4.8 demonstrates the effectiveness of this root update method.

Adaptive Coverage Range

While MCTS is an anytime algorithm, meaning that the construction of the tree can terminate at any point and a solution will be recovered, it only converges to the optimal solution with a sufficient number of simulations. Although it may be infeasible to reach the optimal solution given time constraints, estimates of the action values become increasingly more reliable with more simulations, leading to a higher quality coverage path. In order to find quality solutions at high planning rates, a real-time system must find a good balance between the fidelity of a simulation (e.g. how accurately we model the coverage observation) and the number of simulations.

To maximize the number of simulations within a suitable planning time, we propose an approximation of the coverage model that reduces the time complexity of the generative model $\mathcal{G}$. Our approximate world coverage update obviates the need for expensive ray-tracing operations in Alg. 2. First, we estimate the spaciousness r_{spac} of the local environment [80]. Then we adapt the distance at which a range-finder coverage measurement is performed based on r_{spac}. We denote this adaptive coverage distance by r_{adapt}. See Fig. 4.4 as an example of our adaptive coverage range approach.

Given a range-finder 3D pointcloud scan $\{z_i\}$ where z_i is the point at which a ray intersects an obstacle, we compute *spaciousness* as:

$$r_{\text{spac}} = f\big(\text{median}\{d(z_i)\}\big). \tag{4.9}$$

where $d(z_i)$ is the euclidean distance between the range-finder origin and a ray intersection-point z_i, and f is a low-pass filter: $f(x_t) = \alpha_1 f(x_{t-1}) + \alpha_2 x_t$ with constants $\alpha_1 = 0.95$ and $\alpha_2 = 0.05$. Robust to outliers in a potentially noisy pointcloud, $\text{median}\{d(z_i)\}$ gives a notion of the current scale of the local environment around the robot. Then, given r_{spac}, we compute r_{adapt} as:

$$r_{\text{adapt}} = \begin{cases} \alpha \cdot r_{\text{spac}}, & \text{if } \; r_{\text{spac}} \leq \frac{r_{\max}}{\alpha} \\ r_{\max}, & \text{otherwise,} \end{cases} \tag{4.10}$$

where α is an empirically tuned scaling constant, and $r_{\max}$ is our model-defined maximum sensor range. Equipped with r_{adapt}, we generate a probabilistic coverage mask $\{m_i\}$, detailed in Alg. 3-#1. The mask serves as an input to the generative model econGEN (Alg. 4), which then updates the world coverage state using inexpensive matrix operations. In comparison, expGen (Alg. 4) uses the more exact, but significantly more expensive Function CoverageUpdate (Alg. 2).

Algorithm 3 Coverage Planner

```
Function CoveragePlan
repeat
    Obtain: state $s = (n_q, \mu, G)$
            pointcloud scan $\{z_i\}$
    #1 Generate Coverage Mask
    Compute adaptive coverage range $r_{\text{adapt}}$ in Eq. (4.10)
    $\{m_i\} \leftarrow$ CoverageUpdate$(n_q, O, r_{\text{adapt}})$ ▷ Alg. 2
        ▷ where $O \Rightarrow p_c(n_i) = p_r(n_i) = 0 \; \forall \; n_i \in N$
    #2 Find Planning Root
    $n_\tau, \mu_\tau \leftarrow$ RootNode$(s, a^-_{1:N})$ ▷ see PLGRIM in [29]
    $s \leftarrow (n_\tau, \mu_\tau, G)$ ▷ update robot state to root parameters
    #3 Plan and Execute
    $T_r \leftarrow$ MCTS$(s, \; \{m_i\})$
    Extract action sequence $a^*_{1:N}$ from $T_r$
    #4 Prep for Next Episode
    $a^-_{1:N} \leftarrow a_{1:N}$
until timeout

Function RootNode
Input: state $s = (n_q, \mu, G)$
       previous action sequence $a^-_{1:N}$
Extract path $a^-_{Q:N}$ ▷ $n_Q$ is path node closest to $n_q$
Initialize path risk $\rho_{\text{path}}$ and distance $d_{\text{path}}$ to 0
for action $e_{ij}$ in path $a^-_{Q:N}$ do
    $\rho_{\text{path}}$ += $\rho_{ij}/d_{ij}$;   $d_{\text{path}}$ += $d_{ij}$
    if $\rho_{\text{path}} > \rho_{\max}$ or $d_{\text{path}} > d_{\max}$ then
        Assign root node $n_\tau \leftarrow n_i$
        Find root orientation $\mu_\tau$ ▷ if $n_\tau = n_Q$, then $\mu_\tau \leftarrow \mu$
        return $n_\tau, \mu_\tau$
    end if
end for

Function MCTS
Input: state $s = (n_q, \mu, G)$
       coverage mask $\{m_i\}$
Initialize empty lookahead tree $T_r$
repeat
    $T_r \leftarrow$ SIMULATE$(s; \; \mathcal{G})$
        ▷ estimate generative model $\mathcal{G}$ using efficient
          econGEN$(s, \{m_i\}, \pi_{rollout})$ in Alg. 4
until timeout
return $T_r$
```

Algorithm 4 Estimated Generative Models $\mathcal{G}$

```
Function econGEN ▷ economical model
Input: state $s = (n_q, \mu, G)$
       coverage mask $\{m_i\}$
       policy $\pi$
1: $n_q', \mu' \leftarrow \pi(n_q, \mu)$
2: $\{p_c(n_i)'\} \leftarrow \{\max[m_i,\ p_c(n_i)]\}$ ▷ fast
3: $r \leftarrow R(s, a)$ ▷ Eq. (4.6)
4: return $s', r$
Function expGEN ▷ expensive model
Input: state $s = (n_q, \mu, G)$
       coverage mask $\{m_i\}$
       policy $\pi$
1: $n_q', \mu' \leftarrow \pi(n_q, \mu)$
2: $\{p_c(n_i)'\} \leftarrow$ CoverageUpdate$(n_q, G, r_{\max})$ ▷ Alg. 2 (slow)
3: $r \leftarrow R(s, a)$ ▷ Eq. (4.6)
4: return $s', r$
```

Discussion

A decoupled approach to the coverage planning problem leverages a greedy algorithm for non-myopic viewpoint selection. This approximation relies on the fact that selecting more viewpoints never reduces the total coverage reward, since Eq. 4.3 exhibits monotonicity [75, 81]. While true in theory, this conjecture falters in a real-world exploration domain where the robot only has partial information about the world. In this setting, the inclusion of risky or low quality viewpoints, i.e., those evaluated using unreliable world estimates, can have adverse effects on the final policy and the robot's ability to collect coverage reward over an exploration mission. Since viewpoint interdependency is not critically examined, a policy incorporates *all* viewpoints in order to achieve performance guarantees associated with greedy methods for submodular maximization [75]. The optimal policy is constructed by finding the shortest path through a set of viewpoints. The core issues with this optimization is that it incorrectly assumes a stable and reliable world model. Specifically, the path is constructed without considering that: (*i*) the robot may fail during execution of the path, (*ii*) world coverage and traversability estimates become increasingly unreliable with increasing distance from the robot, and (*iii*) the world model (i.e. Local IRM) changes dynamically as the robot uncovers and maps new regions.

In order to address the aforementioned issues, the proposed approach to the coverage planning problem exhibits the following properties that make it suitable for a real-world exploration domain.

1) *Viewpoint Selectiveness*: A policy is evaluated by computing the marginal coverage reward and path cost for each successive action, or viewpoint, in the policy (Eq. 4.6). Understanding coverage interdependency between successive viewpoints lifts the burden of needing to fully cover the current graph with a single policy – an unproductive and potentially harmful ambition in the presence of uncertainty. As a result, viewpoints that do not provide sufficient coverage utility within a time-budget, or jeopardize the robot's safety, can be discounted from the final policy, while still preserving MCTS optimality.

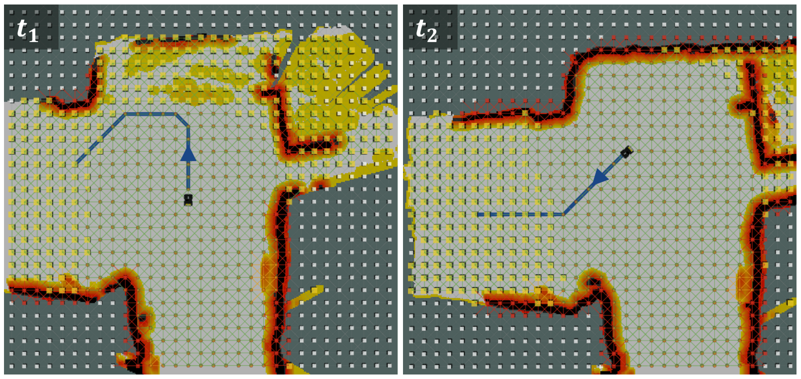
2) *Robustness to Uncertainty*: The lookahead tree is rooted at (or very near to) the robot's current location. Hence, MCTS visits nodes close to the robot more frequently, effectively focusing its search time in areas of the environment where world coverage and traversability risk estimates are more reliable. Moreover, due to a discount factor in the problem objective Eq. (4.8), policies that shift coverage reward earlier in time are more rewarding. By incorporating this near-sighted incentive, the robot accounts for stochasticity in sensing and motion control, as well as the fact that the world model will evolve as undetected areas are exposed.

Fig. 4.6 compares paths constructed by our approach and a decoupled approach in an uncertain world. Recall in the decoupled approach, viewpoints are greedily selected in order of highest marginal coverage reward. Therefore, rather than discounting viewpoints far from the robot where world estimates are poor, the decoupled approach actually prioritizes distant points since there is less sensor overlap at these locations with the robot's current field-of-view. The path planner is then "locked into" these viewpoints, and optimistically reasons over this potentially unreliable search space.

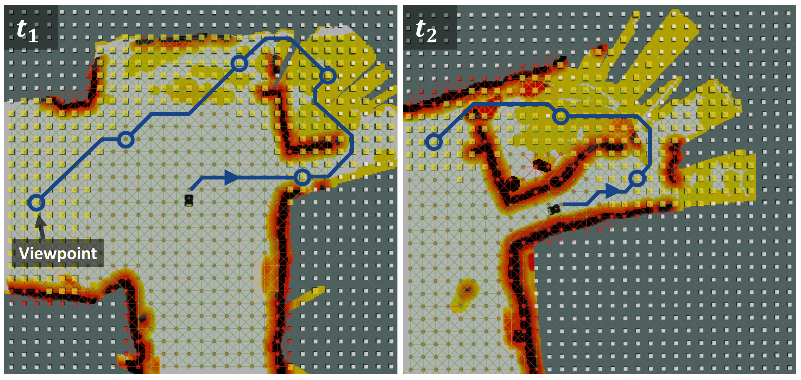
4.8 Experimental Results

In order to evaluate our proposed approach, we perform simulation studies and real-world experiments with a four-wheeled vehicle (Husky robot) and quadruped (Boston Dynamics Spot robot). Both robots are equipped with custom sensing and computing systems [4, 5, 71], and the entire autonomy stack runs in real-time on an Intel Core i7 processor with 32 GB of RAM. The stack relies on a multi-sensor fusion framework, the core of which is 3D point cloud data provided by

LiDAR range sensors [6]. During testing, the proposed approach (or baseline) was integrated as the local planner within the hierarchical planning framework PLGRIM. The frontier-based planner detailed in Chap. 5 provided global guidance.



(a) Unified Coverage Planner (Proposed)



(b) Decoupled Coverage Planner

Figure 4.6: For two planning episodes at t_1 and t_2 during Husky's autonomous exploration of a real-world mine, we show the coverage path constructed using our proposed unified approach (a) and the commonly-adopted decoupled approach (b) for solving the submodular coverage problem. In (a), the robot collects the remaining coverage reward at the end of the passage, before continuing to the large, unexplored passage to the left. In (b) at snapshot t_1, the robot incorrectly detects openings at the end of the passage due to bad sensor measurements and selects a set of viewpoints accordingly. The shortest path through the poorly-selected viewpoints guides the robot through a narrow passage to the right, which is riskier and less rewarding than the passage to the left.

Simulation Evaluation

We evaluated the proposed planner against *ours-LF*: the proposed rollout-based method with a low-fidelity coverage sensor model, i.e. non-probabilistic and static coverage range $r_{\max}$. All tests were performed in a simulated maze environment, as shown in Fig. 4.7. The maze consists of a large irregular network of large spaces and narrow passages, many of which are connected by sharp bends. This geometry exposes the weaknesses of a rollout-based planner where the coverage sensor model does not effectively approximate the actual range finder sensor. The long-range ours-LF planner (r_{max} = 8m) overestimates the coverage sensor range and, therefore, fails to detect openings at the sharp bends. As a result, large swaths of the environment are not exposed, and the robot terminates exploration early. Alternatively, the short-range ours-LF planner (r_{max} = 4m) performs significantly better since it can expose and explore all narrow passages. However, since it underestimates the coverage sensor range, it finds redundant trajectories in the large spaces, which contributes to a slight degradation in performance. Our proposed solution can handle all settings, since it neither over- or under-estimates the true coverage range in exchange for reducing computation. Moreover, since the model is probabilistic, it inherently adjusts its coverage density to the local environment. As a consequence, when the robot approaches a sharp bend, it travels deep enough to "see" uncovered space around the corner, which is critical to exposing the entire environment.

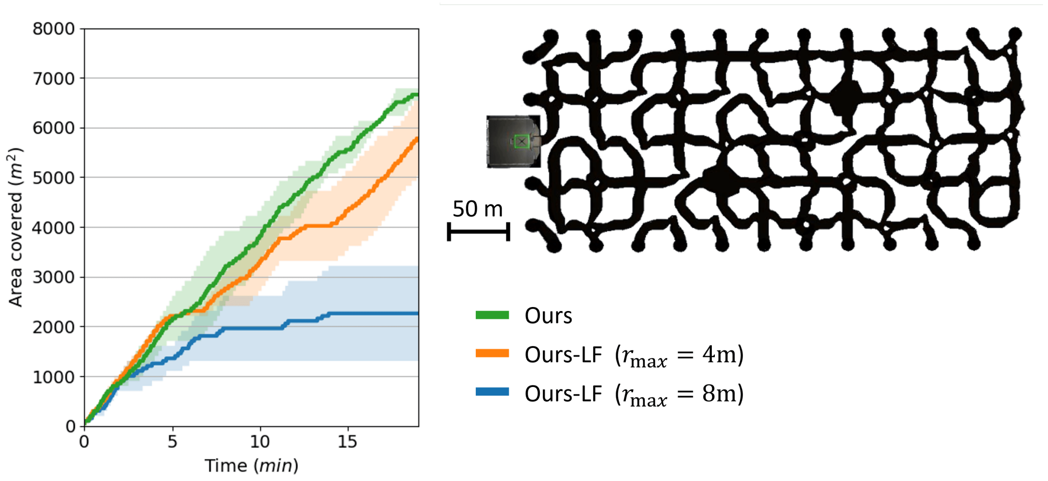
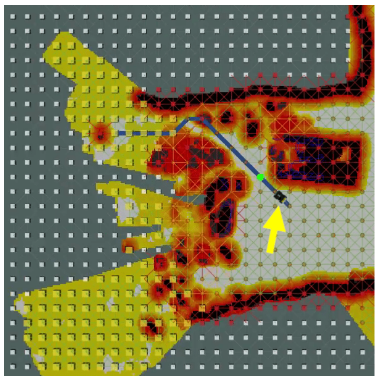
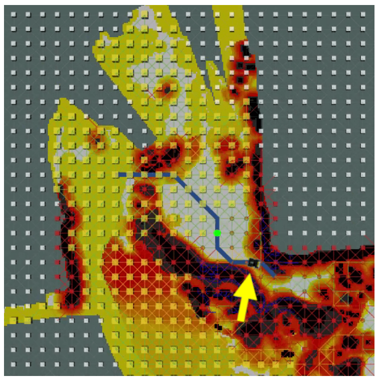


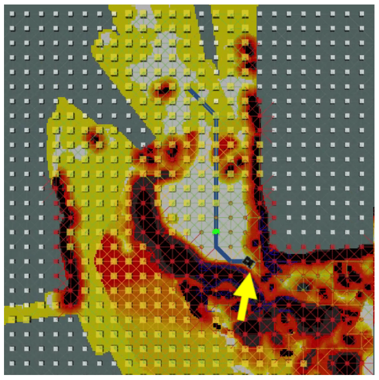
Figure 4.7: Results from simulated exploration runs in the simulated maze (shown at top right). We define our coverage metric to be the accumulated area within an 8m radius of the robot. Each curve is the average of 2 runs.

