Coordinated Aerial Exploration of Subterranean Voids

Rohit Garg

CMU-RI-TR-21-44

August 2021

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

Thesis Committee:
Dr. Sebastian Scherer, *Chair*
Dr. Maxim Likhachev
Dr. Rogerio Bonatti, Microsoft Research

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

To all the people and robots that made this work possible.

Abstract

Subterranean exploration has been thrust into the spotlight by the recent DARPA Subterranean Challenge. Teams are tasked with developing a multi-agent system that can rapidly navigate through unknown underground environments while sending back maps and other semantic information. A system with such capabilities has a wide variety of applications from surveying and mapping mines to search and rescue operations. Aerial platforms in particular have the ability to fly fast and discover large parts of an environment despite their lower operational endurance. They can also change altitude at will to explore narrow nooks in spaces that are otherwise difficult to access. The work done in this thesis aims to leverage these abilities to build a resilient autonomy pipeline that enables multiple drones to coordinate an underground exploration mission with the click of a button. Towards that end, we propose an integrated exploration and coordination pipeline that is designed to generalize across a variety of subterranean environments. An ensemble of planners has been built in to tackle a large set of operating conditions ensuring the robot is never still for too long. Coordination map sharing enables the aerial platforms to minimize the overlap in explored volumes between them and other robots on the team. An extensive set of field experiments show that the approach works well in subterranean spaces such as caves, mines and urban infrastructure. We conclude with a discussion on further work planned for improving the system in the future.

Contents

When this dissertation is viewed as a PDF, the page header is a link to this Table of Contents.

List of Figures

Chapter 1

Introduction

1.1 Motivation

Knowledge about subterranean spaces has significant value in a variety of domains. From surveying and mapping of abandoned mines, to search and rescue applications in places like caves and subway stations. Now that the human race is going back to the Moon, there is new found cause to explore lunar lava tubes through the use of a rover controlled from a basestation several hundred thousand miles away.

In June 2018, 12 boys went to explore the Tham Luang cave in Thailand's Chiang Rai province accompanied by their football coach. They got trapped deep inside the cave underneath a mountain [8]. A massive search and rescue effort was mounted to rescue them from the cave. Amongst one of many challenges faced by the rescuers was an outdated and incomplete map of the cave. A French caving society had last conducted a survey of the cave in the 1980s. Many of the deepest recesses inside remained unmapped [1]. "It was reported that when the search began, estimates of distances between key points were inaccurate and the location of landmarks uncertain, clouding even the most basic assumptions" [1]. Figure 1.1 shows a hand drawn map produced by one of the rescuers that had to be used during the operation.

In retrospect, one can now see the utility in having systems that can autonomously map and identify objects of interest in such treacherous environments that pose so much danger to both the rescuer and the rescuee. High resolution 3D imagery would give an unprecedented level of situational awareness in such spaces that can only make the rescuers' job easier and less risky. The map shown in Figure 1.2 is a creation of the work done in this thesis. This was achieved using a fully autonomous drone that flew the trajectory highlighted in yellow. This map has enough detail and resolution for a rescue team to accurately assess the geometry of the environment and would be a valuable tool in mounting a search and rescue operation.

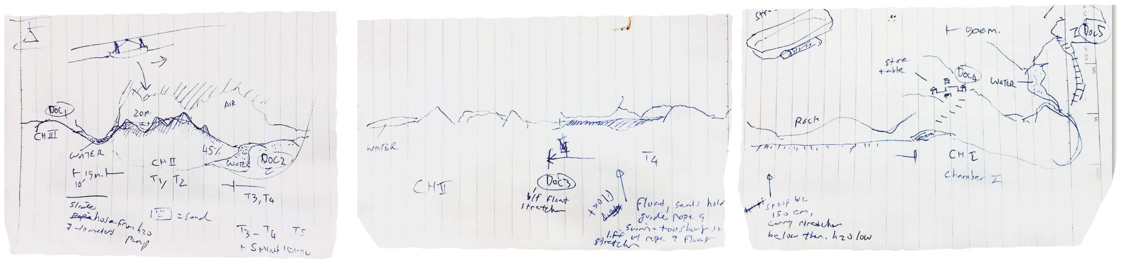


Figure 1.1: A hand drawn map of the Tham Luang cave used by the rescue team. Reproduced from [1].

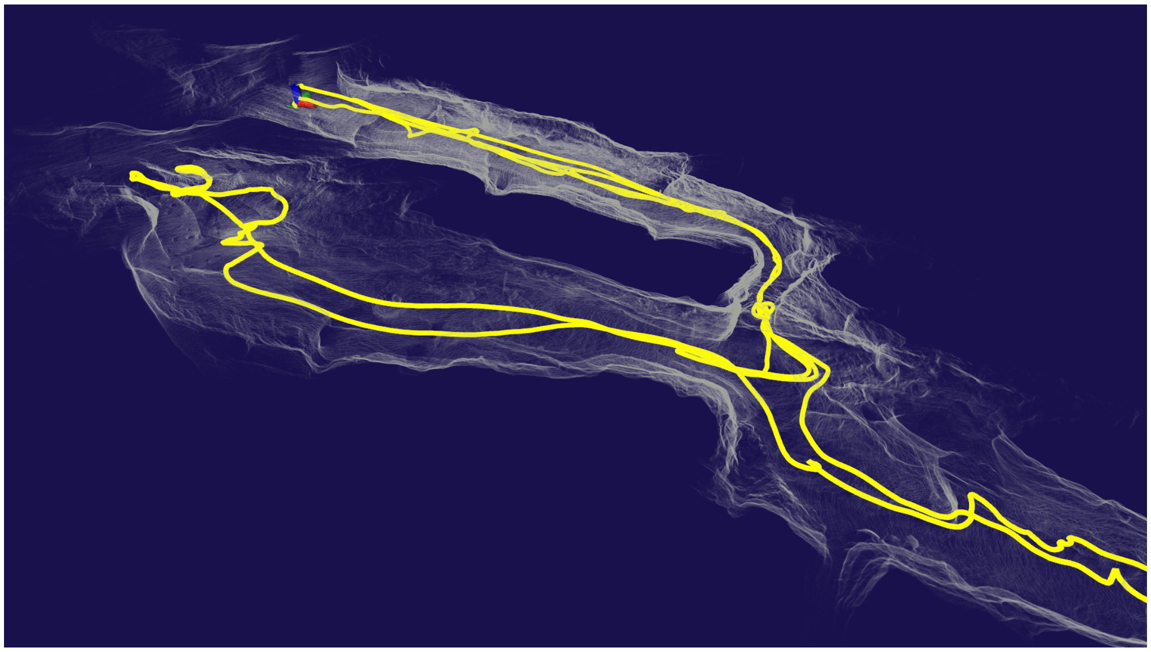


Figure 1.2: Map built from an autonomous exploration flight at Laurel Caverns, PA

