Rapid Subsurface Exploration with Multiple Aerial Robots

Kshitij Goel

CMU-RI-TR-21-23

August 2021

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

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Robotics.

Abstract

This thesis develops a robotic exploration framework that allows for rapid and communication-efficient mapping of subsurface environments with a team of aerial robots.

Aerial robots can provide rapid and agile mobility in diverse environments where ground mobility is either severely constrained or impossible. However, high-speed flight with such robots poses challenges due to limited sensing range, on-board computation, and constrained dynamics. For operation in unknown environments, the planning subsystem must guarantee collision-free operation, and for exploration tasks, the system should also select sensing actions to maximize information gain with respect to the environment. To this end, the first contribution of this thesis is a motion primitive-based, receding-horizon planning approach that maximizes information gain, accounts for platform dynamics, and ensures safe operation. Analysis of motions parallel and perpendicular to frontiers given constraints on sensing and dynamics leads to bounds on safe velocities for exploration. These bounds inform the design of the motion primitive approach. Experimental results on a hexarotor robot demonstrate rapid exploration at state-of-the-art speeds in an outdoor environment.

Deploying a team of these robots can further improve the rate of exploration. Challenges imposed by the communication bottlenecks in such deployments towards human-robot and inter-robot coordination have been left largely unaddressed in prior works. Effective coordination often requires high-quality perceptual feedback, and the gap in the state of the art is the lack of efficiency in the communication of such feedback. To this end, the second contribution of this thesis is a distributed perceptual modeling approach that enables high-fidelity mapping while remaining amenable to low-bandwidth communication channels. The approach yields significant gains in exploration rate for multi-robot teams as compared to state-of-the-art approaches. The approach is evaluated through simulation studies and hardware experiments in a wild cave in West Virginia, USA.

Acknowledgements

I would like to thank my advisors Prof. Nathan Michael and Prof. Wennie Tabib, for their mentorship during the past few years of developing the work presented in this thesis. Nate supported me to work as an RA in his lab, which motivated me to become a better researcher and eventually pursue graduate studies at RI, leading to this thesis. Wennie's leadership and insight helped me contribute to her cave exploration work, without which many chapters of this thesis would not exist. I am also grateful to my other thesis committee members, Prof. David Wettergreen and Paloma Sodhi, for their valuable feedback and important questions on the results presented in this thesis.

I have had the opportunity of collaborating on diverse research themes with many researchers at the Resilient Intelligent Systems Laboratory (RISLab): Dr. Micah Corah, Cormac O'Meadhra, Dr. John Yao, Alex Spitzer, Xuning Yang, Aditya Dhawale, Mosam Dabhi, Matt Collins, and Curtis Boirum. Each of these collaborations has contributed to the system presented in this thesis in some way. Special thanks to Micah and Xuning for being supportive mentors when I started out as a robotics researcher. One of my collaborations with Micah, that built on Xuning's work, is a chapter in this thesis. I am also thankful to my colleagues at RISLab with whom I did not get an opportunity to collaborate, but were always available to provide feedback on paper manuscripts, practice talks, and research ideas: Arjav Desai, Dr. Kumar Shaurya Shankar, Dr. Ellen Cappo, Dr. Vishnu Desaraju, Shobhit Srivastava, and Tim Lee.

This thesis contains field results from robotic deployments in caves. The logistics of these deployments would not have been possible without the help from Brian Osbun, Carroll Bassett, Hope Brooks, Ryan Maurer, David Cale, and Dennis Melko.

I also want to thank the MS program administrators, Barbara (B.J.) Fecich and Prof. George Kantor, for being patient with my endless stream of questions. I am also grateful to Rachel Burcin and Prof. John Dolan for always being available for a chat, ever since my RISS internship days.

Finally, thanks go to my family for supporting me during some of the most testing times in my professional and personal life.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Planetary subsurface voids are known to be of geological and astrobiological significance for future planetary exploration [1]. In particular, exploration of planetary caves can potentially provide records of geological, meteorological, and environmental history of that planet [2]. Moreover, these cave environments provide shelter from ionizing space radiation and stable climatic conditions as opposed to the surface thermal conditions. Such buffered conditions may allow these environments to serve as future human space habitats [3, 4, 5]. About 1036 potential cave entrances have been identified on Mars along with several hundreds on other celestial bodies in our solar system [2, 6]. However, much less is known about these subsurface environments beyond the entrances. This warrants further investigation via robotic missions for data collection and scouting [5]. Robotic precursor missions have engaged in surface exploration of Mars [7] but have not explored subsurface environments. As a result, robotic subsurface exploration has been identified as a key technology for future planetary exploration missions [2, 8, 9].

Autonomous navigation and high-resolution perceptual modeling are critical needs in the context of robotic subsurface planetary exploration [11]. A challenge of operating in subsurface environments is communicating to a surface station. Communication may be limited or impossible due to the inability of radio waves to penetrate rock, impeding data relay to Earth, so compact data transmission is essential. Operating on planets far from Earth introduces additional restrictions on power and compute that may be mitigated by leveraging multiple robots to increase coverage in spatially expansive environments [12].

Exploration frameworks cannot assume *a priori* knowledge about the structure of the environment so the exploration system must operate with unknown locomotion constraints. Aerial robots have recently been leveraged to mitigate these constraints in the subterranean domain [13] and considered for subsurface mapping on Mars [10] (Fig. 1.1). In this thesis,

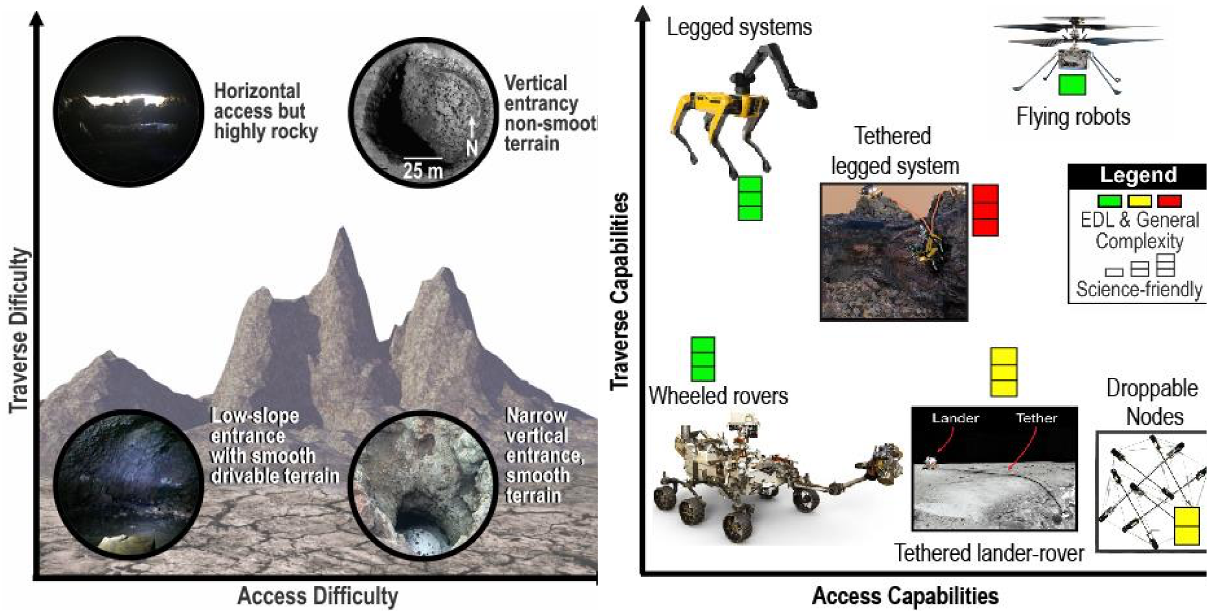


Figure 1.1: (Left) Traversal and access difficulty for caves on Mars. (Right) Traversal and access capabilities of existing robots. While the choice of the robot platform and the deployment site depend largely on the scientific objectives of the mission, flying robots provide the most traversal and access capabilities and the least EDL and general complexity. The robotic exploration framework developed in this thesis uses a team of aerial robots. (*Source: [10], NASA/JPL, Boston Dynamics*)

we consider aerial robots operating in a cave on Earth (Fig. 1.2) as an analog scenario for subsurface exploration on Mars. These robots are often limited by size, weight, and power (SWaP) constraints [13]. Energy constraints on these platforms impose limits on flight endurance necessitating rapid exploration, since the existence of a replenishment infrastructure in the planetary exploration context cannot be guaranteed at these sites [11]. Several frameworks for rapid exploration have been proposed that either use a single fast-moving aerial robot [14, 15, 16, 17] or multiple slow-moving aerial robots [18, 19]; however, a real-world deployable framework that combines the elements from both is desirable. This thesis addresses a key challenge for planetary exploration: enabling rapid multi-robot exploration in subsurface environments by leveraging a perceptual modeling framework amenable to low-bandwidth communication while remaining high-fidelity.

1.1 Thesis Problem

1.1.1 Concept of Operations

This thesis is motivated by a concept of operations for a subsurface mapping mission on Mars using a team of aerial robots, similar to the concept surface mission studied

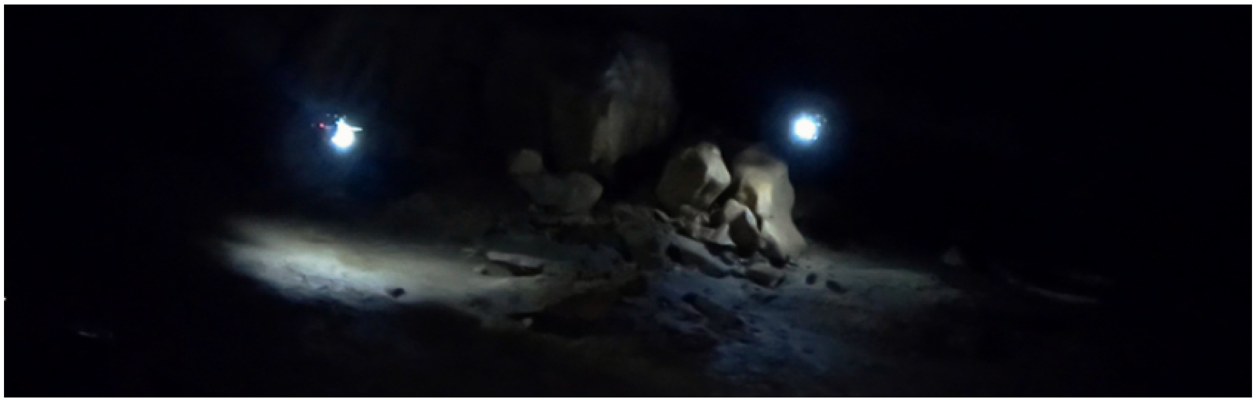


Figure 1.2: Cave exploration with two aerial robots in West Virginia, USA. The robotic exploration framework developed in this thesis is evaluated in Mars-analog caves on Earth. A video of the flight can be accessed at the following link: `https://youtu.be/osko8EKKZUM`.

by Matthies [20] for Titan. It is assumed that the mission can be realized by sending multiple aerial robots ("daughtercraft") from a surface station ("mothership") to perform rapid, effective, and affordable high-resolution mapping of the subsurface environment.

Communication There are three communication channels available in this concept, as shown in Fig. 1.3, with the following assumptions:

- **Daughtercraft - Mothership:** Whittaker et al. [11] suggest the use of either very low frequency (VLF) radios or magneto-inductive (MI) links to achieve limited data rate through thick layers of rock. The MI links in particular can provide approximately 20-25 m dry soil penetration at channel capacity ranging from 0.1-0.25 Mbit/s when using small antennas (coils) [21]. In this thesis, it is assumed that the daughtercraft are equipped with these MI links.
- **Mothership - Orbiter:** Orbiters can communicate at approximately 0.208-0.521 Mbit/s with a surface station for 8 minutes per sol, or Martian day [22].
- **Orbiter - Earth:** To transmit from the orbiter to Earth, the communication rate depends on which orbiter is above the lander to relay the data to Earth. This thesis assumes the lowest data rate from the Mars Odyssey orbiter, which ranges from 0.128-0.256 Mbit/s [22].

The bottleneck in communication is between the subsurface aerial robot (daughtercraft) and surface station (mothership) when the robot is transmitting at depths between 20–25 m below ground, so an analysis of the exploration system is provided for both 0.1 Mbit/s and 0.25 Mbit/s rates.

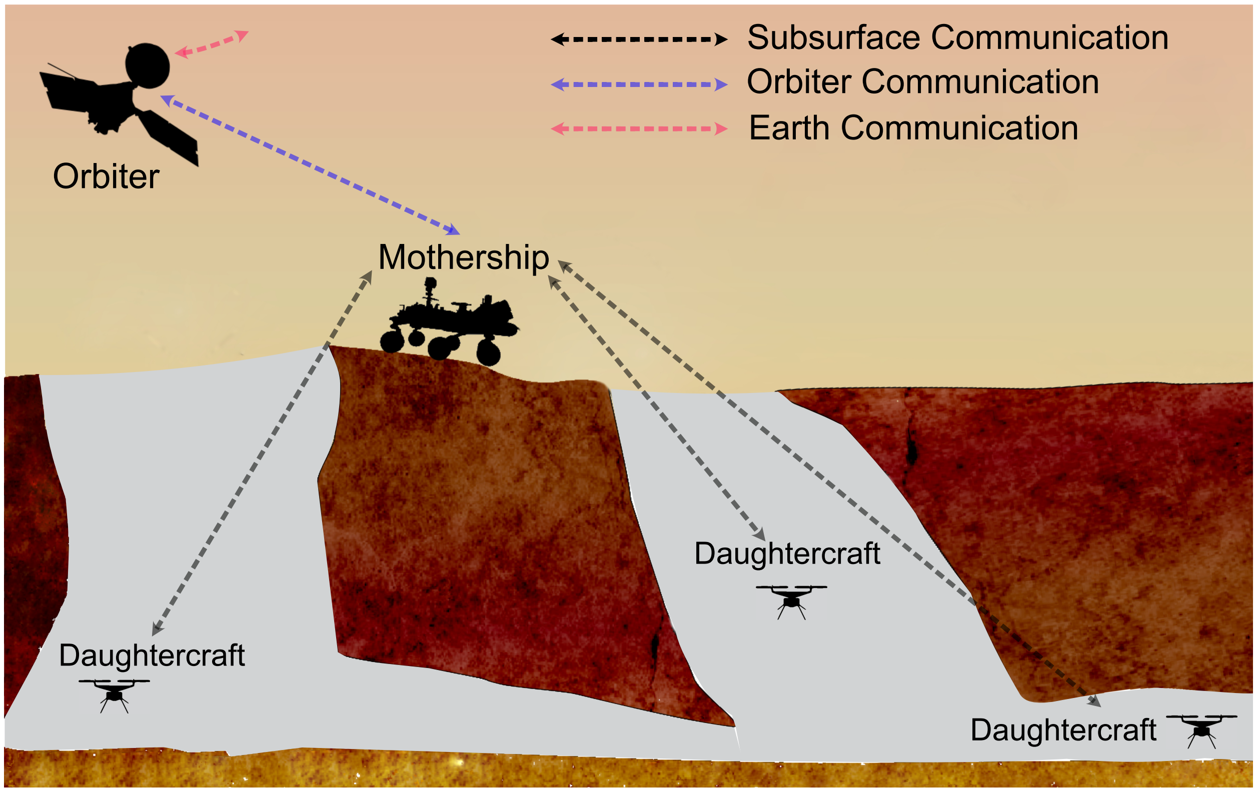


Figure 1.3: Illustration of the concept of operations for planetary subsurface exploration that motivates this thesis. A surface station ("mothership") deploys a team of aerial robots ("daughtercraft") in the subsurface environment for the exploration task. There are three low-bandwidth communication channels available: daughtercraft - mothership (subsurface communication), mothership - orbiter (orbiter communication), and orbiter - Earth (Earth communication). The daughtercraft are assumed to be energy-constrained with respect to the mothership. The multi-robot exploration framework developed in this thesis can allow for rapid navigation of energy-constrained robots in subsurface environments and communication-efficient map transmission over low-bandwidth channels. (*Original image: [2]*)

Sensing The subsurface aerial robots can be equipped with limited-range depth sensors (for example, LiDARs and stereo-depth sensors). Such sensors can have different field-of-view (FoV), for example, LiDAR sensors usually have a $360°$ FoV while depth cameras have an FoV $< 180°$. We assume that either kind of sensors can be equipped on the aerial robots, as has been done on previous exploration systems.

1.1.2 Challenges

Within this concept of operations, the multi-robot exploration system should address the following challenges:

- **Safe motion planning for exploration at high speeds:** Energy-constrained daughtercraft need to explore the environment rapidly to avoid repeated replenishment, while maintaining a collision-free operation. Since daughtercraft are aerial robots

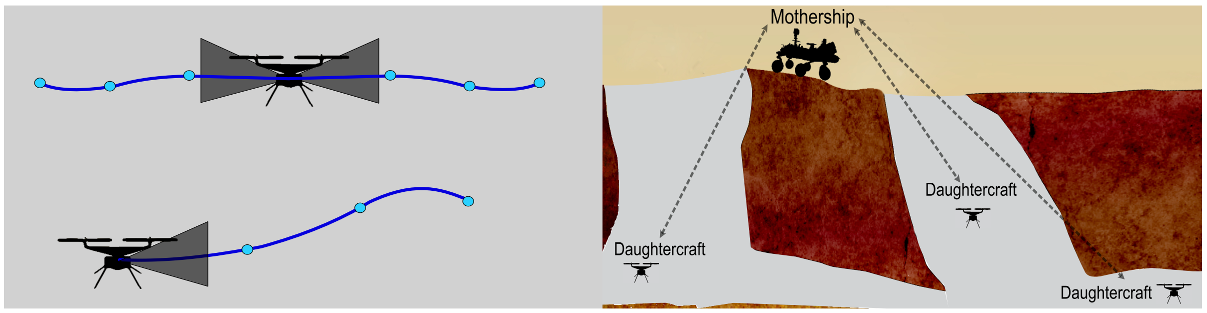


Figure 1.4: Overview of the challenges addressed by the multi-robot exploration framework addressed in this thesis. (Left) Safe informative motion planning at high speeds for both 360° and limited FoV sensing. (Right) Communication-efficient distributed mapping that enables map transmission over low-bandwidth communication channels.

equipped with limited-range depth sensors for mapping, safe informative planning requires maximizing the information gained about the unknown space by operating the robots at high speeds under dynamics and sensing constraints.

- **Sensor-agnostic motion planning for exploration:** Sensors with different FoV can impact the strategy with which daughtercraft explore the subsurface environment. Therefore, the motion planner must be sensor-agnostic and allow for a high rate of information gain per unit time for both 360° LiDARs and limited-FoV depth sensors.
- **Distributed, high-fidelity, and compact mapping:** Communication limitations enforce the requirement to create compact subsurface maps for real-time and low-latency transmission. Scientific objectives require these maps to be of a high perceptual detail. Thus, the multi-robot exploration system must allow for distributed mapping operation that generates a compact and high-fidelity map for efficient transmission.

1.2 Thesis Contributions and Outline

To address these challenges, this thesis proposes a multi-robot exploration framework that enables safe and rapid exploration via an information-theoretic receding horizon motion planner, and leverages prior work in Gaussian Mixture Model (GMM)-based mapping to create a compact distributed mapping system that represents the surface at a high-fidelity and allows for occupancy modeling for information-theoretic exploration. The system is evaluated in real Mars-analog caves on Earth, both in single-robot and multi-robot experimental settings.

The contributions of this thesis are summarized below and are detailed in the subsequent chapters.

Chapter 4: Rapid Motion Primitives-based Single-Robot Exploration A motion primitives-based, receding-horizon planning approach that maximizes information gain, accounts for platform dynamics, and ensures safe operation. Simulation experiments in a complex 3D environment demonstrate the utility of the motion primitive actions for rapid exploration and provide a comparison to a reduced motion primitive library that is appropriate for online planning. Experimental results on a hexarotor robot with the reduced library demonstrate rapid exploration at speeds above 2.25 m/s under varying clutter in an outdoor environment which is comparable to and exceeding the existing state-of-the-art results [15].

Chapter 5: Communication-Efficient Single-Robot Exploration An information-theoretic exploration strategy to explore cave environments that compactly represents sensor observations as Gaussian mixture models and maintains a local occupancy grid map for a motion planner that greedily maximizes an information-theoretic objective function. The approach accommodates both limited field of view depth cameras and larger field of view LiDAR sensors and is extensively evaluated in long duration simulations on an embedded PC. The system is deployed in Laurel Caverns, a commercially owned and operated cave in southwestern Pennsylvania, USA, and a wild cave in West Virginia, USA [13].

Chapter 6: Rapid and Communication-Efficient Multi-Robot Exploration A multi-robot exploration framework that leverages the work from the previous two chapters to enable high-fidelity distributed mapping at high speeds while remaining amenable to low-bandwidth communication channels. The approach yields significant gains in exploration rate for multi-robot teams as compared to state-of-the-art approaches. The system is evaluated through simulation studies and hardware experiments in a wild cave in West Virginia [23].

Chapter 2

Related Work

This thesis develops a robotic exploration framework that allows for rapid and communication-efficient mapping of subsurface environments with a team of aerial robots. The proposed framework addresses research gaps in two subsystems of existing multi-robot exploration frameworks (Section 2.1): (1) motion planning for rapid exploration (Section 2.2) and (2) distributed mapping for exploration (Section 2.3).

2.1 Subsurface Multi-Robot Exploration Frameworks

With the ongoing DARPA Subterranean Challenge [24], there is an increased interest in deploying a team of robots in subsurface environments [25, 26, 27]. At the time of writing this thesis, research and development towards the last phase of the challenge is in progress. This phase requires the teams to demonstrate multi-robot exploration capabilities in cave

Figure 2.1: Motivation to use aerial robots for cave exploration. Delicate formations (left) and challenging terrain (right) in a subterranean cave. Aerial robots are promising robotic platforms that can enable rapid autonomy in such domains [10]. *Credit: C. Bassett*

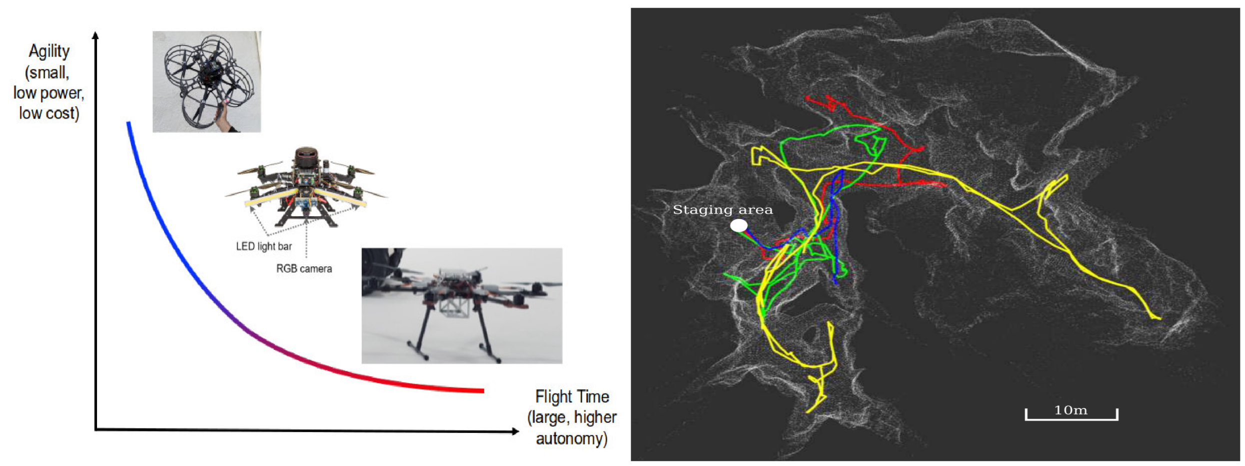

Agility (small, low power, low cost)
LED light bar
RGB camera
Flight Time (large, higher autonomy)
Staging area
10m

Figure 2.2: Trade-offs in sizing of the aerial robots for cave exploration. (Left) Aerial robots developed by Agha et al. [25] plotted with respect to their agility and flight time. (Right) During a cave exploration experiment, Hudson et al. [26] observed that their aerial robot was able to explore a distant part of the cave after passing through a narrow passage that the ground robots could not access (yellow trajectory). There exists an accessibility versus flight time trade-off in aerial robot design for the cave exploration application.

networks, which is the domain of interest for this thesis. Various kinds of robotic platforms have been proposed by the teams (legged, ground, aerial, etc.); we focus on the frameworks that allow for operation with multiple aerial robots due to the reasons stated in Chapter 1.

Agha et al. [25] present the NeBula (Networked Belief-aware Perceptual Autonomy) architecture for cave exploration. Specifically, towards using an aerial robot for exploration, many platforms have been developed (Fig. 2.2) and field results have been reported for mine environments. One representative experiment in the Beckley Mine, Morgan Town, WV, shows the aerial robot being able to explore with an average speed of 1.0 m/s over a flight duration of 45 s [28]. The robot is equipped with a 2D LiDAR for navigation and planning via a discrete grid-based representation of the environment. Hudson et al. [26] present cave exploration results using a heterogeneous team of robots (Fig. 2.2), one of them being an Emesent[1] drone with a custom sensor pack. The exploration strategy uses a point cloud representation of the environment to drive the robot into previously unexplored spaces. In the two cave experiments where this aerial robot was used, the average distance traversed was 271.5 m over an average duration of 684 s (i.e., an average speed of 0.39 m/s). Petracek et al. [27] propose a subsurface exploration system that can use multiple aerial robots. Experiments with a single robot in the Bull Rock Cave system demonstrate large-scale mapping with robot speeds reported up to 2.0 m/s. However, the grid-based representation lacks perceptual detail as the maps built onboard the robot use a 20 cm voxel size.

[1]Emesent: https://www.emesent.io/autonomy-level-2/

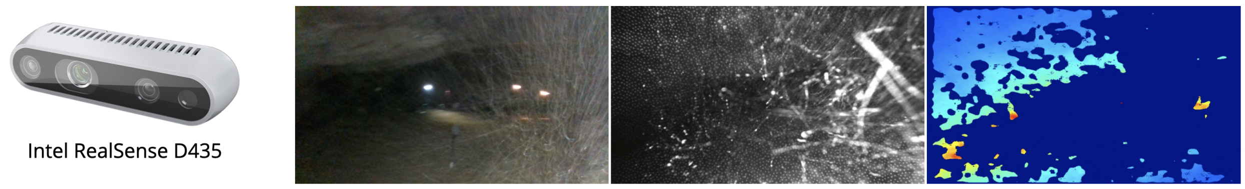


Figure 2.3: Effects of dust on the depth sensor (Intel RealSense D435) data in a cave environment during exploration with an aerial robot (extreme right) [13]. Such effects can have a significant impact on the map accuracy in these environments. Thus, the planning strategy for exploration must account for a map refinement objective in addition to a coverage objective.