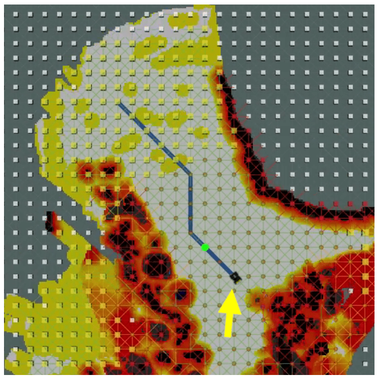
Real-World Evaluation

Our solution was extensively tested on physical robots in real-world environments. In particular, we present results from the exploration of a limestone mine (Figs. 4.9, 4.8) and library (Fig. 4.10) in the Kentucky Underground, Nicholasville, KY. While the mine primarily consists of large passageways (~15m-width) with complex terrain, the library is a network of narrow aisles (~1.5m-width) with smooth flooring.


(a) t = 00:00


(b) t = 00:30


(c) t = 00:31


(d) t = 01:46

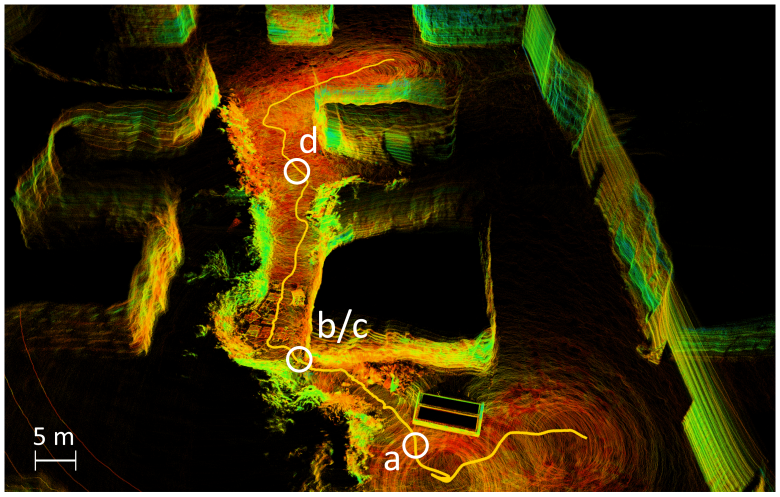


Figure 4.8: Snapshots of robot's navigation through rocks and debris during its exploration of a limestone mine. The bottom figure shows the robot's trajectory (yellow) and snapshot locations on a point cloud map generated by the robot. The coverage path (blue) and the planning root node (green circle) are shown in snapshots (a) – (d). Note that (b) and (c) are from consecutive planning episodes as the robot turns a corner, receives new sensor measurements, and updates the world risk state. The root node is updated according to the receding horizon reconciliation method, introduced in Chap. 3.9.

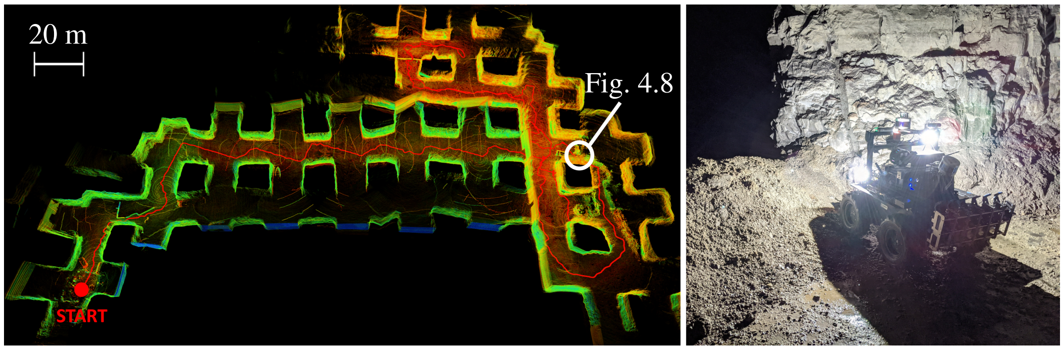


Figure 4.9: Husky's exploration of a limestone mine during a 60 min. mission using the proposed coverage planner. The coverage paths down the main corridor exhibit a wave-like shape. When the robot encounters a junction, it moves toward the corner in order to maximize coverage of both branches, and then re-aligns with the centerline of the main corridor. The starting location of snapshot Fig. 4.8 (a) is indicated on the map. Fig. 5.8 (b) provides the associated coverage plot.

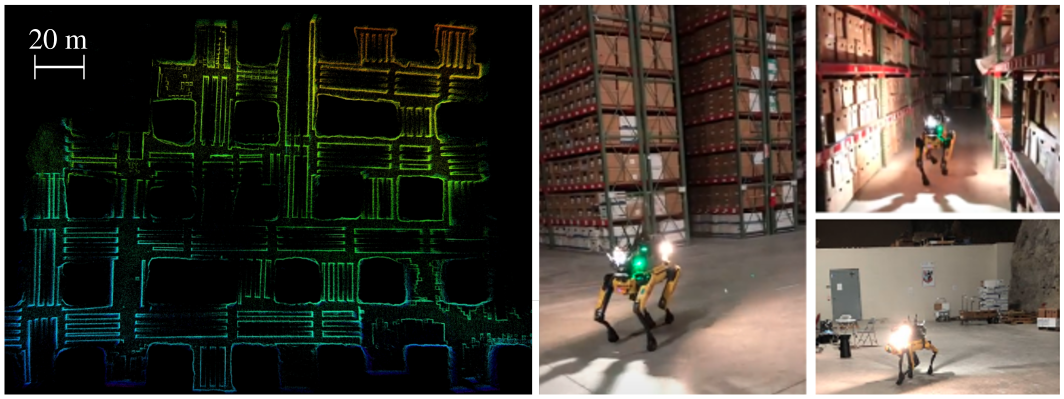


Figure 4.10: Exploration of a library using the proposed coverage planner. Fig. 5.8 (a) provides a coverage plot of Au-Spot's library exploration during a 50 min. mission.

4.9 Summary

This chapter presented an approach for solving the time-limited coverage problem for autonomous exploration of unknown environments. To solve this problem, which is submodular in nature, we used a unified rollout-based search algorithm. This allows us to evaluate how the robot's observation of the world affects the utility of future observations, while simultaneously accounting for traversability risk and the robot's dynamic constraints. In order to adequately investigate the search space, we reduce rollout computation using an effective approximation to the coverage sensor model which adapts the coverage range to the local environment. As a result, we can solve the submodular coverage problem in a unified manner, which we contend is more robust to real-world uncertainty than decoupled methods.

Chapter 5

GLOBAL COVERAGE PLANNER

This chapter presents a method for solving the time-limited exploration problem over a global graph-based representation of the environment. It begins by proposing the *Frontloaded Information Gain Orienteering Problem* (FIG-OP) – a generalization of the traditional orienteering problem where the assumption of a reliable environmental model no longer holds. The FIG-OP addresses model uncertainty by *frontloading* expected information gain through the addition of a greedy incentive, effectively expediting the moment in which previously unseen area is uncovered. In order to reason across multi-kilometer environments, FIG-OP is solved over an information-efficient world representation (Fig. 5.1), constructed through the aggregation of information from a topological and metric map, i.e., the Global Information Roadmap and Risk Map, respectively. The FIG-OP solution exhibits improved coverage efficiency over solutions generated by greedy and traditional orienteering-based approaches (i.e. severe and minimal model uncertainty assumptions, respectively).

This chapter was adapted from:

> O. Peltzer*, **A. Bouman***, S. Kim, R. Senanayake, J. Ott, H. Delecki, M. Sobu, M. Schwager, M. Kochenderfer, J. Burdick, and A. Agha-Mohammadi. “FIG-OP: Exploring Large-Scale Unknown Environments on a Fixed Time Budget”. In: *Submitted to IEEE Robotics and Automation Letters (RA-L)*. IEEE, 2022.

5.1 Introduction

Consider a time-limited mission where a robot, equipped with mapping and localization capabilities, is tasked with autonomously exploring a large-scale unknown environment. The robot must *(i)* maintain an internal environment representation that encodes traversability and task-specific information, and *(ii)* plan risk-mitigating paths that increase the robot’s understanding of the world. All the while, the robot must account for motion and sensing uncertainty in order to plan and execute robust exploratory behaviors. To this end, we introduce a planning framework based on the *orienteering problem* formulation [82] for solving the resource-constrained exploration problem over long horizons.

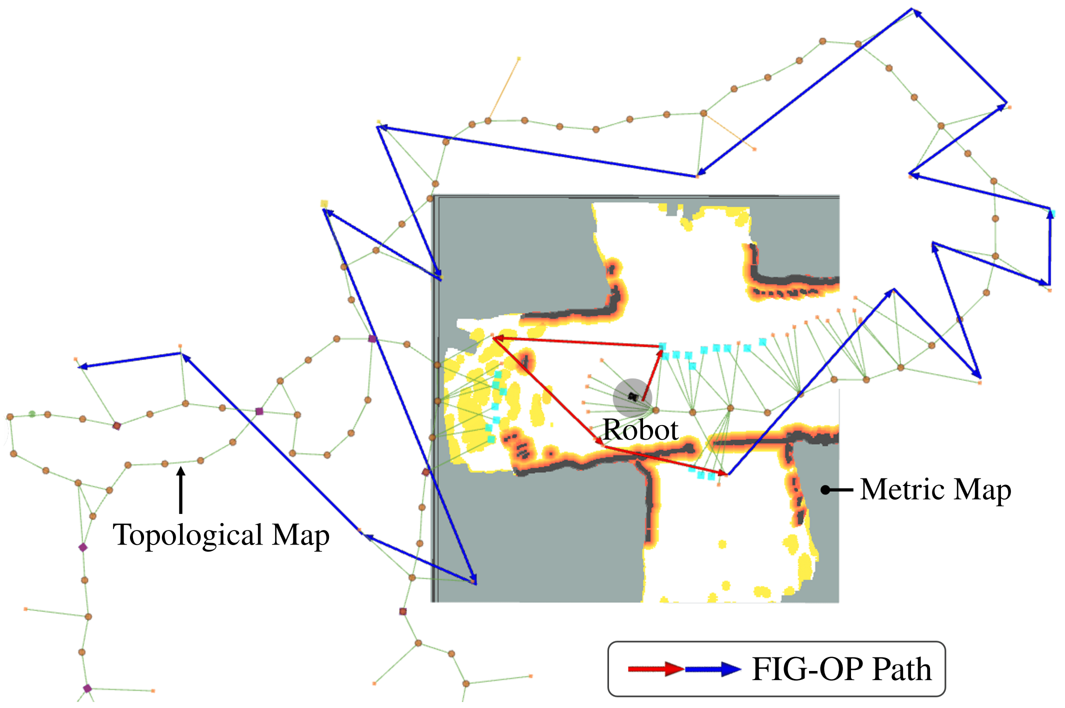


Figure 5.1: FIG-OP path generated during the autonomous exploration of the limestone mine of the Kentucky Underground, Nicholasville, KY, with a Husky robot platform. The environment representation is broken down into two levels: a dense robot-centered local metric map, and a sparse globally-spanning topological map. Arrows show the FIG-OP solution. Path costs are computed on either the metric (red arrow) or topological map (blue arrow).

Due to the computational constraints imposed by real-time systems, large-scale environments are commonly represented by topological graph-based structures. Equipped with this representation, the majority of exploration-driven algorithms apply one-step lookahead strategies to extend the boundary of explored space, ultimately resulting in *globally* sub-optimal plans. More recent approaches have found long horizon solutions using a traveling salesman problem formulation [83]. However, they assume unlimited mission time and complete understanding of the environment during policy execution.

In reality, a topological graph structure representing an environment grows and changes dynamically as the robot uncovers new regions. In addition, the robot faces a risk of failure when traversing edges of the graph that is not accurately captured in the graph's action cost. As a result, the traditional orienteering problem formulation can suffer from overly ambitious planning, which can lead to a delay in the moment where new area is uncovered in favor of maximizing an unreliable reward estimate over the mission horizon. We argue, somewhat counter-intuitively, that a *greedy* or *time discounted* incentive allows for effectively exploring dynamic graphs, to ensure we extend the explored boundary in the near term.

To address the challenges associated with optimal long term planning in unknown environments, we propose a framework consisting of two primary components. First, we construct a multi-fidelity world representation that encodes information about the robot's traversal risk and past sensor coverage. Then, we find long-horizon exploratory paths, robust to representation uncertainty, within the allotted mission time. The specific contributions of this work are as follows.

1. We construct a multi-fidelity, in terms of both time and space, world representation by combining risk and coverage information from *(i)* a large-scale, but outdated and sparse, topological graph-based structure, and *(ii)* a local-scale, but continuously updated and high resolution, metric grid-based structure. By extracting information from both sources, we increase the coverage rate by more than 35% on average in a 30 minute run, when compared to using only the topological graph.

2. We propose a variant of the orienteering problem, called the *Frontloaded Information Gain Orienteering Problem* (FIG-OP), for planning paths. The FIG-OP objective is a function of both information gain and travel distance, resulting in solutions that shift, or *frontload*, information gain earlier in time. We introduce an algorithm for solving FIG-OP in real time based on Guided Local Search [84].

3. The proposed solution was extensively tested on physical robots in various real-world environments. It also served as the top-level planner for team CoSTAR's entry in the Final Circuit of the DARPA SubT Challenge [4]. In addition, we ran comparative experiments in high-fidelity simulation environments that show an improvement in coverage rate over competitive baseline methods.

5.2 Related Work

The objective of the exploration problem is to maximize sensor *coverage* of an unknown environment for a given mission time. The term coverage designates the area swept out by a robot's sensor footprint [15].

Exploration schemes relying on the identification of the boundary between uncovered and covered space, regions termed *frontiers*, was first proposed by Yamauchi

[28]. Since then, frontiers have been used extensively throughout the exploration literature [29–32]. Traditional frontier-based approaches construct one-step lookahead policies that maximize a utility function, accounting for expected gain of information and the cost of motion [28, 54, 85–89].

Several frontier-based approaches have incorporated *art gallery problem* schemes in order to find coverage-optimal paths towards an unexplored boundary. Here, the objective is to find the minimum number of viewpoints that collectively maximizes coverage [24, 53, 83]. To ensure computational efficiency, Heng [53] and Cao et al. [83] approximate the coverage problem, which is inherently submodular [90, 91], as a modular orienteering problem and traveling salesman problem, respectively. While non-myopic, these approaches are limited to a local region around the robot and assume no uncertainty in coverage information during policy construction.

In order to address real-world stochasticity, the coverage problem has also been formulated as a Partially Observable Markov Decision Process (POMDP) [29, 64]. POMDP solvers find robust long-horizon policies, but require a model of the robot's sequential action-observation process under motion and sensing uncertainty. While models have been effectively developed for short-range planning, POMDP planning at multi-kilometer scales suffer from model inaccuracy. As a result, [29] assumes no changes to the state of the environment during global policy construction, significantly simplifying the problem to one in an MDP setting where frontiers are terminal states. As a consequence, the planning horizon is severely limited, resulting in a myopic exploration policy.

The orienteering problem provides long-horizon solutions to resource-constrained problems [82]. Variants of the orienteering problem have been proposed for centralized multi-robot coverage problems [92] and persistent monitoring problems [93]. However, these methods are not exploration-driven and, thus, rely on a prior map of the environment. When applied to the exploration problem, the OP suffers from many sources of model uncertainty (e.g. sensor measurements, hazard assessment, localization, and motion execution). At a high level, we focus on the false assumption that the graph structure is preserved over the planning horizon. On the contrary, during exploration, a robot will uncover new swaths of the environment, extending the horizon of explored space and augmenting its understanding of the world.