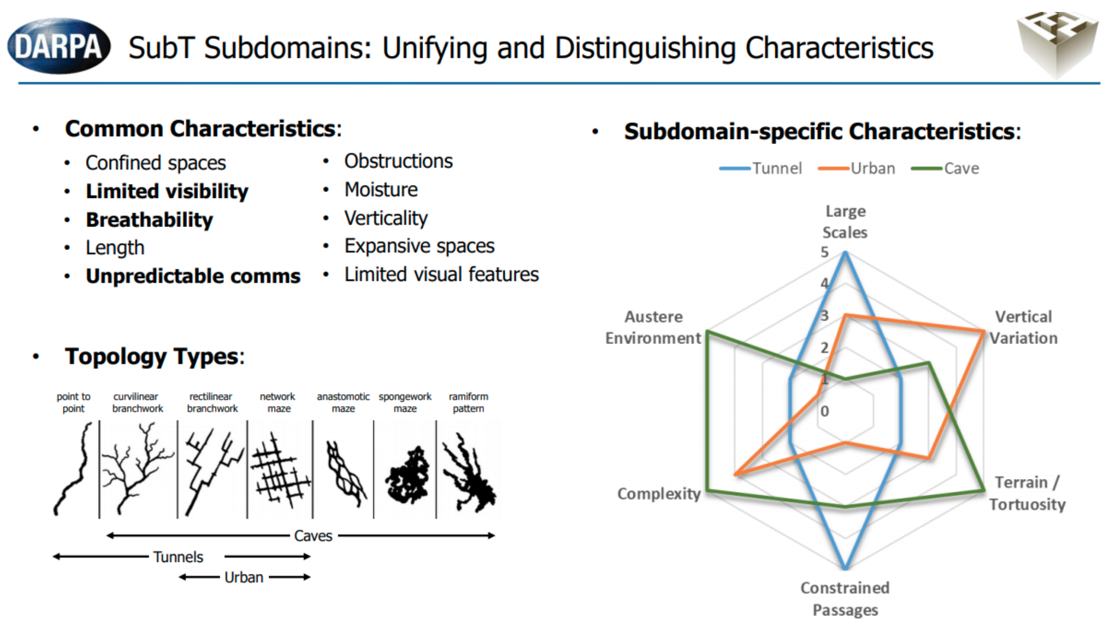
Another area where autonomous exploration and mapping can have an impact is in mapping abandoned mines. In the United States it is believed that there are several thousands of abandoned mines. There are so many in fact, that their number is unknown to even the Department of Surface Mines. In such a situation, with continued infrastructure and mining development taking place both above and below the earth's surface, it becomes imperative to have a *deeper* understanding of these historically unmapped voids [2][21].

DARPA Subterranean Challenge The DARPA Subterranean (SubT) Challenge was started in 2018 as a three year long competition inviting global teams to develop technologies that enable rapid mapping, navigation and search capabilities for mobile agents in complex underground environments [11]. The competition is split into two tracks - systems and virtual. Systems track teams are expected to develop and demonstrate physical systems that can compete in live competitions in environments representative of the three broad categories of subterranean voids. Teams are required to deploy a team of robots that can only be operated by a single base station operator. The circuit environment contains an array of objects of interest - artifacts such as fire extinguishers, cellphones, backpacks, drills amongst others. These artifacts need to be detected, correctly identified and triangulated within an error margin of 1.0m. The teams are scored on the number of correct detections over the course of a timed run. Figure 1.3 (from [11]) shows the different environments and their relative complexity in great detail. One of the key challenges here, which also happens to be what the work done in this thesis is trying to solve, is to build an autonomy pipeline that generalizes to all the three types of environments enabling rapid and efficient exploration through a multi-agent team.

Team Explorer CMU and Oregon State University have a joint entry to the systems track in the competition. Figure 2.4 shows the different systems that have been built as part of this competition. All of them are custom built platforms except the Spot quadruped robot that was acquired from Boston Dynamics. The work done in this thesis will be focusing on the aerial platforms and their performance during field deployments.

The problem this thesis is tackling lies at the intersection of robot perception and planning. What's the best way for a team of robots to explore an apriori unknown environment in a manner that is fast, reliable and generalizes well?

The exploration behavior of our aerial platforms relies on the use of a frontier-


SubT Subdomains: Unifying and Distinguishing Characteristics

- **Common Characteristics**:
  - Confined spaces
  - **Limited visibility**
  - **Breathability**
  - Length
  - **Unpredictable comms**
  - Obstructions
  - Moisture
  - Verticality
  - Expansive spaces
  - Limited visual features
- **Topology Types**:
- **Subdomain-specific Characteristics**:

Figure 1.3: Overview of environments in the Challenge. Credits - Timothy Chung, DARPA [11]

based exploration paradigm to determine what areas to plan to. This information is then fed into an ensemble of planners designed to handle a variety of operating conditions. Additionally, we implemented a system of sharing coarse observed maps and roadmaps between agents in the team that allows the UAVs to coordinate the entire exploration effort between themselves and the ground platforms. The overall objective is to plan and execute trajectories that maximize the chances of observing artifacts within the field of view of the object detection cameras. The exploration pipeline was developed to address challenges unique to aerial platforms in general. These include a limited sensor field of view, exploiting the ability to move vertically, computational constraints, and a relatively short operational endurance compared to ground based platforms.

1.2 Background Literature

Robot exploration approaches can broadly be divided into two categories. The first category belongs to the oldest and still popular approach based on computing frontiers. Frontier based exploration was first introduced in [31] which computed a 2D boundary between known free cells and unknown cells in an occupancy map to figure out what area to explore. Frontier cells that represent this boundary, are clustered, and a collision free path is planned to these cluster centroids. This way the robot continuously stretches this boundary between what is known and what isn't as it navigates its way to successive frontier clusters. In [23], randomly sampled viewpoints near 3D frontiers are evaluated against a series of criteria based on visibility and an information gain utility score. A collision free path is planned to the viewpoint in a greedy fashion, and the robot moves from one viewpoint to another until a termination criterion is met. [28] uses similar 3D frontiers to get candidate viewpoints based on local surface normals projected into known free space. This approach was shown to work for Autonomous Underwater Vehicles. This was further extended in [29], wherein a multisensor input was considered for the task of 3D underwater exploration. This work focuses on optical coverage while taking advantage of a longer range multibeam sonar sensor to inform the exploration. Leading to two kinds of viewpoints - range and camera viewpoints. This ensures coverage with both kinds of sensors. [33] introduces the concept of surface frontiers that enable exploration focused on covering surfaces as opposed to free space. This is something we have made use of in our initial implementations.

Frontier based exploration has also been extended to multiple robots. [32] extended their seminal work to multiple robots that share their local grid maps over the network. This way each robot would know what has been explored globally and could plan paths to as yet unexplored global frontiers. [5] follows a decision theoretic approach to coordinate exploration efforts in a heterogeneous robot team. This is done by computing the utility of unexplored areas and the cost to reach them. These two metrics are then traded off by considering how many robots are headed to the area in consideration. Leading to a maximization of the overall utility and minimization of overlapping explored areas between robots. In the same spirit [13] proposes an integrated multirobot mapping and exploration pipeline that uses a Bayesian decision theoretic strategy. This is done by exchanging sensor data and estimating relative positions between robots when they are within communications range. If position estimates are within an error threshold, then the robots form an exploration cluster and combine their maps to coordinate their actions. Otherwise, the robots focus on their own locally generated frontiers for exploration. [25] looks at the multi-agent exploration problem from a semantic perspective by taking into account the type of place that needs to be explored. The work makes use of a classifier to assign labels to different locations in the environment, arguing that this added knowledge enables the robot team to explore the environment more efficiently.