Notably, all teams use a grid-based or a point cloud representation of the environment and have identified that accounting for limited availability of communication resources within the exploration framework is a key milestone for future work [29, 30, 31]. The contributions in this thesis build upon these works to improve the rate of exploration and allow for communication-efficient distributed mapping.

2.2 Planning for Rapid Exploration

A motion planner designed for exploration of *a priori* unknown and unstructured spaces with an aerial robot must satisfy three key requirements: (1) reduce the amount of unknown space ("Coverage"), (2) correct any inaccuracies in the known space ("Map Refinement", Fig. 2.3), and (3) return collision-free motion plans in real-time.

Many planners have been proposed towards meeting these objectives. Towards meeting coverage objectives, geometric motion planning approaches have been proposed. Cieslewski et al. [14] use a frontier-based motion planning strategy in which the planner biases the motion plans towards frontiers while penalizing a change in direction of velocity. Recent work by Cao et al. [32] improves upon frontier-based coverage strategies and proposes a hierarchical planner that uses surface normals to generate informative viewpoints via a geometric heuristic. These objectives can provide a rapid coverage of the environment with low-noise sensors such as LiDARs. However, since the uncertainty in the map is not explicitly accounted into motion planning, maximizing coverage alone might result in degraded performance when using noisy data from depth sensors.

Towards incorporating uncertainty of the map explicitly into decision making, there is a line of research that uses mutual information between projected measurements and the current map as the objective to drive exploration. However, these methods are computationally expensive when compared with coverage methods. Charrow et al. [33] and Zhang et al. [34] propose a computationally tractable approximation of the Cauchy-Schwarz Quadratic Mutual Information (CSQMI) and Shannon Mutual Information (FSMI) respectively, and use this information metric over a local lattice of feasible trajectories to greedily maximize information in a local map. The problem with local and greedy approaches is that they are susceptible to getting stuck in local extrema of information distribution over the environment. To this end, modifications have been proposed to these information-theoretic strategies that incorporate not only a local information maximization objective, but also allows to reason over the global information distribution. Approaches by Charrow et al. [35] and Tabib et al. [36] use frontiers to model this global information spread, while the work by Corah et al. [19] uses a global library of informative views as a succinct and more accurate representation of the global information content. In the proposed motion planner, we use the sensing objectives proposed by Corah et al. to drive the robot towards reducing uncertainty in the map while escaping local extremums in the information distribution.

With the sensing objectives in place, the planner now needs to ensure the third key requirement: return collision-free motion plans in real-time. Given the increasing application of aerial robots as sensing platforms, recent works have begun to consider the effects of system dynamics on high-speed exploration and navigation in unknown environments. Cieslewski et al. [14] propose a strategy based on maintaining rapid forward motion by driving the system toward frontier cells (cells on boundary of free and unknown space [37]) within the camera field-of-view. While the authors note that reaction time for obstacles avoidance can limit speeds in exploration, they provide little discussion of why this happens or how it can be avoided. This thesis considers a broader variety of sensing actions that can avoid these limitations and also incorporates sensing and planning time into action design. Corah et al. [19] propose a receding horizon motion planning strategy that optimizes the sensing objectives via a Monte Carlo tree search over concatenated motion primitives. However, these concatenated primitives are constrained to limited speeds and do not account for sensing constraints. Liu et al. [38] addressed sensing constraints related issues in there navigation work by accounting for these constraints in an optimization-based planning approach. However, the optimization-based action generation strategy can be computationally expensive to use in an exploration scenario. In contrast, forward-arc motion primitives have been proposed for high-speed assistive teleoperation Spitzer et al. [39]. Forward-arc motion primitives are parameterized by a maximum velocity parameter, and are computationally efficient to generate, as opposed to an optimization-based approach. We build on these works and propose a receding-horizon information-theoretic motion planner that uses forward-arc motion primitives for action generation. Through simulated and hardware experiments using a multirotor we demonstrate that the proposed planner allows the aerial robot to explore at high speeds while using a limited range depth sensor.

2.3 Distributed Mapping for Exploration

Ebadi et al. [29] state that the communication bottlenecks faced during mapping with a multi-robot system due to the use of downsampled point clouds can be addressed by map compression techniques or compact representations for motion planning. Dang et al. [30] use the state-of-the-art, memory-efficient OctoMap [40] approach for map representation but mention efficient map sharing as one of the future challenges. Rouček et al. [31] use elevation maps for mapping but only on wheeled and ground robots because the transmission of these maps requires a physically large communication module. These shared challenges indicate a gap in the state-of-art for communication-efficient distributed mapping methods in rapid aerial multi-robot exploration systems for subterranean domains. Corah et al. [19] highlight the benefits of a distributed mapping strategy that exploits the compactness of Gaussian Mixture Models (GMMs) relative to the occupancy grid approach [41]. However, the approach is computationally prohibitive for real-world deployment, limits robot speeds, and the effects of communication constraints on the exploration performance of the robot team are not discussed. In contrast, this thesis proposes a distributed perceptual modeling approach that enables real-time high-fidelity mapping while remaining amenable to low-bandwidth communication channels. Evaluation using constrained bandwidth experiments suggested that these maps can help in improving exploration performance implicitly.

Chapter 3

Background

This chapter provides an overview of two basic aspects used in the remainder of the thesis. In Section 3.1, the exploration problem is defined for a typical multirotor exploration scenario. In Section 3.2, preliminaries for GMM-based mapping are presented from prior works [13, 36], and are used later in Chapters 5 and 6.

3.1 Exploration Problem

Figure 3.1 depicts a typical multirotor exploration scenario. Environment is modeled by an occupancy grid map with independent Bernoulli occupancy probabilities for voxels. Depending on user-specified thresholds for occupancy, the map is partitioned into three subsets: free space ($\mathcal{X}_{\text{free}}$, white voxels in Fig. 3.1), unknown space ($\mathcal{X}_{\text{unk}}$, black voxels in Fig. 3.1), and occupied space ($\mathcal{X}_{\text{occ}}$). The voxels at the boundary of $\mathcal{X}_{\text{free}}$ and adjacent to voxels in $\mathcal{X}_{\text{unk}}$ are called frontier voxels, denoted by the set $\mathcal{X}_{\text{frt}}$ [37]. The objective of a motion planner in such exploration scenarios is to minimize the amount of unknown space $\mathcal{X}_{\text{unk}}$ in the minimum time possible. For information-theoretic techniques, this rate is quantified as the rate of reduction of entropy of the occupancy map [42]. The entropy of the map decreases as the occupancy values of previously unknown voxels are updated based on sensor observations and become more certain, and there is at most one bit of entropy per voxel.

3.2 Gaussian Mixture Models for Exploration

Gaussian Mixture Models (GMMs) can be leveraged to compactly encode sensor observations for transmission over low-bandwidth communications channels [13, 36]. The

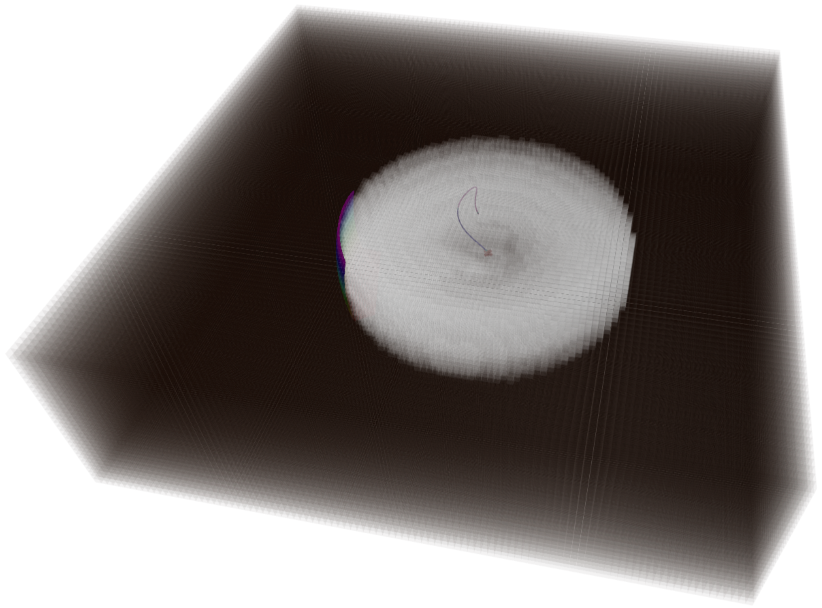


Figure 3.1: An ideal multirotor exploration scenario with no obstacles in the environment. The objective is to design a planner that takes actions such that over time, all of the unknown space (black voxels) is explored (white voxels) using a time-of-flight sensor while ensuring that the robot motion and planned trajectories are always within the free space (white voxels).

GMM provides a generative model of the sensor observations from which occupancy may be reconstructed by resampling from the distribution and raytracing through a local occupancy grid map. Formally, the GMM is a weighted sum of M Gaussian probability density functions (PDFs). The probability density of the GMM is expressed as

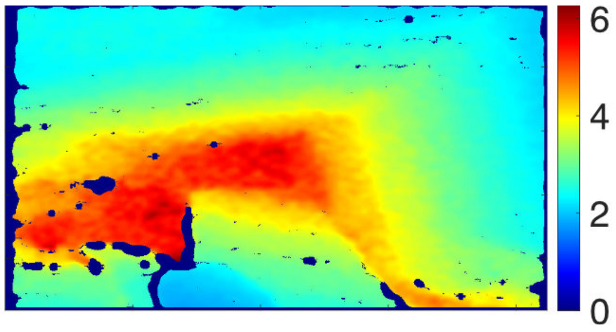
$$p(\boldsymbol{x}|\boldsymbol{\Theta}) = \sum_{m=1}^{M} \pi_m \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_m, \boldsymbol{\Lambda}_m)$$

where $p(\boldsymbol{x}|\boldsymbol{\Theta})$ is the probability density for the D-dimensional random variable $\boldsymbol{x}$ and is parameterized by $\boldsymbol{\Theta} = \{\pi_m, \boldsymbol{\mu}_m, \boldsymbol{\Lambda}_m\}_{m=1}^{M}$. $\pi_m \in \mathbb{R}$ is a weight such that $\sum_{m=1}^{M} \pi_m = 1$ and $0 \leq \pi_m \leq 1$, $\boldsymbol{\mu}_m$ is a mean, and $\boldsymbol{\Lambda}_m$ is a covariance matrix for the m^{th} D-dimensional Gaussian probability density function of the distribution. The multivariate probability density for $\boldsymbol{x}$ is written as

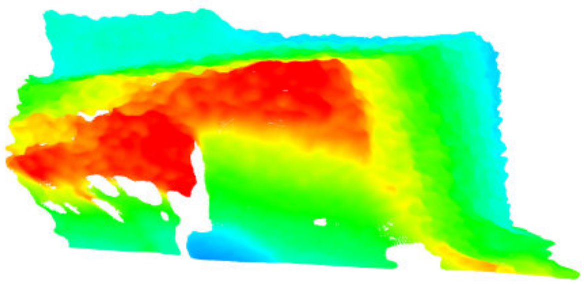
$$\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_i, \boldsymbol{\Lambda}_i) = \frac{|\boldsymbol{\Lambda}_i|^{-1/2}}{(2\pi)^{D/2}} \exp\Big(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_i)^T \boldsymbol{\Lambda}_i^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_i)\Big).$$

In this work, a depth observation taken at time t and consisting of N points, $\mathcal{Z}_t = \{z_t^1, \ldots, z_t^n, \ldots, z_t^N\}$, is used to learn a GMM. Estimating optimal GMM parameters $\boldsymbol{\Theta}$ remains an open area of research [43]. The Expectation Maximization (EM) algorithm

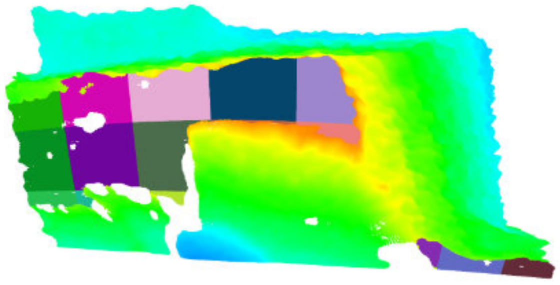
(a) Color Image



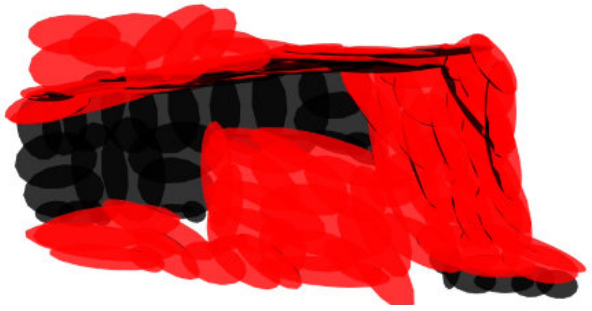
(b) Depth Image



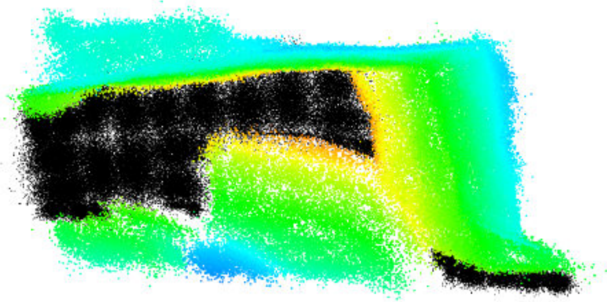
(c) Point Cloud



(d) Free Space Windows



(e) GMM



(f) Resampled Data

Figure 3.2: Overview of the approach to transform a sensor observation into free and occupied GMMs. (a) A color image taken onboard the robot exploring Laurel Caverns. (b) A depth image corresponding to the same view as the color image with distance shown as a heatmap on the right hand side (in meters). (c) illustrates the point cloud representation of the depth image. (d) In the mapping approach, points at a distance smaller than a user-specified max range r_d (in this case $r_d = 5\,\text{m}$) are considered to be occupied, and a GMM is learned using the approach detailed in Section 3.2.1. Points at a distance further than r_d are considered free, normalized to a unit vector, and projected to r_d. The free space points are projected to image space and windowed using the technique detailed in Section 3.2.2 to decrease computation time. Each window is shown in a different color. (e) The GMM representing the occupied-space points is shown in red and the GMM representing the free space points is shown in black. Sampling 2×10^5 points from the distribution yields the result shown in (f). The number of points to resample is selected for illustration purposes and to highlight that the resampling process yields a map reconstruction with an arbitrary number of points.

is usually employed to solve the maximum-likelihood parameter estimation problem, which is guaranteed to find a local maximum of the log likelihood function [44]. To make the optimization tractable, EM introduces latent variables $\mathbf{C} = \{c_{nm}\}$ for each point z_t^n and cluster m and iteratively performs two steps: expectation (E) and maximization (M) [44, 45, 46].

The E step calculates the expected value of the complete-data log-likelihood $\ln p(\mathcal{Z}_t, \mathbf{C}|\boldsymbol{\Theta})$ with respect to the unknown variables $\mathbf{C}$ given the observed data $\mathcal{Z}_t$ and current parameter estimates $\boldsymbol{\Theta}^i$, which is written as $E[\ln p(\mathcal{Z}_t, \mathbf{C}|\boldsymbol{\Theta})|\mathcal{Z}_t, \boldsymbol{\Theta}^i]$ [45]. This amounts to evaluating the posterior probability, β_{nm}, using the current parameter values $\boldsymbol{\Theta}^i$ (shown in Eq. (3.1)) [44]

$$\beta_{nm} = \frac{\pi_m \mathcal{N}(z_t^n|\boldsymbol{\mu}_m^i, \boldsymbol{\Lambda}_m^i)}{\sum\limits_{j=1}^{M} \pi_j \mathcal{N}(z_t^n|\boldsymbol{\mu}_j^i, \boldsymbol{\Lambda}_j^i)}, \tag{3.1}$$

where β_{nm} denotes the responsibility that component m takes for point z_t^n. The M step maximizes the expected log-likelihood using the current responsibilities, β_{nm}, to obtain updated parameters, $\boldsymbol{\Theta}^{i+1}$ via the following:

$$\boldsymbol{\mu}_m^{i+1} = \sum_{n=1}^{N} \frac{\beta_{nm} \mathbf{z}_t^n}{\sum_{n=1}^{N} \beta_{nm}} \tag{3.2}$$

$$\boldsymbol{\Lambda}_m^{i+1} = \sum_{n=1}^{N} \frac{\beta_{nm}(\mathbf{z}_t^n - \boldsymbol{\mu}_m^{i+1})(\mathbf{z}_t^n - \boldsymbol{\mu}_m^{i+1})^T}{\sum_{n=1}^{N} \beta_{nm}} \tag{3.3}$$

$$\pi_m^{i+1} = \frac{\sum_{n=1}^{N} \beta_{nm} \mathbf{x}_n}{\sum_{n=1}^{N} \beta_{nm}}. \tag{3.4}$$

Every iteration of EM is guaranteed to increase the log likelihood and iterations are performed until a local maximum of the log likelihood is achieved [44].

The E step is computationally expensive because a responsibility β_{nm} is calculated for each cluster m and point z_t^n, which amounts to NM responsibility calculations. In the M step, every parameter must be updated by iterating over all N samples in the dataset. In practice, a responsibility matrix $\mathbf{B} \in \mathbb{R}^{N\times M}$ is maintained whose entries consist of the β_{nm} to estimate the parameters $\boldsymbol{\Theta}$.

Following the work of OMeadhra et al. [47], distinct occupied $\mathcal{G}(\mathbf{x})$ (detailed in Section 3.2.1) and free $\mathcal{F}(\mathbf{x})$ (detailed in Section 3.2.2) GMMs are learned to compactly represent the density of points observed in the environment (Fig. 3.2). The process by which $\mathcal{F}(x)$ and $\mathcal{G}(x)$ are created is illustrated in Figs. 3.2c and 3.2d. Because the GMM is a generative model, one may sample from the distribution (Fig. 3.2f) to generate points associated with the surface model and reconstruct occupancy (detailed in Section 3.2.3).

3.2.1 Occupied Space

For points with norms less than a user-specified maximum range r_d, the EM approach is adapted from [48] to accept points that lie within a Mahalanobis distance of λ. Because Gaussians fall off quickly, points far away from a given density will have a small effect on the updated parameters for that density. By reducing the number of points, this decreases the computational cost of the EM calculation. Only points that have a value smaller than λ are considered (i.e., points larger than λ are discarded):

$$\lambda > \sqrt{(\boldsymbol{x}_n - \boldsymbol{\mu}_m^1)^T(\boldsymbol{\Lambda}_m{}^1)^{-1}(\boldsymbol{x}_n - \boldsymbol{\mu}_m^1)} \tag{3.5}$$

where the superscript 1 denotes the initialized values for the mean, covariance, and weight. This approach differs from our prior work Tabib et al. [36]; we utilize the approach in [48] as it yields greater frame-to-frame registration accuracy in practice. Frame-to-frame registration is not used in this work and is left as future work.

3.2.2 Free Space

To learn a free space distribution, points with norms that exceed the maximum range r_d are projected to r_d. The EM approach from Section 3.2.1 is used to decrease the computational cost of learning the distribution. To further decrease the cost, the free space points are split into windows in image space and GMMs consisting of n_f components are learned for each window. The windowing strategy is employed for learning distributions over free space points because it yields faster results and the distributions cannot be used for frame-to-frame registration. The number of windows and components per window is selected empirically. Fig. 3.2d illustrates the effect of the windowing using colored patches and Fig. 3.2e illustrates the result of this windowing technique with black densities. Once the free space distributions are learned for each window the windowed distributions are merged into a single distribution.

Let $\mathcal{G}_i(\boldsymbol{x})$ be a GMM trained from N_i points in window i and let $\mathcal{G}_j(\boldsymbol{x})$ be a GMM trained from N_j points in window j, where $\sum_{w=1}^{W} N_w = N$ for sensor observation $\mathcal{Z}_t$ and W windows. $\mathcal{G}_j(\boldsymbol{x}) = \sum_{k=1}^{K} \tau_k \mathcal{N}(\boldsymbol{x}|\boldsymbol{\nu}_k, \boldsymbol{\Omega}_k)$ may be merged into $\mathcal{G}_i(\boldsymbol{x}) = \sum_{m=1}^{M} \pi_m \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_m, \boldsymbol{\Lambda}_m)$ by concatenating the means, covariances, and weights. However, care must be taken when

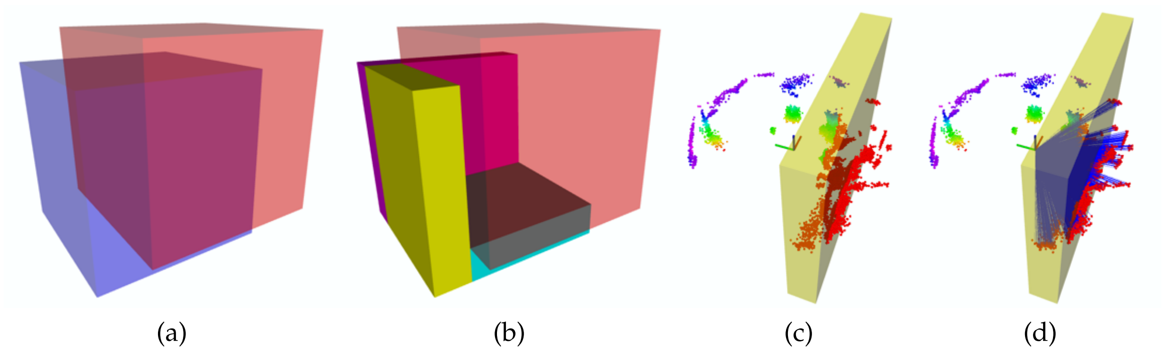


Figure 3.3: Overview of the method by which occupancy is reconstructed. (a) The blue bounding box b_{t+1} is centered around $\mathbf{X}_{t+1}$ and red bounding box b_t is centered at $\mathbf{X}_t$. (b) illustrates the novel bounding boxes in solid magenta, teal, and yellow colors that represent the set difference $b_{t+1} \setminus b_t$. (c) Given a sensor origin shown as a triad, resampled pointcloud, and novel bounding box shown in yellow, each ray from an endpoint to the sensor origin is tested to determine if an intersection with the bounding box occurs. The endpoints of rays that intersect the bounding box are shown in red. (d) illustrates how the bounding box occupancy values are updated. Endpoints inside the yellow volume update cells with an occupied value. All other cells along the ray (shown in blue) are updated to be free.

merging the weights as they must be renormalized to sum to 1 [49]. The weights are renormalized via Eqs. (3.6) and (3.7):

$$N^* = N_i + N_j \tag{3.6}$$

$$\boldsymbol{\pi}^* = \begin{bmatrix} \frac{N_i \pi_1}{N^*} & \dots & \frac{N_i \pi_m}{N^*} & \frac{N_j \tau_1}{N^*} & \dots & \frac{N_j \tau_k}{N^*} \end{bmatrix}^T \tag{3.7}$$

where $m \in [1, \dots, M]$ and $k \in [1, \dots, K]$ denote the mixture component in GMMs $\mathcal{G}_i(\boldsymbol{x})$ and $\mathcal{G}_j(\boldsymbol{x})$, respectively. $N^* \in \mathbb{R}$ is the sum of the support sizes of $\mathcal{G}_i(\boldsymbol{x})$ and $\mathcal{G}_j(\boldsymbol{x})$. $\boldsymbol{\pi}^* \in \mathbb{R}^{M+K}$ are the renormalized weights. The means and covariances are merged by concatenation.

3.2.3 Local Occupancy Grid Map

The occupancy grid map [50] is a probabilistic representation that discretizes 3D space into finitely many grid cells $\mathbf{m} = \{m_1, ..., m_{|\mathbf{m}|}\}$. Each cell is assumed to be independent and the probability of occupancy for an individual cell is denoted as $p(m_i|\mathbf{X}_{1:t}, \mathcal{Z}_{1:t})$, where $\mathbf{X}_{1:t}$ represents all vehicle states up to and including time t and $\mathcal{Z}_{1:t}$ represents the corresponding observations. Unobserved grid cells are assigned the uniform prior of 0.5 and the occupancy value of the grid cell m_i at time t is expressed using log odds notation for numerical stability.

$$l_{t,i} \triangleq \log\left(\frac{p(m_i|\mathcal{Z}_{1:t}, \mathbf{X}_{1:t})}{1 - p(m_i|\mathcal{Z}_{1:t}, \mathbf{X}_{1:t})}\right) - l_0$$

When a new measurement $\mathcal{Z}_t$ is obtained, the occupancy value of cell m_i is updated as

$$l_{t,i} \triangleq l_{t-1,i} + L(m_i|\mathcal{Z}_t)$$

where $L(m_i|\mathcal{Z}_t)$ denotes the inverse sensor model of the robot and l_0 is the prior of occupancy [50].

Instead of storing the occupancy grid map $\mathbf{m}$ that represents occupancy for the entire environment viewed since the start of exploration onboard the vehicle, a local occupancy grid map $\bar{\mathbf{m}}_t$ is maintained centered around the robot's pose $\mathbf{X}_t$. The local occupancy grid map moves with the robot, so when regions of the environment are revisited, occupancy must be reconstructed from the surface models $\mathcal{G}(x)$ and $\mathcal{F}(x)$. To reconstruct occupancy at time $t+1$ given $\bar{\mathbf{m}}_t$, the set difference of the bounding boxes b_t and b_{t+1} for $\bar{\mathbf{m}}_t$ and $\mathbf{m}_{t+1}$, respectively, are used to compute at most three non-overlapping bounding boxes (see Figs. 3.3a and 3.3b for example). The intersection of the bounding boxes remains up-to-date, but the occupancy of the novel bounding boxes must be reconstructed using the surface models $\mathcal{G}(x)$ and $\mathcal{F}(x)$. Raytracing is an expensive operation [51], so time is saved by removing voxels at the intersection of b_t and b_{t+1} from consideration.