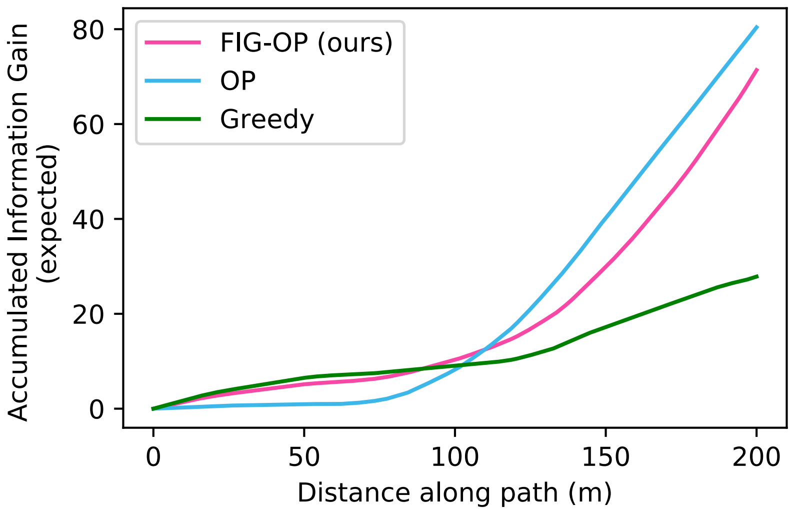


Figure 5.2: Expected accumulated information gain for paths constructed according to FIG-OP, OP, and a greedy objective in the real-world mine (Fig. 5.8). The curves are an average of 50 consecutive planning episodes with a planning horizon of 200 m. Early in the path, FIG-OP is competitive with the greedy algorithm, and thus quickly collects information gain. Later in the path, the greedy algorithm suffers due to an accumulation of globally suboptimal decisions. However, since FIG-OP encourages long-term efficiency, its solution stays competitive with the OP solution, which is designed to maximize information gain over the entire planning horizon.

5.3 Problem Formulation

Given an *a priori* unknown environment, our objective is to construct a long-horizon path that provides global guidance to frontier regions so as to maximize information gathered over a predefined time budget. To solve this problem, we propose a variant of the Orienteering Problem that produces solutions that *frontload* information. We call this framework the *Frontloaded Information Gain Orienteering Problem (FIG-OP)*.

We assume a graph-based environment representation $G = (N, E)$ with nodes N and edges E. Nodes are discrete areas in space that represent frontiers, and edges represent actions. More precisely, we define an action to be a traversal from node $n_i \in N$ to a node $n_j \in N$, connected by an edge $e_{ij} \in E$. Each edge e in the graph has an action cost $a(e)$.

Path p is a non-repeating sequence of nodes. $\mathcal{N}(p)$ and $\mathcal{E}(p)$ denote the set of nodes and edges along path p, respectively. We define $a_p(n_i)$ to be the total action cost associated with traversal from the root node n_0 to node n_i along the edges in p.

More specifically, for any node $n_i \in p$, we define

$$a_p(n_i) = \sum_{e \in \mathcal{E}(p_{0:i})} a(e),$$

where $a(e)$ is the action cost associated with edge e, and $p_{0:i}$ is the contiguous subsequence of p from the root node n_0 to node n_i.

When visiting a node n_i on graph G, the robot collects information gain $IG(n_i) \geq 0$, i.e., the amount of new area uncovered. Let us now consider a *frontloading* function F that inflates information gain based on the accumulated action cost until the time of collection. We wish to solve the optimization problem for FIG-OP as:

$$\begin{aligned} \underset{p}{\text{maximize}} \quad & \sum_{n_i \in \mathcal{N}(p)} \underbrace{\boldsymbol{F}\big(a_p(n_i)\big) \cdot \boldsymbol{IG}\big(n_i\big)}_{FIG} \\ \text{s.t.} \quad & \sum_{e \in \mathcal{E}(p)} a(e) \leq a_{max} \;\text{ and }\; p_0 = n_0, \end{aligned} \tag{5.1}$$

where a_{max} is the robot's action cost budget. The purpose of the function F is to favor solutions where frontiers are visited in the near term. To this end, we define F as follows:

$$F\big(a\big) = 1 + k_1 \cdot S\left(\frac{a - k_2}{k_3}\right), \tag{5.2}$$

where S is the reversed logistic function $S(x) = \frac{1}{1+e^x}$, and amplitude k_1, inflection point k_2, and steepness k_3 are positive shaping parameters of F (see Fig. 5.4) The frontloading function F exhibits the following properties that make it suitable for long-horizon exploration planning where model uncertainty is high.

1) *Model Uncertainty Compensation*: To account for uncertainty in the time required to map the area beyond a frontier region, F inflates information gain for frontiers within a local neighborhood of the robot, effectively expediting the moment in which new area is uncovered. As a result, policies are constructed according to a more optimistic model of the environment where frontiers likely lead to large unexplored swaths. By biasing towards short term information gain, we visit frontiers earlier in time with the expectation that they will significantly alter our world understanding by exposing new opportunities for information gathering. See Fig. 5.3.

2) *Long-Term Efficiency*: As action cost a goes to infinity, F converges to 1, reducing Eq. (5.1) to the standard OP formulation. By maintaining this long-term reward incentive, we assume a level of reliability in our environment model. We find that this policy-encoded foresight helps the robot gather more information over the mission time horizon. In essence, FIG-OP strikes a strategic balance between short and long-term information gain, as illustrated in Fig. 5.2.

3) *Solution Regularization*: In contrast with the widely-adopted exponentially decaying function $E(a_p) = \gamma^{\frac{a_p}{k}}$, F is most sensitive to action cost a_p at its inflection point k_2. This is a critical feature since local action costs, computed on the metric map are continuously updated, and thus are prone to fluctuations when estimates of traversability risk unexpectedly change (discussed further in Sec. 5.4). With this is mind, the inflection point k_2 can be selected to regularize the solution, i.e. lessen the path's susceptibility to locally fluctuating estimates. We find that this logistic form reduces detrimental oscillatory behavior and improves coverage performance, as shown in Fig. 5.4.

We solve the FIG-OP using a receding horizon approach, where a model of the environment is constructed from the current state and used for planning at each iteration. By replanning regularly, the robot can adapt to unforeseen changes in the environment, such as newly uncovered areas or changing action costs.

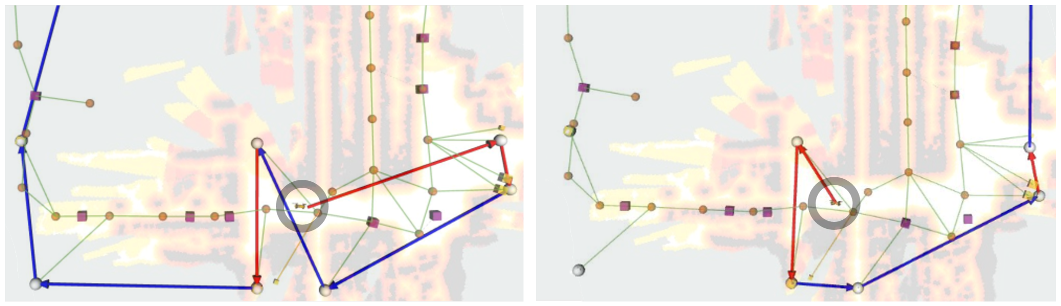
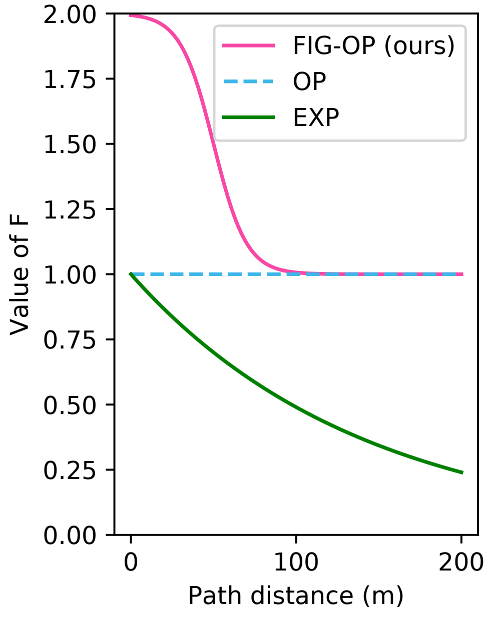
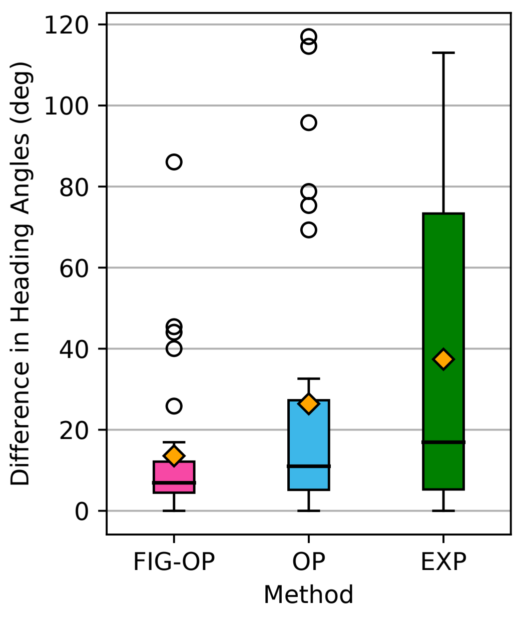

(a) Orienteering Solution (b) FIG Orienteering Solution

Figure 5.3: Comparison of the OP and FIG-OP solutions generated during Spot robot's autonomous exploration of a storage facility. Note that FIG-OP frontloads information gain, i.e. the first frontier in the solution path returned by FIG-OP has a lower action cost than that of the first frontier in the OP solution. By following the FIG-OP path, the robot will rapidly expose a new corridor in the library.



(a) *F* objective shapes



(b) Heading sensitivity comparison

Figure 5.4: Frontloading function characteristics. (a): The frontloading function F in Eq. (5.2) is shown for FIG-OP (ours), orienteering problem (OP), and the exponentially discounted (EXP) problem objectives. FIG-OP parameters are set to $k_1 = 1$, $k_2 = 50$, $k_3 = 10$. For EXP, we choose $k = 50$ and $\gamma = 0.7$, that empirically maximizes expected accumulated information gain over the 200 meter horizon. (b): We compare the absolute difference in the robot's heading direction to the first waypoint in the path during consecutive planning episodes over a 30 s interval. This data is extracted from the Husky robot's exploration mission in a limestone mine where complex terrain causes unexpected changes in traversability risk estimates. The plot illustrates the high sensitivity of the exponentially discounting objective (EXP) to locally changing risk estimates compared to our logistic objective form (FIG-OP). Note: the box extends from the first quartile to the third quartile of the data, with a line at the median and a diamond at the mean.

5.4 World Representation

We introduce a multi-fidelity world representation for efficient evaluation of FIG-OP in Eq. (5.1). First, we define the two sources of information, namely a *local metric map* and a *global topological maps* at the base of our methodology. Then, we introduce our proposed graph structure for effectively solving the FIG-OP problem. After defining the information gain metric adopted, we provide our approach for computing action costs for the edges in the graph.

The robot must always maintain an internal representation of its environment. Traditionally, these representations have fallen under one of two categories: *topological* maps or *metric* maps [94]. Topological maps are graph-based structures constructed by separating a space into a set of non-intersecting regions based on sensed features (e.g. frontiers). Since its resolution is dependent upon the complexity of the environment, topological maps are typically compact and scale well to large environments. Metric maps, meanwhile, are agnostic to environment complexity and divide the space into identical cells to form a grid structure. When compared to topological maps, metric maps are easily constructed and maintained. However, since the grid must resolve detailed features in the environment, metric maps can suffer from large space and time complexities [18]. For compactness, versatility and fidelity, we extract and combine information from a local metric map and a global topological map for computation of the information gain and travel cost.

Local Metric Map

To capture local traversability risk at a high resolution, we employ a rolling, fixed-sized grid structure $M = \{m_i\}$, which is centered at the robot's current position [11], introduced as the *Risk Map* in Chap. 2. This metric map, which is constructed by aggregating local point-cloud sensor measurements, captures mobility stressing features in the environment, such as obstacles, slopes, and ground roughness. We denote m_i as a cell in the grid-based map. Then, using the geometric planner proposed by Fan et al. [11], we define $\rho^{\ell}(m_i, m_j)$ as the instantaneous cost of the risk-minimized path between cells m_i and m_j.

Global Topological Map

To capture the global exploration state at multi-kilometer scales, we use a sparse bidirectional graph structure $G^g = (N^g, E^g)$, introduced as the *Global Information Roadmap* in Chap. 3. This topological map is globally fixed and consists of two mutually exclusive subsets of nodes: *breadcrumbs* $n^g_{i,b} \in N^g$ and*frontiers* $n^g_{i,f} \in N^g$. Breadcrumbs are generated in the robot's wake and capture covered traversable spaces in the environment. Alternatively, frontiers are generated at the boundary between explored and unexplored areas and, thus, capture uncovered traversable spaces. Given the topological map G^g, we define the action cost $\rho^g(n^g_i, n^g_j)$ of traversal between two nodes n^g_i and n^g_j as the distance associated with the shortest path, computed by applying Dijkstra's algorithm over the weighted graph. The edge weights are computed between two nodes within a neighborhood using the above geometric planner over the metric map. Due to range and computational limitations, edge weights are computed only once, despite changes in risk assessment over time.

5.5 FIG-OP Graph Structure

We propose a multi-fidelity *complete graph* structure $G = (N, E)$, which combines information from both the local metric map and the global topological map (Fig. 5.1). To reduce the size of the search space, a node $n_i \in N$ designates a cluster of frontiers in the topological map, determined using the DBSCAN algorithm [95]. Functions $f_\ell()$ and $f_g()$ map a cluster centroid to a metric cell and topological node, respectively. Every node pair (n_i, n_j) is connected by an edge $e_{ij} \in E$. Practically, we assume that by visiting one frontier in the cluster, any neighboring frontier in the cluster is accessible within a predefined distance.

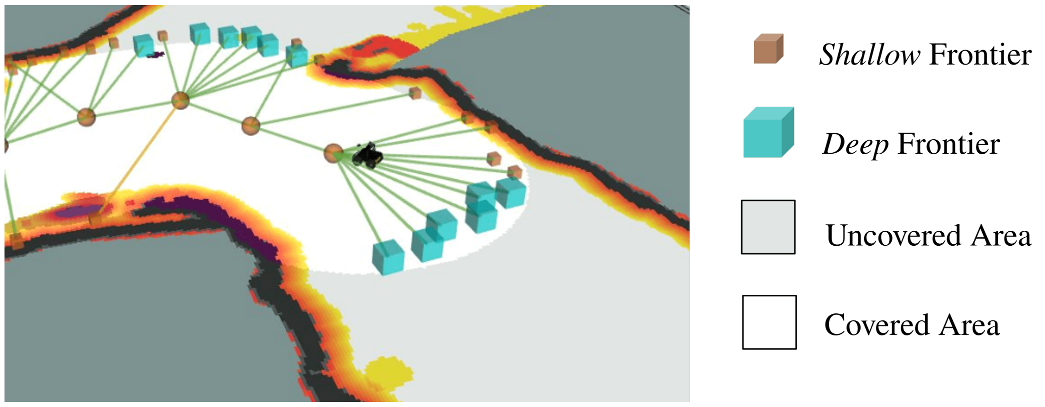


Figure 5.5: The estimated depth of uncovered area is one metric used to compute frontier information gain. Here, frontiers are classified into two categories: *deep* and *shallow*. Deep frontiers indicate entrances to unexplored passages.

Information Gain

See Chapter 3.8 for a theoretical description of information gain based on a reduction of entropy formulation. Here, we define information gain $IG(n_i^g)$ to be the expected area that can be uncovered by reaching a frontier node n_i^g on the topological map. Frontier segments are detected on the metric map using an omnidirectional LiDAR sensor at a range r_{sense}, based on the Fast Frontier Detector method [33]. Since the occupancy of cells beyond r_{sense} cannot be reliability determined [96], we do not count the number of cells expected to be visible to the robot as it travels along an edge. Instead, we define $IG(n_i^g)$ as a function of the approximated breadth and depth of the uncovered area beyond a frontier node. Frontier breadth is estimated by measuring the length (i.e. number of metric map cells) of the frontier segment. Frontier depth is estimated by attempting to associate a frontier segment with segments detected at longer ranges using connected component analysis. Frontiers that can be associated with long-range segments have large depth values, as shown in Fig. 5.5.

Multi-Fidelity Action Cost

Since the topological map consists of sparsely sampled nodes and maintains edge weights based on potentially outdated risk information, it can only provide a crude estimate of the traversal cost between two clusters n_i and n_j. To overcome this weakness, we compute traversal costs over the metric map for clusters located within its bounds based on the most up-to-date traversal risk assessment. We combine both metric and topological information to form a multi-fidelity cost estimate:

$$a(e_{ij}) = \begin{cases} \rho^{\ell}\big(f_{\ell}(n_i), f_{\ell}(n_j)\big), & \text{if } f_{\ell}(n_i), f_{\ell}(n_j) \in M \\ \rho^{g}\big(f_{g}(n_i), f_{g}(n_j)\big), & \text{otherwise.} \end{cases}$$

Fig. 5.6 illustrates the benefits of combining action costs computed on the metric and topological maps. Specifically, it shows how the second action cost in Fig. 5.1 is computed in order to “close the loop" on the topological map.