The second category of exploration algorithms are ones which optimize an information theoretic objective function such as Shannon's entropy or mutual information [4, 16, 18, 19]. Cauchy-Schwarz Quadratic Mutual Information has been shown to be a computationally efficient alternative to using Shannon Mutual Information as a measure of uncertainty in the map [7]. This work introduces a control policy based on maximizing CSQMI between the measurements received by the robot over a time horizon and the map. Other works that make use of CSQMI as an information theoretic measure to guide the robot to uncertain regions in the map include [6, 12, 14, 15, 26, 27].

We chose to opt for a frontier based approach rather than an information theoretic approach as frontier-based approaches are generally more computationally efficient while yielding similar exploration behavior. Our frontier definition extends the standard definition of exploration frontiers [31] to also include information beyond the boundary using observations from another sensor. Similar techniques have been proposed for other multi-sensor exploration planners [3, 29]. Additionally, we focus on frontiers of the surface rather than free space, since we believe it is more likely that artifacts are on or near surfaces.

1.3 Contributions

While there is plenty of existing research to highlight the multitude of approaches that can be taken to tackle a problem such as this. It is still an open question as to how well they generalize across different kinds of environments, and how scalable they are with respect to the number of robots. The work done in this thesis tries to tackle that aspect. We have developed an integrated exploration and coordination pipeline that has been shown to work across 3 distinctly different environments (tunnels, caves and urban spaces) with large variations in topology and local terrain geometry. An extensive set of field experiments have been conducted without changing any algorithmmic parameters between the environments, highlighting the generalizability of the approach. The coordination aspect of the approach has been shown to work with a heterogeneous multi-agent team in the same environments, and can scale seamlessly to any number of robots.

1.4 Chapters Overview

The following Chapters are organized as follows. Chapter 2 talks about the system architecture. Going into a little more detail about the overall autonomy pipeline and some of the design decisions that were taken. It also describes the hardware platform and the entire physical system consisting of ground and aerial platforms. Chapter 3 will discuss the single robot exploration approach and provide the overall high level algorithm driving the approach. Chapter 4 will then talk about the map sharing approach that enables coordination between the robots. Chapter 5 will show results from field experiments and simulation runs. Chapter 6 will impart concluding remarks and discuss future work.

Chapter 2

System Architecture and Hardware Overview

2.1 Architecture Details and Rationale

An overview of our exploration and coordination software architecture is provided in Fig. 2.1. The overall pipeline is discussed here along with the rationale behind its intended design.

Map Building

Starting with the map building module, we used occupancy grid maps as the underlying map representation because of its probabilistic modeling of occupancy and ray-casting voxel updates that can be helpful in eliminating noise in degraded conditions such as flying in dust. Additionally, the map representation needed to be fast and computationally efficient owing to a compute constrained aerial platform as well as scalable in order to accommodate extremely large environments such as caves and tunnels. This is why we came up with a custom built occupancy grid mapping solution (VDBMap) that uses an open source data structure optimized for volumetric data (OpenVDB [22]). We found that our OpenVDB based solution gave us on average constant time random access of voxel data as compared to logarithmic time random access provided by a competing grid mapping pipeline called OctoMap

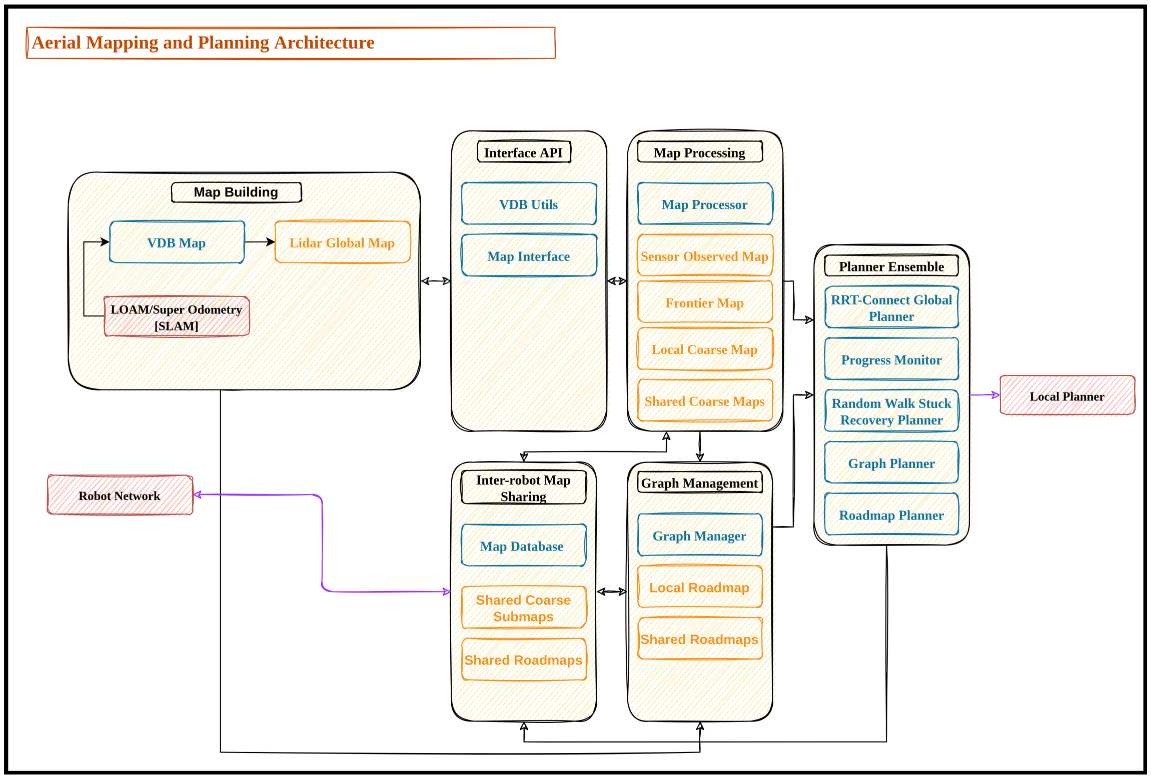


Figure 2.1: Autonomy Architecture. Individual ROS Nodes are shown in blue boxes. The maps they work with are shown in Yellow boxes.

[17] that is quite popular within the robotics community.

Custom built Occupancy Grid Maps using OpenVDB

This subsection sheds a little more light on the specifics behind the data structure used as the basis of the map representation. OpenVDB [22] was conceived at DreamWorks as an efficient solution for working with level set data that underpins a majority of rendered animations in their movies.