The local occupancy grid map at time $t+1$, $\bar{\mathbf{m}}_{t+1}$, is initialized by copying the voxels in local grid $\bar{\mathbf{m}}_t$ at the intersection of b_{t+1} and b_t. In practice, the time to copy the local occupancy grid map is very low (on the order of a few tens of milliseconds) as compared to the cost of raytracing through the grid. Not all Gaussian densities will affect the occupancy reconstruction so to identify the GMM components that intersect the bounding boxes a KDTree [52] stores the means of the densities. A radius equal to twice the sensor's max range is used to identify the components that could affect the occupancy value of the cells in the bounding box. A ray-bounding box intersection algorithm [53] checks for intersections between the bounding box and the ray from the sensor origin to the density mean. Densities that intersect the bounding box are extracted into local submaps $\bar{\mathcal{G}}(x)$ and $\bar{\mathcal{F}}(x)$. Points are sampled from each distribution and raytraced to their corresponding sensor origin to update the local grid map (example shown in Figs. 3.3c and 3.3d).

As the number of mixture components in the distribution increases over time in one region, updating the occupancy becomes increasingly expensive as the number of points needed to resample and raytrace increases. Tabib et al. [36] detail a method for limiting the potentially unbounded number of points from a 360° LiDAR sensor. However, this thesis assumes robots that are equipped with a low-cost limited field-of-view depth sensor. A geometric method to limit sensor observations for this case is described in Chapter 5.

Chapter 4

Rapid Motion Primitives-based Single-Robot Exploration

This chapter presents the motion primitives-based receding-horizon motion planner developed in this thesis. The action space used by the motion planner is represented as a collection of forward-arc motion primitives with velocity bounds derived from dynamics and sensing constraints. Finite-horizon trajectory selection from this action space is performed via Monte Carlo tree search (MCTS). This motion planner is later used in Chapter 6 on each robot of a multi-robot exploration system that can rapidly explore a cave.

4.1 Approach

4.1.1 Steady-State Velocity Analysis

This section presents an analysis of the exploration performance for an aerial robot operating for steady-state conditions such as continuous motion toward a frontier. This analysis produces bounds on velocity and rates of entropy reduction, given the constraints on dynamics and sensing. We leverage these insights in Section 4.1.2 to design motion primitive actions for rapid exploration.

System Model and Safety Constraints

This work applies a simplified double-integrator quadrotor model with acceleration and velocity constraints for analysis of limits on exploration performance, which can be thought of as a relaxation of dynamics models that are commonly used for position and attitude control of multirotor vehicles [54, 55]. Let $\mathbf{r} = [x, y, z]^\top$ be the position of the vehicle in an inertial world frame $\mathcal{W} = \{\mathbf{x}_\mathcal{W}, \mathbf{y}_\mathcal{W}, \mathbf{z}_\mathcal{W}\}$, and let the body frame be $\mathcal{B} = \{\mathbf{x}_\mathcal{B}, \mathbf{y}_\mathcal{B}, \mathbf{z}_\mathcal{B}\}$. Assuming small roll and pitch angles, and given the yaw angle ψ, the system state is $\xi = [\mathbf{r}^\top, \psi, \dot{\mathbf{r}}^\top, \dot{\psi}]^\top$. The derivatives of position and yaw satisfy bounds on velocity and acceleration

$$||\dot{\mathbf{r}}||_2 \leq V_\text{max} \qquad ||\ddot{\mathbf{r}}||_2 \leq A_\text{max} \qquad |\dot{\psi}| \leq \Omega, \tag{4.1}$$

where $||\cdot||_2$ is the L^2-norm.

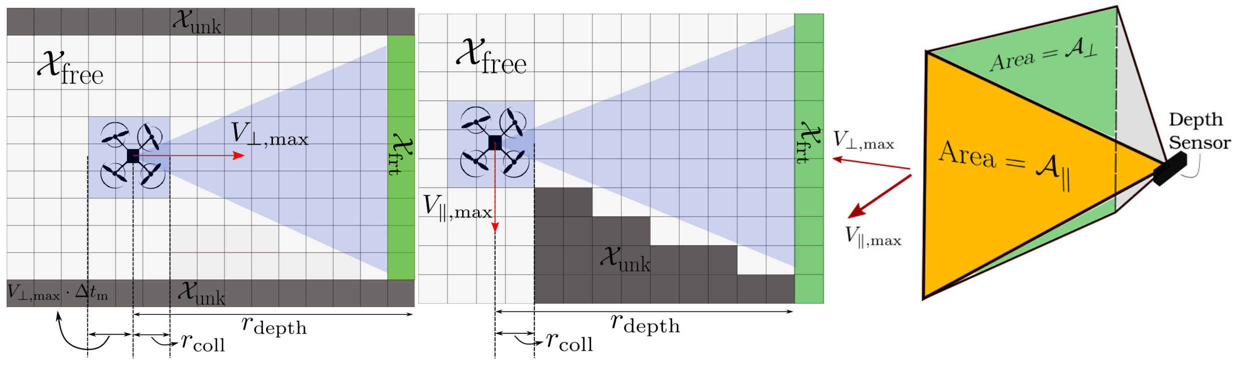
Further, the robot is equipped with a forward-facing depth sensor with range of r_depth for use in mapping. However, while navigating the environment, the robot must also be able to avoid collisions with obstacles.

The requirement for collision-free operation restricts the set of actions that a multirotor can safely execute while navigating in an unknown environment. A planning policy can ensure collision-free operation by guaranteeing that the robot is able to stop entirely within unoccupied space $\mathcal{X}_\text{free}$, given an appropriate collision radius r_coll, such as in the work of Janson et al. [56]. In the worst case, any voxel in the unknown space $\mathcal{X}_\text{unk}$ may be revealed to be occupied and so possibly force the robot to stop within $\mathcal{X}_\text{free}$.

The robot plans once every Δt_p seconds, and there is also a latency Δt_m for acquiring depth information and integrating it into the occupancy map. The sensor data is Δt_m seconds old at the beginning of planning, and once the planner is done, the robot executes the selected action for another Δt_p so that the total effective latency is no more than $\Delta t_l = \Delta t_m + 2\Delta t_p$. Note that, although latency may be unpredictable in practice, the robot will not depend on consistent latency to maintain safe operation.

Steady-State Exploration Scenarios

Figure 4.1 illustrates two possible scenarios for steady-state motion with respect to a frontier. For the perpendicular case (Fig. 4.1a) the robot moves continuously toward a frontier and may have to avoid obstacles at the edge of the sensor range. As discussed in Section 4.1.1, the robot must be able to avoid collisions with obstacles in the unknown environment even if there are not any there. This means that the the robot must always be prepared to stop before reaching the frontier. For the parallel case (Fig. 4.1b) the robot moves along the frontier through space that has already been mapped. When known space is free of obstacles, the robot may continue to do so at the maximum allowable velocity. This scenario can also be thought of as the limit for a spiral motion—which will arise later in the experimental results—as the curvature becomes very small.


$\mathcal{X}_{\text{unk}}$
$\mathcal{X}_{\text{free}}$
$V_{\perp,\max}$
$\mathcal{X}_{\text{frt}}$
$V_{\perp,\max} \cdot \Delta t_{\text{m}}$
r_{depth}
r_{coll}
$V_{\parallel,\max}$
Area $= \mathcal{A}_{\perp}$
Area $= \mathcal{A}_{\parallel}$
Depth Sensor

(a) Perpendicular scenario (b) Parallel scenario (c) Sensor Cone

Figure 4.1: Steady-state exploration scenarios. Analysis in Section 4.1.1 presents upper bounds on velocities are for a double integrator system (a) for motion perpendicular to a frontier ($V_{\perp,\text{max}}$) and (b) for motion parallel to it ($V_{\|,\text{max}}$). To ensure safety in the perpendicular case, the robot has to stop within a user-specified collision radius from the frontier ($\mathcal{X}_{\text{frt}}$), i.e. within $r_{\text{depth}} - r_{\text{coll}}$ from the current state. For the parallel case, no such restrictions exist since the robot is moving in the explored space which, ideally, is free ($\mathcal{X}_{\text{free}}$). (c) Combining the area of the projection of the sensor cone in the direction of motion with the bounds on velocity leads to upper bounds on rates of novel voxels observed during exploration.

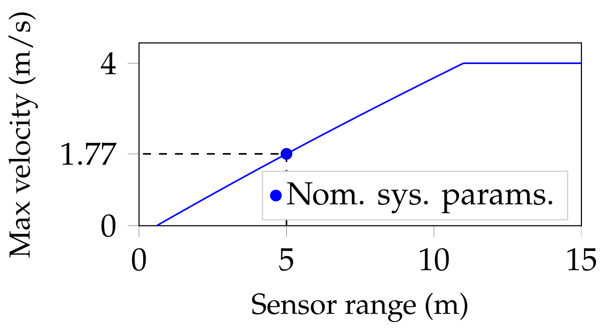
Bounds on Velocity

Given the system model and constraints for exploration scenarios, we now proceed with calculation of the velocity bounds for motion perpendicular and parallel to the frontier, $V_{\perp,\text{max}}$ and $V_{\|,\text{max}}$ respectively. Maximum velocity toward the frontier is computed based on motion at a constant velocity followed by stopping at maximum deceleration to satisfy the safety constraint. The expression for $V_{\perp,\text{max}}$ is a function of acceleration (A_{max}), maximum sensing range (r_{depth}), the collision radius (r_{coll}), and the latency in planning Δt_l (see Fig. 4.1a) and is given by

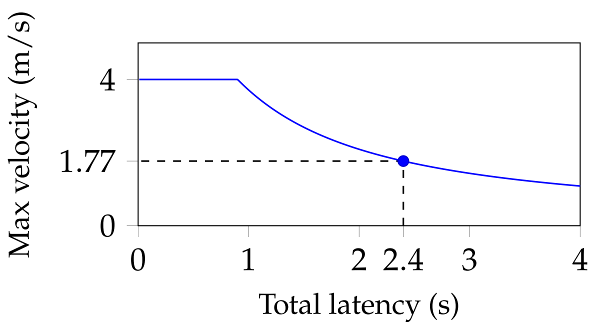
$$\begin{aligned} V_{\perp,\text{max}} &= \min(V_{\text{max}}, V'_{\perp,\text{max}}) \\ V'_{\perp,\text{max}} &= A_{\text{max}} \cdot \left(\sqrt{\Delta t_l^2 + 2\frac{r_{\text{depth}} - r_{\text{coll}}}{A_{\text{max}}}} - \Delta t_l \right). \end{aligned} \tag{4.2}$$

Fig. 4.2 shows the variation of this bound with r_{depth} and Δt_l for the parameters used in this work. For motion parallel to a frontier (see Section 4.1.1 and Fig. 4.1b), there are no obstacles in the direction of motion. Therefore, the steady-state upper bound on the velocity moving parallel to the frontier is identical to the maximum achievable by the system, i.e. $V_{\|,\text{max}} = V_{\text{max}}$.

The entropy reduction then can also be bounded for each scenario terms of the sensor geometry (see Fig. 4.1c) and steady-state velocities by projecting the sensing volume in the



(a) Velocity ($V_{\perp,\max}$) vs. sensor range (r_{depth})



(b) Velocity ($V_{\perp,\max}$) vs. total latency (Δt_l)

Figure 4.2: Maximum achievable velocity moving toward a frontier ($V_{\perp,\max}$) according to Eq. (4.2) based on parameters used in Table 4.1 and varying (a) sensor range and (b) total latency (which consists of latency for the mapping system and time for planning). The circle marks the maximum velocity at which the robot can move toward unknown space for the parameters used in this paper (see Table 4.1) which is less than half the overall velocity bound. Approaching this velocity limit requires either sensor range exceeding 10 m, both decreased planning time and mapping latency, or some combination of the two.

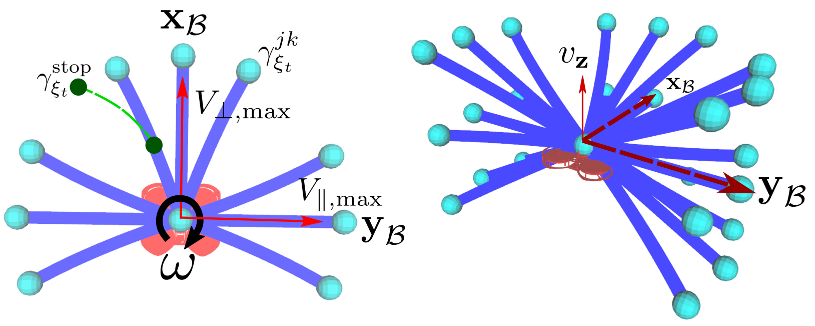
Table 4.1: Steady-state upper bounds on velocity and rate of entropy reduction for the scenarios described in Section 4.1.1. All values are computed for a planning time of 1 Hz, mapping latency 0.4 s, sensor point cloud of size 9.93 m × 5.68 m based on the Intel RealSense D435 depth sensor with image size 424px × 240px and a maximum depth r_{depth} of 5 m. Occupancy grid resolution is 20 cm with an overall bound on top speed $V_{\max}$ at 4 m/s, and collision radius r_{coll} set at 0.6 m.

Value/Cases	Area (m^2)	Velocity (m/s)	Volume rate (m^3/s)	Entropy rate (bits/s)
Perpendicular ($\perp$)	56.4	1.77	99.83	1.25×10^4
Parallel ($\|$)	57.19	4.00	228.8	2.86×10^4
$\perp$, rapid yaw	56.8	1.77	100.5	1.26×10^4
$\|$, rapid yaw	78.05	4.00	312.2	3.90×10^4

direction of motion. Here, we also introduce the possibility of rapid yaw motion during either motion. Results are shown in Table 4.1. Note that moving parallel to the frontier can provide significantly improved performance.

4.1.2 Action Representation

This section details the design of available actions for the proposed motion planning framework. We define a trajectory generation scheme, related parameters and conventions, and action design specifics leveraging insights gained in Section 4.1.1.



(a) Variation in ω (b) Variation in $v_\mathbf{z}$

Figure 4.3: Actions in $\mathbf{x}_\mathcal{B} - \mathbf{y}_\mathcal{B}$ plane at the multirotor state $\boldsymbol{\xi}_t$, $\gamma^{jk}_{\xi_t}$ (blue), are generated using discretized velocity bounds obtained from the analysis in Section 4.1.1. The set of such primitives at each state is termed a motion primitive library (MPL) Γ_{ξ_t}. MPLs are sampled in directions perpendicular ($\mathbf{x}_\mathcal{B}$) and parallel ($\mathbf{y}_\mathcal{B}, -\mathbf{y}_\mathcal{B}$) to the sensor scans with speeds bounded by $V_{\perp,\max}$ and $V_{\parallel,\max}$ respectively, see (a). Variation in $\mathbf{z}_\mathcal{B}$ direction using a user-specified bound on vertical velocity, $V_\mathbf{z}$, yields the final action space shown in (b). Dynamically feasible stopping trajectories $\gamma^{\text{stop}}_{\xi_t}$ are available for each primitive (green) for safety (only one shown for brevity).

Motion Primitive Library Generation

Safe and accurate high-speed flight requires large and smooth linear acceleration and angular velocity references. Smoothness for such references depends on higher derivatives of position, jerk and snap [57]. For this work, the actions that are available to the robot are snap-continuous, forward arc [58] motion primitives, which have previously been applied to high-speed teleoperation of multirotors [39]. Given the differentially-flat state of the multirotor at time t, $\boldsymbol{\xi}_t = [x, y, z, \psi]^\top$, denote an available action parameterization as $\mathbf{a} = [v_\mathbf{x}, v_\mathbf{z}, \omega]$ where $v_\mathbf{x}$ and $v_\mathbf{z}$ are velocities in the body frame $\mathbf{x}_\mathcal{B}$ and $\mathbf{z}_\mathcal{B}$ directions, and ω is the body yaw rate. Actions are discretized using user-specified maximum velocity bounds in $\mathbf{x}_\mathcal{B} - \mathbf{y}_\mathcal{B}$ plane (ω variation, N_ω primitives) and in $\mathbf{z}_\mathcal{B}$ direction ($v_\mathbf{z}$ variation, $N_\mathbf{z}$ primitives) to obtain a motion primitive library (MPL) Γ_{ξ_t} given by (Fig. 4.3):

$$\Gamma_{\xi_t} = \{\gamma^{jk}_{\xi_t} \mid j \in [1, N_\omega], k \in [1, N_\mathbf{z}], \|[v_\mathbf{x}, v_\mathbf{y}]\| \le V_{\max}, \|v_\mathbf{z}\| \le V_\mathbf{z}, \|\omega\| \le \Omega\} \tag{4.3}$$

where, $\|\cdot\|$ denotes L^2-norm of a vector, $V_{\max}$ and $V_\mathbf{z}$ are the bounds on speed in $\mathbf{x}_\mathcal{B} - \mathbf{y}_\mathcal{B}$ plane and $\mathbf{z}$ direction respectively, and Ω is the bound on body yaw rate. For a given action discretization, the motion primitive $\gamma^{jk}_{\xi_t}$ is an 8th order polynomial in time with fixed start and end point velocities and unconstrained position at the end. The velocity at the end point, at time τ, follows by forward propagating unicycle kinematics using the current state and the action parameterization while higher order derivatives up to snap, endpoints are kept zero:

$$\dot{\boldsymbol{\xi}}_\tau = [v_\mathbf{x}\cos\psi, v_\mathbf{x}\sin\psi, v_\mathbf{z}, \omega]^\top,\ \psi = \omega\tau,\ \boldsymbol{\xi}_\tau^{(j)} = \mathbf{0},\ \text{for } j \in \{2,3,4\} \tag{4.4}$$

where $\boldsymbol{\xi}^{(j)}$ denotes the j^{th} time derivative of $\boldsymbol{\xi}$. The stopping trajectories at any $\boldsymbol{\xi}_t$ ($\gamma_{\boldsymbol{\xi}_t}^{\text{stop}}$, Fig. 4.3) can be sampled by keeping $\dot{\boldsymbol{\xi}}_t = \mathbf{0}$. Further detail on forward-arc motion primitive generation can be found in [59]. Dynamic feasibility check is based on pre-specified empirically observed bounds on linear acceleration and linear jerk L^2-norms. This search achieves having the trajectory in the desired end point velocity $\dot{\boldsymbol{\xi}}_t$ in the minimum time possible from the current state.

Action Space for Fast Exploration

The action space for the proposed planner is a collection of MPLs, defined by Eq. (4.3), generated using linear velocities based on bounds obtained in Section 4.1.1, $V_{\perp,\max}$ and $V_{\parallel,\max}$. The planner uses 6 MPLs to represent the action space, $\mathcal{X}_{\text{act}} = \{\Gamma_{\boldsymbol{\xi}_t}^i \mid i \in [1,6]\}$, and sets of upper bounds on linear velocities (Table 4.2) define each of these different MPLs. These MPLs include both high-speed actions for navigating the environment and actions that mimic steady-state conditions described Section 4.1.1. Later, in Section 4.2, we highlight effects on exploration performance due to these components, especially the high speed parallel and low speed perpendicular MPLs.

4.1.3 Action Selection

We formulate the action selection problem as a finite-horizon optimization seeking to maximize cumulative information gain [33], and build upon previous work [19, 60, 61] on robotic exploration using Monte Carlo tree search (MCTS).

Most MCTS-based planners follow four steps: selection, expansion, simulation playout, and backpropagation of statistics [62, 63]. Such planners usually construct a search tree iteratively by random rollout from a previously unexpanded node selected based on upper-confidence bounds for trees (UCT) [63]. Prior works [19, 60] have applied MCTS in planning for exploration using multirotors using a UCT-based selection policy, information gain rewards, and random simulation playout over a finite horizon. We extend this approach by adding considerations for model constraints into the node expansion phase of MCTS.

Information-Theoretic Exploration Objectives

Following a similar approach as our prior work [19], the planner optimizes an objective with two components: a local information reward based on Cauchy-Schwarz quadratic mutual information (CSQMI) [33], and a global reward for decrease the shortest path distance to points in the state space that are expected to provide significant information gain [19]. For any candidate action, γ_{ξ_t}, we compute the local information gain $\mathcal{I}_\gamma$ over user-specified time intervals and treat the joint information gain as a reward for the MCTS planner. The distance reward serves to direct the robot toward unexplored and possibly distant regions of the environment once the local environment is exhausted of information causing the local information reward to decrease.

Safety at High Speeds

Given the action representation described in Sect. 4.1.2, we require the planner to ensure safety while expanding nodes. Specifically, the trajectory tracked by the controller should both respect constraints on the dynamics and remain in known free space for all time. To satisfy this condition, before sending any trajectory to the robot, we require knowledge of a trajectory that will bring the robot to a stop—and potentially keep it there—afterward. As such, the robot will avoid collision, even if the planner fails to provide any output. If ever planning fails, the known stopping trajectory is sent to the robot, and the robot will continue to replan from a future reference point.

4.2 Results and Analysis

This section describes hardware and simulation results for the proposed exploration approach. The simulation results evaluate performance in a warehouse-like environment which serves as a representative example of a large-scale exploration task. The hardware results demonstrate exploration at high speeds using a hexarotor platform under various degrees of clutter. Unless otherwise noted, the configuration of the robot for simulation matches the hardware.

4.2.1 Aerial Platform

Platform used for experiments is a hexarotor aerial robot (Fig. 4.7a) with a forward-facing depth camera for mapping (Realsense D435) with a 89.57° by 59.24° field of view and a range of $r_{\text{depth}} = 5.0\,\text{m}$. The robot itself weighs 55.37 N, has a thrust to weight ratio

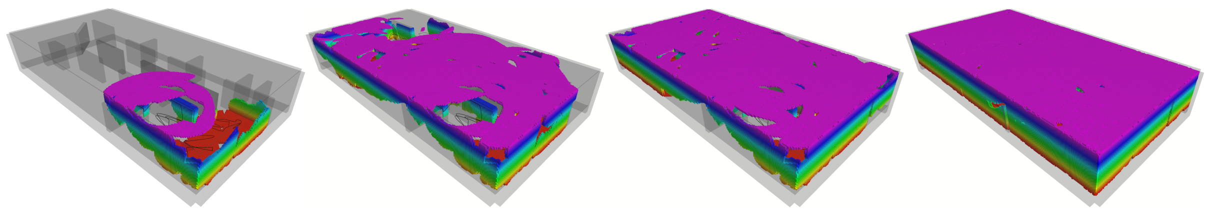

(a) $t = 100\,\mathrm{s}$ (b) $t = 500\,\mathrm{s}$ (c) $t = 1000\,\mathrm{s}$ (d) $t = 1500\,\mathrm{s}$

Figure 4.4: Occupied map at different stages of exploration of a simulated three-dimensional 60 m×30 m×11 m warehouse environment used for experiments. The map is colored based on Z height.

of 3.5, and has a diameter of 0.89 m from motor to motor. Limits on acceleration and jerk are set to $A_{\max} = 10\,\mathrm{m/s^2}$ and $J_{\max} = 35\,\mathrm{m/s^3}$ respectively, based on empirical data available for the platform. Unless otherwise stated, the planning horizon is kept at 4 seconds for all experiments. For both simulation and hardware experiments, mapping and planning run on a computationally constrained quad-core embedded CPU Gigabyte Brix 6500U. The robot obtains odometry estimates via a downward-facing camera and IMU using a monocular Visual-Inertial Navigation System (VINS-Mono [64]), previously used for high-speed teleoperation of a multirotor [39]. This state estimation component, only present for hardware experiments, is executed on a quad-core NVIDIA Tegra TX2 on-board the vehicle. Contrary to perfect state estimation in simulation experiments, for the hardware experiments the robot only has access to odometry for navigation and is susceptible to drift. For the purpose of this work, we will continue to emphasize the role of planning and speed in the exploration experiments and will comment briefly on ramifications of drift on outcomes. Future iterations of this platform will seek to combine a local mapping strategy [19] with a complete SLAM system.

4.2.2 Simulated Exploration of a Warehouse Environment

The simulations demonstrate exploration of a large warehouse environment (pictured in Fig. 4.4). These trials are repeated for three system configurations which vary the motion primitive library and the computational setting:

- **High branching factor (BF), Sim-Time**: The planner uses the large motion primitive library (Table 4.2a) for exploration. The simulation and clock pause at the end of each planning round until the MCTS planner completes a user-defined number (3000) of iterations. The simulation time then does not include this additional time spent in planning.
- **High BF, Real-Time**: The planner uses the large motion primitive library for explo-

Table 4.2: Motion primitive libraries used to construct the action space using Eq. (4.3) from Section 4.1.2 and the bounds obtained after applying analysis presented in Section 4.1.1. The vertical velocity bound ($V_{\mathbf{z}}$) and the speed bound in $\mathbf{x}_{\mathcal{B}} - \mathbf{y}_{\mathcal{B}}$ plane ($V_{\max}$) are kept at 0.3 m/s and 4.0 m/s respectively. The total number of primitives for a MPL is $N_{\text{prim}} = N_{\omega} \cdot N_{\mathbf{z}}$.