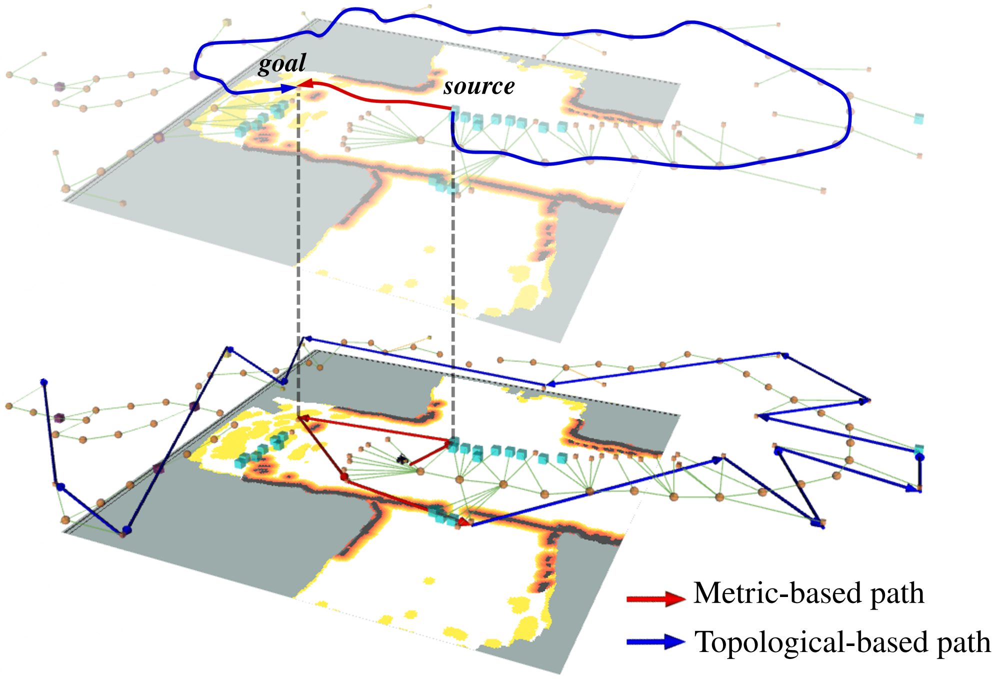


Figure 5.6: The FIG-OP graph combines low-fidelity action costs computed on the topological map (blue) and high-fidelity action costs computed on the metric map (red). Here, the topological-based and metric-based paths between two frontiers (*source* and *goal* located in the metric map) is displayed on the top layer. By using the more accurate metric-based path, FIG-OP finds a path that “closes a loop" on the topological map – a critical feature for accurately estimating action costs globally.

Algorithm 5 FIG-GLS: Front-loading variant of Guided Local Search

```
Input: Graph $G$, previous solution $\mathcal{B}_{t-1}$, new frontiers $\mathcal{F}$
  $\mathcal{S} \leftarrow$ Construct($G, \mathcal{B}_{t-1}, \mathcal{F}$)
  $\mathcal{B}$.solution $\leftarrow \mathcal{S}$
  $\mathcal{B}$.cost $\leftarrow$ cost($\mathcal{S}$)   ▷ Using equation (5.1)
  AlgLoop $\leftarrow 0$
  while AlgLoop $\leq$ MaxAlg
    AlgLoop $\leftarrow$ AlgLoop + 1
    LsLoop $\leftarrow 0$
    while Solution improved and LsLoop $\leq$ MaxLs
      LsLoop $\leftarrow$ LsLoop + 1
      $\mathcal{S} \leftarrow$ TSP($\mathcal{S}$)
      $\mathcal{S} \leftarrow$ Swap($\mathcal{S}$)
      $\mathcal{S} \leftarrow$ BackwardSwap($\mathcal{S}$)
      $\mathcal{S} \leftarrow$ Insert($\mathcal{S}$)
      $\mathcal{S} \leftarrow$ Replace($\mathcal{S}$)
    end while
    if cost($\mathcal{S}$) > $\mathcal{B}$.cost:
      $\mathcal{B}$.solution $\leftarrow \mathcal{S}$
      $\mathcal{B}$.cost $\leftarrow$ cost($\mathcal{S}$)   ▷ Using equation (5.1)
    else if $\mathcal{S} = \mathcal{B}$
      if not disturbed before
        $\mathcal{S} \leftarrow$ Disturb($\mathcal{S}$)
      else
        $\mathcal{S} \leftarrow$ Swap($\mathcal{S}$)
        $\mathcal{S} \leftarrow$ BackwardSwap($\mathcal{S}$)
        return $\mathcal{B}$
      end if
      if AlgLoop = MaxAlg/2
        $\mathcal{S} \leftarrow$ Disturb($\mathcal{S}$)
      end if
    end if
  end while
  return $\mathcal{B}$
```

5.6 Online Planning

Guided Local Search is a state-of-the-art heuristic method for solving the Orienteering Problem [82, 84]. We extend the method to allow for the modified OP objective function in Eq. (5.1). Alg. 5 describes the search procedure at a high level. We introduce three notable changes to [84]:

1) Solutions are evaluated, i.e. the path cost is computed, according to the revised FIG-OP objective function Eq. (5.1). See Alg. 5.

2) Two procedures, *Swap* and *BackwardSwap*, are introduced to explore different orderings of nodes within a solution and increase the objective value. *Swap* iterates over the path in a forward direction. To encourage frontloading of high information gain frontiers, *BackwardSwap* considers swapping nodes in reverse order. If the path objective remains the same, the swap is conducted if the total path cost is decreased.

3) Every iteration, we seed FIG-OP with the previous solution updated with newly generated frontiers inserted at the front of the path.

The complexity of this sub-optimal solver grows quadratically with the number of frontier clusters in the graph, which scales similarly to commonly adopted polynomial-time motion planners.

5.7 Experimental Results

We perform simulation studies and real-world experiments with a four-wheeled vehicle (Husky robot) and a quadruped (Boston Dynamics Spot robot) in order to evaluate our proposed algorithm. The robot is equipped with custom sensing and computing systems [4, 5, 71], and the entire autonomy stack runs in real-time on an Intel Core i7 processor with 32 GB of RAM. The stack relies on a multi-sensor fusion framework, the core of which is 3D point cloud data provided by LiDAR range sensors [6]. To evaluate planner performance in both simulation and real-world, we compute *coverage* $[m^2]$ as the accumulated area within the robot's sensor footprint during a run. Throughout this section, we detail how our experimental findings validate core features of our proposed method.

Simulations

We demonstrate FIG-OP's performance in a simulated subway and maze environment. We compare the performance of FIG-OP against the following frontier-based exploration planners in the simulated subway and maze environments.

1) *FIG-OP:* Proposed method where FIG-OP ($k_1 = 1$, $k_2 = 50$, and $k_3 = 10$) is solved over the multi-fidelity complete graph G using Alg. 5.

2) *FIG-OP with Low-Fidelity Action Costs (FIG-LF)*: Modification to the proposed method where all action costs are computed on the topological graph G^g.

3) *Greedy:* Myopic planner that selects the frontier with the smallest action cost based on G [28].

4) *OP*: Long-horizon planner where the orienteering problem is solved over G using the GLS algorithm in [84].

Simulation results are provided in Fig. 5.7 and Table 5.1.

Model Uncertainty Compensation: The effectiveness of FIG-OP's greedy incentive is most evident in the maze environment. The maze consists of a large irregular network of passages. Most passages lead to long unexplored branches, a fact which is not encoded in the robot's model of the environment due to sensing limitations. As a result, in many cases the information gain assigned to a frontier underestimates the frontier's true value. Due to its false assumption of no model uncertainty, the OP method suffers in this setting since it plans over the full mission horizon by maximizing an unreliable reward estimate. On the other hand, both FIG-OP and greedy approaches perform well since frontiers, leading to long unexplored branches, are visited earlier in the plan.

Long-Term Efficiency: The effectiveness of FIG-OP's long-horizon planning is most evident in the subway environment. The subway consists of interconnected, polygonal rooms. Near the beginning of the mission, the environment model is inaccurate since frontiers represent large swaths of space, and the model changes drastically as these frontiers are visited. Hence, FIG-OP and the greedy algorithm exhibit a higher coverage rate than OP, as shown in Fig. 5.7b. As the mission progresses and frontiers no longer represent large areas (i.e. model uncertainty reduces), then the coverage rate for the greedy method decreases. Meanwhile, OP and FIG-OP exhibit high coverage rates as they efficiency collect the remaining information in the environment. By strategically balancing greedy incentive and long-horizon efficiency, FIG-OP explores 95% of the subway faster (on average) than the greedy and OP methods, as shown in Fig. 5.7a.

Multi-fidelity model validation: We compare FIG-OP with its low-fidelity action cost counterpart FIG-LF, Table 5.1. We find that integrating information from the metric map significantly improves coverage capabilities in both simulation environments, with a notable 35% improvement in coverage rate in the maze environment.

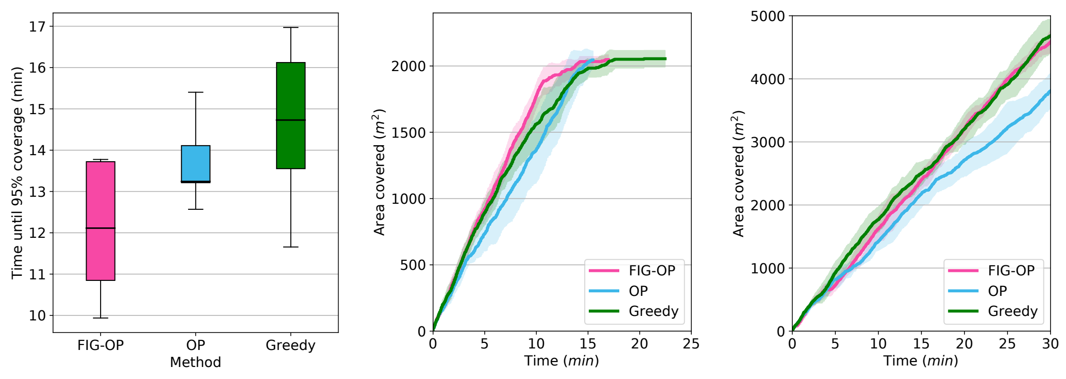
Table 5.1: For each listed algorithm, the average coverage metric over 5 runs in a simulated maze and subway environment is displayed. Standard deviations are provided in parenthesis.

Simulated Maze

Method	Coverage Rate (m^2/min)	30 min Coverage (m^2)	Planning Time (s)
FIG-OP (ours)	150.3 (7.4)	4577 (165)	0.22 (0.42)
OP	121.7 (6.3)	3822 (275)	0.12 (0.34)
Greedy	**154.3 (8.0)**	**4646 (257)**	0.02 (0.04)
FIG-LF	113.5 (7.0)	3468 (250)	0.15 (0.42)

Simulated Subway Station

Method	Coverage Rate (m^2/min)	95% Coverage Time (min)	Planning Time (s)
FIG-OP (ours)	**163.7 (20.5)**	**12.07 (1.53)**	0.08 (0.08)
OP	142.8 (14.1)	13.71 (0.98)	0.05 (0.06)
Greedy	135.4 (17.2)	14.64 (1.83)	0.01 (0.01)
FIG-LF	130.8 (19.4)	14.47 (2.15)	0.09 (0.56)



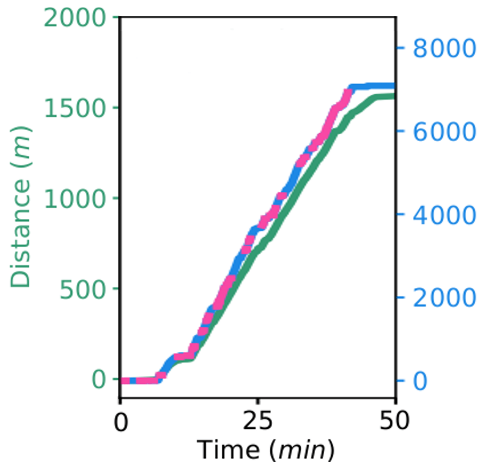
(a) Simulated **Subway** (b) Simulated **Subway** (c) Simulated **Maze**

Figure 5.7: Exploration by our proposed FIG-OP and baseline methods in simulated subway and maze environments, as displayed in Fig. 3.7. The exploration metrics for each method consist of five runs, with curve (b) and (c) displaying the averages. Refer to Fig. 5.4 for details about the box plot.

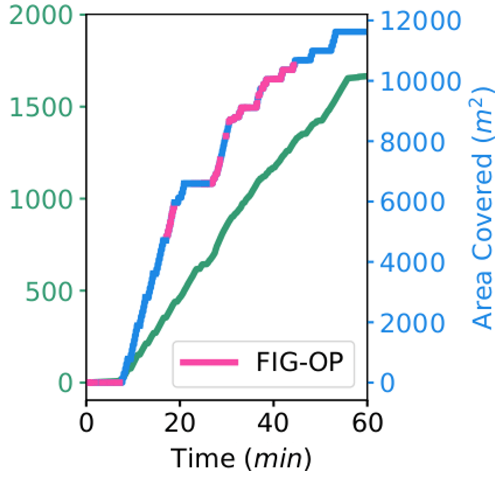
Field Tests

We extensively tested our FIG-OP solution on physical robots in a subway system, mine, and storage facility. During hardware tests, FIG-OP was integrated within the larger autonomy framework PLGRIM, introduced in [29]. That is, the planning system onboard the robot alternates between a local viewpoint-based planner and a global frontier-based planner (i.e. FIG-OP). Fig. 5.8 show the findings from an autonomous exploration run in a mine and library.

Solution Regularization: The benefits of the solution regularization provided by FIG-OP is most notably demonstrated in a real-world environment where traversability risk estimates fluctuate. Using traversability risk estimates from the mine run (Fig. 5.8), we evaluated path regularization based on different forms of the front-loading function F in Eq. (5.2). Our findings in Fig. 5.4b indicate that our proposed logistic form regularizes the solution over consecutive planning episodes. As a result, the robot can maintain continuous velocities, which is essential for rapid exploration.



(a) Real-world **Library**



(b) Real-world **Mine**

Figure 5.8: Autonomous exploration of a library by Au-Spot (a), and a limestone mine by Husky (b). In this exploration metric, pink denotes time intervals where FIG-OP was directly guiding the robot. The local planner presented in Chap. 4 has control otherwise. Note that after the robot has been outside the communication range of the base station for more than 10 minutes, it retraces its steps until communication is reestablished. During this time, the covered area does not increase. Figs. 4.9 and 4.10 show point cloud maps of the mine and subway.

5.8 Summary

This chapter presented a novel planning framework for autonomous exploration of large-scale complex environments under mission time constraints. Our proposed formulation FIG-OP is a generalization of the orienteering problem where the robot does not have access to a reliable environment model. FIG-OP compensates for model uncertainty by incorporating a greedy incentive that shifts information gain earlier in time. We solve FIG-OP over a multi-fidelity world representation, and demonstrate its ability to strike an effective balance between near- and long-term planning through an extensive test campaign.

Chapter 6

SQUID

Aircraft that can launch ballistically and convert to autonomous, free-flying drones have applications in many areas such as emergency response, defense, and space exploration, where they can gather critical situational data using onboard sensors. This chapter presents a ballistically-launched, autonomously-stabilizing multirotor prototype (*SQUID – Streamlined Quick Unfolding Investigation Drone*) with an onboard sensor suite, autonomy pipeline, and passive aerodynamic stability. The vehicle demonstrates autonomous transition from passive to vision-based, active stabilization, confirming the ability of multirotors to autonomously stabilize after a ballistic launch in a GPS-denied environment.

This chapter was adapted from:

> **A. Bouman**, P. Nadan, M. Anderson, D. Pastor, J. Izraelevitz, J. Burdick, and B. Kennedy. "Design and Autonomous Stabilization of a Ballistically-Launched Multirotor". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 8511–8517.

6.1 Introduction

Unmanned fixed-wing and multirotor aircraft are usually launched manually by an attentive human operator. Aerial systems that can instead be launched ballistically without operator intervention will play an important role in emergency response, defense, and space exploration where situational awareness is often required, but the ability to conventionally launch aircraft to gather this information is not available.

Firefighters responding to massive and fast-moving fires could benefit from the ability to quickly launch drones through the forest canopy from a moving vehicle. This eye-in-the-sky could provide valuable information on the status of burning structures, fire fronts, and safe paths for rapid retreat. Likewise, military personal in active engagements could quickly deploy aerial assets to gather information as the situation evolves.