It's name comes from the fact that it is a Volumetric, Dynamic grid that is closely related to B+ Trees. It allows for an unbounded 3D grid space that supports fast (average O(1)) random access into the stored voxel data [34]. The data structure has a fixed hierarchy consisting of a root node, and two layers of internal nodes, which then terminate into leaf nodes. The leaf nodes are the nodes that actually encode the individual voxels. Figure 2.2 gives a visual representation of the underlying B+

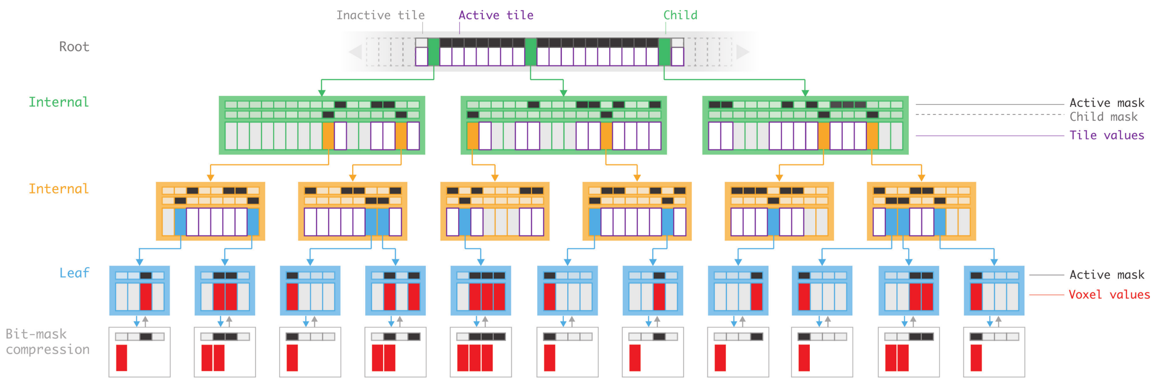


Figure 2.2: A VDB Tree with one Root Node, two Internal Nodes and Leaf Nodes [22].

Tree that forms the basis of this data structure. We'd like to direct the reader to [22] in order to get a detailed analysis of this data structure.

Map Processing

Built on top of the core occupancy grid mapping representation are specific algorithms such as frontier cluster extraction, viewpoint sampling and selection that process information in the maps and provide actionable inputs to the planning module. Since these algorithms work directly with the grid maps, they are also able to take advantage of OpenVDB's fast random access and sparse grid traversal methods. It is also worth noting here that a few approximations were made within the module handling frontier cluster extraction and viewpoint sampling. For instance, when sampling for viewpoints, we uniformly sample poses facing a frontier cluster centroid. We employ an approximating assumption here that if a robot faces a cluster centroid, it is able to observe all frontiers within the cluster. This allowed us to avoid using an explicit camera model to compute viewpoints, which helps us maintain a low computational burden on the system. This does mean that tracking viewpoints does not guarantee coverage in that small chunks of frontier clusters may be left unobserved. It happens to be a worthwhile tradeoff to make in the interest of computational efficiency.

Planner Ensemble

In order to build a robust planning strategy, an ensemble of planners was created as planning ensembles can better account for failure modes as well as different mission objectives [10]. There are four planners that the robot can choose from, namely the RRT-Connect planner, the Random Walk Planner, the Graph Planner and the Roadmap Planner. At any time, the robot is running one of these planners and has the ability to instantly switch to another planner if the required conditions are met.

The overall planning architecture was designed in hierarchical fashion in the sense that the global planners that form the planner ensemble feed a lower level local planner that generates local trajectories based on dynamically feasible motion primitives. This separation between the two tiers of planners was done so that an unforeseen failure in any of the global planners does not affect the local planner - ensuring flight safety at all times.

The RRT-Connect planner is in charge of carrying out the primary exploration mission that enables the drone to track viewpoints pointed towards previously unobserved volume. More details on this in Chapter 3. We needed to handle the situation in which the drone doesn't make progress along the generated global plan which could be due to a number of reasons such as getting stuck in dust or a narrow passage. In such a case, the local planner finds it difficult to find valid trajectories to track the global plan. The Random Walk Planner was conceived to handle such a situation. By feeding the local planner with a randomly sampled goal, it gets the robot *unstuck* from its current position, and then hands the control back to the primary exploration planner to continue exploration.

Once the robot is at a point wherein it needs to return home with the remaining battery life, it switches to the Graph Planner that plans a path back to the takeoff location using a graph generated from previously visited poses. Lastly, we incorporated a Roadmap Planner to enable the robots to find paths to each others' unexplored frontiers using a roadmap that is similar to the graph generated by the Graph Planner. This was done because there are situations in which a robot has no local frontiers left to explore due to them having already been observed by other robots. In which case, we needed a way for the robot to find a way to visit frontiers of other robots in the team. More details on this in Chapter 4.

Now that the rationale behind the design choices made to construct the system architecture is clear, we'll have a quick overview of the hardware used and the different robots that are going to be used in the final SubT Challenge Event.

2.2 Hardware Overview

Figure 2.3 shows the primary aerial robot used for the work in this thesis. It's a custom built platform that carries about a 1kg heavy perception payload. Table 2.1 lists out the key components of the payload that are used.

Part Name	Description
Intel NUC	Primary computer with a quad core Intel i7 8559U CPU
Velodyne VLP-16	16 Line Lidar with an FOV of 30 deg and 100 m range
IDS UI-3271	RGB Camera for object detection
Intel RealSense L515	Short range lidars mounted upward and downward facing

Table 2.1: Payload Components on the UAV

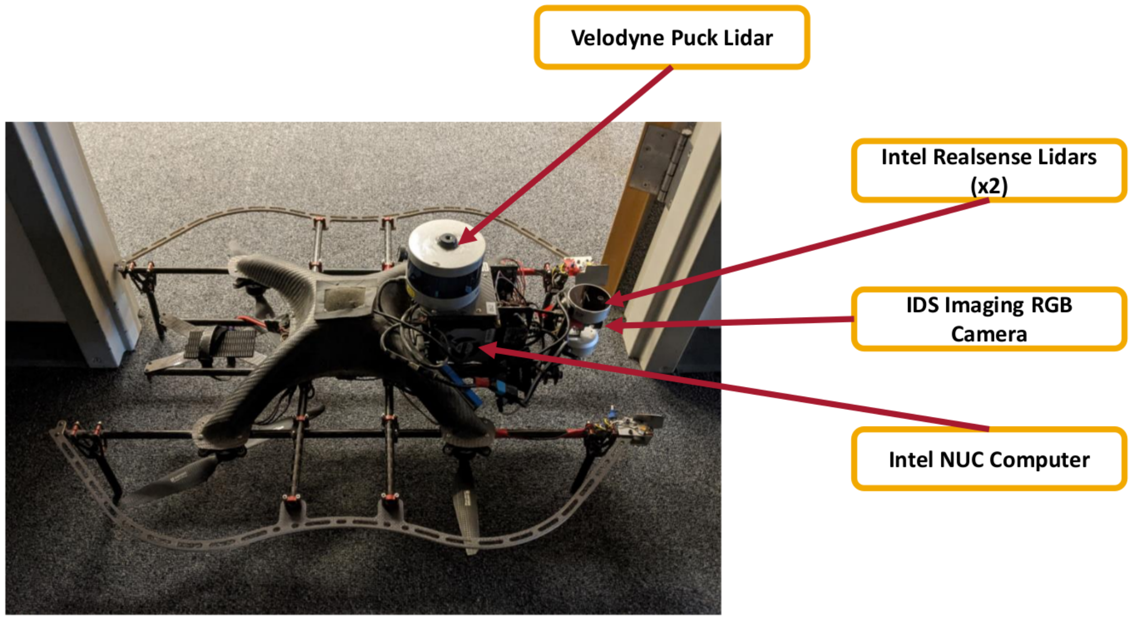
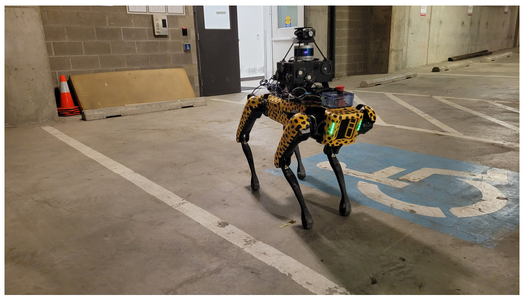