(a) Large Library

ID	**Type**	**Max. Speed**	Dir.	N_{ω}	$N_{\mathbf{z}}$	N_{prim}
1	Yaw	0	ψ	1	1	1
2	$\perp$	$V_{\perp,\max}$	$\mathbf{x}_{\mathcal{B}}$	9	5	45
3	$\perp$	$V_{\max}$	$\mathbf{x}_{\mathcal{B}}$	9	5	45
4	$\parallel$	$V_{\max}$	$\mathbf{y}_{\mathcal{B}}$	9	5	45
5	$\parallel$	$V_{\max}$	$-\mathbf{y}_{\mathcal{B}}$	9	5	45
6	Z	$V_{\mathbf{z}}$	$\mathbf{z}$	1	5	5

(b) Minimal Library

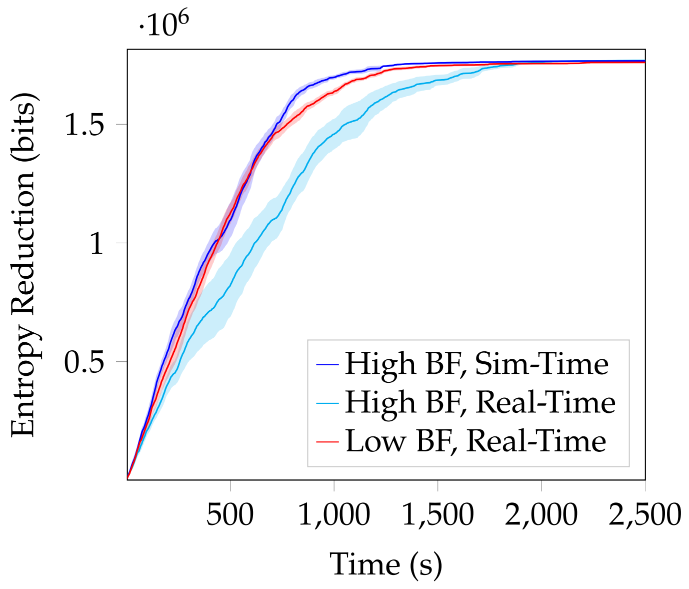
ID	**Type**	**Max. Speed**	Dir.	N_{ω}	$N_{\mathbf{z}}$	N_{prim}
1	Yaw	0	ψ	1	1	1
2	$\perp$	$V_{\perp,\max}$	$\mathbf{x}_{\mathcal{B}}$	3	3	9
3	$\perp$	$V_{\max}$	$\mathbf{x}_{\mathcal{B}}$	3	3	9
4	$\parallel$	$V_{\max}$	$\mathbf{y}_{\mathcal{B}}$	3	3	9
5	$\parallel$	$V_{\max}$	$-\mathbf{y}_{\mathcal{B}}$	3	3	9
6	Z	$V_{\mathbf{z}}$	$\mathbf{z}$	1	3	3

ration, but the simulation of the multirotor runs in real time. The planner runs in an anytime fashion on a computer comparable to the on-board computer used for flight experiments presented in Section 4.2.3 while simulators for the camera and dynamics run on a separate computer.

- **Low BF, Real-Time**: The planner uses the minimal motion library (Table 4.2b) for exploration and the computational configuration is the same as the above.

These experiments first establish baseline performance (High BF, Sim-Time) given access to a variety of motion primitives and a relatively large amount of planning time. The latter two configurations demonstrate online planning in a configuration that closely matches the hexarotor platform used in this paper. These experiments seek to demonstrate the role of computational constraints in design of the motion primitive library. For each configuration, we provide several trials, one for each of 5 given starting locations.

Each trial lasts 2500 seconds which provides ample time to explore the entire environment. For all trials, the perpendicular velocity $V_{\perp,\max}$ is further limited to 1.25 m/s, below



Time to reach (s):	**25%**	**50%**	**75%**	**90%**
High BF, Sim-Time	**157.7**	**365.7**	**602.5**	**788.3**
High BF, Real-Time	267.4	501.6	866.5	1183.2
Low BF, Real-Time	193.2	382.5	604.8	901.1

Figure 4.5: (top) Entropy reduction vs. time for each of the simulation trials. Transparent patches show standard-error. (bottom) Average time to reach fractions of the maximum entropy reduction over all trials (1.766×10^6 bits). While the High BF, Sim-Time case dominates in terms of entropy reduction, the Low BF, Real-Time case is able to provide similar performance. Note that the different configuration are described at the beginning of Section 4.2.2 and that BF denotes branching factor.

the value of 1.77 m/s obtained in Section 4.1.1 which we find admits forward motion at a constant speed given the trajectory generation approach used for motion primitive design. The maximum speed is set to more than three times greater at $V_{\max} = 4.0\,\mathrm{m/s}$.

Figure 4.5 summarizes exploration performance for each experiment. The high branching factory Sim-Time case which has access to extra planning time performs at least as well or better than the other configurations in terms of entropy reduction. However, the configuration with same motion primitive library and real time is significantly impaired and requires between approximately 1.3 to 1.8 times as long to reach reported levels of entropy reduction. The lower branching factor case matches the first configuration much more closely. As such, this latter configuration is appropriate for deployment on the compute-constrained hexarotor platform.

In addition to being able to explore an environment rapidly and completely, we characterize the roles of the motion primitive actions in the exploration task. Figure 4.6 shows density estimates plots for speeds, yaw rates, and entropy rate labelled by the type of action selected by the MCTS planner for execution for periods when the entropy reduction rate is significant (greater than 600 bits/s) so as to separate exploration actions from traversal of the environment and time periods after exploration is effectively complete. This threshold corresponds to a knee point in the overall distribution of entropy reduction rates: 94.4% of all entropy reduction occurs above this threshold but during only 27.9% of time during the trials. Interestingly, the time rate of entropy reduction is largely consistent across action types. However, as expected, motions perpendicular to the frontier primarily

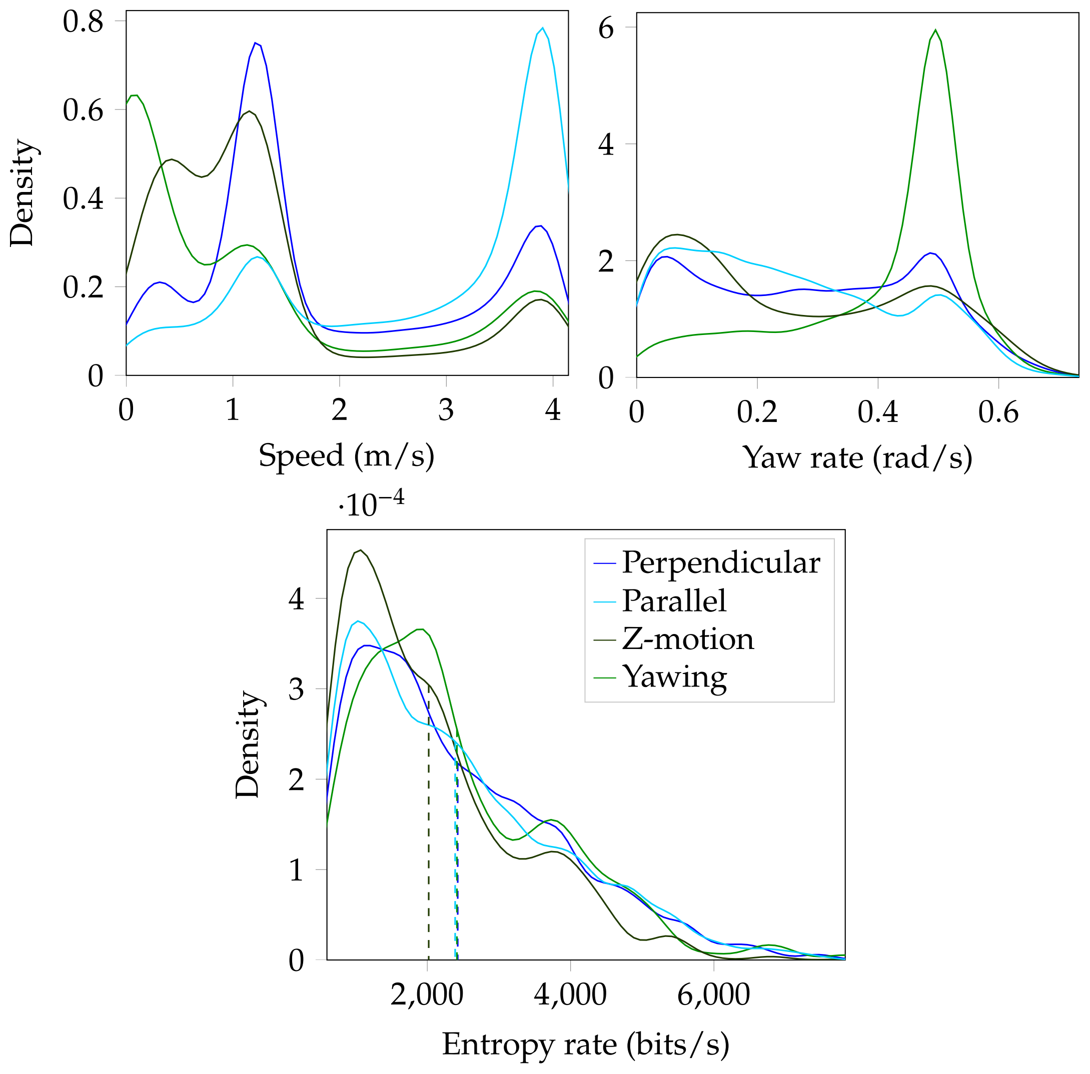


Figure 4.6: Exploration performance by action. Plots provide estimates of probability densities (via kernel density estimation) for speed, yaw rate, and entropy rate conditional on the type of action (Table 4.2) being executed by the robot. All densities are also conditional on a significant entropy reduction rate (greater than 600 bits/s) in order to emphasize performance characteristics for actions that directly contribute to the map rather than traversal of known space or time periods after exploration is effectively complete. Note that, even though the option can has access to high-speed motions perpendicular to frontiers, entropy reduction for perpendicular actions occurs primarily at lower speeds (1.25m/s) in accordance with the analysis in Section 4.1.1.

Table 4.3: Performance statistics for the high branching factor, Sim-Time simulation study. Unless otherwise stated, rates refer to average performance over time-periods when tracking a given action type. All statistics include any time the robot is tracking a given action, except for the entropy reduction rate in the last row, which is conditional on a significant entropy reduction rate (greater than 600 bits/s). Best (or nearly equivalent) values are bolded.

Actions:	$\perp$	$\|\|$	**Z**	**Yaw**	**Stop**
Selection frequency	0.343	**0.479**	0.089	0.088	0.001
Total entropy red, norm.	**0.40**	**0.41**	0.06	0.12	0.01
Average speed (m/s)	2.163	**2.778**	0.959	1.114	1.611
Average yaw rate (rad/s)	0.286	0.234	0.170	**0.381**	0.080
Entropy red. rate (bits/s):	**2425**	**2389**	2020	**2414**	1283

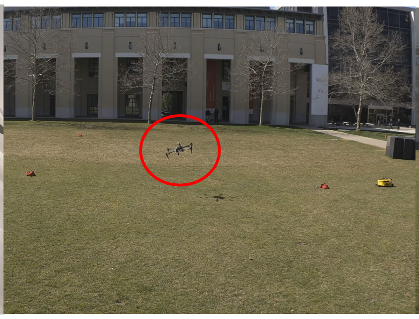
contribute to entropy reduction at reduced speed despite both low-speed and high-speed primitives being available. Table 4.3 shows provides further statistics for the different kinds of actions. Even though entropy reduction rates are similar across actions when the entropy reduction rate is significant, the planner selects motions parallel and perpendicular to frontiers most frequently, and those actions account for more than 80% of all entropy reduction.

4.2.3 Hardware Experiments under Varied Conditions

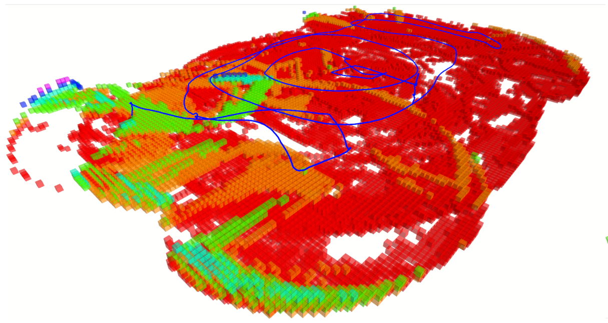
Real-world autonomous exploration experiments are conducted using the aerial platform described in Section 4.2.1 (Fig. 4.7a) in two outdoor scenarios: (1) Open space (Fig. 4.7b), and (2) Scattered obstacles (Fig. 4.7c). Total exploration duration is limited to 90 seconds to minimize the effects of state estimation drift on the resulting map. During each scenario, the robots explores while confined to $12\,\mathrm{m} \times 24\,\mathrm{m} \times 2\,\mathrm{m}$ bounding box. The robot starts at the same position in the bounding box for each trial in both scenarios. The bounds on the speed for perpendicular ($V_{\perp,\max}$) and parallel ($V_{\|,\max}$) motions, are set at 1.25 m/s and 2.25 m/s respectively. The explored maps and robot trajectory for two experiments, one from each scenario, are shown in Figs. 4.7d and 4.7e. Speeds achieved by the vehicle during the experiments are shown in Fig. 4.8 Fig. 4.9 provides plots of reduction of map entropy as well as summary statistics. Even though the trials were relatively short, the odometry often drifted significantly by the end This drift likely contributed to the robot getting stuck behind an obstacle during the trial **S2**. For this reason, we only use the first for the first 40 seconds of each trial when computing summary statistics unless otherwise noted.

As shown in Fig. 4.8, the odometry system reports that the robot reaches and slightly

(a) Multirotor with depth sensor


(b) Open space scenario

(c) Obstacles scenario


(d) Open space scenario exploration after 90 s

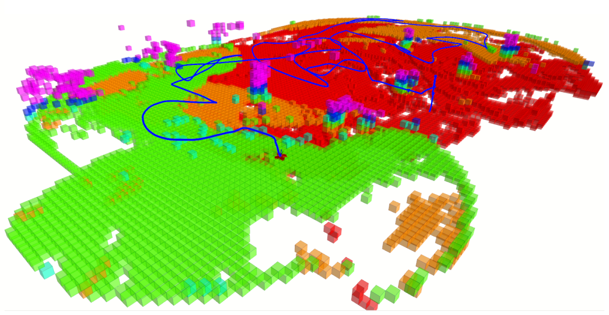

(e) Obstacles scenario exploration after 90 s

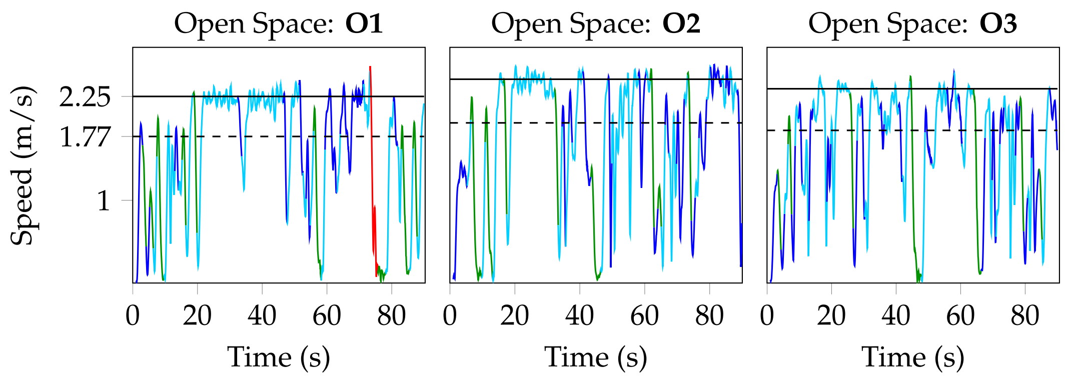
Figure 4.7: Rapid exploration with a hexarotor in an outdoor environment. (a) Multirotor used for hardware experiments in two real world scenarios: (b) open space and (c) space with scattered obstacles. (d), (e) show the explored maps (color gradient based on Z height) and overall paths after 90 s of 3D exploration using the proposed exploration approach. A video of the experiments can be accessed here: `https://youtu.be/YXt4yiTp0Ac`.

exceeds the maximum desired reference speed[1] in each trial, primarily while executing motions parallel to the frontier. Fig. 4.7d shows a particularly notable example of this behavior where the robot executed an outward spiraling motion soon after the start of the trial.

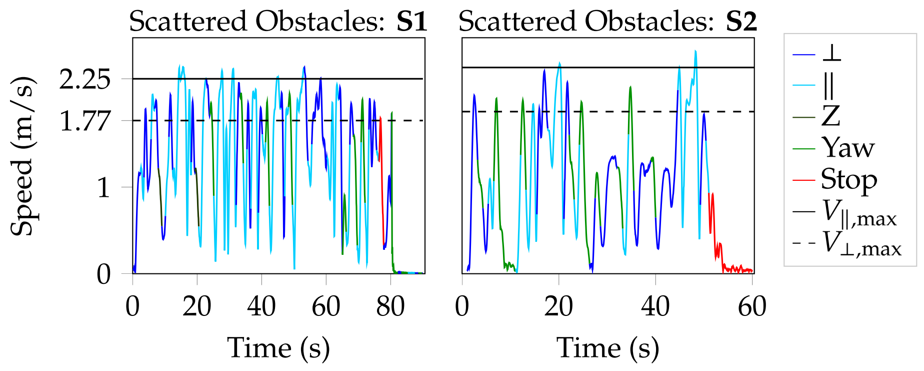
4.3 Summary

We have investigated how the dynamics of aerial platforms and the geometry of common sensors impact selection of control actions in robotic exploration. We have obtained bounds on the velocity for different kinds of motions and applied this analysis to the design of a motion primitive library and information-based planner suitable for rapid exploration with aerial robots. We have demonstrated this approach both in simulated exploration of a large warehouse environment and in outdoor experiments with only on-board computation.

[1]The robot may exceed reference speeds due to error in tracking the position reference because of environment disturbances and inaccuracies in the system model.



(a) Speeds by type of action for open space scenario.

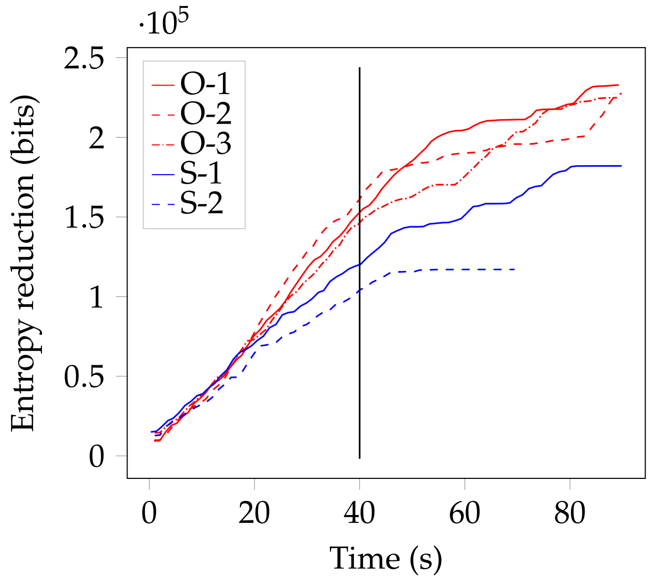


(b) Speeds by type of action for obstacles scenario.

Figure 4.8: Speeds attained during the hardware experiments. A video of the experiments can be accessed here: `https://youtu.be/YXt4yiTp0Ac`.

This system produces interesting and intuitive motions in practice such as outward spirals for rapid coverage of open space. Further, the experimental results demonstrate speeds exceeding 2.25 m/s for both open and cluttered environments, which matches and slightly exceeds prior state-of-the-art results [14].

The analysis illuminates competing directions for improvements to speed and entropy reduction performance. Decreasing planning time and latency, such as by improved efficiency or reactive planning [14] or simply increasing sensor range, can improve speeds moving perpendicular to frontiers which may be especially important in highly cluttered environments where motion parallel to frontiers is not viable. At the same time, the ability to safely and rapidly navigate known environments is also tightly coupled to exploration performance both for motions parallel to frontiers and when traversing known space to new unexplored regions. Thus, improvements to state-estimation, mapping, and planning under uncertainty are also critical to incrasing speed and entropy reduction rates in exploration.



(a) Exploration Performance

Trials:	**O1**	**O2**	**O3**	**S1**	**S2**
Entropy red. at 40s (bits×10^5)	1.53	1.61	1.46	1.2	1.04
Entropy red. final (bits×10^5)	2.32	2.27	2.25	1.82	1.16
Average speed (m/s)	1.50	1.56	1.56	1.42	0.98
Average yaw rate (rad/s)	0.39	0.39	0.33	0.33	0.31
Max. speed (m/s)	2.39	2.38	2.38	2.40	2.29
Max. yaw rate (rad/s)	0.66	0.59	0.55	0.60	0.63

(b) Exploration Statistics

Figure 4.9: Entropy reduction for hardware trials and summary statistics. Except for the final entropy reduction, all statistics are computed over the first 40 second of each trial (shown by the black bar in the entropy reduction plot).

Chapter 5

Communication-Efficient Single-Robot Exploration

This chapter describes a communication-efficient robotic exploration system that uses a single aerial robot equipped with a low-cost limited field-of-view (FoV) depth sensor for cave mapping. Building upon prior work in communication-efficient GMM-based occupancy modeling [36], the computational-efficiency of the occupancy update step is extended to limited FoV sensors via a geometric approximation. Safe and informative path planning with a limited FoV sensor is enabled via the action representation for exploration developed in the previous chapter. Results for both planning and mapping subsystems are presented in simulated and real cave environments using a single aerial robot.

5.1 Approach

The exploration system consists of mapping, information-theoretic planning, and a monocular visual-inertial navigation system (Fig. 5.1).

5.1.1 GMM-based Mapping with Limited-FoV Sensor

The limited FoV sensor model is directional, so it is approximated by two non-intersecting tetrahedra such that their union forms a rectangular pyramid (shown in Fig. 5.2). For two sensor FoVs the intersection between the four pairs of tetrahedra is calculated and the intersection points found. The convex hull of the intersection points is converted to a polyhedron mesh with triangular facets. The volume of the convex hull is found by summing individual volumes of the tetrahedrons that make up the polyhedron [65]. The

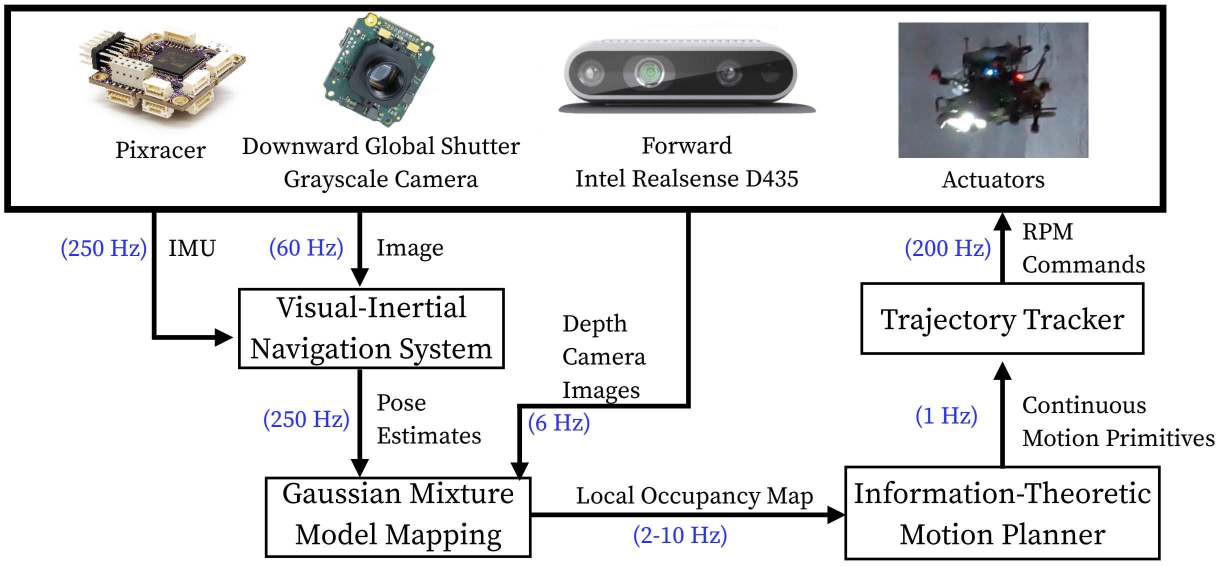

Pixracer
Downward Global Shutter Grayscale Camera
Forward Intel Realsense D435
Actuators
(250 Hz) IMU
(60 Hz) Image
Visual-Inertial Navigation System
Depth Camera Images
(6 Hz)
(250 Hz) Pose Estimates
Gaussian Mixture Model Mapping
Local Occupancy Map
(2-10 Hz)
Information-Theoretic Motion Planner
(1 Hz) Continuous Motion Primitives
Trajectory Tracker
(200 Hz) RPM Commands

Figure 5.1: Overview of the autonomous exploration system presented in this chapter. Using pose estimates from a visual-inertial navigation system (**??**) and depth camera observations, the mapping method (**??** and Section 3.2.3) builds a memory-efficient approximate continuous belief representation of the environment while creating local occupancy grid maps in real-time. A motion primitives-based information-theoretic planner (Section 6.1.2) uses this local occupancy map to generate snap-continuous forward-arc motion primitive trajectories that maximize the information gain over time.

overlap is estimated as a percentage of overlapping volume between two sensor FoVs and a sensor observation is only stored if its overlap exceeds a user-defined threshold. In this way, the number of components that represent a given location can be reduced while ensuring that the environment is covered.

5.1.2 Planning for Exploration

This work utilizes an information-theoretic planning strategy using CSQMI as the primary reward function, extending the prior work [36] to support limited FoV sensors in addition to 360° FoV sensors. The proposed framework can be divided into two stages: (1) action space generation and (2) action selection. At the start of any planning iteration, the planner uses the action generation strategy (detailed in Section 5.1.3) to generate a set of candidate actions up to a user-specified planning horizon using motion primitives. The action selector evaluates the collision-free and dynamically feasible subset of the action space using CSQMI as a reward function, returning the most informative plan to execute during the next planning iteration (see Sections 5.1.4 and 5.1.5).

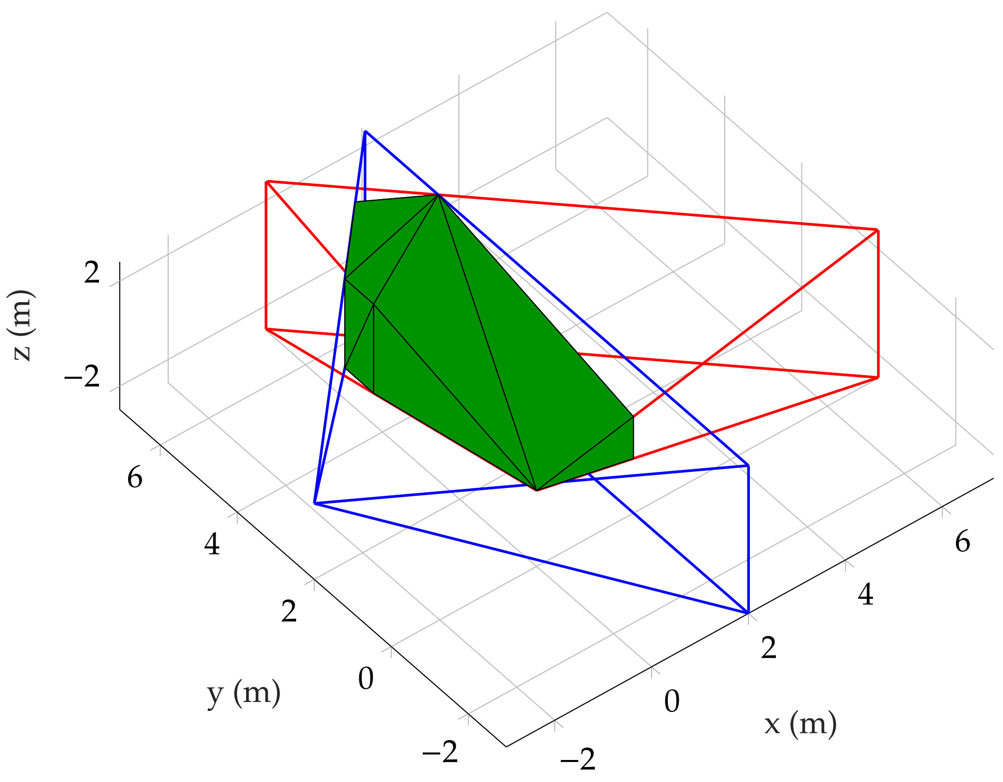


Figure 5.2: For limited FoV sensors, the FoV is approximated by the illustrated blue and red rectangular pyramids. These FoVs may also be represented as tetrahedra. To determine if a sensor position should be stored, the overlapping volume between the two approximated sensor FoVs is found.