Ballistic launches also provide unique opportunities in the exploration of other bodies in the solar system. The *Mars Helicopter Scout* (MHS), deployed from the Mars 2020 rover, provided the first powered flight on another solar system body in

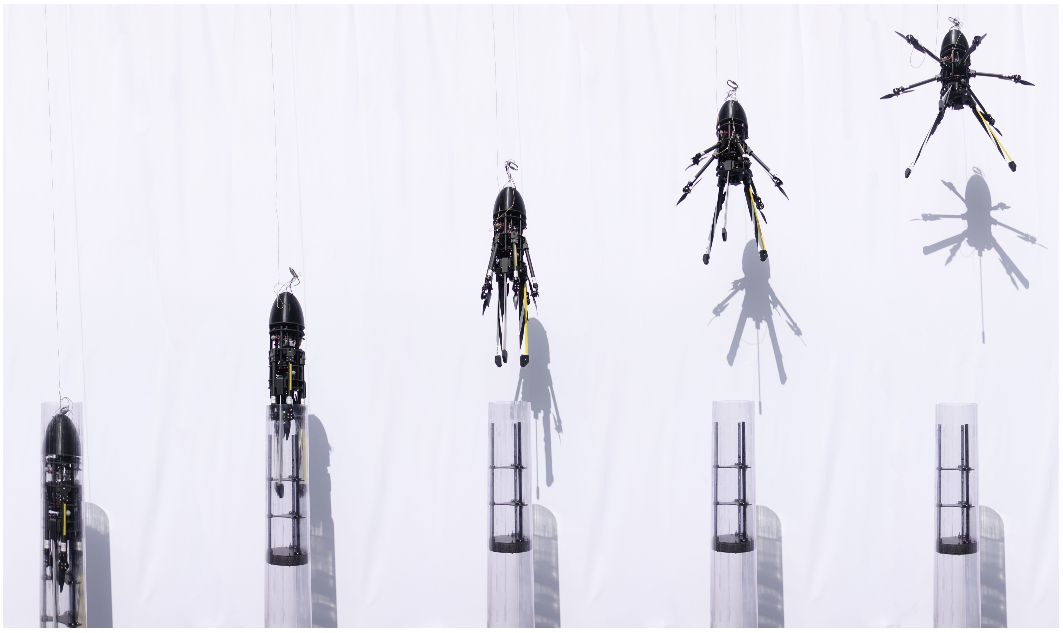


Figure 6.1: Launching *SQUID*: starting from inside the launcher tube, then deploying its arms and fins, before reaching its fully-deployed configuration. Note the slack in the development safety tether and how the carriage assembly remains in the tube throughout launch. Each snapshot is 41 ms apart

history [97]. MHS greatly expands the data collection range of the rover, however, it has a multistep launch sequence that requires flat terrain. The addition of a ballistic, deterministic launch system for future rovers or landers would physically isolate small rotorcraft from the primary mission asset. Aerial launch technology would even enable the aircraft to deploy directly from the entry vehicle during decent and landing, enabling it to land and explore sites that are at a great distance from the rover.

Multirotor aircraft are advantageous over fixed-wing systems as they can hover in place and aggressively maneuver in cluttered environments to achieve greater vantage points. However, the rotating blades of the multirotor are a hazard to nearby personnel (who may be distracted by other obligations), a problem which is particularly present if the system is to launch autonomously without human supervision. In these situations, multirotor aircraft operating in crowded and rapidly changing environments need a precise, highly deterministic, and fully autonomous takeoff method to achieve a safe operating altitude away from assets and personnel.

In the application scenarios described above, ideally the multirotor is stored for extended periods of time ("containerized") before being launched quickly, safely, and autonomously. Furthermore, when deployed from a moving vehicle, the drone must be aerodynamically stable to avoid tumbling when exposed to sudden crosswinds. Most current drone designs however are slow to deploy, require user intervention prior to takeoff, and cannot be deployed from fast-moving vehicles. Current foldable designs also require the user to manually unfold the arms which slows the process and puts the user at risk if the multirotor prematurely activates. A multirotor that can launch from a simple tube and autonomously transition to flight would solve many of the shortcomings of conventional drone deployment strategies.

While mature tube-launched fixed-wing aircraft are already in active use [98–100], tube-launched rotorcraft (both co-axial and multirotor) are much rarer and primarily still in development. Several consumer drones (e.g., the DJI *Mavic* series [101] and Parrot *Anafi* [102]) can be folded to occupy a small volume, but these designs cannot fit smoothly inside a launch system, and the unfolding is manual. Other manually unfolding rotorcraft can achieve a cylindrical form factor like *SQUID*: the *Power Egg* from Power Vision folds into an egg shape [103], the *LeveTop* drone folds into a small cylinder [104], and the coaxially designed *Sprite* from Ascent Aerosystems packs into a cylinder shape [105]. Automatic in-flight unfolding mechanisms for quadrotors, using both active [106] and passive [107] actuation, have been developed for the traversal of narrow spaces. However, to enable the ability to ballistically launch like *SQUID*, these existing foldable platforms must be redesigned to withstand launch loads and maintain passive aerodynamic stability post-launch. Ballistically-launched aerial systems that combine an aerodynamically stable structure and a foldable airfoil system have been developed in coaxial rotorcraft [108] and multirotor [109] formats, but both designs are still in the theoretical design phase, and have yet to demonstrate a transition from ballistic to stabilized flight.

In previous work [110], we introduced a small *SQUID* prototype, a folding quadrotor that launches from a 3-inch tube to a height of 10 m or more, and then passively unfolds to a fully functional multirotor when triggered by a nichrome burn wire release mechanism. This prior work introduced the basic aerodynamic principles and structural design concepts required to sustain the g-forces associated with a ballistic launch. A prototype was fabricated and ballistically launched from a vehicle moving at speeds of 80 km/h (22 m/s). However, the multirotor was stabilized by a remote pilot after the ballistic launch phase.

This chapter advances the line of investigation started in [110] and presents the design, development and testing of a full-scale *SQUID* prototype. Capable of carrying a significant sensor payload, *SQUID* transitions from a folded, 6 inch-diameter (152.4 mm) launch configuration to an autonomous, fully-controllable hexacopter after launch (Fig. 6.1). The entire process from launch to stabilization requires no user input and demonstrates the viability of using ballistically-launched multirotors for useful missions.

We review the full-scale *SQUID* design (Section 6.2), focusing on key changes from the first prototype [110]. Section 6.3 then describes the ballistic launch phase, Section 6.3 describes scale-model testing used to validate *SQUID*'s passive stabilization design, and Section 6.4 details the autonomous stabilization procedure. Experiments summarized in Section 6.4 demonstrate the passive-to-active stabilization pipeline. Conclusions are found in Section 6.5.

6.2 Mechanical Design

The mechanical design of the new *SQUID* prototype (hereafter termed *SQUID*, while *μSQUID* will refer to the earlier 3-inch *SQUID* prototype) is dictated by three broad functional requirements. The multirotor must: (i) launch from a tube (6-inch diameter for this prototype), (ii) travel ballistically to a predetermined height, and (iii) autonomously transition into stable, multirotor flight. To satisfy these non-traditional flight requirements, *SQUID* blends design elements from both ballistic and multirotor platforms. The multirotor's central rigid body houses a battery and the perception and control systems, and interfaces with six fold-out arms with rotors and three fold-out fins which passively stabilize the multirotor during ballistic motion. The layout of key *SQUID* components is given in Fig. 6.2 and the configuration in folded and deployed states are shown in Fig. 6.3. Table 6.1 and Table 6.2 provide a list of key *SQUID* components and main design attributes.

Table 6.1: *SQUID* System Properties

Property	Value	Units
Mass	3.3	kg
Length	79	cm
Folded Diameter	15	cm
Unfolded Diameter (propeller tip-to-tip)	58	cm
Thrust at Hover	56	%
Launch Speed	12	m/s

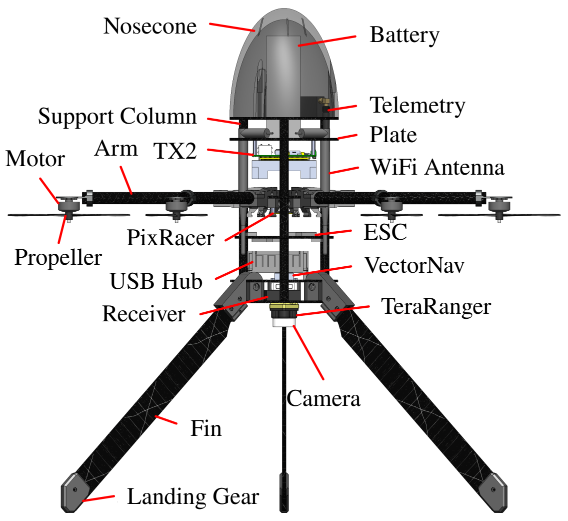


Figure 6.2: An annotated schematic of *SQUID*.

Central Rigid Body

In contrast to conventional multirotors, *SQUID*'s central body must sustain high transient forces during ballistic launch. Unlike prior *μSQUID*, which was manually stabilized by a pilot, *SQUID* also requires a perception system comprising a camera (FLIR Chameleon3), rangefinder (TeraRanger Evo 60m), IMU/barometer (VectorNav VN-100), and onboard computer (NVIDIA Jetson TX2) to achieve full autonomous stabilization. Due to these added components, the original 3D-printed aeroshell structure was abandoned in favor of a hollow carbon fiber frame in order to maximize volume, increase strength, and allow easy access to the perception and control systems.

The frame consists of six thick carbon fiber plates separated by support columns (made of aluminum standoff pins surrounded by carbon fiber tubes) that transmit the launch loads. A 3D printed nosecone reduces drag by approximately 50% compared to a bluff body nose. The placement of the heavy LiPo battery in the nosecone shifts the center of mass (COM) upward. This placement ensures that *SQUID*'s aerodynamic center (AC) trails behind the COM, which improves the passive ballistic stabilization. Passive stabilization is further addressed in Section 6.2.

Rotor Arms

The six rotors are mounted on carbon fiber tubes which attach to the central body with passive, spring-loaded hinges to allow 90° of rotation. The arms can exist in

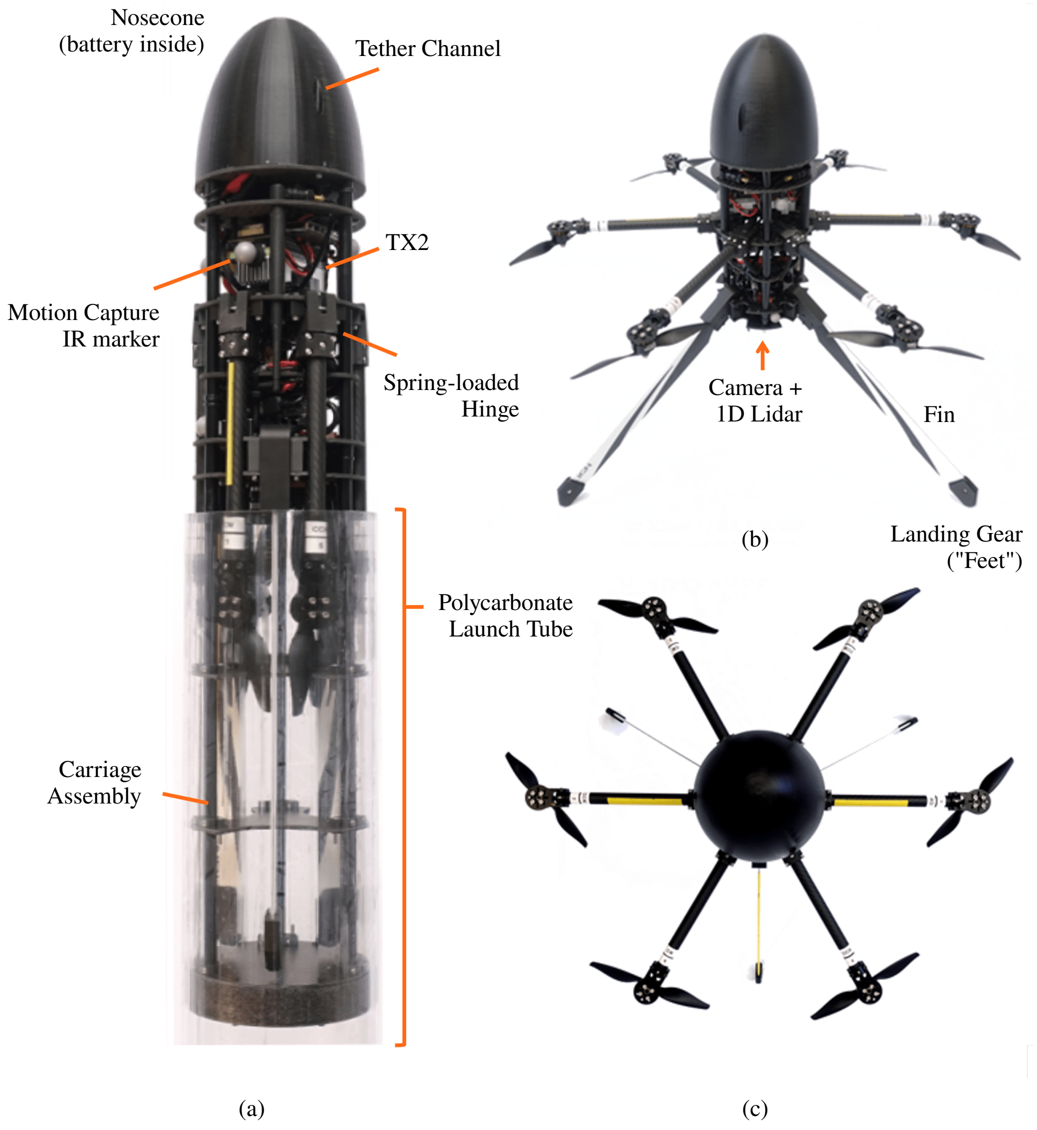


Figure 6.3: *SQUID* partially inside the launcher tube and interfacing with the carriage (a), and with its arms and fins fully deployed from a side (b) and top perspective (c).

two states: constrained by the launch tube to be parallel to the body axis (closed), or extending radially outward perpendicular to the central axis (open). For *μSQUID*, the timing of the transition was controlled by an arm release mechanism [110]. For *SQUID* however, the transition from closed to open state occurs immediately after the multirotor leaves the launch tube, reducing mechanical complexity.

A torsional spring inside the hinge generates 1.04 N·m of torque when the arm is closed, and half that amount when the arm is open. Vibration in the motor arms during flight dictates the addition of a spring-loaded latch to keep the arms rigidly open after deployment.

Fins

SQUID's fins provide aerodynamic stabilization during ballistic flight to ensure the vehicle maintains the launch direction before active stabilization is engaged. Aerodynamic forces on the fins shift the multirotor's AC downward behind the COM, enabling *SQUID* to passively weathercock and align with the direction of flight. Folding fins, rather than fixed fins, are a major design change from *μSQUID* [110] and were driven by a compromise between competing requirements of aerodynamic stability, low drag, constrained tube volume, and design simplicity. This design change was guided by the use of literature-derived expressions [111, 112] and scale model testing.

Fixed fins have a number of disadvantages. Any fin requires clean, unseparated flow to operate as designed. Therefore, fins that remain fixed within the tube area must also be paired with a streamlined tailbox in order to have access to said flow. This tailbox streamlining however reduces the wake drag and hence also reduces the stabilizing force it provides. Additionally, small fins which fit within the tube can only be partially effective as they have a limited wingspan. Expanding the fins along the tube only further lowers their aspect ratio (and therefore lift coefficient), reducing their capacity to move the AC. Deploying fins radially is therefore a much more effective means of enhancing stability, improving *SQUID*'s ability to predictably rotate upwind.

SQUID's tubular cross section and foldout fins increase stability relative to *μSQUID* and simplify launch packaging issues with a simple cylindrical geometry, but do so at the cost of more ballistic drag. For most *SQUID* applications however, ballistic efficiency can be sacrificed for these gains. Foldout fins can be tailored to provide a desired stability margin between the COM and AC, and provides margin for swappable payloads that may shift the COM. Given our selected 30 cm fins, the AC is located 38 cm from the nose, with a margin of 14 cm from the COM. Uncertainties in aerodynamic coefficients, drag on the arms, and the dynamics of the unfolding components can lead to substantial deviations from this calculated margin however. Accordingly, we validated our aerodynamic stability with a 3:1 scale model (50 mm diameter, 150 grams) using an open air wind tunnel (see Section 6.3) prior to full-scale tests.