Figure 2.3: Aerial Platform Hardware

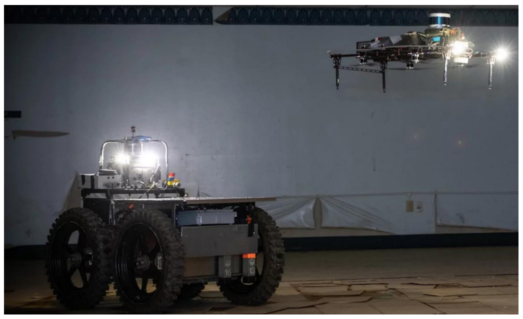
2.2.1 Robot Team

While the work described in this thesis focuses on aerial deployment, it's important to get a sense of the entire physical system. We have two types of platforms - ground and aerial. The ground platforms consist of wheeled robots as shown in Figures 2.4c, 2.4d, and quadrupeds as shown in Figure 2.4b. The aerial platform can be seen in Figures 2.4a, 2.4c. The latter also shows the ability of the drones to take off from the back of the UGVs. We have the ability to carry the drones on the back of the UGVs deep into the environment (Figure 2.4d) and launch them autonomously or remotely by the basestation operator. This allows the drone to leverage their ability to explore at higher altitudes as compared to the ground platforms. Coordination map sharing between the robots additionally enables behaviors that allow each platform to better utilize its strengths during an exploration mission. For instance, the ground platforms, having higher endurance, can focus on traveling far along the ground and covering volume visible upto a limited height. During that time, they can launch drones in spaces that are vertically expansive, allowing the drones to fly up and explore what would not be observable by the ground vehicles. Having the twin exploration planners in the planning ensemble enables the aerial vehicles to be used in a variety of ways. As described before, they can be launched inside the environment from the UGVs where they use the RRT-Connect planner to explore. They can also be launched from a starting area, where they can quickly fly to areas that have not yet been explored by the other robots by employing the Roadmap Planner. These use cases will become more clear with the proceeding chapters.

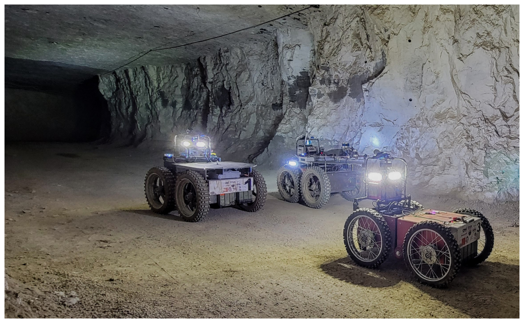
(a)



(b)



(c)



(d)

Figure 2.4: The Robot Team (Team Explorer - DARPA SubT Challenge)

Chapter 3

Single Robot Exploration

As alluded to earlier, the overall approach to exploration can be split into two parts. The first part solves the single robot exploration problem that enables each individual agent within the multi-robot team to explore and cover the environment effectively. This behavior leads into the second part that scales the approach to multiple agents and attempts to bias their exploration regions away from one another.

The single robot exploration approach centers on creating viewpoints that face clustered frontiers, picking a viewpoint based on a selection heuristic and then generating a global plan to the selected viewpoint.

An overview of the exploration pipeline is provided in Algorithm 1. It broadly follows the flow of the autonomy architecture diagram shown earlier in Figure 2.1. Starting with the map building that uses dense registered point clouds to create a global occupancy grid, we then move on to the map processing part involving generating frontiers, and sampling viewpoints. Once a list of valid viewpoints has been created, it is passed to the RRT-Connect based global planner that creates a global plan for the local planner to follow.

3.1 Frontier Detection

Surface Frontiers Our goal in generating frontiers is to focus on exploring unknown surfaces, and not free space. Which is why we diverge from the traditional definition of frontiers [31] that places them in free space. Instead, in our initial implementation, we generated frontiers using occupied cells, similar to [33]. Figure 3.3 (reproduced from [33]) illustrates this distinction. S_k^* represents the observable surface from the free space that the UAV can navigate through. S_k is the surface already observed. ∂S_k represents the surface frontiers, or the boundary between the surface that is observed and the surface that has not yet been observed. Although this approach helps the UAV focus its efforts towards observing surfaces, it is prone to noise. The frontiers are generated based on thresholding counts of unknown, free and occupied voxels inside a cube centered around each occupied voxel. This can produce several spurious frontiers on the outer surfaces of walls for instance. In an indoor or subterranean setting, these outer frontiers are not reachable and therefore need to be cleared. Which is achieved through aggressive thresholding of the aforementioned cell type counts. But when we reduce the false positive frontier voxels this way, we also create false negatives i.e. frontiers in unexplored spaces also get cleared out. Figure 3.1 illustrates the issue. One can see noisy frontier clusters that don't get cleared out in areas that have been observed by the drones cameras. This is because they lie on the outer surface of the wall and hence cannot be marked as observed by the camera model. There is an opening to a tunnel that we know exists since it shows up in the lidar map, but no frontier clusters are generated there because the noise suppression is clearing them out.

Frontier Generation based on subtracted maps Amongst several maps that are being processed, there is a *sensor observed map* that tracks the volume observed by the camera (V_{camera}), and a *lidar global map* that maintains a global geometric map of the volume observed by the longer range lidar sensor (V_{lidar}). We wanted to leverage the fact that the lidar sensor has a longer range and incorporate that into the frontier detection algorithm. This led to an improved technique that uses a subtracted volume V_s to generate frontiers.

$$V_s = V_{lidar} - V_{camera} \tag{3.1}$$

The volume computed in 3.1 is what constitutes the *frontier map*. Once we get the frontier map, we cluster the frontiers based on Euclidean distance, and compute centroids for all the clusters. The improvement in the generated frontier clusters is evident in Figure 3.2. The noisy frontier clusters present in already observed spaces get cleared out, and we get well placed spaced out frontier clusters in all the unexplored tunnels.

Figure 3.4 shows the two maps that are generated during an exploration flight inside a cave. The coverage done by the camera is tracked by the Sensor Observed Map shown in green. As the drone explores, it clears out the volume that constitutes the Frontier Map in pink.