5.1.3 Action Space Generation

This section describes the design of the action space for the exploration planner and how it can be modified for operation with different field-of-view sensors.

Designing the Action Space

The final action space, $\mathcal{X}_{\text{act}}$, is a collection of MPLs selected according to three criteria: (1) rate of information gain, (2) safety, and (3) limitations in compute. Prior work [36] provides such a design for a 360° FoV sensor (LiDAR). Goel et al. [15] present an analysis on how these three factors influence $\mathcal{X}_{\text{act}}$ for a limited FoV depth sensor. This work extends [36] using the analysis in [15], yielding a motion planner amenable for exploration with either a LiDAR or a depth sensor and that ensures similar exploration performance in either case (see Section 4.2).

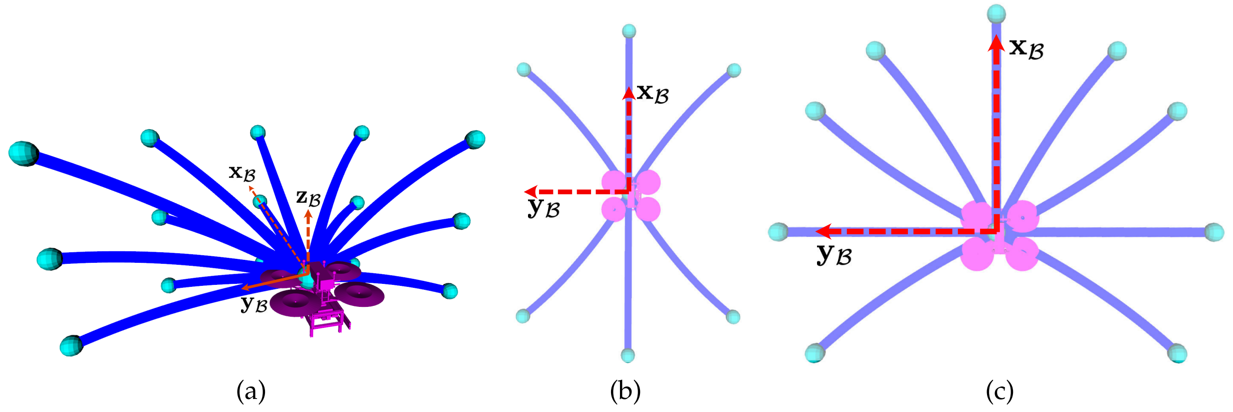


Figure 5.3: Action space design for the proposed information-theoretic planner. (a) shows a single motion primitive library generated using bounds on the linear velocity along $\{\mathbf{x}_\mathcal{B}, \mathbf{z}_\mathcal{B}\}$ and the angular velocity along $\{\mathbf{z}_\mathcal{B}\}$. (b) and (c) show top-down views of the motion primitive library collections used when the sensor model is a LiDAR [36] and a depth camera [15] respectively (off-plane primitives are not shown). The proposed planner can be used with either of these sensors using the appropriate action space designs explained in Section 5.1.3.

Action Space for 360° FoV Sensors 360° FoV sensors are advantageous in an exploration scenario because of three factors: (1) 360° depth data from the sensor allows for visibility in all azimuth directions, (2) a larger volume is explored per unit range when compared to a limited FoV sensor, and (3) yaw in-place motion does not help gain information. The first factor enables backward and sideways motion into the action space $\mathcal{X}_{\text{act}}$ without sacrificing safety (Fig. 5.3b). The second factor influences the entropy reduction: for the same trajectory, a sensor with a larger FoV will explore more voxels compared to the limited FoV case. The third factor reduces the number of motion primitive libraries in the action space to yield increased planning frequency. An example of an action space designed while considering these factors is presented in [36] and the corresponding parameters are shown in Table 5.1a.

These factors indicate that the same action space $\mathcal{X}_{\text{act}}$ cannot be used for limited FoV cameras if comparable exploration performance is to be maintained. This motivates the need for an alternate and informed action design for the limited FoV cameras.

Action Space for Limited FoV Sensors Goel et al. [15] show that for an exploration planner using a limited FoV sensor, the design of the action space $\mathcal{X}_{\text{act}}$ can be informed by the sensor model. The authors consider a depth sensor to design $\mathcal{X}_{\text{act}}$ by incorporating the sensor range and FoV, among other factors. This work follows a similar approach yielding an action space that contains MPLs in both the $\mathbf{x}_\mathcal{B}$ and $\mathbf{y}_\mathcal{B}$ directions (Fig. 5.3c). The parameters to construct the MPL collection comprising $\mathcal{X}_{\text{act}}$ are shown in Table 5.1b.

MPL ID	Vel., Time	N_ω, $N_\mathbf{z}$	N_prim
1	$v_{\mathbf{x}_\mathcal{B}}, \tau$	3, 5	15
2	$v_{\mathbf{x}_\mathcal{B}}, 2\tau$	3, 5	15
3	$-v_{\mathbf{x}_\mathcal{B}}, \tau$	3, 5	15
4	$-v_{\mathbf{x}_\mathcal{B}}, 2\tau$	3, 5	15

(a) **LiDAR**

MPL ID	Vel., Time	N_ω, $N_\mathbf{z}$	N_prim
1	$v_{\mathbf{x}_\mathcal{B}}, \tau$	3, 5	15
2	$v_{\mathbf{x}_\mathcal{B}}, 2\tau$	3, 5	15
3	$v_{\mathbf{y}_\mathcal{B}}, \tau$	3, 5	15
4	$v_{\mathbf{y}_\mathcal{B}}, 2\tau$	3, 5	15
5	$-v_{\mathbf{y}_\mathcal{B}}, \tau$	3, 5	15
6	$-v_{\mathbf{y}_\mathcal{B}}, 2\tau$	3, 5	15
7	$\omega_{\mathbf{z}_\mathcal{B}}, \tau$	1, 5	5

(b) **Depth Camera**

Table 5.1: Discretization used to construct the action space $\mathcal{X}_\text{act}$ for the simulation experiments for (a) LiDAR and (b) depth camera cases. Total number of primitives for a MPL are denoted by $N_\text{prim} = N_\omega \cdot N_\mathbf{z}$. The base duration τ was kept at 3 s for all experiments.

Note that there is an additional MPL corresponding to a yaw-in-place motion, unlike the 360° FoV case, to compensate for the limited FoV of the depth camera. For further detail on how to obtain these parameters, please refer to [15, 66].

5.1.4 Information-Theoretic Objective

The action selection policy uses CSQMI as the information-theoretic objective to maximize the information gain over time. CSQMI is computed at k points along the primitive γ_{ξ_t}, and the sum is used as a metric to measure the expected local information gain for a candidate action $\mathcal{I}_\gamma$. However, this design may result in myopic decision-making. Therefore, frontiers are also incorporated to model the global spatial distribution of information [37]. This global reward, denoted by $\mathcal{V}_\gamma$, is calculated based on the change in distance towards a frontier along a candidate action. Using the node state ξ_0, end point state ξ_τ, and a distance field constructed based on the position of the frontiers, this reward can be calculated as $\mathcal{V}_\gamma = d(\xi_0) - d(\xi_\tau)$, where $d(\xi_t)$ denotes the distance to the nearest voxel in the distance field from state ξ_t [19].

5.1.5 Action Selection

Using the rewards described in the preceding section, the objective for the motion planner is defined as follows [15, 19]:

$$\begin{aligned} &\underset{\gamma_{\xi_t}}{\operatorname{argmax}}\ \mathcal{I}_\gamma + \alpha \mathcal{V}_\gamma \\ &\text{s.t. } \gamma_{\xi_t} \in \mathcal{X}_\text{act} \end{aligned} \tag{5.1}$$

Algorithm 1 Overview of Action Selection for Exploration

```
1:  input: $\mathcal{X}_{\text{act}}, \mathcal{X}_{\text{free}}$
2:  output: $\gamma^*_{\xi_t}$                                    ▷ best action
3:  for $\Gamma_{\xi_t} \in \mathcal{X}_{\text{act}}$ do
4:      for $\gamma_{\xi_t} \in \Gamma_{\xi_t}$ do
5:          $feasible \leftarrow$ SafetyCheck($\gamma_{\xi_t}, \gamma^{\text{stop}}_{\xi_t}, \mathcal{X}_{\text{free}}$)
6:          if $feasible$ then
7:              $\mathcal{I}_\gamma \leftarrow$ InformationReward($\gamma_{\xi_t}$)
8:              $\mathcal{V}_\gamma \leftarrow$ FrontierDistanceReward($\gamma_{\xi_t}$)
9:          else
10:             $\mathcal{I}_\gamma \leftarrow 0.0, \mathcal{V}_\gamma \leftarrow 0.0$
11:         end if
12:     end for
13: end for
14: return $\gamma^*_{\xi_t} \leftarrow \underset{\gamma_{\xi_t} \in \mathcal{X}_{\text{act}}}{\text{argmax}}[\mathcal{I}_\gamma + \mathcal{V}_\gamma]$
```

where α is a weight that adjusts the contribution of the frontier distance reward. Recall, the goal is to maximize this reward function in real-time on a compute-constrained aerial platform. Previous information-theoretic approaches that construct a tree and use a finite-horizon planner either do not use a global heuristic [67] or are not known to be amenable for operation on compute-constrained platforms [19]. In this work, a single-step planner is used with the action space $\mathcal{X}_{\text{act}}$ consisting of motion primitives of varying duration for real-time performance (see Table 5.1). Due to this choice, the planner computes rewards over candidate actions that extend further into the explored map from the current position. In this manner, longer duration candidate actions provide a longer lookahead than the case when all candidate actions are of the same duration even in single-step planning formulations (see Table 5.1).

The action selection procedure is detailed in Algorithm 1. For every candidate action γ_{ξ_t} in the action space $\mathcal{X}_{\text{act}}$, a safety check procedure is performed to ensure that this candidate and the associated stopping action ($\gamma^{\text{stop}}_{\xi_t}$) are dynamically feasible and lie within free space $\mathcal{X}_{\text{free}}$ (Line 5). The free space check is performed using a Euclidean distance field created from locations of occupied and unknown spaces in the robot's local map given a fixed collision radius [60]. Checking that the stopping action is also feasible ensures that the planner never visits an inevitable collision state, which is essential for safe operation [56]. If the action is feasible, the local information reward ($\mathcal{I}_\gamma$, Line 7) and frontier distance reward ($\mathcal{V}_\gamma$, Line 8) are determined as described in Section 5.1.4. The planner then returns the action with the best overall reward (Line 14).

5.2 Results and Analysis

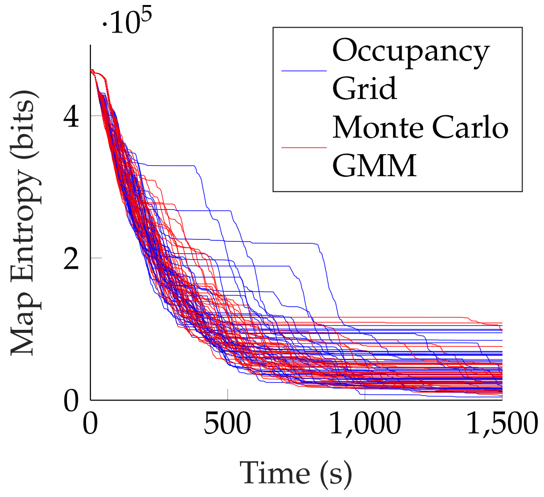
This section details the experiments to validate the approach. Results are reported for both real-time simulation trials and field tests in caves. The following shorthand is introduced for this section only: MCG will refer to the Monte Carlo GMM mapping approach and OG mapping will refer to the Occupancy Grid mapping approach. The mapping and planning software is run on an embedded Gigabyte Brix 8550U with eight cores and 32 GB RAM, for both hardware and simulation experiments. Unless otherwise noted, the parameters for simulation and hardware experiments are equal.

5.2.1 Comparison Metrics

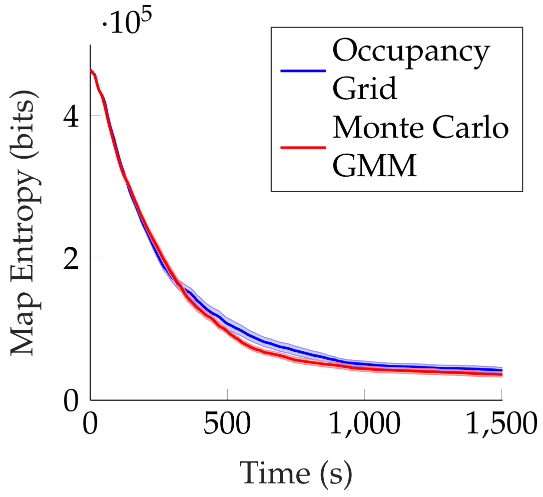
To calculate the memory requirements for the OG mapping approach, the incremental OG map is transmitted as a changeset pointcloud where each point consists of 4 floating point numbers: $\{x, y, z, \text{logodds}\}$. The changeset is computed after insertion of every pointcloud. A floating point number is assumed to be four bytes, or 32 bits. For the MCG approach, the cumulative data transferred is computed by summing the cost of transmitted GMMs. Each mixture component is transmitted as 10 floating point numbers: six numbers for the symmetric covariance matrix, three numbers for the mean, and one number for the mixture component weight. One additional number is stored per GMM that represents the number of points from which the GMM was learned. The transform between the sensor origin reference frame and the global reference frame is stored for each GMM using six numbers to represent the three translational and three rotational degrees of freedom. To ensure a fair comparison of exploration performance between the two approaches, a global occupancy grid serves as a referee and is maintained in the background with a voxel resolution of 0.2 m. This occupancy grid is used to compute map entropy over time [42], thus measuring exploration progress during a simulation or a hardware experiment.

Table 5.2: Reconstruction error for Fig. 5.5. The error is calculated as the PointCloud-to-Mesh distance between the environment reconstructions and mesh.

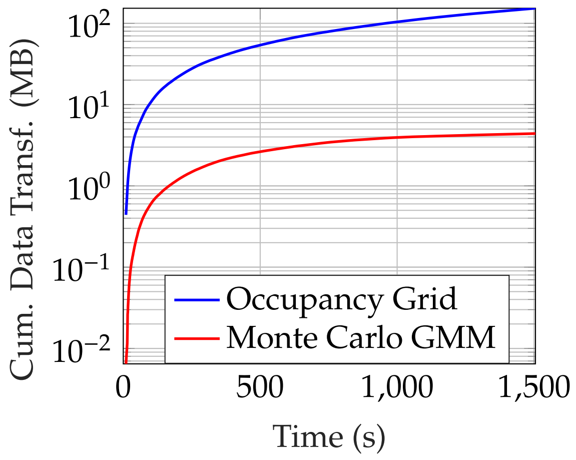
	Mean (m)	**Std (m)**
MCG	1.3×10^{-2}	1.9×10^{-2}
OG	6.3×10^{-2}	3.9×10^{-2}



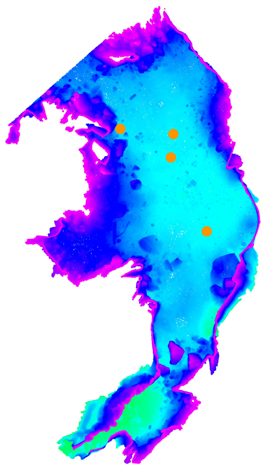
(a)



(b)



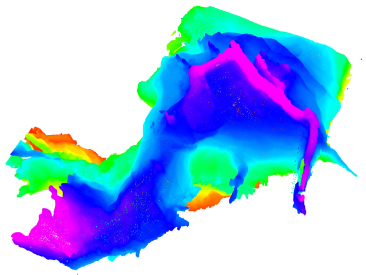
(c)

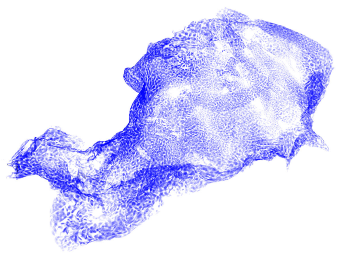


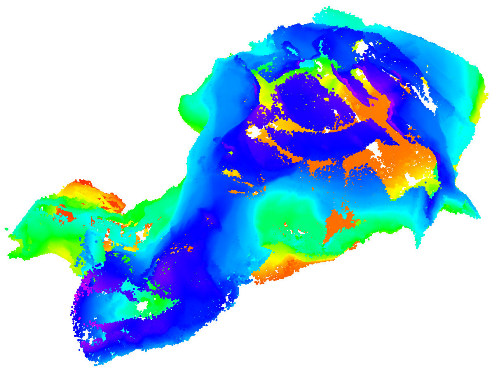
(d)

Figure 5.4: Exploration statistics for simulation experiments. (a) illustrates the map entropy over time for 80 trials (40 trials per mapping method), (b) illustrates the average map entropy over time. Although both mapping methods achieve similar entropy reduction, MCG uses significantly less memory according to the average cumulative data transferred shown in (c). The average cumulative data transferred at the end of 1500 s is 4.4 MB for the MCG approach and 153 MB for the OG approach. The MCG method represents a decrease of approximately two orders of magnitude as compared to the OG method. The experiments are conducted in the simulated cave environment shown in (d). The four starting positions are shown as orange dots.

(a) Simulated Cave Environment


(b) Points from the PLY


(c) GMM map (1σ covariances)


(d) Resampled pointcloud

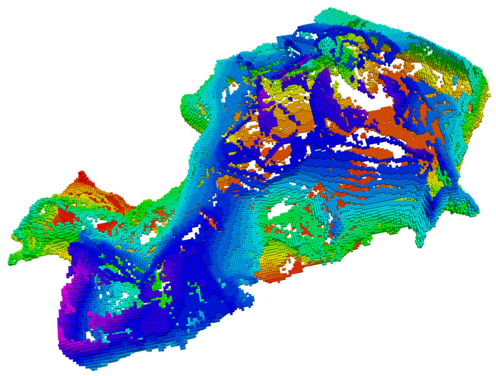

(e) Dense voxel map

Figure 5.5: The colorized mesh used in simulation experiments is shown in (a) and produced from FARO scans of a cave in West Virginia. (b) illustrates the pointcloud from the mesh shown in (a). (c) illustrates the MCG map with 1σ covariances, which is densely resampled with 1×10^6 points, to obtain the reconstruction shown in (d). (e) illustrates the dense voxel map with 20 cm voxels after 1500 s of exploration with the depth camera sensor model. The reconstruction accuracy for (d), and (e) are shown in Table 5.2. All pointclouds shown are colored from red to purple according to z-height.

5.2.2 Simulation Experiments

The exploration strategy is evaluated with 80 real-time simulation trials over approximately 67 hours in a $30\,\text{m} \times 40\,\text{m} \times 6\,\text{m}$ environment constructed from colorized FARO pointclouds of a cave in West Virginia (see Fig. 5.5a). In each simulation, the multirotor robot begins exploration from one of four pre-determined starting positions and explores for 1500 s. Ten exploration tests for each of the four sensor configurations are run from each of the four starting positions leading to a total of 80 trials. The end time of 1500 s is empirically set based on the total time required to fully explore the cave. Note that ground truth state estimates are used for these simulation experiments, while the hardware experiments in Section 5.2.3 rely on visual-inertial odometry [68]. The reconstruction error (Table 5.2 and Figs. 5.5d and 5.5e) is computed as pointcloud-to-mesh distances between reconstructed pointclouds from each trial and the environment mesh. In the case the MCG approach, the GMM map is densely resampled to produce a pointcloud. For the OG case, the occupancy grid map is converted to a pointcloud by assuming the points to be at the center of each voxel.

Depth Camera Simulations

The depth camera sensor model also has a max range of 5.0 m and operates at 10 Hz for all simulation experiments. For all simulation trials, the maximum speed in the $\mathbf{x}_{\mathcal{B}} - \mathbf{y}_{\mathcal{B}}$ plane is $\|V_{\max}\| = 0.75\,\text{m/s}$, the maximum speed along the $\mathbf{z}_{\mathcal{B}}$ axis is $V_{\mathbf{z}} = 0.5\,\text{m/s}$, and the maximum yaw rate is $\Omega = 0.25\,\text{rad/s}$. CSQMI is computed at the end point of the candidate action ($k = 1$). $\lambda = 5$ and $n_f = 2$ for all simulations and hardware trials.

The MCG approach outperforms OG in terms of memory efficiency while maintaining similar exploration performance. Figure 5.4c depicts the cumulative amount of data transfer in this case. After 1500 s, transferring the MCG map requires 4.4 MB an 153 MB to incrementally transfer the OG map. Further, the MCG approach has lower average reconstruction error as compared to the OG approach (see Fig. 5.5e) as shown in Table 5.2.

5.2.3 Hardware Experiments

Flight Arena Experiments

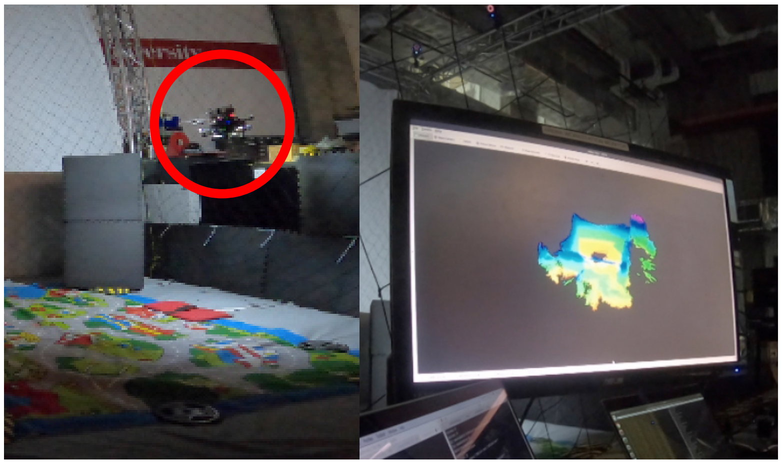

(a) (b)

Figure 5.6: Flight arena exploration scenario. (a) is a still image of the robot flying during one of the MCG trials (full video of the trial may be found at `https://youtu.be/egwjv7YwHPE`) and (b) illustrates the live map transmitted to the base station from the same trial.

Experiments were conducted in a flight arena with the aerial system. Each approach (MCG and OG) was flown five times for 150 s. The results are shown in Fig. 5.7 and a video of one MCG trial with the live map displayed on the base station may be found at `https://youtu.be/egwjv7YwHPE`.

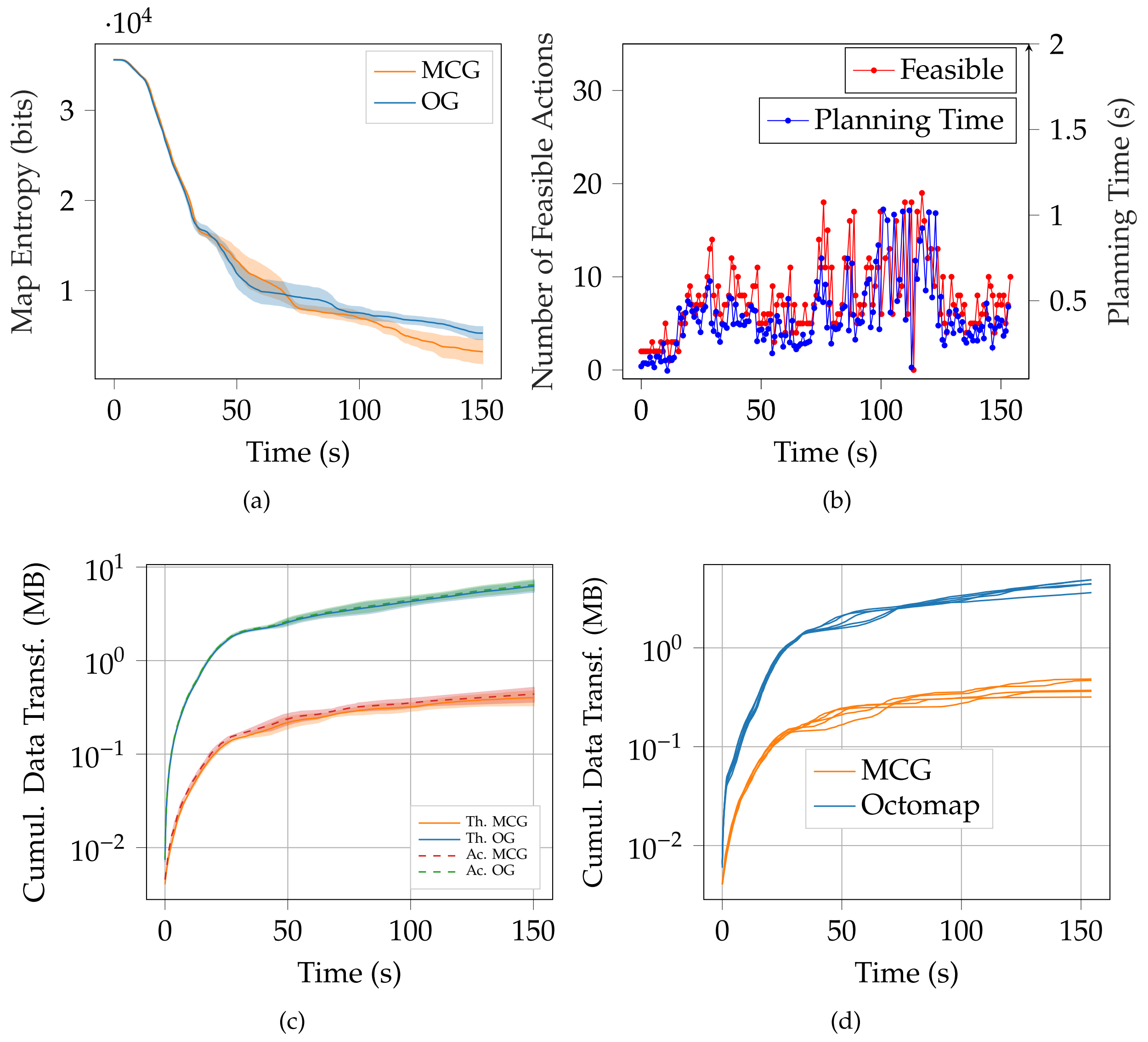


Figure 5.7: Results of repeatability trials for the MCG and OG approaches in a flight arena. (a) plots the mean entropy reduction (with the associated standard error) for five trials for the MCG and OG methods. The mean and the standard error for the cumulative data transferred is provided for each approach in (c). The theoretical (Th. OG and Th. MCG) communications is compared to actual (Ac. OG and Ac. MCG) communications transmitted to the base station using UDP. (b) provides a plot of the number of feasible actions in red with the planning time shown in blue. (d) Uses data from the MCG flights and generates an Octomap in postprocessing to compare the communications required. The Octomap performance is similar to that of the OG approach. More details about this analysis is provided in Section 5.2.3.