While the hinges connecting the fins to the body are similar to the arm hinges, the

Table 6.2: Key *SQUID* components

Component	Description	Mass (g)
Flight Electronics		
Motors	T-Motor F80 Pro, 1900kv	36 (x6)
ESCs	T-Motor F30A 2-4S	6 (x6)
Propellers	7" diameter x 4" pitch	8 (x6)
Flight Controller	mRo PixRacer (PX4 Flight Stack)	11
Receiver	X8R 8-Channel	17
Telemetry	HolyBro 100 mW, 915 MHz	28
Battery	4S LiPo, 6000 mAh, 50C	580
Perception System		
Onboard Computer	NVIDIA TX2	144
Carrier Board	Orbitty Carrier Board	41
Rangefinder	TeraRanger Evo 60mm	9
IMU/Barometer	VectorNav VN-100	4
Camera	FLIR Chameleon3 w/ 3.5 mm Lens	128

fins do not use a latching mechanism because vertical vibrations have little impact on their functionality. “Feet" attached to the ends of the fins protect the tips and enable them to double as landing gear.

6.3 Ballistic Launch Process and Transition to Stabilized Flight

SQUID’s mechanical design and onboard active controls manage the deployment sequence (Fig. 6.4). The deployment pipeline comprises two primary phases: passive stabilization and active stabilization. In the first phase, the multirotor’s aerodynamic design ensures attitude stability as it travels along a ballistic trajectory after launch. Active stabilization begins once the arms are fully deployed and occurs before the trajectory’s apogee. The following sections provide details on the launch stabilization process and our experimental validation of these concepts.

Ballistic Launch Process

SQUID is ballistically launched to a minimum height that depends on both the safety requirements of the assets near the launch site and the altitude required for the targeted investigation. All the energy needed to loft the multirotor to the desired height, as well as to overcome the drag of the passive stabilization process, must be generated over the launching tube’s very short length. Consequently, the airframe experiences very large acceleration forces while being launched.

The core of the launch mechanism is a re-purposed T-shirt cannon [113]. Pressure is supplied by a liquid CO_2 canister that is regulated between 5.5 bar (indoor, to stay within ceiling clearance) and 6.9 bar (outdoor, maximum safe) chamber pressure in gas phase. An aluminum stand holds the launch tube in place and allows adjustment of the launch angle. Accordingly, both the launch height and angle can be adjusted to avoid local hazards.

Prior to launch, *SQUID* rests in a folded state inside the launch tube, which is generally pointed upwards. A 300 gram carriage assembly sits between *SQUID* and the tube base, transmitting launch loads generated by the compressed gas directly to the frame's support columns. A 25 mm-thick polyethylene foam disk at the base of the carriage creates a low-friction seal which maximizes the transfer of energy from the compressed gas into kinetic energy and also prevents the carriage from leaving the tube during launch.

This launching mechanism has a number of inefficiencies. After launch is triggered, the compressed gas accelerates *SQUID* through the tube at approximately 21 g's (estimated from video as the IMU saturates at 16 g's), but short of the unlimited valve throughput prediction of $\approx$350 g's. The maximum height achieved with this system is also 32 m (or 1 kJ potential energy), less than a third of the imparted energy as calculated from the ideal adiabatic expansion of the CO_2 chamber. Discrepancies between the predicted and estimated values are thought to be from friction within the tube, losses from a valve throughput, and air drag.

Passive Stabilization - Launch without Wind

After exiting the launch tube, the arms and fins deploy immediately due to the spring-loaded hinges. This deployment has four effects on the aerodynamic stability: the COM is shifted towards the nose, the AC is shifted rearward due to the fin lift, the fins increase aerodynamic damping in yaw, and mass moves outwards which increases yaw inertia.

As described in Section 6.2, the lower AC helps *SQUID* maintain orientation and follow the intended flight path until active stabilization begins. The large displacement between the COM and AC, coupled with the launch momentum, causes *SQUID* to orient robustly into the apparent wind. When the launch tube is stationary and roughly vertical, this effect helps *SQUID* to passively maintain orientation during the ballistic phase, which simplifies the transition to active stabilization.

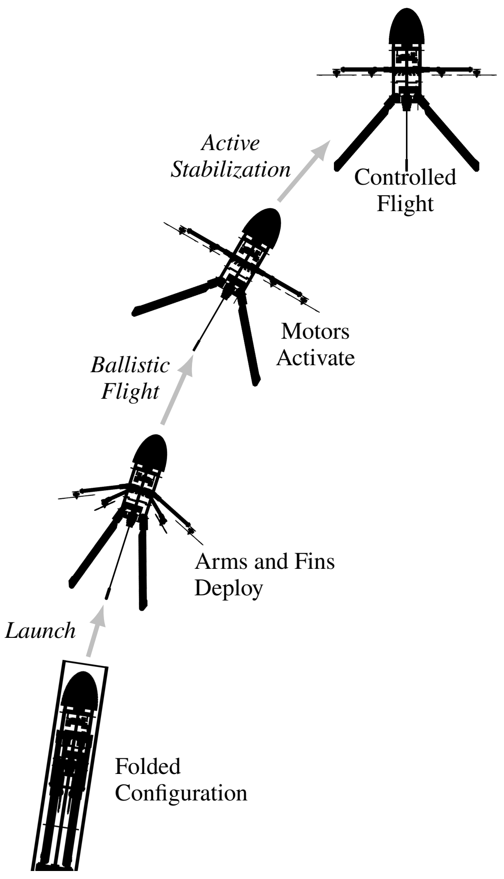


Figure 6.4: *SQUID* deployment sequence.

Passive Stabilization - Launch in Crosswind

During launch from a moving vehicle, *SQUID* experiences a strong crosswind, and will weathercock its nose in the direction of the launch platform's motion. Accordingly, *SQUID*'s passive stabilization design ensures that the multirotor travels smoothly during the ballistic phase and that its orientation at the beginning of the active stabilization phase is predictable.

To validate *SQUID*'s expected passive aerodynamic behavior before field testing, sub-scale wind tunnel tests were performed at the Center for Autonomous Systems and Technologies (CAST) at Caltech. These tests were intended to prove that the new folding fin architecture could provide a sufficient stabilizing effect in the presence of a crosswind.

The sub-scale wind tunnel tests were performed using a 1/3 scale model of *SQUID*. Scaling for ballistically-launched drones near apogee, presented in [110], primarily depends upon the Froude number ($U/\sqrt{gL}$), launch- to wind-velocity ratio, geo-

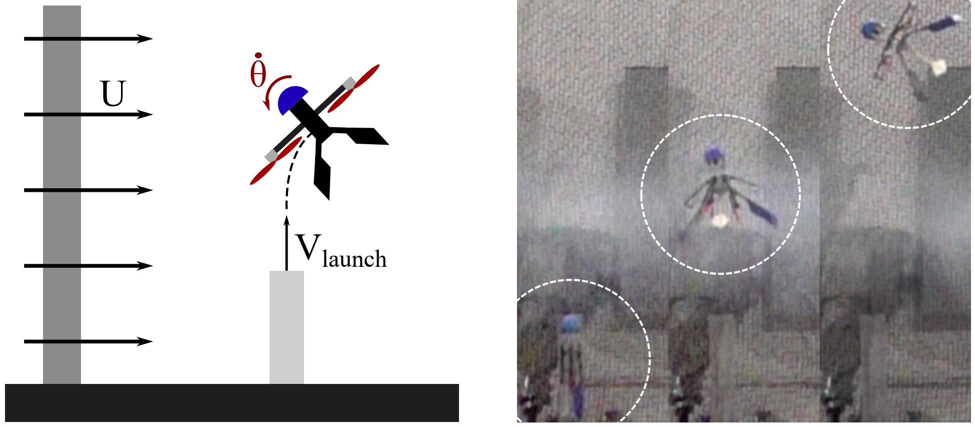


Figure 6.5: Wind Tunnel Testing. Left: definition of experiment parameters. Right: snapshot sequence showing stable upwind pitching of the 1/3 *SQUID* model.

metric parameters, and launch angle. Since *SQUID*'s tailbox is a bluff-body disc, separation at the base is virtually guaranteed, meaning Reynolds effects can be neglected [111]. To correct the sub-scale results to be representative of the full-scale model, the trajectories and velocities were scaled by a factor of 3 and $\sqrt{3}$, respectively [110].

Accordingly, the performance of a vertical launch of 4.5 m/s in 10 m/s crosswinds (Fig. 6.5) can be extrapolated to the behavior of a full-sized drone launched at 7.8 m/s in a 17 m/s crosswind. The aerodynamically stable behavior, as indicated by the upwind turn, illustrates that the multirotor with deployed fins and motor arms produces a sufficient righting moment to predictably orient the multirotor upwind on launch. While not perfectly analogous (full-scale tests were performed at 12 m/s and a slightly different geometry), these sub-scale trajectories had a similar one-third scaled stability margin (5cm) and provided confidence that the full-sized *SQUID* would have a predictable trajectory if launched from a moving vehicle (a goal for future work).

Transition from Passive to Active Stabilization

SQUID commences the autonomy pipeline once the distance sensor indicates the vehicle has cleared the launch tube. The passive-to-active transition occurs after the vehicle has exited the tube and the arms are fully deployed, allowing the motors to spin. Starting the motors early in the ballistic phase of launch is important as the motors need to be fully spooled up and stabilizing the multirotor before apogee. At apogee, the airspeed may not be sufficient to provide enough aerodynamic stabilization, risking the multirotor entering a tumbling state from which is may not recover.

6.4 Active Stabilization

Our active stabilization solution is based upon previous research into autonomously recovering a monocular vision-based quadrotor after its state-estimator fails due to a loss of visual tracking [114, 115]. For our visual inertial odometry pipeline, we utilize the open-source Robust Visual Inertial Odometry (ROVIO), an extended Kalman Filter that tracks both 3D landmarks and image patch features [116]. Since it tightly integrates image intensity information with intertial data to produce odometry estimates, ROVIO is capable of operating in stark, low-texture environments such as over pavement, water, and the surface of other planets.

The first stage of the active stabilization phase controls the attitude to a nominal zero-roll/pitch orientation using the IMU-based attitude estimate. As the air pressure around the multirotor spikes on launch, the barometric altitude estimates become unreliable and the altitude must be maintained open-loop, biased upwards for safety. The barometric readings stabilize within three seconds of launch, and at this point, *SQUID* begins actively controlling its altitude and attempts to reduce the vertical velocity to zero. As no horizontal position or velocity information is available, active control of the lateral position is not possible and *SQUID* continues to drift in plane until the VIO can be initialized.

Several conditions need to be met before the VIO can be successfully initialized. Firstly, the pitch and roll rates need to be near-zero to ensure that the camera captures frames with low motion blur. Secondly, the vertical velocity needs to be near-zero so the distance between the multirotor and the ground remains constant and the initial feature depth can be well established using rangefinder measurements. Finally, the lateral velocity must be small (once again to minimize motion blur), so the multirotor is allowed to drift for 10 s post spool up to enable aerodynamic drag to bleed off excess speed. Future iterations of the autonomy pipeline will sense when to initialize VIO directly from the detected motion blur, enabling the vehicle to enter position stabilization sooner after launch.

The VIO is considered initialized when the cumulative variance of the VIO's x- and y-position estimates drop below a preset threshold. The pose estimates are then fed into the flight controller state estimator filter to be fused with the IMU. At this point, *SQUID* has full onboard state estimation and can now control both altitude and lateral position.

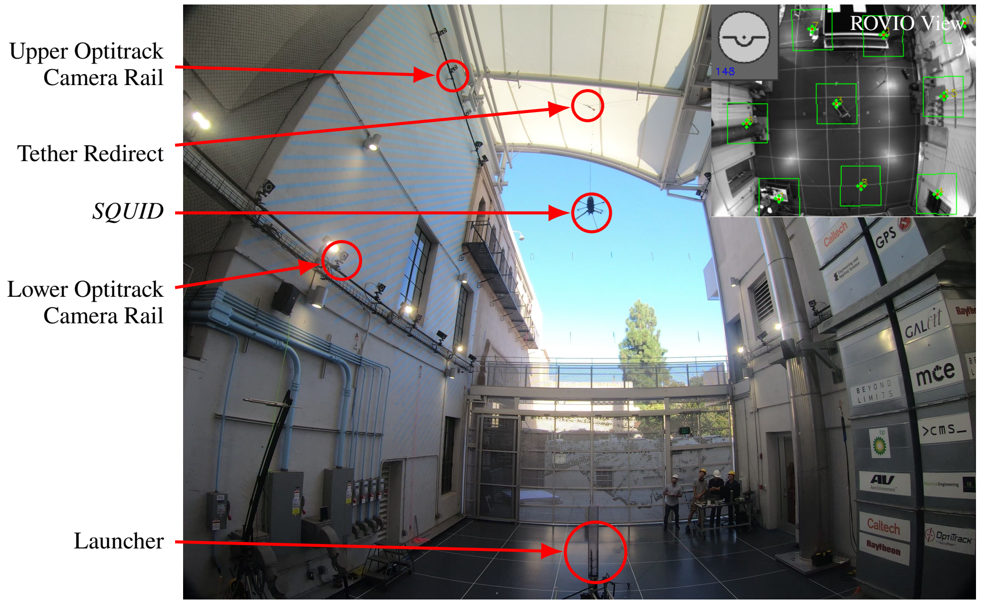


Figure 6.6: Launching *SQUID* in 42 foot-tall flying arena. The arena has two tiers of motion capture cameras.

Experimental Validation

To demonstrate the proposed passive-to-active stabilization pipeline, we launched *SQUID* in a 42 foot-tall flying arena at CAST (Fig. 6.6). The arena has two tiers of Optitrack motion capture cameras allowing *SQUID*'s position and orientation to be tracked throughout the duration of a flight for offline analysis. During initial development, a tether system was constructed inside the arena to prevent the multirotor from damaging the facility in the event of a launch failure. A small weight was used to passively eliminate any slack in the tether. As SQUID accelerates significantly faster than the 1 g of the counterweight (note the slack in the tether in Fig. 6.1), it is unlikely that the tether interfered with the critical passive-to-active attitude stabilization phase.

Fig. 6.8 shows the position tracking of a full launch to active position stabilization test flight. At launch (t=0), altitude is quickly gained as the multirotor accelerates. The motors turn on at Point 1 and begin actively stabilizing the attitude. By Point 2, the barometer has recovered from the launch and closed-loop altitude control commences. Ten seconds after the motors are turned on (Point 3), VIO initialization begins. At Point 4, the VIO is initialized and starts to feed pose estimates to the flight controller, which then actively controls the position of the multirotor, completing the pipeline. The pipeline was successfully demonstrated across several days, lighting conditions, and launch pressures. Footage of the launches can be found at `https://youtu.be/mkotvIK8Dmo`.

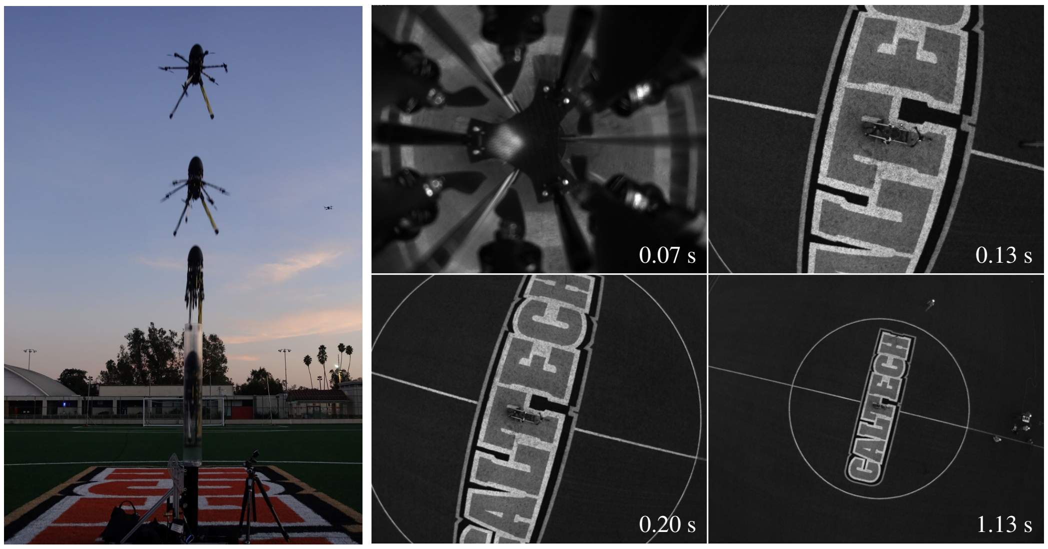


Figure 6.7: Preliminary outdoor free-flight *SQUID* testing.