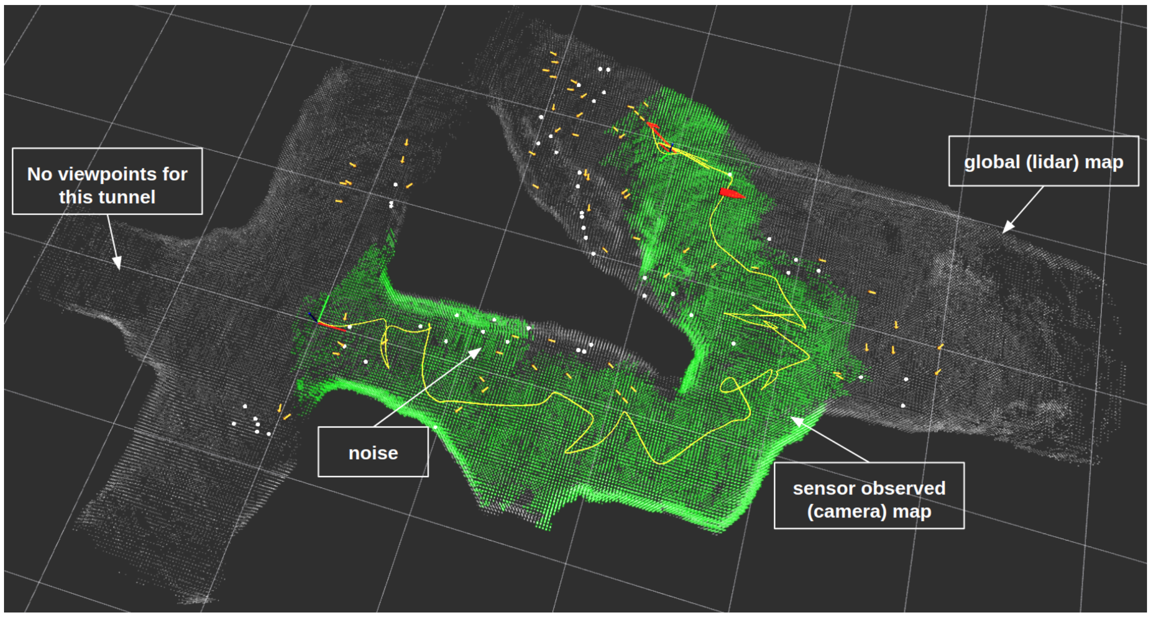


Figure 3.1: [Regular Surface Frontiers] Solid white circles represent frontier clusters. Yellow arrows depict sampled viewpoints facing the frontier clusters. The green map shows the volume observed by the camera. Flight trajectory is shown in yellow. The faint white map is the volume observed by the longer range Lidar sensor.

3.2 Viewpoint Sampling

The next step in the map processing pipeline is generating viewpoints. This is done for each individual frontier cluster. One key assumption that is made here is that having a clear line of sight from a viewpoint to a cluster centroid ensures that the entire frontier is observed. This is a simplifying assumption that was made to lower

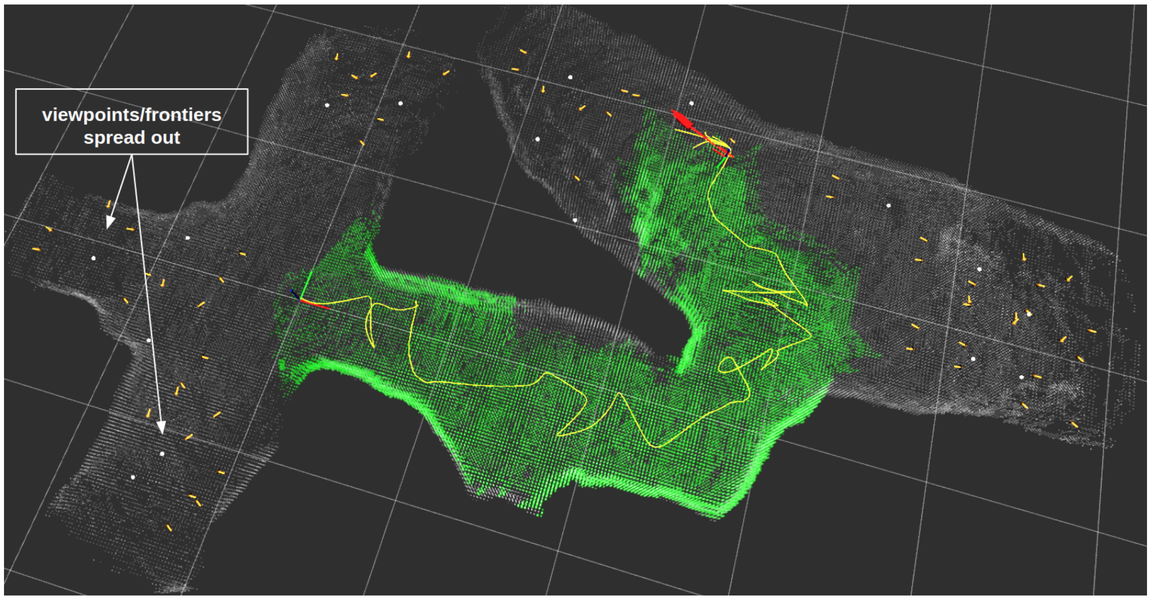


Figure 3.2: Frontiers generated from the subtracted map approach. Clusters in white, viewpoints in yellow. Sensor observed map in green, and lidar observed map in white.

the computational load as compared to an approach in which an explicit 3D camera frustum projection is used to compute cluster voxel visibility.

To generate a set of viewpoints for each cluster, we uniformly sample points that lie on a cylinder centered on the cluster centroid. These points correspond to a discrete set of directions extending from the cluster centroid. Figure 3.5 illustrates this approach. The figure is top-down view of a slice of that cylinder. Sampled viewpoints are evaluated for collisions within the global map, and checked for a collision free path from the viewpoint to the centroid (visibility check). Ones that meet both criteria are put into a list that is then sent to the RRT-Connect based global planner to evaluate and select.

3.3 Viewpoint Selection

The next module in the pipeline prioritizes the viewpoints according to a scoring function. Our proposed scoring function is a weighted sum of (1) the Euclidean distance to the viewpoint, (2) a momentum reward, and (3) a penalty for observing regions that have already been observed by other aerial robots. The momentum

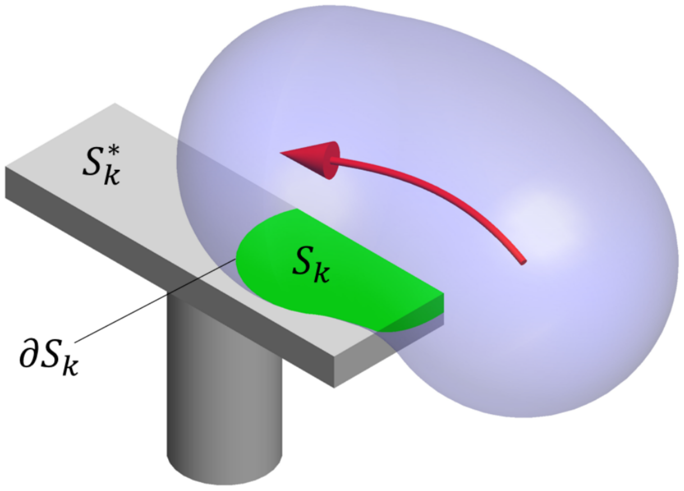


Figure 3.3: Computing surface frontiers. Credits [33]

reward favors selecting viewpoints that are in the current direction of movement to discourage excessive back and forth motions that reduce the speed of the aerial vehicle. A large penalty is given to any viewpoint that is within any of the shared observed submaps received from other robots. Extensive simulation and hardware testing was performed to empirically select appropriate weights for the three terms. We have also investigated adding other heuristic terms, such as the frontier size, heading reward and non-Euclidean distance measures, but we found the proposed solution to be most effective for this task.

$$score = w_1 d + w_2 r_m + w_3 r_s \tag{3.2}$$

Here, w_i refer to the weights, d is the Euclidean distance to the viewpoint, r_m is the momentum reward, r_s is the reward/penalty for the viewpoint based on shared coordination maps received from other robots.