Figure 5.7a provides the mean and standard error for the map entropy over time for the 10 trials (5 for each approach). The communications plot shown in Fig. 5.7c illustrates the theoretical cost to transmit the data (labeled as *Th. OG* and *Th. MCG*) that was calculated during the simulation trials (shown in Fig. 5.4c) as well as the actual transmitted data (labeled as *Ac. OG* and *Ac. MCG* in dashed lines). The actual transmitted data is calculated

Trial	Occ. GMM		Free GMM		Occ. Recon.	
	Mean (s)	Std. (s)	Mean (s)	Std. (s)	Mean (s)	Std. (s)
0	0.64	0.51	0.03	0.02	0.11	0.13
1	0.53	0.47	0.03	0.02	0.08	0.09
2	0.54	0.47	0.04	0.02	0.11	0.13
3	0.52	0.34	0.03	0.03	0.13	0.15
4	0.48	0.25	0.03	0.02	0.11	0.14

(a) Mapping statistics

Trial	Autonomy		Realsense	
	CPU (%)	Mem. (B)	CPU (%)	Mem. (B)
0	186.52	6.76×10^8	36.34	1.29×10^8
1	153.16	7.76×10^8	30.43	1.32×10^8
2	182.52	6.33×10^8	31.69	1.33×10^8
3	203.55	7.61×10^8	29.58	1.24×10^8
4	178.78	6.13×10^8	30.24	1.31×10^8

(b) Memory and compute usage on Gigabyte Brix

Trial	Feasible Actions				Infeasible Actions				Planning			
	Mean (#)	Std. (#)	Max. (#)	Min. (#)	Mean (#)	Std. (#)	Max. (#)	Min. (#)	Mean (s)	Std. (s)	Max. (s)	Min. (s)
0	7.60	3.82	19	0	37.89	5.59	46	9	0.40	0.20	1.03	0.09
1	6.47	2.72	16	1	39.29	3.86	45	12	0.34	0.14	1.03	0.09
2	8.01	3.94	20	1	37.84	4.39	45	13	0.43	0.19	1.05	0.10
3	10.52	5.18	24	1	35.41	5.40	45	12	0.50	0.22	1.03	0.08
4	7.26	4.54	24	2	38.38	5.98	44	3	0.38	0.22	1.05	0.12

(c) Planning statistics

Trial	VINS		BlueFox		PX4		Control		Comms	
	CPU (%)	Mem. (B)	CPU (%)	Mem. (B)	CPU (%)	Mem. (B)	CPU (%)	Mem. (B)	CPU (%)	Mem. (B)
0	154.84	1.03×10^8	27.98	8.50×10^7	16.98	2.16×10^7	16.10	3.15×10^7	0.19	2.10×10^7
1	155.15	1.05×10^8	27.82	8.37×10^7	16.92	2.14×10^7	15.96	3.12×10^7	0.18	1.91×10^7
2	153.76	1.02×10^8	27.86	8.55×10^7	16.88	2.15×10^7	16.04	2.91×10^7	0.19	1.91×10^7
3	154.45	1.01×10^8	27.85	8.59×10^7	16.87	2.14×10^7	15.99	3.12×10^7	0.19	2.10×10^7
4	151.46	1.03×10^8	27.46	8.56×10^7	16.88	1.93×10^7	16.02	2.91×10^7	0.20	2.10×10^7

(d) Memory and compute usage on Nvidia Jetson TX2

Figure 5.8: Comupational evaluation of the exploration framework. (c) and (b) provide timing, compute, and memory statistics for each subsystem for each of the five MCG flights. The figures reported in Fig. 5.8b are averages.

as the cumulative sum of the size of the UDP packets sent over the WiFi router to the base station. This plot demonstrates that the theoretical estimate closely matches the actual transmitted amount of data. No communications dropouts occurred in these trials because the base station and router were stationary and close to the aerial system's flight volume. Section 5.2.3 analyzes the effect of the robot moving away from the base station and router to force a communications dropout.

Figure 5.6a contains still images of the flight and Fig. 5.6b depicts a live map. Fig. 5.7b plots the number of feasible actions available to the robot at a given time on the left and the time to plan on the right.

Figure 5.7d plots the theoretical data transferred and compares it to the OctoMap volumetric map [40]. To generate this plot, a 0.2 m leaf size OctoMap that matches the OG approach's voxel size is created. For each scan added to the map, the set of voxels whose occupancy values (note: not state) have changed are used to generate an OctoMap submap (i.e., a new OctoMap that contains the change set). Submaps for this analysis are created by storing the incremental change set as an OctoMap to enable the exact map to be reconstructed on the receiving computer. The full probabilistic model is saved to disk because occupancy probabilities must be preserved to enable calculation of the mutual information [34] for information-theoretic exploration. The serialized stream does not contain any 3D coordinates. Instead, the spatial relationships between the nodes are stored in the encoding. Eight bits per node are used to specify whether a child node exists and an additional floating point number stores the occupancy value for that node. Due to this design there is some overhead to encode the multi-resolution nature of the data structure. The results demonstrate that the MCG approach also outperforms the OctoMap approach in terms of communication efficiency. Figure 5.8c provides statistics for the planning and mapping components for the MCG trials. The planning times and statistics about the planning actions are provided. Out of all the trials, the planning subsystem triggers the stopping action only once (in Trial 0). Timing results for the generated occupied and free GMMs as well as the time to reconstruct occupancy are provided. Figures 5.8b and 5.8d provide mean memory and compute statistics for the autonomy subsystem, which consists of mapping and planning. It also provides statistics for the realsense, bluefox, and PX4 subsystems to quantify the memory and CPU utilization to stream images and IMU measurements as well as transmitting cascaded commands to the flight controller. Statistics are also provided for communication and state estimation.

Communication Dropout Experiment

A communication dropout was triggered by performing an experiment where the aerial system is carried away from the WiFi router until it is out of range. Figures 5.9a and 5.9b depict the data sent from the aerial system and received by the base station over time (note: the clocks on the aerial system and base station are not synchronized). A view of the operating environment and map is shown in Fig. 5.9c. While data is dropped at around 35 s, the communications are re-established at around 55 s when the robot reapproaches the base station. A video at `https://youtu.be/UVn2BbMQRJg` illustrates the experiment.

Laurel Caverns

The approach is tested in total darkness at Laurel Caverns[1], a commercially operated cave system in Southwestern Pennsylvania consisting of over four miles of passages[2][4]. Figure 5.10a illustrates a composite image from several still images of the robot exploring the Laurel Caverns Dining Room. Two experiments were conducted, one for each of the MCG and OG approaches for a 95 s duration. The map entropy reduction over time is shown in Fig. 5.11c and is similar for both approaches, while the cumulative data transferred (Fig. 5.11d) to represent the maps is more than an order of magnitude lower for the MCG approach as compared to the OG approach. Note, however, that the communication reported for this experiment represents the theoretical, or estimated, communications needed to transmit the data. The data was not transmitted to a base station. The data transfer rate in Fig. 5.11e is calculated using Euler differentiation but note that the accuracy is affected by the limited number of samples. During hardware trials, a bounding box was used to constrain the exploration volume. To put the localization accuracy into perspective, the drift in position is about 0.53 m during a 50.9 m cave flight and the rotation drift is about 0.32 rad over 33.5 rad which is about a 1% drift in both translation and rotation. Position drift may be approximated as the difference between the initial and final position estimates because the robot takes off and lands at the same location.

West Virginia Cave

The approach was also tested in total darkness in a cave in West Virginia[5]. Figure 5.12a illustrates the map entropy reduction over time with a maximum duration of 150 s. The MCG and OG approaches perform similarly in these trials. The actual data transferred between the robot and base station is shown in Fig. 5.12b. Figure 5.12c is a composite image

[1]http://laurelcaverns.com/

[2]The authors acknowledge that caves are fragile environments formed over the course of tens of thousands to millions of years. Laurel Caverns was chosen as a test site because it has relatively few speleothems[3]due to its sandstone overburden and the high silica content of the Loyalhanna limestone [71]. The authors worked with cave management to select a test site that contained low speleothem growth to minimize risk of damage to the cave. Cave management monitored all flights. No flights were executed near delicate formations.

[3]Speleothems are mineral formations found in limestone caves (e.g., stalagmites, stalagtites, and flowstone) that are composed of calcium carbonate, precipitated from groundwater that has percolated through adjacent carbonate host rock [72].

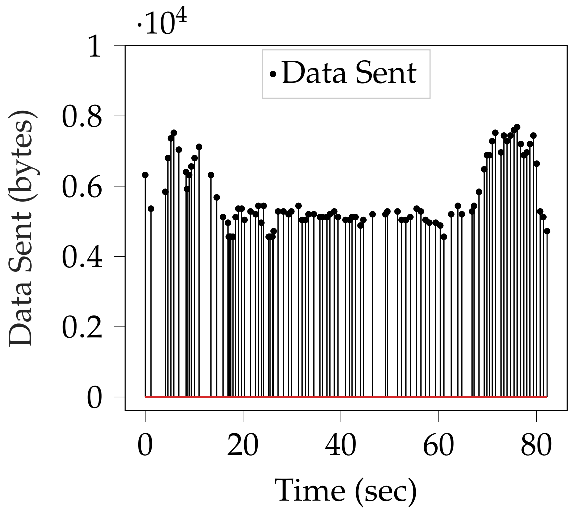
[4]Bat populations in the northeastern U.S. have been decimated with the onset of White-nose Syndrome in the winter of 2007-2008 [73]. Great care was taken not to disturb bats with the aerial systems during the hibernating season.

[5]The region of the cave where flight experiments were conducted contained speleothems that have ceased growing. Speleothems growth may terminate due to geologic, hydrologic, chemical, or climatic factors that cause water percolation to cease at a particular drip site [72]. The authors worked with cave management to select a test site that had neither actively growing speleothems or bats.

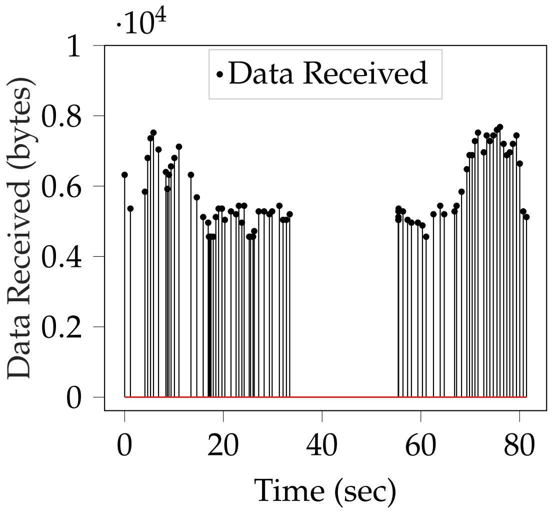
from several still images of the robot exploring the cave. A video of one exploration run may be found at the following link: `https://youtu.be/H8MdtJ5VhyU`. In these experiments a bounding box was used to constrain the exploration volume. The drift in position is about 1.28 m during the 82.6 m flight and rotation drift is about 0.55 rad over 41.8 rad which is about a 1.5% drift in position and 1.3% drift in rotation.

5.3 Summary

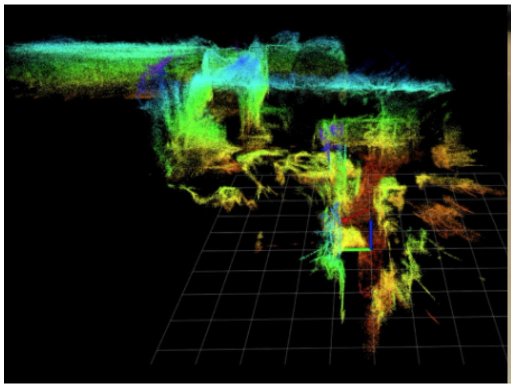
The results presented in this paper comprise the beginning of an promising line of research for autonomous cave surveying and mapping by aerial systems. A high-fidelity model amenable to transmission across low-bandwidth communications channels is achieved by leveraging GMMs to compactly represent the environment. The method is demonstrated with limited field of view sensors utilizing a motion primitives-based planning framework.



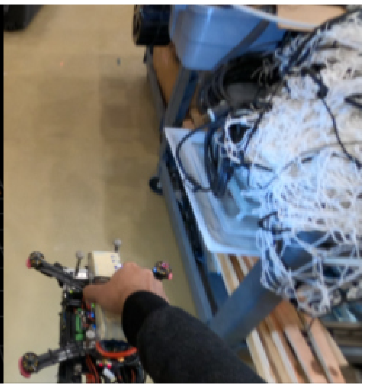
(a)



(b)

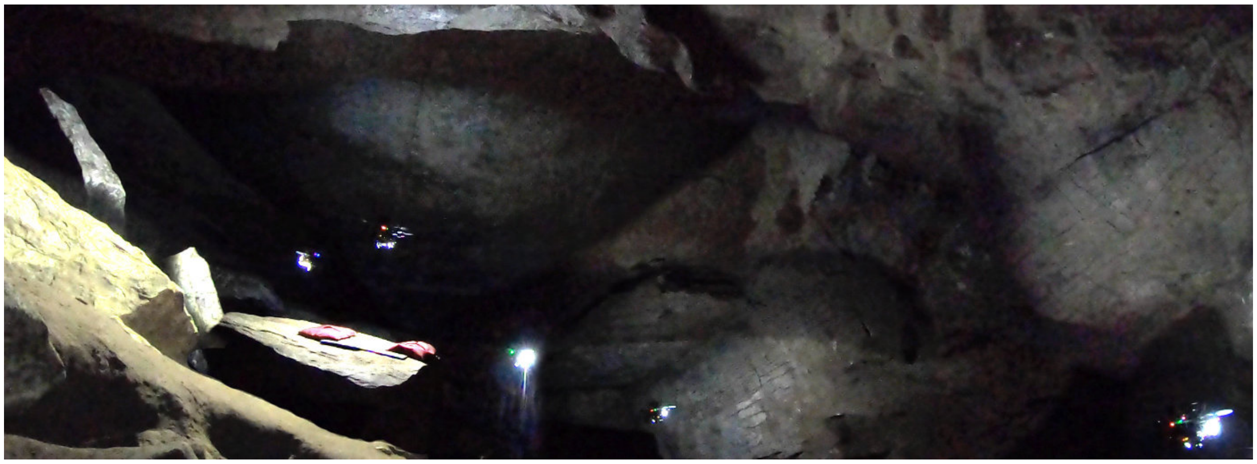


(c)

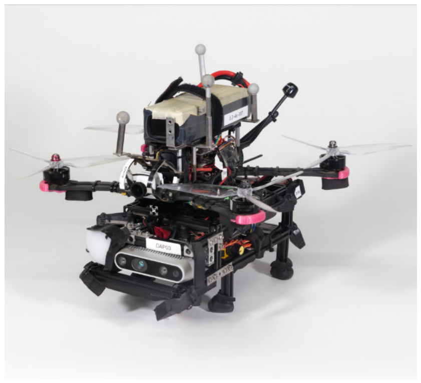


(d)

Figure 5.9: Results of forcing a communication dropout on the system. The aerial system is carried through a research lab and down a hallway away from the base station and router to force a communications dropout. The accompanying video may be found at `https://youtu.be/UVn2BbMQRJg`. (a) illustrates the data sent from the robot and (b) is the data received by the base station (note: the base station and robot do not have their clocks synced). (c) illustrates the live map produced by the base station. (d) illustrates a view of the aerial system at the start of the experiment from a camera mounted on the operator's helmet.

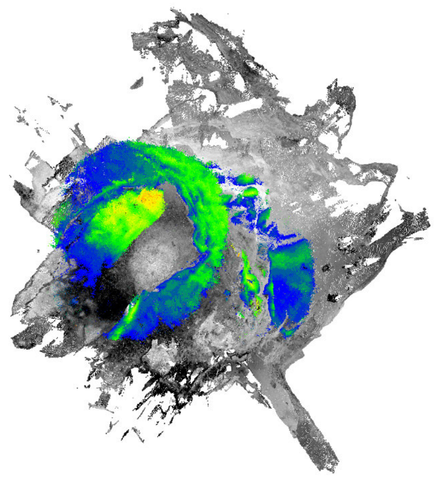


(a)

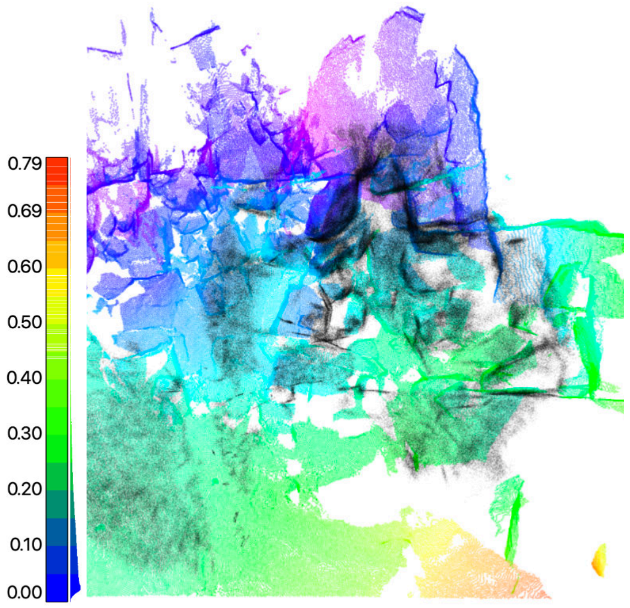


(b)

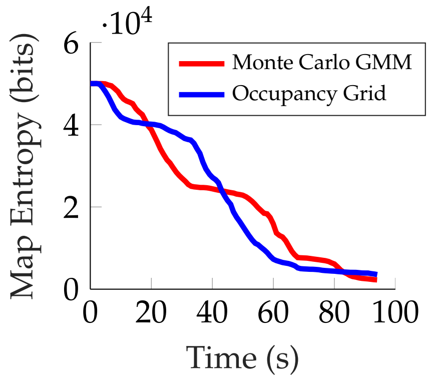
Figure 5.10: (a) A single aerial system explores the Dining Room of Laurel Caverns in Southwestern Pennsylvania. Still images of the robot exploring the environment are super-imposed to produce this figure. (b) The aerial system with dimensions 0.25 m × 0.41 m × 0.37 m including propellers carries a forward-facing Intel Realsense D435 for mapping and downward-facing global shutter MV Bluefox2 camera (not shown). The pearl reflective markers are used for testing in a motion capture arena but are not used during field operations to obtain hardware results. Instead, a tightly-coupled visual-inertial odometry framework is used to estimate state during testing at Laurel Caverns.



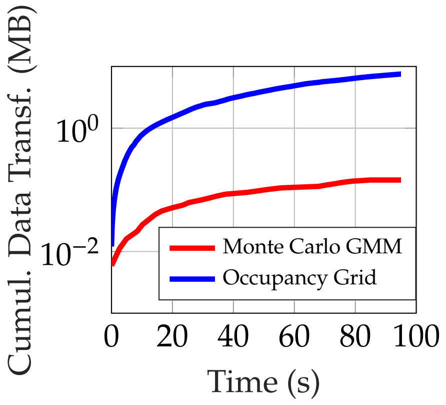
(a)



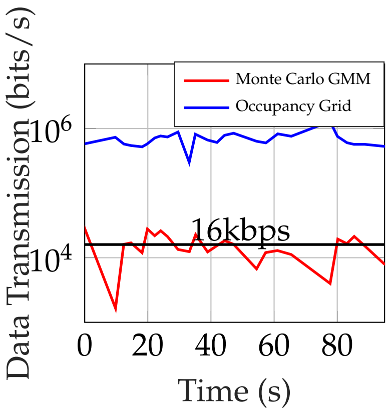
(b)



(c)

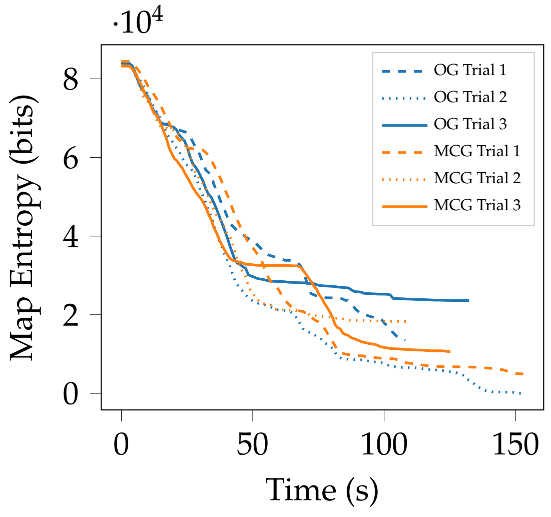


(d)

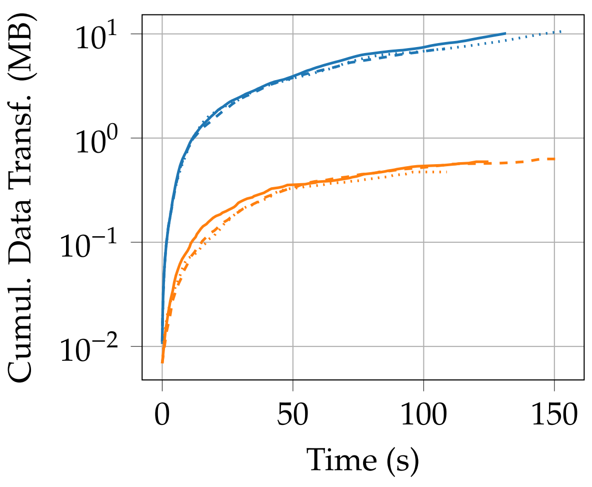


(e)

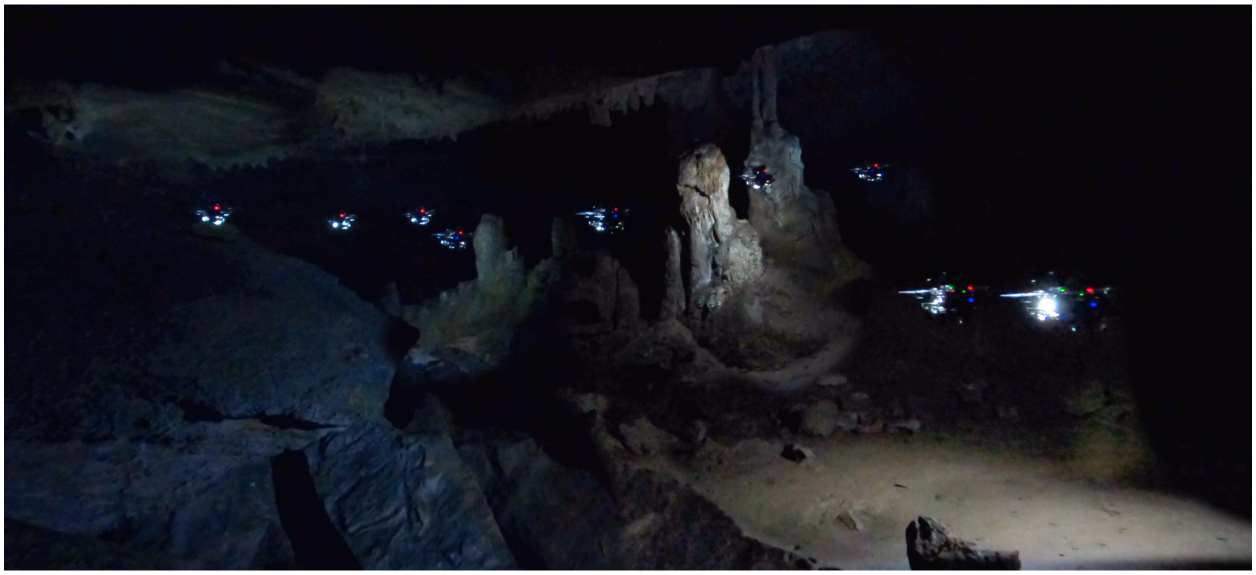
Figure 5.11: Exploration statistics from the experiments at Laurel Caverns. (a) illustrates the reconstruction error of the resampled GMM map as compared to the FARO map by calculating point-to-point distances. The distribution of distances is shown on the right-hand side. The mean error is 0.14 m with a standard deviation of 0.11 m. In particular, there is misalignment in the roof due to pose estimation drift. (b) A subset of the resampled GMM map (shown in black) is overlaid onto the FARO map (shown in colors ranging from red to purple) that displays the breakdown in the middle of the Dining Room. (c) The entropy reduction and (d) cumulative data transferred for one trial for each of the Monte Carlo GMM mapping and OG mapping approaches are shown. The communication is a theoretical calculation – not actual transmitted data. While the map entropy reduction for each approach is approximately similar, the GMM mapping approach transmits significantly less memory than the OG mapping approach (0.1 MB as compared to 7.5 MB). (e) illustrates the bit rate for each approach in a semi-logarithmic plot where the vertical axis is logarithmic. The black line illustrates how the approaches compare to 16kbps. For comparison, 16kbps is sufficient to transmit a low resolution (176 × 144 at 5 fps compressed to 3200 bit/frame) *talking heads* video [69, 70].