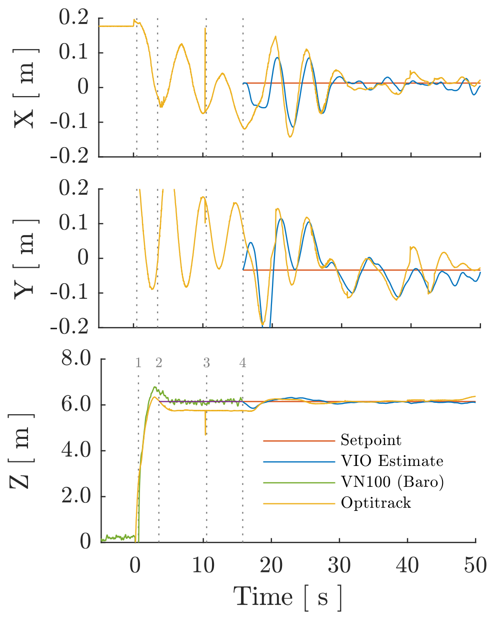


Figure 6.8: Onboard state estimates and ground truth during launch. 1: Motors on, 2: Closed-Loop altitude control, 3: VIO initialization, 4: Position control.

6.5 Summary

SQUID has successfully demonstrated the ability to ballistically launch and transition into autonomous onboard control. In particular, we demonstrate:

1. A 3.3 kg hexacopter with a payload of an advanced sensor package and mission computer.
2. An airframe strong enough to carry and transmit launch loads without damaging onboard components.
3. Passive aerodynamic stability generated by folding fins that set the necessary preconditions for transition to autonomous flight.
4. Wind tunnel testing that validates the proposed multirotor design in cross-wind launches.
5. An autonomy pipeline that carries the platform from launch detection to full 6-degree of freedom stabilization using only onboard sensing (IMU, barometer, rangefinder, and camera) and without the need for GPS.

To further validate the robustness of the presented system, future development of *SQUID* will include outdoor launches in windy/gusty conditions (Fig. 6.7) and launches from a moving vehicle. Planned hardware improvements include a delayed fin- and arm-release trigger to extend the ballistic range.

This proof-of-concept system validates the viability of a ballistically-launched multirotor that deploys without human involvement, opening up new applications in fields such as disaster response, defense, and space exploration.

Chapter 7

CONCLUSION

Inspired by real-world emergency situations, this thesis seeks to develop autonomous robotic systems capable of: (*i*) robustly reasoning about a time-limited mission across a complex environment, and (*ii*) negotiating extreme physical conditions during mission execution. These objectives motivated the development of two proposed autonomy systems: a coverage planning strategy for exploration of large-scale unknown environments (Chaps. 3-5), and a ballistically-launched autonomously-stabilizing multirotor for rapid aerial surveillance (Chap. 6). The specific contributions of this thesis are listed as follows.

- **Hierarchical information-rich structures for spatial and temporal approximation of the coverage policy space.** Spatially, the policy space is approximated by a task-dependent structure enriched with environment map estimates, termed an Information Roadmap (IRM). Temporally, the space is approximated by the aggregation of multiple IRMs, each spanning a different spatial range. Throughout this thesis, two IRMs were discussed and maintained: a local grid-based IRM that captures high-fidelity information in a neighborhood of the robot, and a global graph-based IRM that can span multiple kilometers and encodes free-space connectivity.
- **Cascaded coverage decision making over a stratified policy space.** To bridge the gap between local, risk-aware resiliency and global, reward-seeking mission objectives, a policy constructed globally serves as an input parameter to the local coverage planner. Higher-level policies guide lower-level policies, resulting in a cascaded decision process.
- **Receding-horizon scheme for reconciling consecutive coverage policies.** As the robot explores an environment, it receives new sensory information, updates its belief, and constructs a new coverage policy in a receding horizon fashion. Policies generated during consecutive planning episodes must respect the kinodynamic constraints of the robot, while simultaneously adapting to unexpected hazards in the environment. This reconciliation scheme finds an effective balance between these two, sometimes opposed, objectives.

- **Method to solve the time-limited submodular coverage problem for exploration of unknown environments, modeled by rolling grids.** The time-limited coverage problem on a dense grid exhibits a *diminishing returns* property known as submodularity. Consequently, the robot must evaluate the how its observation of the world affects the utility of future observations, while simultaneously accounting for traversability risk and uncertainty. The proposed method satisfies these requirements by rolling out future robot-world states using an effective coverage sensor model. In particular, the range of the coverage sensor model adapts to the local environment in order replicate ray-surface interactions that regulate surface visibility. This approach obviates the need for expensive ray-tracing operations that make forward rollout algorithms prohibitively slow for a real-time system.

- **Method to solve the time-limited modular coverage problem for exploration of unknown environments, modeled by dynamic graphs.** The time-limited coverage problem on a sparse graph is modular; that is, the marginal benefit of visiting a frontier does not decrease as the set of already visited frontiers increases. While this reduces the complexity of the problem, one must still consider the fact that the graph structure grows and changes dynamically as the robot uncovers new regions. To address this point, a generalization of the orienteering problem (OP), where the robot does not have access to a reliable environment model, is proposed and solved. This OP variant compensates for model uncertainty by incorporating a greedy incentive that shifts information gain earlier in time. By biasing towards short term information gain, the robot visits frontiers earlier with the expectation that they will significantly alter its world understanding by exposing new opportunities for information gathering.

- **Design and autonomous stabilization of a ballistically-launched multirotor.** A foldable hexacopter with an onboard sensor suite, autonomy pipeline, and passive aerodynamic stability was designed, fabricated, and tested. The vehicle demonstrates autonomous transition from passive to vision-based, active stabilization. This confirms the ability of multirotors to autonomously stabilize in a GPS-denied environment after a ballistic launch from a moving or unstable platform.

Future Coverage Planning Work

The proposed coverage planning strategy was tested and evaluated with an understanding that it would need to perform robustly during the Final Circuit of the DARPA Subterranean Challenge. With this in mind, real-world behavior was the primary performance metric. An extensive field test campaign was conducted to evaluate and robustify the proposed coverage planning strategy. While field testing is expensive and time-consuming, when in conjunction with a concrete deadline, it is uniquely effective at exposing fundamental shortcomings in a system. From the field perspective, coverage planning performance degradation was primarily attributed to two major areas: system modeling and global guidance.

System Modeling: Within the field of exploration-driven coverage planning, much focus has been placed on achieving a high fidelity system model (e.g. modeling a coverage sensor's range, incidence angle, etc.), without a critical examination of the overall performance benefits of such a design choice. Rather than strictly attempting to model a system more exactly, further efforts could be directed towards developing clever modeling approximations so that computation can be distributed more effectively. Chapter 4 takes a step in that direction by proposing an adaptive coverage sensor model, which reduces rollout computation. A similar philosophy can be applied to approximating the world model. For the proposed local planner, a careful balance was struck between the Local IRM dimension and the size of the lookahead tree; a large, dense IRM requires that a big tree be constructed in order to find high quality coverage paths. As a consequence for ensuring a reasonably sized tree, the Local IRM was often too sparse to capture risky areas in the environment, creating a domino effect with the introduction of many new issues that had to be addressed. Alternatively, one could design a world model where its fidelity varies according to the underlying Risk Map. Areas of the graph within a neighborhood of risky terrain would have higher resolution, thereby (*i*) increasing the likelihood that high-risk regions are encoded in the model, (*ii*) providing more navigational directions to the robot, and (*iii*) obviating the need for a large lookahead tree.

Global Guidance: Another important avenue of investigation should focus on how to include global guidance in a local plan. While Chapter 3 formalizes a close-knit relationship between a global and local policy, the practical implementation was fairly unsophisticated – if the global planning guidance is within the bounds of the local planning domain, then the local planner has full control. A literature review reveals that a similar switching scheme is used in the majority of other local-global planning proposals. Without a global perspective, the local planner tends to make globally suboptimal decisions, like guiding the robot away from areas of good communications and towards the vicinity of its robot teammates. Attempts were made to encode high fidelity global information into the Local IRM using global distance transforms and Voronoi skeletons, but local performance suffered when the planner was faced with competing objectives.

BIBLIOGRAPHY

[1] *Thai cave rescue: Drones, dogs, drilling and desperation.* https://www.bbc.com/news/world-asia-44652397. 2018.

[2] *Thailand cave rescue: 'Four-day window' for boys to escape.* https://www.bbc.com/news/world-asia-44754335. 2018.

[3] L. Pham. *775183603LP047_Thailand_ Cav.* www.gettyimages.com, Getty Images News Collection #986095734. 2018.

[4] A. Agha-Mohammadi, K. Otsu, B. Morrell, D. Fan, R. Thakker, A. Santamaria-Navarro, S. Kim, A. Bouman, and [63 others]. "NeBula: Quest for Robotic Autonomy in Challenging Environments; TEAM CoSTAR at the DARPA Subterranean Challenge." In: *Journal of Field Robotics (JFR)* (2021).

[5] A. Bouman∗, M. Ginting∗, N. Alatur∗, M. Palieri, D. Fan, T. Touma, T. Pailevanian, S. Kim, K. Otsu, J. Burdick, and A. Agha-Mohammadi. "Autonomous Spot: Long-Range Autonomous Exploration of Extreme Environments with Legged Locomotion." In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020.

[6] K. Ebadi, Y. Chang, M. Palieri, A. Stephens, A. Hatteland, E. Heiden, A. Thakur, B. Morrell, L. Carlone, and A. Agha. "LAMP: Large-scale Autonomous Mapping and Positioning for Exploration of Perceptually-degraded Subterranean Environments." In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2020.

[7] A. Santamaria-navarro, R. Thakker, D. D. Fan, B. Morrell, and A. Agha-mohammadi. "Towards Resilient Autonomous Navigation of Drones." In: *International Symposium on Robotics Research*. 2019.

[8] M. Palieri, B. Morrell, A. Thakur, K. Ebadi, J. Nash, L. Carlone, C. Guaragnella, and A. Agha-mohammadi. "LOCUS: A Multi-Sensor Lidar-Centric Solution for High-Precision Odometry and 3D Mapping in Real-Time." In: *IEEE Robotics and Automation Letters* 5.2 (2020), pp. 2123–2130.

[9] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard. "Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age." In: *IEEE Transactions on robotics* 32.6 (2016), pp. 1309–1332.

[10] J. G. Mangelson, D. Dominic, R. M. Eustice, and R. Vasudevan. "Pairwise Consistent Measurement Set Maximization for Robust Multi-Robot Map Merging." In: *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2018, pp. 2916–2923.

[11] D. D. Fan, K. Otsu, Y. Kubo, A. Dixit, J. Burdick, and A. Agha-mohammadi. "STEP: Stochastic Traversability Evaluation and Planning for Safe Off-Road Navigation." In: *arXiv preprint arXiv:2103.02828*. 2021.

[12] H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwart, and J. Nieto. "Voxblox: Incremental 3D Euclidean Signed Distance Fields for on-board MAV planning." In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Vancouver, BC, 2017.

[13] M. Himmelsbach, F. v. Hundelshausen, and H. .-. Wuensche. "Fast Segmentation of 3D Point Clouds for Ground Vehicles." In: *2010 IEEE Intelligent Vehicles Symposium*. 2010.

[14] P. Krüsi, P. Furgale, M. Bosse, and R. Siegwart. "Driving on Point Clouds: Motion Planning, Trajectory Optimization, and Terrain Assessment in Generic Nonplanar Environments." In: *JFR* 34.5 (2017), pp. 940–984.

[15] H. Choset. "Coverage for Robotics–a Survey of Recent Results." In: *Annals of Mathematics and Artificial Intelligence.* 31.1 (2001), pp. 113–126.

[16] T. M. Cabreira, L. B. Brisolara, and P. R. Ferreira Jr. "Survey on Coverage Path Planning with Unmanned Aerial Vehicles." In: *Drones* 3.1 (2019), p. 4.

[17] Y. Li, H. Chen, M. J. Er, and X. Wang. "Coverage Path Planning for UAVs Based on Enhanced Exact Cellular Decomposition Method." In: *Mechatronics* 21.5 (2011), pp. 876–885.

[18] S. Thrun. "Learning metric-topological maps for indoor mobile robot navigation." In: *Artificial Intelligence* 99.1 (1998), pp. 21–71.

[19] H. Choset and P. Pignon. "Coverage Path Planning: The Boustrophedon Cellular Decomposition." In: *Field and service robotics*. Springer. 1998, pp. 203–209.

[20] E. U. Acar and H. Choset. "Sensor-Based Coverage of Unknown Environments: Incremental Construction of Morse Decompositions." In: *The International Journal of Robotics Research* 21.4 (2002), pp. 345–366.

[21] Z. L. Cao, Y. Huang, and E. L. Hall. "Region Filling Operations with Random Obstacle Avoidance for Mobile Robots." In: *Journal of Robotic systems* 5.2 (1988), pp. 87–102.

[22] E. Rimon. "Construction of C-space Roadmaps from Local Sensory Data. What Should the Sensors Look For?" In: *Algorithmica* 17.4 (1997), pp. 357–379.

[23] M. Coombes, W.-H. Chen, and C. Liu. "Boustrophedon Coverage Path Planning for UAV Aerial Surveys in Wind." In: *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE. 2017, pp. 1563–1571.

[24] J. Faigl and M. Kulich. “On Determination of Goal Candidates in Frontier-based Multi-Robot Exploration.” In: *2013 European Conference on Mobile Robots*. IEEE. 2013, pp. 210–215.

[25] R. Bormann, F. Jordan, J. Hampp, and M. Hägele. “Indoor Coverage Path Planning: Survey, Implementation, Analysis.” In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 1718–1725.

[26] R. Bormann, F. Jordan, W. Li, J. Hampp, and M. Hägele. “Room segmentation: Survey, implementation, and analysis.” In: *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2016, pp. 1019–1026.

[27] S. Garrido, L. Moreno, M. Abderrahim, and F. Martin. “Path planning for Mobile Robot Navigation using Voronoi Diagram and Fast Marching.” In: *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2006, pp. 2376–2381.

[28] B. Yamauchi. “A Frontier-Based Approach for Autonomous Exploration.” In: *Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA’97.’Towards New Computational Principles for Robotics and Automation’*. IEEE. 1997, pp. 146–151.

[29] S. Kim∗, A. Bouman∗, G. Salhotra, D. Fan, K. Otsu, J. Burdick, and A. Agha-Mohammadi. “PLGRIM: Hierarchical Value Learning for Large-scale Exploration in Unknown Environments.” In: *International Conference on Automated Planning and Scheduling (ICAPS)*. 2021.

[30] A. Howard, L. E. Parker, and G. S. Sukhatme. “Experiments with a Large Heterogeneous Mobile Robot Team: Exploration, Mapping, Deployment and Detection.” In: *The International Journal of Robotics Research* 25.5-6 (2006), pp. 431–447.

[31] W. Burgard, M. Moors, C. Stachniss, and F. E. Schneider. “Coordinated Multi-Robot Exploration.” In: *IEEE Transactions on robotics* 21.3 (2005), pp. 376–386.

[32] H. Umari and S. Mukhopadhyay. “Autonomous Robotic Exploration based on Multiple Rapidly-Exploring Randomized Trees.” In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2017, pp. 1396–1402.

[33] M. Keidar and G. A. Kaminka. “Robot Exploration with Fast Frontier Detection: Theory and Experiments.” In: *Autonomous Agents and Multi-Agent Systems (AAMAS)*. 2012, pp. 113–120.

[34] H. Moravec and A. Elfes. “High Resolution Maps from Wide Angle Sonar.” In: *Proceedings. 1985 IEEE international conference on robotics and automation*. Vol. 2. IEEE. 1985, pp. 116–121.

[35] A. Elfes. “Sonar-Based Real-World Mapping and Navigation.” In: *IEEE Journal on Robotics and Automation* 3.3 (1987), pp. 249–265.

[36] A. Zelinsky, R. A. Jarvis, J. Byrne, S. Yuta, et al. “Planning Paths of Complete Coverage of an Unstructured Environment by a Mobile Robot.” In: *Proceedings of international conference on advanced robotics*. Vol. 13. Citeseer. 1993, pp. 533–538.