3.4 Path Planning and Execution

Downstream from the viewpoint selection stage, a *global plan* is generated to each viewpoint in order of their respective scores. This plan is computed using the RRT-Connect algorithm first introduced in [20]. The path is composed of a sequence of 3D

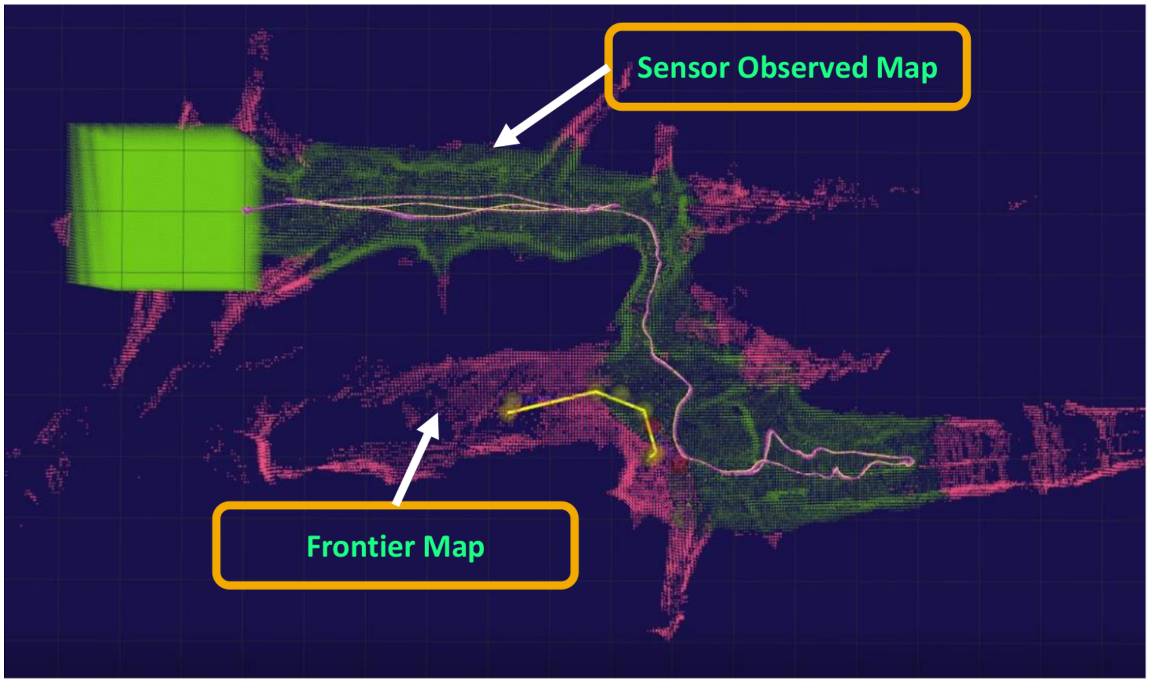


Figure 3.4: Sensor Observed Map (in green) and Frontier Map (in pink). Flight trajectory in pink. Data generated from a flight at Laurel Caverns.

positions using an Euclidean distance metric. The yaw along the trajectory is not set (reliant on the local planner to pick appropriately according to the state of the robot), up until a certain distance before the viewpoint. At which point, the yaw is set to the yaw of the viewpoint. The feasible path that is found is then sent to be executed by the local planner until a switching condition is met. Switching can occur due to multiple factors, when the frontier associated with the goal viewpoint is marked as observed, a viewpoint with a significantly higher score appears, the drone is making insufficient progress along the trajectory, or the viewpoint has been reached by the UAV.

When it is found that the UAV is not making sufficient progress along the trajectory, a backtracking or random walk behavior is briefly executed to resolve the issue, then a new viewpoint is selected, switching back to the RRT-Connect planner. This is handled by the Random Walk Planner within the ensemble.

The Return to home behavior needed at the end of the exploration phase is handled by the Graph Planner. If the remaining battery life falls below a threshold

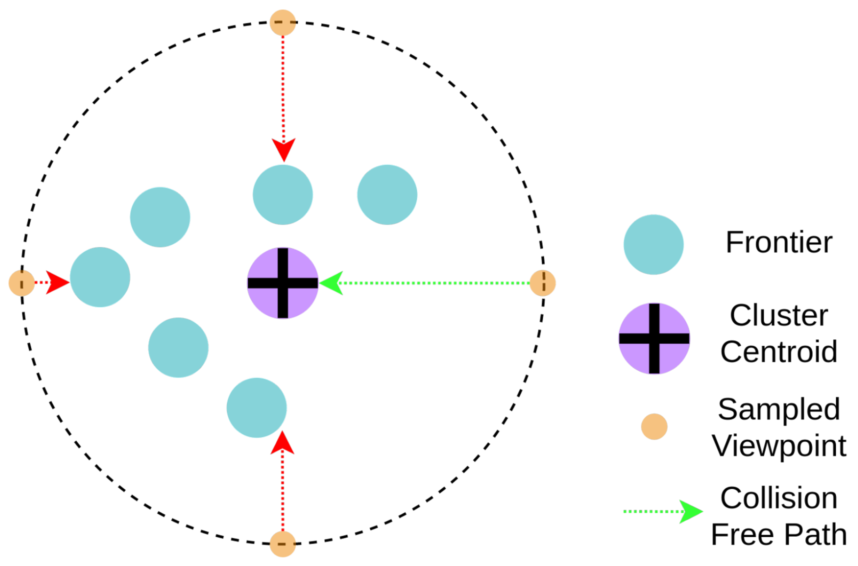


Figure 3.5: Viewpoint Sampling

or if there are no frontier clusters left to explore, then the aerial vehicle plans a path on a roadmap (consisting of keyposes visited by the robot) that takes it back to the take-off location.

RRT-Connect was picked as the planning algorithm mainly because of its fast convergence rates and low computational overhead. In this case, we are not too concerned about suboptimal path lengths because the drone is usually flying into unobserved spaces. Zigzag path trajectories that are suboptimal from an Euclidean distance metric can lead to better exploration of unknown spaces.