(a)



(b)



(c)

Figure 5.12: Overview of the results from experiments in a cave in West Virginia. (a) The map entropy over time for three trials of the MCG and OG approaches. (b) The data transferred between a robot and base station for each trial. The communication reported is actual transmitted data over UDP to a base station. Note that while the exploration performance is similar for both approaches, the data transferred for the MCG approach is substantially less. (c) A composite image of one exploration trial composed of still images.

Chapter 6

Rapid and Communication-Efficient Multi-Robot Exploration

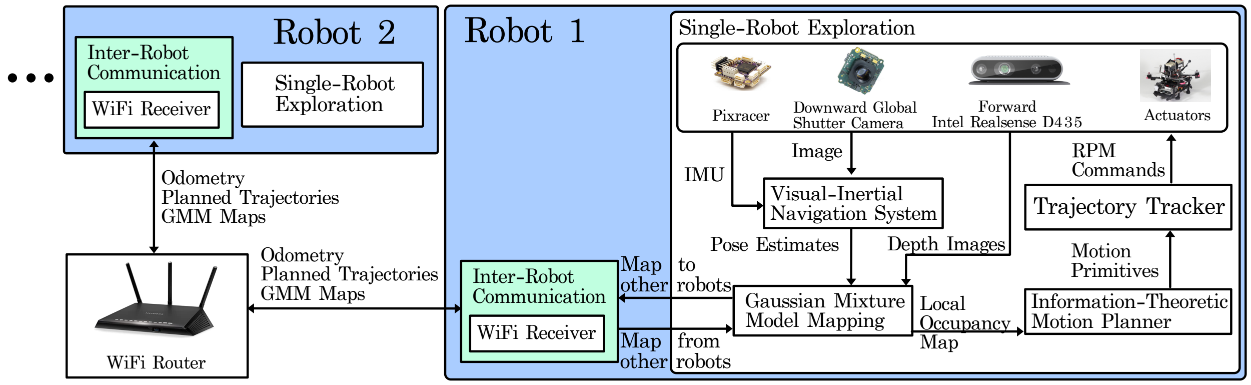
This chapter extends the communication-efficient mapping strategy developed in Chapter 5 to distributed mapping. Using the rapid exploration planner from Chapter 4 on each robot, the combined multi-robot system is experimentally evaluated in a real cave using two aerial robots.

6.1 Approach

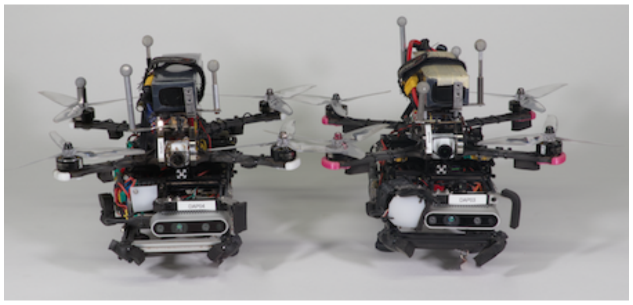
An overview of the system is shown in Fig. 6.1. Each robot is equipped with single-robot exploration and inter-robot communication modules. The exploration module consists of four major subsystems: GMM mapping, information-theoretic motion planning, visual-inertial state estimation, and trajectory tracking. The inter-robot communication module enables sharing information between robots or other computers on the network. The GMM mapping and planning subsystems together with the communication module constitute distributed mapping (Section 6.1.1) and multi-robot planning (Section 6.1.2), respectively. In this section, the following mathematical notation is used: lower-case letters represent scalar values, lower-case bold letters represent vectors, upper-case bold letters represent matrices, and script letters represent sets.

6.1.1 GMM-based Distributed Mapping

This section details the distributed mapping approach to share environment models between robots. Consider a team of N robots. At timestep t robot $i \in N$ receives the depth

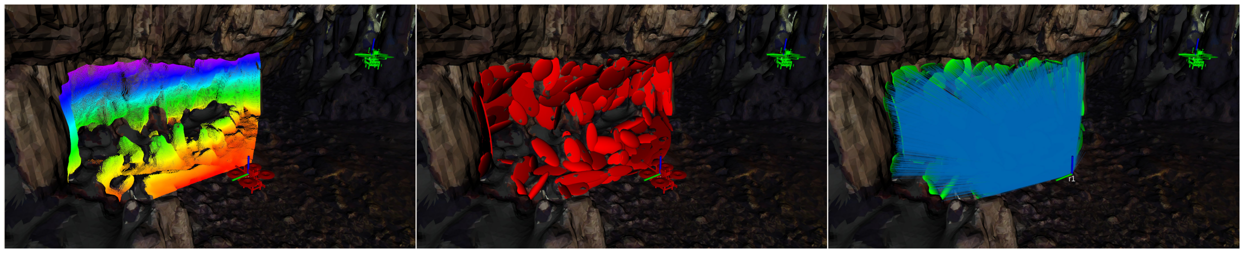


(a) Multi-robot exploration framework



(b) Multirotors used in this work

Figure 6.1: (a) Overview of the rapid multi-robot exploration framework and (b) aerial systems used in experiments in this work.



(a) (b) (c)

Figure 6.2: Overview of the distributed mapping approach. (a) Robot i shown in red, takes a sensor observation shown in colors varying from red to purple and (b) learns a GMM (shown in red). If the GMM is determined to be a keyframe both the GMM and sensor pose are transmitted to robot j (shown in green). (c) The GMM and the sensor pose are transformed into the frame of robot j and used to update the occupancy.

sensor observation, $\mathcal{Z}_t^i$, which represents a set of points. A Gaussian mixture model (GMM) is learned from these points following the approach from [13]. The GMM is parameterized by $\boldsymbol{\Theta} = \{\pi_m, \boldsymbol{\mu}_m, \boldsymbol{\Lambda}_m\}_{m=1}^M$ where $\boldsymbol{\mu}_m \in \mathbb{R}^3$ is a mean, $\boldsymbol{\Lambda}_m \in \mathbb{R}^{3\times3}$ is a covariance, and $\pi_m \in \mathbb{R}$ is a weight such that $\sum_{m=1}^M \pi_m = 1$. A GMM representing point set $\mathcal{Z}_t^i$ is denoted as $\boldsymbol{\Theta}_{\mathcal{Z}_t^i}$.

Keyframe GMMs

To reduce redundant observations, keyframe GMMs are identified for transmission to other robots. A keyframe GMM, $\hat{\mathbf{\Theta}}_{\mathcal{Z}_t^i}$, is determined by approximating the field of view for the current sensor observation as a rectangular pyramid and calculating the overlapping volume with other keyframe fields of view. If the volume is smaller than a user-defined threshold, λ, the sensor observation is considered to be a keyframe. $\hat{\mathbf{\Theta}}_{\mathcal{Z}_t^i}$ and the sensor pose, $\mathbf{S}_t^i \in \mathrm{SE}(3)$, are transmitted to the other robots or computers on the network.

Each robot maintains its own environment representation and relative initial transforms between robots are assumed to be known. When robot j receives $\hat{\mathbf{\Theta}}_{\mathcal{Z}_t^i}$, it is received in the frame of robot i. To transform it into the frame of robot j, the relative initial rotation $\mathbf{R}_0^{ji} \in \mathbb{R}^{3\times 3}$ and translation $\mathbf{x}_0^{ji} \in \mathbb{R}^3$ parameters are applied to the means and covariances of the distribution using the following equations.

$$\boldsymbol{\mu}^j = \mathbf{R}_0^{ji}\boldsymbol{\mu}^i + \mathbf{x}_0^{ji} \qquad \mathbf{\Lambda}^j = \mathbf{R}_0^{ji}\mathbf{\Lambda}^i(\mathbf{R}_0^{ji})^T, \tag{6.1}$$

The transformed GMM is incorporated into robot j's existing GMM map following the approach from [13, 36].

Occupancy Reconstruction

A local occupancy grid map $\mathbf{m}_t^i$ is maintained and centered around the robot's current position $\mathbf{x}_t^i$ for use in information-theoretic motion planning. To generate $\mathbf{m}_t^i$, a number of points $z \in \mathbb{R}^3$ equal to the support size, or number of points used to learn the distribution, is sampled and raytraced to the sensor pose $\mathbf{x}_t^i$. The probability of occupancy along the ray is updated.

Multi-robot Map Updates

Care must be taken to update $\mathbf{m}_t^j$ when receiving $\hat{\mathbf{\Theta}}_{\mathcal{Z}_t^i}$. In addition to applying the transformation parameters so that $\hat{\mathbf{\Theta}}_{\mathcal{Z}_t^i}$ is transformed into the frame of robot j, $\mathbf{m}_t^j$ must also be updated by sampling points from the transformed $\hat{\mathbf{\Theta}}_{\mathcal{Z}_t^i}$ and raytracing through $\mathbf{m}_t^j$ to the sensor pose, $\mathbf{S}_t^i$, which must also be transformed into the frame of robot j. This ensures the occupancy is updated with observations from both robots. A visualization of this is shown in Fig. 6.2. Robot i takes a sensor observation (Fig. 6.2a) and learns $\hat{\mathbf{\Theta}}_{\mathcal{Z}_t^i}$ (Fig. 6.2b). This keyframe GMM is transmitted to robot j, transformed into the frame of robot j, and then used to update $\mathbf{m}_t^j$ (Fig. 6.2c).

6.1.2 Planning for Rapid Multi-Robot Exploration

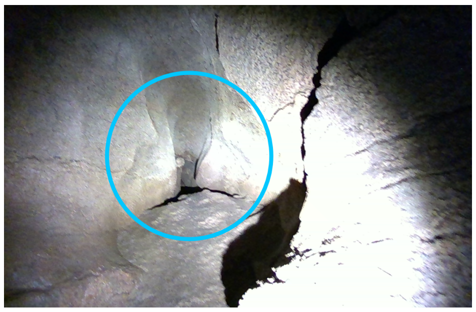
Robot i uses $\mathbf{m}_t^i$ for information-theoretic receding-horizon planning via the strategy presented in [15], which accounts for perception latencies and kinodynamic constraints of the robot. The approach uses Monte Carlo tree search (MCTS) [62] to evaluate the Cauchy-Schwarz Quadratic Mutual Information (CSQMI) [33] for a set of motion primitives over a user-specified time horizon. An informative primitive sequence is selected that maximizes the CSQMI over the MCTS tree. Safety is ensured by checking for collisions with the environment.

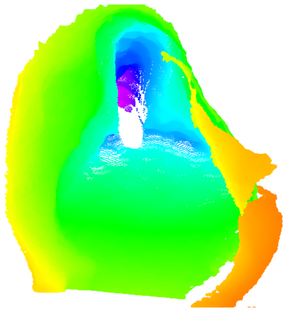
The informative trajectories are shared with other robots and inter-robot collision avoidance is enabled through a standard priority-based collision checker assuming a cylindrical robot model [74]. The priorities are assigned manually before the exploration run and remain constant throughout. To reduce the computational complexity for lower priority robots, three optimizations are applied. First, the collision checking is only active when a pair of robots are within a pre-specified radius. To enable this on each robot without assuming a centralized oracle, the robots share odometry information at a sufficiently high rate (10 Hz) compared to the planning frequency (1 Hz). Second, the number of cylinders sampled over the planned trajectory is limited to a pre-specified maximum to cap the number of cylinder-cylinder collision checks. This maximum value and the associated cylinder collision radius are selected conservatively based on the length of the motion primitive assuming the robot starts at hover and achieves a top speed at the endpoint. Third, for each robot the collision checks are performed only with the candidate motion primitive and the associated stopping motion primitive at the first depth of the MCTS tree because each depth of the tree is of a sufficiently long duration (2 s) as compared to the planning time (1 s). The inter-robot collision checker is used in the constrained-bandwidth simulation study (Section 6.2.3).

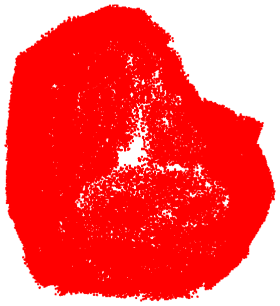
6.2 Results and Analysis

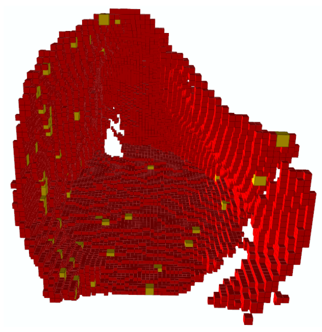
The experimental evaluation is motivated through a concept of operations for a multi-robot exploration mission in a Martian cave. Two robotic systems explore a Martian cave, transmit their maps to a surface station, which serves as a relay to an orbiter, and the orbiter transmits the data to operators on Earth. Three evaluations are conducted to quantify the system performance through this concept of operations: first, the perceptual fidelity and memory usage of the map is compared to state-of-the-art approaches in a representative cave environment (Section 6.2.1); second, a hardware experiment is demonstrated with two rapidly exploring aerial systems and the communication requirement for each mapping approach is compared (Section 6.2.2); and third, a simulation study is conducted to study the effects of the bandwidth constraints on exploration performance (Section 6.2.3).

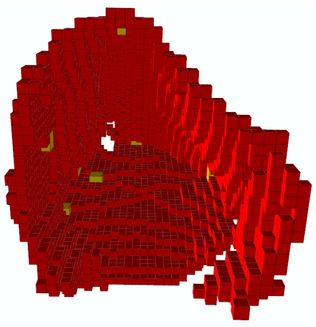
To correctly analyze the performance of the simulation study, the bottleneck in data transmission rate is identified and bounds on the rates are determined for the concept of operations presented in Section 1.1.1.Throughout this section the shorthand OG is used to refer to the occupancy grid mapping approach [41] while OM refers to OctoMap [40].

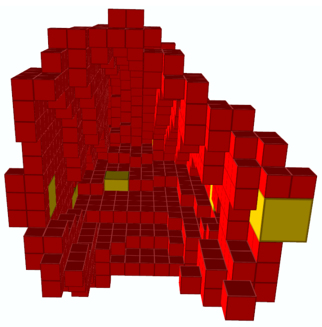

(a) RGB Image


(b) Point Cloud


(c) Resampled GMM


(d) OM (0.025 m)


(e) OM (0.05 m)


(f) OM (0.1 m)

	0.025 m	0.05 m	0.1 m
(bytes)	(bytes)	(bytes)	(bytes)
GMM	**Occupancy Grid (OG)**		
4028	1.3×10^6	1.8×10^5	2.7×10^4
GMM	**OctoMap (OM)**		
4028	2.2×10^5	5.8×10^4	1.4×10^4

(g) Memory

Figure 6.3: Fidelity and memory usage evaluation of several mapping approaches. (a) and (b) illustrate data from a representative environment the robot may encounter in the cave. A potential passage is circled in cyan. (g) highlights significant reduction in memory usage required by the GMM approach as compared to the OG and OM approaches. (c) Resampled points from the GMM are shown in red. (d)–(f) illustrate the OctoMap representation with leaf sizes varying from 0.025 m to 0.1 m. Leaf voxels are shown in red and larger voxels in yellow.

6.2.1 Perceptual Detail Evaluation

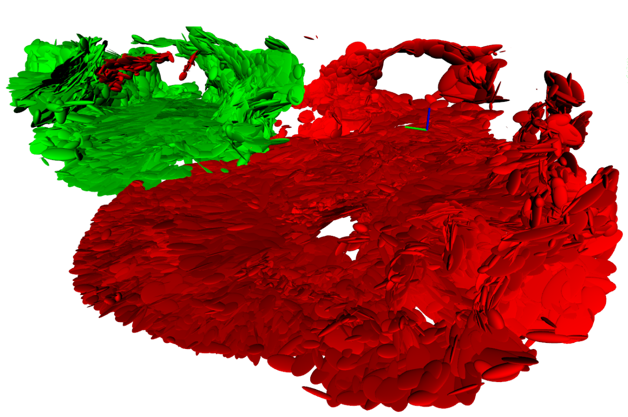
The first evaluation compares the perceptual fidelity of different environment representations in the context of memory usage. An RGB image and point cloud of a crevice in the cave are shown in Figs. 6.3a and 6.3b respectively. It is not clear from the image and depth information if the passage continues or there is a lack of data due to insufficient accuracy in the sensor observation. In either case, additional views are required to determine the exact nature of the passage. Figure 6.3g demonstrates that as the resolution of the OG and OM approaches increases, the memory demands also substantially increase. By comparison, the GMM approach requires substantially less memory. When using the GMM approach, the resulting resampled point cloud is shown in Fig. 6.3c, where a hole in the data is visible. This approach is compared to OM with varying leaf sizes in Figs. 6.3d to 6.3f.

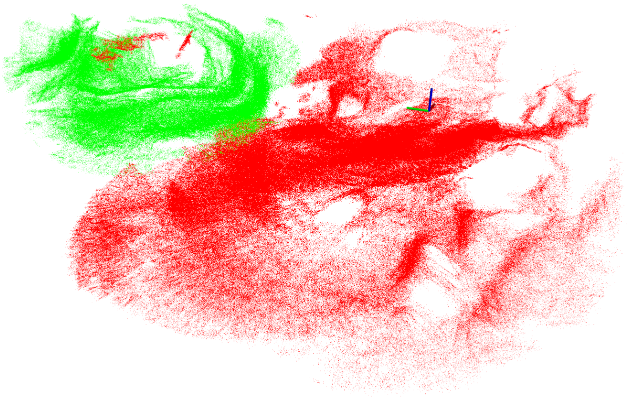
To obtain these results, a GMM was learned consisting of 100 components. Each component requires 10 floating point numbers which includes six floating point numbers to represent the symmetric covariance, three floating point numbers for the mean, and one floating point number to represent the mixing weight. Additional memory was used to represent the pose via six floating point numbers (three each for translation and rotation) where each floating point number is assumed to be four bytes. A 32-bit unsigned integer (four bytes) is also used to represent the support size of the GMM. In the OG case, one floating point number is used to store the logodds value and one unsigned integer (four bytes) is used to represent the index for each voxel in the change set. The total change set of N voxels is transmitted along with meta-data to reconstruct the grid. The meta-data consists of three unsigned integers to represent the dimensions of the grid in width, height, and length as well as three floating point numbers to represent the origin for a total of 24 bytes. The total data required to represent the sensor observation with an OG is $8N + 24$ bytes. For OM, the full probabilistic model is serialized and stored to disk. The size of the file is reported in the table. The motivation for retaining the logodds values in the OG and OM representations is to enable information-theoretic planning. The advantage of the GMM approach is that the probability of occupancy can be reconstructed at an arbitrary voxel resolution [36, 47], which significantly reduces the memory requirements as compared to the OG and OM approaches. The OG and OM approaches must retain the probability of occupancy to enable information-theoretic exploration [33, 34].

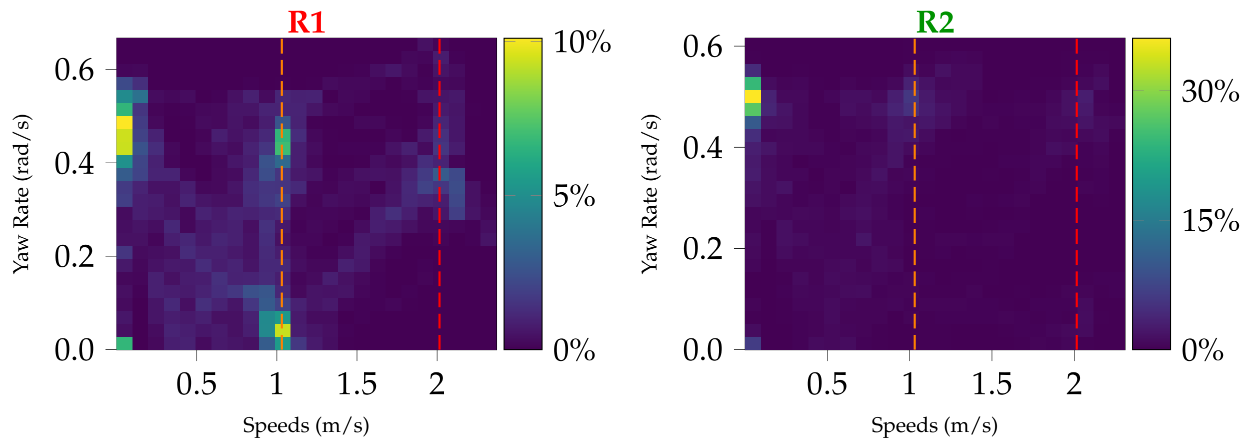
6.2.2 Hardware Experiments

The second evaluation consists of hardware experiments for two aerial systems exploring the cave. The experiment demonstrates (1) each robot generates informative plans with

(a) Robots (circled) deployed in a cave. Communication router shown via dotted line.


(b) Combined GMM map


(c) Resampled points from the GMM map



(d) Speed bounds shown by dashed lines.

Figure 6.4: Rapid and communication efficient exploration of a cave with a team of two aerial robots. (a) illustrates the environment with the two robots (**R1** and **R2**) and the WiFi router used for communication. (b) illustrates the final GMM maps generated on the base-station. (d) shows the percentage density plots for linear speeds and yaw rates as measured by the visual-inertial navigation system during flight. A video of the flight can be accessed here: `https://youtu.be/osko8EKKZUM`.

linear speeds up to 2.37 m/s and yaw rates up to 0.6 rad/s while maintaining safety and (2) the communication required to transmit the map from robots to a base station is substantially less as compared to the OG and OM approaches. For the purposes of this experiment, the robots are deployed in disjoint bounding boxes and the coordination between robots is not studied. What follows is a description of the experimental setup (including the implementation details) and results.

Table 6.1: This figure highlights that the GMM approach requires significantly less memory to represent the combined map as compared to state-of-the-art approaches. In the context of transmitting this data using a channel with capacity 0.25 Mbit/s, it would take significantly less time for the GMM approach as compared to the other approaches.

Completion %:	45%		65%		85%	
Case	Map Size (Mbit)	Time (hours)	Map Size (Mbit)	Time (hours)	Map Size (Mbit)	Time (hours)
GMM	$\mathbf{0.7 \times 10^{1}}$	$\mathbf{8.0 \times 10^{-3}}$	$\mathbf{1.4 \times 10^{1}}$	$\mathbf{1.6 \times 10^{-2}}$	$\mathbf{1.9 \times 10^{1}}$	$\mathbf{2.2 \times 10^{-2}}$
OG (0.1 m)	9.2×10^{1}	1.0×10^{-1}	1.57×10^{2}	1.7×10^{-1}	2.0×10^{2}	2.3×10^{-1}
OG (0.05 m)	5.4×10^{2}	6.0×10^{-1}	9.1×10^{2}	0.1×10^{1}	1.2×10^{3}	0.1×10^{1}
OG (0.025 m)	3.9×10^{3}	0.4×10^{1}	6.7×10^{3}	0.7×10^{1}	8.9×10^{3}	0.9×10^{1}
OM (0.1 m)	2.4×10^{2}	2.6×10^{-1}	3.9×10^{2}	4.4×10^{-1}	5.2×10^{2}	5.8×10^{-1}
OM (0.05 m)	1.6×10^{3}	0.2×10^{1}	2.6×10^{3}	0.3×10^{1}	3.4×10^{3}	0.4×10^{1}
OM (0.025 m)	9.8×10^{3}	1.1×10^{1}	1.6×10^{4}	1.8×10^{1}	2.1×10^{4}	2.4×10^{1}

Each robot in the multi-robot system employs the navigation and control technique outlined in prior work [13]. The robots communicate with other computers on the network via WiFi and use the User Datagram Protocol (UDP) to transfer packets over the network. Before the start of each experiment, the SE(3) transform between the takeoff positions of the robots is measured manually using the navigation approach. The relative initial transform is used by the distributed mapping subsystem to align the GMM map fragments in the frames of other robots to the current robot's local frame.

The maximum speed[1] of the robots in the xy-plane is 2.0 m/s, the maximum speed towards unknown space is 1.0 m/s, the maximum z-direction speed is 0.25 m/s, and the maximum yaw rate is constrained to 0.5 rad/s. One of the metrics used to assess the planning performance is quantifying the maximum speed and yaw rate achieved by the robot while ensuring collision free operation. Both linear and yawing motions are exploratory actions for an aerial robot equipped with a limited field of view depth sensor [13, 15]. The data transmitted from the robots to the base station is used to quantify the success of the mapping approach. The GMM results of Fig. 6.4 are generated in flight during an actual trial in the cave. To enable a fair comparison, the depth images collected from the GMM exploration trial in the cave are post-processed using the OG and OM approaches. This ensures that variation in the other subsystems does not unduly affect the results. An analysis to quantify the memory required for each approach similar

[1]The speed limits and the operational volumes were chosen based on the cave passage dimensions. The authors worked with cave management to select a test site that contained neither actively growing speleothems or bats. Possible effects of imperfect trajectory tracking and state estimation were also taken into account.

to Section 6.2.1 is presented. The OG and OM results are generated by updating the map using the depth information for the current image and publishing the change set. For the OM approach, the change set is serialized to file as the full probabilistic model to enable the base station and other robot to exactly recreate the map for information-theoretic exploration.

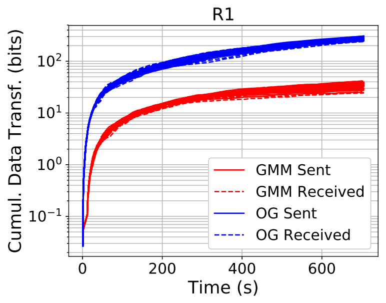
The two deployed robots are denoted by **R1** and **R2** in Fig. 6.4. The robots achieve high exploration rates by selecting actions that enable safe operation at linear speeds up to 2.37 m/s and yaw rates up to 0.6 rad/s, which are of the same order as state-of-the-art fast exploration works[2] [15, 16, 17]. Moreover, note that since **R1** operates in a relatively open space compared to **R2**, a larger percentage of high speed actions are selected (Fig. 6.4d). In contrast, the planner selects the yawing motion and slow linear actions towards frontiers more often for **R2** to allow for safe operation in a constrained space (Fig. 6.4d). Both of these behaviors in the multi-robot system arise automatically due to the choice of the action representation for single-robot planning in [15]. These behaviors show that the same action representation can be used on every robot in the team without any change in parameters and still allow for intelligent speed adaptation for rapid and safe exploration.