[37] J. S. Oh, Y. H. Choi, J. B. Park, and Y. F. Zheng. “Complete Coverage Navigation of Cleaning Robots using Triangular-Cell-Based Map.” In: *IEEE Transactions on Industrial Electronics* 51.3 (2004), pp. 718–726.

[38] Y. Gabriely and E. Rimon. “Spanning-tree based coverage of continuous areas by a mobile robot.” In: *Annals of mathematics and artificial intelligence* 31.1 (2001), pp. 77–98.

[39] E. Gonzalez, O. Alvarez, Y. Diaz, C. Parra, and C. Bustacara. “BSA: A complete coverage algorithm.” In: *proceedings of the 2005 IEEE international conference on robotics and automation*. IEEE. 2005, pp. 2040–2044.

[40] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. “Planning and acting in Partially Observable stochastic domains”. In: *Artificial Intelligence* 101 (1998), pp. 99–134.

[41] M. J. Kochenderfer, T. A. Wheeler, and K. H. Wray. *Algorithms for decision making*. MIT Press, 2022.

[42] S. Ross, J. Pineau, S. Paquet, and B. Chaib-Draa. “Online planning algorithms for POMDPs.” In: *Journal of Artificial Intelligence Research* 32 (2008), pp. 663–704.

[43] J. G. Blank. “Robotic Mapping and Exploration of a Terrestrial Lava Tube: A Structured Planetary Cave Mission Simulation with a Remote Astrobiology Science Team.” In: *3rd International Planetary Caves Conference*. 2020.

[44] K. Nagatani, S. Kiribayashi, Y. Okada, K. Otake, K. Yoshida, S. Tadokoro, T. Nishimura, T. Yoshida, E. Koyanagi, M. Fukushima, et al. “Emergency response to the nuclear accident at the Fukushima Daiichi Nuclear Power Plants using mobile rescue robots.” In: *JFR* 30.1 (2013), pp. 44–63.

[45] G. E. Monahan. “State of the art—a survey of partially observable Markov decision processes: theory, models, and algorithms.” In: *Management science* 28.1 (1982), pp. 1–16.

[46] J. Pineau, G. Gordon, and S. Thrun. “Point-based value iteration: An anytime algorithm for POMDPs.” In: *IJCAI*. 2003, pp. 1025–1032.

[47] D. Silver and J. Veness. “Monte-Carlo planning in large POMDPs.” In: *NeurIPS*. 2010, pp. 2164–2172.

[48] A. Somani, N. Ye, D. Hsu, and W. S. Lee. “DESPOT: Online POMDP Planning with Regularization.” In: *NeurIPS*. 2013, pp. 1772–1780.

[49] B. Bonet and H. Geffner. “Learning Sorting and Decision Trees with POMDPs.” In: *Proceedings of the International Conference on Machine learning (ICML)*. 1998, pp. 73–81.

[50] S.-K. Kim, O. Salzman, and M. Likhachev. “POMHDP: Search-based belief space planning using multiple heuristics”. In: *International Conference on Automated Planning and Scheduling (ICAPS)*. Vol. 29. 2019, pp. 734–744.

[51] A. Agha-Mohammadi, S. Chakravorty, and N. Amato. “FIRM: Sampling-based Feedback Motion Planning Under Motion Uncertainty and Imperfect Measurements.” In: *The International Journal of Robotics Research (IJRR)* 33.2 (2014), pp. 268–304.

[52] T. Tao, Y. Huang, F. Sun, and T. Wang. “Motion planning for slam based on frontier exploration.” In: *2007 International Conference on Mechatronics and Automation*. 2007, pp. 2120–2125.

[53] L. Heng, A. Gotovos, A. Krause, and M. Pollefeys. “Efficient visual exploration and coverage with a micro aerial vehicle in unknown environments.” In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2015, pp. 1071–1078.

[54] H. H. González-Banos and J.-C. Latombe. “Navigation strategies for exploring indoor environments.” In: *The International Journal of Robotics Research* 21.10-11 (2002), pp. 829–848.

[55] R. Grabowski, P. Khosla, and H. Choset. “Autonomous exploration via regions of interest.” In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Vol. 2. 2003, pp. 1691–1696.

[56] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell. “Curiosity-driven Exploration by Self-supervised Prediction.” In: *arXiv preprint arXiv:1705.05363*. 2017.

[57] Y. Burda, H. Edwards, D. Pathak, A. Storkey, T. Darrell, and A. Efros. “Large-Scale Study of Curiosity-Driven Learning.” In: *arXiv:1808.04355*. 2018.

[58] Y. Burda, H. A. Edwards, A. J. Storkey, and O. Klimov. “Exploration by Random Network Distillation.” In: *arXiv preprint arXiv:1810.12894*. 2018.

[59] N. Savinov, A. Raichuk, R. Marinier, D. Vincent, M. Pollefeys, T. Lillicrap, and S. Gelly. “Episodic Curiosity through Reachability.” In: *arXiv preprint arXiv:1810.02274*. 2018.

[60] H. Kurniawati, Y. Du, D. Hsu, and W. S. Lee. “Motion planning under uncertainty for robotic tasks with long time horizons”. In: *The International Journal of Robotics Research (IJRR)* 30.3 (2011), pp. 308–323.

[61] H. Bai, S. Cai, N. Ye, D. Hsu, and W. S. Lee. “Intention-aware online POMDP planning for autonomous driving in a crowd.” In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2015, pp. 454–460.

[62] V. Indelman, L. Carlone, and F. Dellaert. “Planning in the continuous domain: A generalized belief space approach for autonomous navigation in unknown environments.” In: *The International Journal of Robotics Research (IJRR)* 34.7 (2015), pp. 849–882.

[63] R. Martinez-Cantin, N. De Freitas, E. Brochu, J. Castellanos, and A. Doucet. “A Bayesian exploration-exploitation approach for optimal online sensing and planning with a visually guided mobile robot.” In: *Autonomous Robots* 27.2 (2009), pp. 93–103.

[64] M. Lauri and R. Ritala. “Planning for Robotic Exploration Based on Forward Simulation.” In: *Robotics and Autonomous Systems* 83 (2016), pp. 15–31.

[65] L. P. Kaelbling and T. Lozano-Pérez. “Planning in the know: Hierarchical belief-space task and motion planning.” In: *Workshop on Mobile Manipulation, IEEE ICRA*. 2011.

[66] T. Dang, S. Khattak, F. Mascarich, and K. Alexis. “Explore Locally, Plan Globally: A Path Planning Framework for Autonomous Robotic Exploration in Subterranean Environments.” In: *Proceedings of the International Conference on Advanced Robotics (ICAR)*. 2019, pp. 9–16.

[67] S.-K. Kim, R. Thakker, and A. Agha-Mohammadi. “Bi-directional value learning for risk-aware planning under uncertainty.” In: *IEEE Robotics and Automation Letters* 4.3 (2019), pp. 2493–2500.

[68] N. A. Vien and M. Toussaint. “Hierarchical Monte-Carlo Planning.” In: *Twenty-Ninth AAAI Conference on Artificial Intelligence*. 2015.

[69] A. Bircher, M. Kamel, K. Alexis, H. Oleynikova, and R. Siegwart. “Receding horizon "next-best-view" planner for 3D exploration.” In: *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2016, pp. 1462–1468.

[70] M. Littman, A. Cassandra, and L. Kaelbling. “Learning policies for partially observable environments: Scaling up.” In: *Machine Learning Proceedings*. 1995, pp. 362–370.

[71] K. Otsu, S. Tepsuporn, R. Thakker, T. S. Vaquero, J. A. Edlund, W. Walsh, G. Miles, T. Heywood, M. T. Wolf, and A. Agha-mohammadi. “Supervised autonomy for communication-degraded subterranean exploration by a robot team.” In: *IEEE Aerospace Conference*. 2020.

[72] T. Dang, M. Tranzatto, S. Khattak, F. Mascarich, K. Alexis, and M. Hutter. “Graph-based subterranean exploration path planning using aerial and legged robots.” In: *Journal of Field Robotics* 37.8 (2020), pp. 1363–1388.

[73] C. Witting, M. Fehr, R. Bähnemann, H. Oleynikova, and R. Siegwart. “History-aware autonomous exploration in confined environments using MAVs.” In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 1–9.

[74] S. K. Ghosh. *Visibility algorithms in the plane*. Cambridge University Press, 2007.

[75] A. Krause and D. Golovin. “Submodular function maximization.” In: *Tractability* 3 (2014), pp. 71–104.

[76] C. Cao, H. Zhu, H. Choset, and J. Zhang. “TARE: A Hierarchical Framework for Efficiently Exploring Complex 3D Environments.” In: *Robotics: Science and Systems Conference (RSS), Virtual*. 2021.

[77] C. H. Papadimitriou. “The complexity of the Lin–Kernighan heuristic for the traveling salesman problem.” In: *SIAM Journal on Computing* 21.3 (1992), pp. 450–465.

[78] J. Current, H. Pirkul, and E. Rolland. “Efficient algorithms for solving the shortest covering path problem.” In: *Transportation Science* 28.4 (1994), pp. 317–327.

[79] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton. “A survey of Monte Carlo Tree Search methods.” In: *IEEE Transactions on Computational Intelligence and AI in games* 4.1 (2012), pp. 1–43.

[80] K. Chen, B. T. Lopez, A. Agha-mohammadi, and A. Mehta. “Direct LiDAR Odometry: Fast Localization With Dense Point Clouds.” In: *IEEE Robotics and Automation Letters* 7.2 (2022), pp. 2000–2007. DOI: 10.1109/LRA.2022.3142739.

[81] M. Roberts, D. Dey, A. Truong, S. Sinha, S. Shah, A. Kapoor, P. Hanrahan, and N. Joshi. “Submodular trajectory optimization for aerial 3d scanning.” In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 5324–5333.

[82] I.-M. Chao, B. L. Golden, and E. A. Wasil. “The team orienteering problem.” In: *European Journal of Operational Research* 88.3 (1996), pp. 464–474.

[83] C. Cao, H. Zhu, H. Choset, and J. Zhang. “TARE: A Hierarchical Framework for Efficiently Exploring Complex 3D Environments.” In: 2021.

[84] P. Vansteenwegen, W. Souffriau, G. V. Berghe, and D. Van Oudheusden. “A guided local search metaheuristic for the team orienteering problem.” In: *European Journal of Operational Research* 196.1 (2009), pp. 118–127.

[85] B. Fang, J. Ding, and Z. Wang. “Autonomous robotic exploration based on frontier point optimization and multistep path planning.” In: *IEEE Access* 7 (2019), pp. 46104–46113.

[86] C. Wang, W. Chi, Y. Sun, and M. Q.-H. Meng. “Autonomous robotic exploration by incremental road map construction.” In: *IEEE Transactions on Automation Science and Engineering* 16.4 (2019), pp. 1720–1731.

[87] T. Dang, F. Mascarich, S. Khattak, C. Papachristos, and K. Alexis. "Graph-based path planning for autonomous robotic exploration in subterranean environments." In: IEEE. 2019, pp. 3105–3112.

[88] T. Roucek, M. Pecka, P. Čížek, T. Petrıcek, J. Bayer, V. Salanský, T. Azayev, D. Hert, M. Petrlık, T. Báca, V. Spurný, V. Krátký, P. Petrácek, D. Baril, M. Vaidis, V. Kubelka, F. Pomerleau, J. Faigl, K. Zimmermann, M. Saska, T. Svoboda, and T. Krajnık. "System for multi-robotic exploration of underground environments CTU-CRAS-NORLAB in the DARPA Subterranean Challenge." In: *arXiv preprint arXiv:2110.05911* (2021).

[89] J. Williams, S. Jiang, M. O'Brien, G. Wagner, E. Hernandez, M. Cox, A. Pitt, R. Arkin, and N. Hudson. "Online 3D Frontier-Based UGV and UAV Exploration Using Direct Point Cloud Visibility." In: *IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*. IEEE. 2020, pp. 263–270.

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

[91] J. Binney, A. Krause, and G. S. Sukhatme. "Informative path planning for an autonomous underwater vehicle." In: IEEE. 2010, pp. 4791–4796.

[92] B. Liu, X. Xiao, and P. Stone. "Team Orienteering Coverage Planning with Uncertain Reward." In: *arXiv preprint arXiv:2105.03721* (2021).

[93] J. Yu, M. Schwager, and D. Rus. "Correlated orienteering problem and its application to persistent monitoring tasks." In: *IEEE Transactions on Robotics* 32.5 (2016), pp. 1106–1118.

[94] D. Filliat and J.-A. Meyer. "Map-based navigation in mobile robots: I. a review of localization strategies." In: *Cognitive Systems Research* 4.4 (2003), pp. 243–282.

[95] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, et al. "A density-based algorithm for discovering clusters in large spatial databases with noise." In: *kdd*. Vol. 96. 34. 1996, pp. 226–231.

[96] A. Visser, M. v. Ittersum, G. Jaime, A. Luis, L. A. Stancu, et al. "Beyond frontier exploration." In: *Robot Soccer World Cup*. Springer. 2007, pp. 113–123.

[97] NASA. *Mars Helicopter to Fly on NASA's Next Red Planet Rover Mission*. May 2018 (accessed on January 2019). URL: `www.nasa.gov/press-release/mars-helicopter-to-fly-on-nasa-s-next-red-planet-rover-mission`.

[98] Raytheon. *Coyote UAS*. (Accessed July 2019). URL: `https://www.raytheon.com/capabilities/products/coyote`.

[99] UVision. *Hero UAV*. (Accessed on July 2019). URL: `https://uvisionuav.com/main-products/`.

[100] Leonardo. *Horus-detail-Leonardo*. (Accessed on July 2019). URL: `https://www.leonardocompany.com/en/allproducts`.

[101] DJI. *Mavic 2-DJI Store*. (Accessed on 01/2019). URL: `https://store.dji.com/product/mavic-2`.

[102] Parrot. *Drone Camera 4k HDR ANAFI*. (Accessed on 01/2019). URL: `https://www.parrot.com/us/drones/anafi`.

[103] Powervision. *PowerEgg Camera Drone, Fly To The Future*. (accessed on January 2019). URL: `https://www.powervision.me/en/product/poweregg`.

[104] LeveTop. *The Foldable & Portable Drone*. URL: `https://www.levetop.com/`.

[105] A. AeroSystems. *Ascent AeroSystems*. 2019 (accessed on January 2019). URL: `http://www.ascentaerosystems.com/`.

[106] D. Falanga, K. Kleber, S. Mintchev, D. Floreano, and D. Scaramuzza. "The Foldable Drone: A Morphing Quadrotor That Can Squeeze and Fly." In: *IEEE Robotics and Automation Letters*. IEEE. 2019.

[107] N. Bucki and M. Mueller. "Design and Control of a Passively Morphing Quadcopter." In: *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2019.

[108] P. Gnemmi, S. Changey, K. Meder, E. Roussel, C. Rey, C. Steinbach, and C. Berner. "Conception and manufacturing of a projectile-drone hybrid system." In: *IEEE/ASME Transactions on Mechatronics* 22.2 (2017), pp. 940–951.

[109] L. Henderson, T. Glaser, and F. Kuester. "Towards bio-inspired structural design of a 3D printable, ballistically deployable, multi-rotor UAV." In: *Aerospace Conference, 2017 IEEE*. IEEE. 2017, pp. 1–7.

[110] D. Pastor, J. Izraelevitz, P. Nadan, A. Bouman, J. Burdick, and B. Kennedy. "Design of a Ballistically-Launched Foldable Multirotor." In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2019.

[111] S. F. Hoerner. *Fluid-dynamic Drag: practical information on aerodynamic drag and hydrodynamic resistance*. Hoerner Fluid Dynamics, 1958.

[112] S. Hoerner. "Fluid-dynamic lift." In: *Hoerner Fluid Dynamics* (1985).

[113] tshirtguns.com. *Bleacher Reacher Mega T-Shirt Launcher*. URL: `http://tshirtgun.com/bleacher%5C_reacher%5C_mega%5C_2014.pdf`.

[114] M. Faessler, F. Fontana, C. Forster, and D. Scaramuzza. “Automatic Re-Initialization and Failure Recovery for Aggressive Flight with a Monocular Vision-Based Quadrotor.” In: *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2015.

[115] R. Brockers, M. Humenberger, D. Weiss, and L. Matthies. “Towards autonomous navigation of miniature UAV.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. IEEE. 2014.

[116] M. Bloesch, M. Burri, S. Omari, M. Hutter, and R. Siegwart. “Iterated extended Kalman filter based visual-inertial odometry using direct photometric feedback.” In: *The International Journal of Robotics Research* 36.10 (2017), pp. 1053–1072.