The combined map from **R1** and **R2** requires significantly less time to transmit under the bandwidth constraint when measuring at various points during exploration (Table 6.1). An implication of this in the context of the concept of operations is that at 100% exploration completion it will take about 104.40 seconds to transmit the GMM map, 12.30 hours to transmit the 0.025 m resolution OG map, and 1.25 days to transmit the 0.025 m resolution OM map to Earth. It is important to note why the OM approach requires more memory than the OG approach for this result while it required less memory than the OG approach in Fig. 6.3g. The change set must be encoded as an OctoMap before serializing to file. The approach presented by Hornung et al. [40] requires that the spatial relationships between nodes be implicitly stored in the encoding. This means that the serialized stream does not contain any 3D coordinates and additional data must be stored to preserve the structure of the octree. This is in contrast to the OG approach that stores a logodds value and index from which 3D coordinates can be recovered. Therefore, for small change sets, the OM approach has much higher overhead than the OG approach.

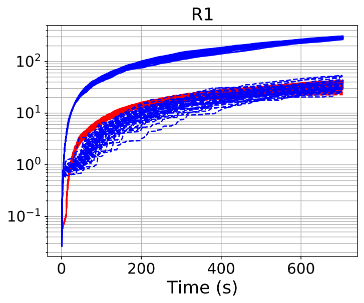
6.2.3 Effects of Constrained Communication

For this study the assumption on the robots operating in disjoint spaces is relaxed and a priority-based inter-robot collision checker is implemented for shared space operation. The

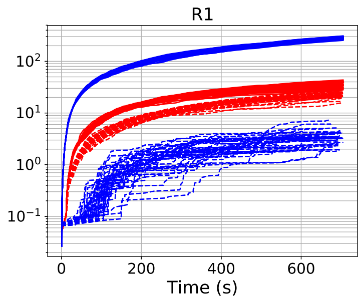
[2]The attained speeds exceed the limits slightly due to imperfect trajectory tracking and state estimation.



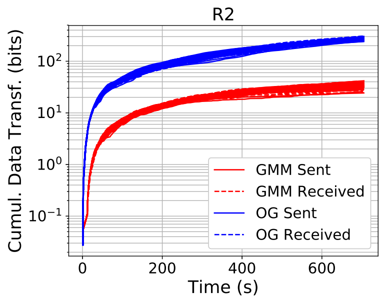
(a) R1 - No limit



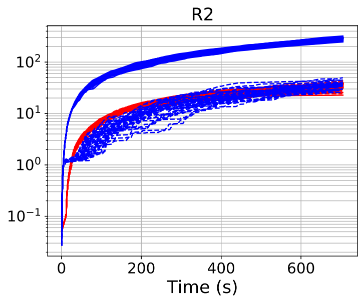
(b) R1 - 0.25 Mbit/s



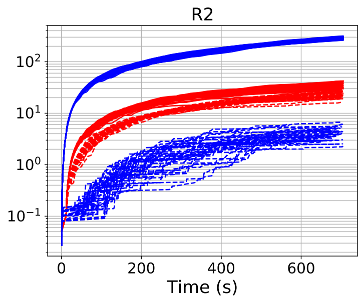
(c) R1 - 0.1 Mbit/s



(d) R2 - No limit



(e) R2 - 0.25 Mbit/s



(f) R2 - 0.1 Mbit/s

Completion %:	45%			65%			85%		
Comm. Limit	GMM (s)	OG (s)	Δ (%)	GMM (s)	OG (s)	Δ (%)	GMM (s)	OG (s)	Δ (%)
No limit	80.19	81.51	1.62	130.73	131.98	0.95	225.8	237.51	4.93
0.25 Mbit/s	79.91	92.38	**13.5**	129.1	160.15	**19.39**	214.86	282.11	**23.84**
0.1 Mbit/s	86.51	93.43	7.41	142.95	165.88	13.82	247.83	270.15	8.27

(g) Exploration completion times

Figure 6.5: Variation of exploration performance with inter-robot communication limits. (a) to (f) plot the cumulative map data sent and received for the GMM and OG approaches under different data rate constraints for the two robots. The received data is impacted significantly for the OG approach at 0.25 Mbit/s while both approaches are affected at 0.1 Mbit/s. Note that in all experiments the planning and coordination methodology is kept the same for a fair comparison. (g) compares the time to achieve a certain percentage of environment coverage. We observe that at the 0.25 Mbit/s constraint, the GMM approach improves the performance of the team by up to 23.84%.

simulation consists of a two-robot team that explores the cave environment. Two approaches are tested: GMM and OG. The OM approach is not compared for this experiment because to the best of our knowledge there is no existing open-source implementation of the Shannon mutual information used for planning by Zhang et al. [34]. Further, this enables us to retain the same planning subsystem for a fair comparison of the GMM and OG approaches. The communication rate is varied among 0.1 Mbit/s, 0.25 Mbit/s, and unconstrained. Each configuration is tested in 40 experiments with a 700 s duration. The duration of the exploration is chosen based on the top speed of the robots and the spatial dimensions of the environment. The exploration software is run on separate computers in a distributed fashion over a wired connection. The simulations are run on two desktop computers running Ubuntu 18.04 with Intel i7-6700K CPUs. One computer has 32 GB RAM and the other has 16 GB of RAM. For the wired connection, the data rate is limited via the network traffic control tool in Linux that uses the Token Bucket Filter (TBF) to maintain the specified rate value [75]. Figure 6.5 illustrates the results from the simulation study. As the communication bandwidth is reduced from no limit in Fig. 6.5a to 0.25 Mbit/s the OG approach begins to drop packets and the exploration performance of the multi-robot approach decreases as compared to the GMM approach (see Fig. 6.5g). At this rate, the GMM approach achieves 85% environment coverage in less than 80% of the time that it takes the OG approach. However, as the communication rate decreases further to 0.1 Mbit/s the GMM approach also suffers though it is able to outperform the OG approach.

6.3 Summary

This chapter leveraged the compactness of Gaussian mixture models for high-fidelity perceptual modeling to increase the rate of multi-robot exploration in reduced bandwidth scenarios such as autonomy in caves. The mapping approach enables retention of environment details while remaining amenable to low-bandwidth transmission. The advantage of this mapping strategy is that it enables a substantial increase in exploration rate of the multi-robot team as compared to state-of-the-art mapping techniques even as the communication bandwidth of the connection between robots decreases.

Chapter 7

Conclusion

This thesis develops a robotic exploration framework that allows for rapid and communication-efficient mapping of unknown environments with a team of aerial robots.

7.1 Summary of Contributions

Chapter 4: Rapid Motion Primitives-based Single-Robot Exploration A motion primitive-based, receding-horizon planning approach that maximizes information gain, accounts for platform dynamics, and ensures safe operation. Simulation experiments in a complex 3D environment demonstrate the utility of the motion primitive actions for rapid exploration and provide a comparison to a reduced motion primitive library that is appropriate for online planning. Experimental results on a hexarotor robot with the reduced library demonstrate rapid exploration at speeds above 2.25 m/s under a varying clutter in an outdoor environment which is comparable to and exceeding the existing state-of-the-art results [15].

Chapter 5: Communication-Efficient Single-Robot Exploration An information-theoretic exploration strategy to explore cave environments that compactly represents sensor observations as Gaussian mixture models and maintains a local occupancy grid map for a motion planner that greedily maximizes an information-theoretic objective function. The approach accommodates both limited field of view depth cameras and larger field of view LiDAR sensors and is extensively evaluated in long duration simulations on an embedded PC. The system is deployed in Laurel Caverns, a commercially owned and operated cave in southwestern Pennsylvania, USA, and a wild cave in West Virginia, USA [13].

Chapter 6: Rapid and Communication-Efficient Multi-Robot Exploration A multi-robot exploration framework that leverages the work from the previous two chapters to enable high-fidelity distributed mapping at high speeds while remaining amenable to low-bandwidth communication channels. The approach yields significant gains in exploration rate for multi-robot teams as compared to state-of-the-art approaches. The system is evaluated through simulation studies and hardware experiments in a wild cave in West Virginia [23].

7.2 Future Work

Future work will improve perceptual detail in the environment and develop hierarchical strategies that adapt the fidelity of the model based on the sensor data. Multi-modal mapping (for example, thermal, RGB, etc.) may also be beneficial in these scenarios. Finally, coordination strategies can be developed to enable robots to share communication-efficient policies and improve the rate of exploration.

Bibliography

[1] V Stamenkovic et al. The next frontier for planetary and human exploration. *Nature Astronomy*, 3(2):116–120, 2019. 1

[2] Timothy N Titus, J Judson Wynne, Michael J Malaska, Ali-akbar Agha-Mohammadi, Peter B Buhler, E Calvin Alexander, James W Ashley, Armando Azua-Bustos, Penelope J Boston, Debra L Buczkowski, et al. A roadmap for planetary caves science and exploration. *Nature Astronomy*, 5(6):524–525, 2021. 1, 1.3

[3] Friedrich Horz. Lava tubes-potential shelters for habitats. In *Lunar bases and space activities of the 21st century*, pages 405–412, 1985. 1

[4] Samuel B Kesner, Jean-Sébastien Plante, Penelope J Boston, Tibor Fabian, and Steven Dubowsky. Mobility and power feasibility of a microbot team system for extraterrestrial cave exploration. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 4893–4898. IEEE, 2007. 1

[5] Timothy Titus, J Judson Wynne, Penny Boston, Pablo de Leon, Cansu Demirel-Floyd, Heather Jones, Francesco Sauro, Kyle Uckert, Ali Aghamohammadi, Calvin Alexander, et al. Science and technology requirements to explore caves in our solar system. *Bulletin of the American Astronomical Society*, 53(4):167, 2021. 1

[6] Glen E Cushing. Candidate cave entrances on mars. *Journal of Cave and Karst Studies*, 74(1):33–47, 2012. 1

[7] Mark Maimone, Yang Cheng, and Larry Matthies. Two years of visual odometry on the mars exploration rovers. *Journal of Field Robotics*, 24(3):169–186, 2007. 1

[8] David Miranda. 2020 nasa technology taxonomy, 2020. 1

[9] B. M. Jakosky et al. *Mars, the Nearest Habitable World – A Comprehensive Program for Future Mars Exploration*. Mars Architecture Strategy Working Group (MASWG), 2020. 1

[10] Charity M Phillips-Lander, Ali Agha-Mohammadi, JJ Wynne, Timothy N Titus, Nancy Chanover, Cansu Demirel-Floyd, Kyle Uckert, Kaj Williams, Danielle Wyrick, Jen Blank, et al. Mars astrobiological cave and internal habitability explorer (macie): A new frontiers mission concept. *arXiv preprint arXiv:2105.05281*, 2021. 1, 1.1, 2.1

[11] Red Whittaker et al. Exploration of planetary skylights and tunnels, 2014. 1, 1, 1.1.1

[12] Jorge Cortés and Magnus Egerstedt. Coordinated control of multi-robot systems: A survey. *SICE Journal of Control, Measurement, and System Integration*, 10(6):495–503, 2017. 1

[13] Wennie Tabib, Kshitij Goel, John Yao, Curtis Boirum, and Nathan Michael. Autonomous cave surveying with an aerial robot. *arXiv preprint arXiv:2003.13883*, 2020. 1, 1.2, 2.3, 3, 3.2, 6.1.1, 6.1.1, 6.2.2, 7.1

[14] Titus Cieslewski, Elia Kaufmann, and Davide Scaramuzza. Rapid exploration with multi-rotors: A frontier selection method for high speed flight. In *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.*, Vancouver, Canada, September 2017. 1, 2.2, 4.3

[15] Kshitij Goel, Micah Corah, Curtis Boirum, and Nathan Michael. Fast exploration using multirotors: Analysis, planning, and experimentation. In *Conf. on Field and Service Robot.*, Tokyo, Japan, Aug. 2019. 1, 1.2, 5.1.3, 5.3, 5.1.3, 5.1.5, 6.1.2, 6.2.2, 7.1

[16] Anna Dai, Sotiris Papatheodorou, Nils Funk, Dimos Tzoumanikas, and Stefan Leutenegger. Fast frontier-based information-driven autonomous exploration with an mav. *arXiv preprint arXiv:2002.04440*, 2020. 1, 6.2.2

[17] Mihir Dharmadhikari, Tung Dang, Lukas Solanka, Johannes Loje, Huan Nguyen, Nikhil Khedekar, and Kostas Alexis. Motion primitives-based path planning for fast and agile exploration using aerial robots. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 179–185. IEEE, 2020. 1, 6.2.2

[18] Kyle Cesare, Ryan Skeele, Soo-Hyun Yoo, Yawei Zhang, and Geoffrey Hollinger. Multi-uav exploration with limited communication and battery. In *2015 IEEE international conference on robotics and automation (ICRA)*, pages 2230–2235. IEEE, 2015. 1

[19] M. Corah, C. OMeadhra, K. Goel, and N. Michael. Communication-efficient planning and mapping for multi-robot exploration in large environments. *IEEE Robotics and Automation Letters*, 4(2):1715–1721, April 2019. ISSN 2377-3766. doi: 10.1109/LRA.2019.2897368. 1, 2.2, 2.3, 4.1.3, 4.1.3, 4.2.1, 5.1.4, 5.1.5, 5.1.5

[20] Larry Matthies. Niac phase 1 final study report on titan aerial daughtercraft, 2017. 1.1.1

[21] Steven Kisseleff, Ian F Akyildiz, and Wolfgang H Gerstacker. Survey on advances in magnetic induction-based wireless underground sensor networks. *IEEE Internet of Things Journal*, 5(6):4843–4856, 2018. 1.1.1

[22] NASA. Mars science laboratory data rates/returns, 2019. URL `https://marsmobile.jpl.nasa.gov/msl/mission/communicationwithearth/data/`. 1.1.1

[23] Kshitij Goel, Wennie Tabib, and Nathan Michael. Rapid and high-fidelity subsurface exploration with multiple aerial robots. In Bruno Siciliano, Cecilia Laschi, and Oussama Khatib, editors, *Experimental Robotics*, pages 436–448, Cham, 2021. Springer International Publishing. ISBN 978-3-030-71151-1. 1.2, 7.1

[24] DARPA. Darpa subterranean challenge, 2020. URL `https://www.subtchallenge.com/`. 2.1

[25] Ali Agha, Kyohei Otsu, Benjamin Morrell, David D Fan, Rohan Thakker, Angel Santamaria-Navarro, Sung-Kyun Kim, Amanda Bouman, Xianmei Lei, Jeffrey Edlund, et al. Nebula: Quest for robotic autonomy in challenging environments; team costar at the darpa subterranean challenge. *arXiv preprint arXiv:2103.11470*, 2021. 2.1, 2.2, 2.1

[26] Nicolas Hudson, Fletcher Talbot, Mark Cox, Jason Williams, Thomas Hines, Alex

Pitt, Brett Wood, Dennis Frousheger, Katrina Lo Surdo, Thomas Molnar, et al. Heterogeneous ground and air platforms, homogeneous sensing: Team csiro data61's approach to the darpa subterranean challenge. *arXiv preprint arXiv:2104.09053*, 2021. 2.1, 2.2, 2.1

[27] Pavel Petracek, Vit Kratky, Matej Petrlik, Tomas Baca, Radim Kratochvil, and Martin Saska. Large-scale exploration of cave environments by unmanned aerial vehicles. *IEEE Robotics and Automation Letters*, 2021. 2.1, 2.1

[28] CoSTAR Team. Scout tunnel experiment, 2018. URL `https://www.youtube.com/watch?v=MdYXuImXik`. 2.1

[29] Kamak Ebadi, Yun Chang, Matteo Palieri, Alex Stephens, Alex Hatteland, Eric Heiden, Abhishek Thakur, Benjamin Morrell, Sally Wood, Luca Carlone, et al. Lamp: Large-scale autonomous mapping and positioning for exploration of perceptually-degraded subterranean environments. *arXiv preprint arXiv:2003.01744*, 2020. 2.1, 2.3

[30] Tung Dang, Marco Tranzatto, Shehryar Khattak, Frank Mascarich, Kostas Alexis, and Marco Hutter. Graph-based subterranean exploration path planning using aerial and legged robots. *Journal of Field Robotics*, 2020. 2.1, 2.3

[31] Tomáš Rouček, Martin Pecka, Petr Čížek, Tomáš Petříček, Jan Bayer, Vojtěch Šalanskỳ, Daniel Heřt, Matěj Petrlík, Tomáš Báča, Vojěch Spurnỳ, et al. Darpa subterranean challenge: Multi-robotic exploration of underground environments. In *International Conference on Modelling and Simulation for Autonomous Systems*, pages 274–290. Springer, 2019. 2.1, 2.3

[32] Chao Cao, Hongbiao Zhu, Howie Choset, and Ji Zhang. Tare: A hierarchical framework for efficiently exploring complex 3d environments. In *Proceedings of Robotics: Science and Systems (RSS '21)*, July 2021. 2.2

[33] B. Charrow, S. Liu, V. Kumar, and N. Michael. Information-theoretic mapping using Cauchy-Schwarz quadratic mutual information. In *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, Seattle, WA, May 2015. 2.2, 4.1.3, 4.1.3, 6.1.2, 6.2.1

[34] Zhengdong Zhang, Theia Henderson, Sertac Karaman, and Vivienne Sze. Fsmi: Fast computation of shannon mutual information for information-theoretic mapping. *The International Journal of Robotics Research*, 39(9):1155–1177, 2020. 2.2, 5.2.3, 6.2.1, 6.2.3

[35] Benjamin Charrow, Gregory Kahn, Sachini Patil, Sikang Liu, Ken Goldberg, Pieter Abbeel, Nathan Michael, and Vijay Kumar. Information-theoretic planning with trajectory optimization for dense 3D mapping. In *Proc. of Robot.: Sci. and Syst.*, Rome, Italy, July 2015. 2.2

[36] Wennie Tabib, Kshitij Goel, John Yao, Mosam Dabhi, Curtis Boirum, and Nathan Michael. Real-time information-theoretic exploration with gaussian mixture model maps. In *Proc. of Robot.: Sci. and Syst.*, FreiburgimBreisgau, Germany, June 2019. doi: 10.15607/RSS.2019.XV.061. 2.2, 3, 3.2, 3.2.1, 3.2.3, 5, 5.1.2, 5.1.3, 5.3, 5.1.3, 6.1.1, 6.2.1

[37] B. Yamauchi. A frontier-based approach for autonomous exploration. In *Proc. of the Intl. Sym. on Comput. Intell. in Robot. and Autom.*, Monterey, CA, July 1997. 2.2, 3.1, 5.1.4

[38] Sikang Liu, Michael Watterson, Sarah Tang, and Vijay Kumar. High speed navigation

for quadrotors with limited onboard sensing. In *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, Stockholm, Sweden, May 2016. 2.2

[39] Alex Spitzer, Xuning Yang, John Yao, Aditya Dhawale, Kshitij Goel, Mosam Dabhi, Matt Collins, Curt Boirum, and Nathan Michael. Fast and agile vision-based flight with teleoperation and collision avoidance on a multirotor. In *Proc. of the Intl. Sym. on Exp. Robot.*, Buenos Aires, Argentina, 2018. Springer. to be published. 2.2, 4.1.2, 4.2.1

[40] Armin Hornung, Kai M Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. Octomap: An efficient probabilistic 3d mapping framework based on octrees. *Autonomous Robots*, 34(3):189–206, 2013. 2.3, 5.2.3, 6.2, 6.2.2

[41] Alberto Elfes. Using occupancy grids for mobile robot perception and navigation. *IEEE Computer Society*, 22(6):46–57, 1989. 2.3, 6.2

[42] T M Cover and J A Thomas. *Elements of Information Theory*. John Wiley & Sons, New York, NY, 2012. 3.1, 5.2.1

[43] Reshad Hosseini and Suvrit Sra. An alternative to em for gaussian mixture models: Batch and stochastic riemannian optimization. *Mathematical Programming*, pages 1–37, 2017. 3.2

[44] C. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag New York, New York, 2 edition, 2007. 3.2, 3.2

[45] Jeff A Bilmes et al. A gentle tutorial of the em algorithm and its application to parameter estimation for gaussian mixture and hidden markov models. *Intl. Comput. Sci. Inst.*, 4(510):126, 1998. 3.2

[46] Ben Eckart, Kihwan Kim, Alejandro Troccoli, Alonzo Kelly, and Jan Kautz. Mlmd: Maximum likelihood mixture decoupling for fast and accurate point cloud registration. In *Intl. Conf. on 3D Vision*, pages 241–249. IEEE, 2015. 3.2

[47] C. OMeadhra, W. Tabib, and N. Michael. Variable resolution occupancy mapping using gaussian mixture models. *IEEE Robot. Autom. Letters*, 4(2):2015–2022, April 2019. ISSN 2377-3774. doi: 10.1109/LRA.2018.2889348. 3.2, 6.2.1

[48] W. Tabib and N. Michael. Simultaneous localization and mapping of subterranean voids with gaussian mixture models. In *Conf. on Field and Service Robot.*, Tokyo, Japan, Aug. 2019. 3.2.1, 3.2.1

[49] Shobhit Srivastava. Efficient, multi-fidelity perceptual representations via hierarchical gaussian mixture models. Master's thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh PA, August 2017. 3.2.2

[50] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*. MIT press, 2005. 3.2.3

[51] John Amanatides, Andrew Woo, et al. A fast voxel traversal algorithm for ray tracing. In *Eurographics*, volume 87, pages 3–10, 1987. 3.2.3

[52] Jose Luis Blanco and Pranjal Kumar Rai. nanoflann: a C++ header-only fork of FLANN, a library for nearest neighbor (NN) with kd-trees. `https://github.com/jlblancoc/nanoflann`, 2014. 3.2.3

[53] Amy Williams, Steve Barrus, R Keith Morley, and Peter Shirley. An efficient and robust ray-box intersection algorithm. In *ACM SIGGRAPH 2005 Courses*, page 9. ACM, 2005. 3.2.3

[54] Nathan Michael, Daniel Mellinger, Quentin Lindsey, and Vijay Kumar. The grasp multiple micro-uav testbed. *IEEE Robot. Autom. Mag.*, 17(3):56–65, 2010. 4.1.1

[55] Robert Mahony, Vijay Kumar, and Peter Corke. Multirotor aerial vehicles. *IEEE Robot. Autom. Mag.*, 20(32), 2012. 4.1.1

[56] Lucas Janson, Tommy Hu, and Marco Pavone. Safe motion planning in unknown environments: Optimality benchmarks and tractable policies. In *Proc. of Robot.: Sci. and Syst.*, Pittsburgh, PA, July 2018. 4.1.1, 5.1.5

[57] Daniel Mellinger and Vijay Kumar. Minimum snap trajectory generation and control for quadrotors. In *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, Shanghai, China, May 2011. 4.1.2

[58] Xuning Yang, Ayush Agrawal, Koushil Sreenath, and Nathan Michael. Online adaptive teleoperation via motion primitives for mobile robots. *Autonomous Robots*, April 2018. 4.1.2

[59] Xuning Yang, Koushil Sreenath, and Nathan Michael. A framework for efficient teleoperation via online adaptation. In *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, pages 5948–5953. IEEE, 2017. 4.1.2

[60] Micah Corah and Nathan Michael. Distributed matroid-constrained submodular maximization for multi-robot exploration: theory and practice. *Auton. Robots*, 2018. 4.1.3, 5.1.5

[61] Mikko Lauri and Risto Ritala. Planning for robotic exploration based on forward simulation. *Robot. Auton. Syst.*, 83, 2016. 4.1.3

[62] Guillaume Chaslot. *Monte-Carlo Tree Search*. PhD thesis, Universiteit Maastricht, 2010. 4.1.3, 6.1.2

[63] Cameron Browne, Edward Powley, Daniel Whitehouse, Simon Lucas, Peter I. Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of Monte Carlo tree search methods. *IEEE Trans. on Comput. Intell. and AI in Games*, 4(1):1–43, 2012. 4.1.3

[64] Tong Qin, Peiliang Li, and Shaojie Shen. Vins-mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Transactions on Robotics*, 34(4):1004–1020, 2018. 4.2.1

[65] E. W. Weisstein. *Tetrahedron*. URL `http://mathworld.wolfram.com/Tetrahedron.html`. 5.1.1

[66] Kshitij Goel, Micah Corah, and Nathan Michael. Fast exploration using multirotors: Analysis, planning, and experimentation. Technical Report CMU-RI-TR-19-03, The Robotics Institute, Carnegie Mellon University, 2019. 5.1.3

[67] Wennie Tabib, Micah Corah, Nathan Michael, and Red Whittaker. Computationally efficient information-theoretic exploration of pits and caves. In *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.*, Daejeon, Korea, October 2016. 5.1.5

[68] John W. Yao. *Resource-Constrained State Estimation with Multi-Modal Sensing*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, April 2020. 5.2.2

[69] L. Contin and S. Battista. Performance evaluation of video coding schemes working at very low bit rates. In *Proceedings of ICASSP '94. IEEE International Conference on Acoustics, Speech and Signal Processing*, volume v, pages V/409–V/412 vol.5, April 1994. doi: 10.1109/ICASSP.1994.389401. 5.11

[70] M. Masry and S. S. Hemami. An analysis of subjective quality in low bit rate video. In *Proceedings 2001 International Conference on Image Processing (Cat. No.01CH37205)*, volume 1, pages 465–468 vol.1, Oct 2001. doi: 10.1109/ICIP.2001.959054. 5.11

[71] Kevin Joseph Patrick. *Pennsylvania Caves and other Rocky Roadside Wonders*. Stackpole Books, 2004. 2

[72] R. S. Bradley. *Paleoclimatology: Reconstructing Climates of the Quaternary*. Elsevier Science & Technology, Saint Louis, 2014. Online: accessed 3 November 2020. 3, 5

[73] Winifred F Frick, Jacob F Pollock, Alan C Hicks, Kate E Langwig, D Scott Reynolds, Gregory G Turner, Calvin M Butchkoski, and Thomas H Kunz. An emerging disease causes regional population collapse of a common north american bat species. *Science*, 329(5992):679–682, 2010. 4

[74] Chengtao Cai, Chunsheng Yang, Qidan Zhu, and Yanhua Liang. Collision avoidance in multi-robot systems. In *2007 International Conference on Mechatronics and Automation*, pages 2795–2800. IEEE, 2007. 6.1.2

[75] Bert Hubert, Thomas Graf, Greg Maxwell, Remco van Mook, Martijn van Oosterhout, P Schroeder, Jasper Spaans, and Pedro Larroy. Linux advanced routing & traffic control. In *Ottawa Linux Symposium*, volume 213. sn, 2002. 6.2.3