Algorithm 1 Overview of the aerial exploration and coordination pipeline. The implementation executes many of the listed functions in parallel within separate ROS nodes.

```
 1: ▷ Initialize Maps
 2: $global\_map, observed\_map, frontier\_map, shared\_coarse\_maps \leftarrow$ InitMaps
 3: ▷ Update Plan at a Fixed Frequency
 4: for each timestep do
 5:     ▷ Receive Data
 6:     $odometry, lidar \leftarrow$ ReadSensors
 7:     $shared\_coarse\_maps \leftarrow$ ReceiveSharedMaps
 8:     ▷ Mapping
 9:     $global\_map \leftarrow$ BuildOccupancyMap($lidar, odometry, global\_map$)
10:     $observed\_map \leftarrow$ Raycasting($global\_map, odometry, observed\_map$)
11:     $frontier\_map \leftarrow$ ExtractFrontiers($observed\_map, odometry$)
12:     ▷ Frontier Generation and Viewpoint Sampling
13:     $clusters \leftarrow$ Clustering($frontier\_map$)
14:     $viewpoints \leftarrow$ GenerateViewpoints($clusters$)
15:     $coordination\_scores \leftarrow$ EvaluateMaps($viewpoints, clusters, shared\_coarse\_maps$)
16:     ▷ Viewpoint Selection
17:     $sorted\_viewpoints \leftarrow$ ScoreAndSort($viewpoints, odometry, coordination\_scores$)
18:     ▷ Progress Monitor
19:     $request\_new\_path \leftarrow$ CheckProgress($path, odometry$)
20:     if $request\_new\_path$ then
21:         ▷ Path Planning
22:         for each $viewpoint \in sorted\_viewpoints$ do
23:             $path \leftarrow$ Bi-RRT-Connect($odometry, viewpoint, global\_map$)
24:             if PathFound($path$) then
25:                 break
26:     ▷ Publish Incremental Goals to Local Planner
27:     $waypoint \leftarrow$ ExecutePath($odometry, path$)
28:     PublishWaypoint($waypoint$)
29:     ▷ Share Coarse Sensor Observed Maps with Other Robots
30:     $outgoing\_coarse\_maps \leftarrow$ GenerateCoarseMap($observed\_map, frontier\_map$)
31:     CommunicateSharedMap($outgoing\_coarse\_maps$)
```

Chapter 4

Multi-Robot Exploration

The coordination between robots is done by sharing two kinds of maps using a mesh based wireless communications network (similar to [9]) that is laid out by the ground robots as they traverse the environment. The robots themselves are also nodes in the network and can directly communicate with each another if they happen to be in communications range. Since the ground robots run a different autonomy architecture, it was decided to keep the coordination one-sided when it came to coordinating exploration between them and the drones. This way the ground robots only send information to the drones and don't act on the information they receive from the aerial robots. The drones are free to act on the information they receive irrespective of the kind of robot in the system. This reduced the overall system complexity, keeping field testing and system deployments tractable.

The maps being shared are of two types. The first type of map is the *coarse sensor observed map* that contains a coarse version of information stored in the *sensor observed maps*. The second type are *roadmaps* that contain a history of keyposes along the robot's already traversed trajectory. The second type was developed to counter the limitations imposed by the first approach. More details in the sections below.

4.1 Coarse Sensor Observed Maps

These maps are binary occupancy grids consisting of large voxels (grid resolution $2.0m$) that encode two states - *true* indicates that the volume has been observed by a robot, and *false* indicates that the volume is a frontier volume (i.e. has been identified as geometry that exists but has not yet been observed by the camera).

Each robot creates its own local version of a coarse sensor observed map and shares it with other robots over the network. The same robot receives these maps shared by other robots and merges them into a global coarse map that is then used to score frontier clusters and viewpoints. Figure 4.1 shows this in action. The map visualized is that from UAV1 based on coarse maps it receives from UAV2. The coarse maps UAV1 generates locally are shown in green and blue. Blue voxels denote that the volume has been observed by UAV1's camera, while the Green voxels denote volume that consists of frontiers. The coarse maps received from UAV2 are shown as Red voxels. More on how that information is used follows below.

Going back to the viewpoint selection Eq.3.2, the last term r_s (viewpoint reward) is based on these shared coarse maps. When frontier clusters are being generated, they are evaluated for what kind of volume they lie in based on the information in the global coarse map. Each viewpoint associated with a cluster gets that score. If the cluster lies in volume that has already been observed by another robot, it is given the lowest score. The next higher score is assigned for clusters that lie in volume that has been marked as a frontier volume by other robots. The highest score goes to clusters that lie in completely unknown volume in other robots' sensor observed maps. Figure 4.1 shows the different frontier cluster scores as solid circles with distinct colors - Red, Yellow and Green. Each cluster is represented by its centroid. Red colored centroids mean that the cluster lies in volume that has already been observed by another robot. Yellow colored centroids denote a cluster that lies in volume that also happens to be frontier volume for the other robot. Green colored centroids denote volume that has not been picked up by the other robots' lidar - it is completely unknown.

This way of scoring clusters and their associated viewpoints allows for biasing the robots' exploration regions away from each other. This can be quite effective in environments with a high branching factor near the starting/takeoff area because it directs the robots down different directions from the very beginning. Figure 4.2

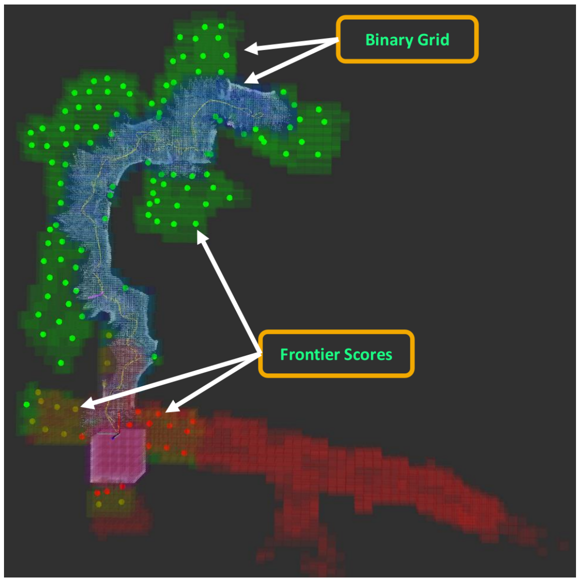


Figure 4.1: UAV1 - Scoring local frontier clusters using Coarse Sensor Observed Maps. A top view of a map built during an exploration run inside a limestone mine at Brady's Bend. Red voxels show incoming coarse maps from UAV2.

shows this approach in action. This is a coordinated exploration run that was carried out with two UAVs in simultaneous flight. This experiment was performed inside the mine at Brady's Bend, PA. Each flight lasted approximately 8 minutes in duration. In the left half of the figure, the green and faint blue voxels represent two voxel states transmitted in the outgoing coarse submaps generated by UAV1 during its exploration run. These submaps are shared over the network and are received by UAV2 (depicted as red voxels in the right half of the figure). UAV2 is launched a minute after UAV1 starts its exploration run. We can observe that UAV2 is able to successfully explore regions that were not explored by UAV1 and avoid regions previously explored by UAV1.

The solid green points that are seen in Figure 4.2 represent the frontier clusters with the highest score - ones which lie in volume that is unknown by the other robot. The Red points indicate frontier clusters that lie in volume that has already been observed by the other robot. The clusters with the middle score are the ones that are colored in yellow - they lie in volume that also happens to be a frontier volume in the other robot's observed map.

Algorithm 1 also includes the coarse sensor observed maps sharing and processing

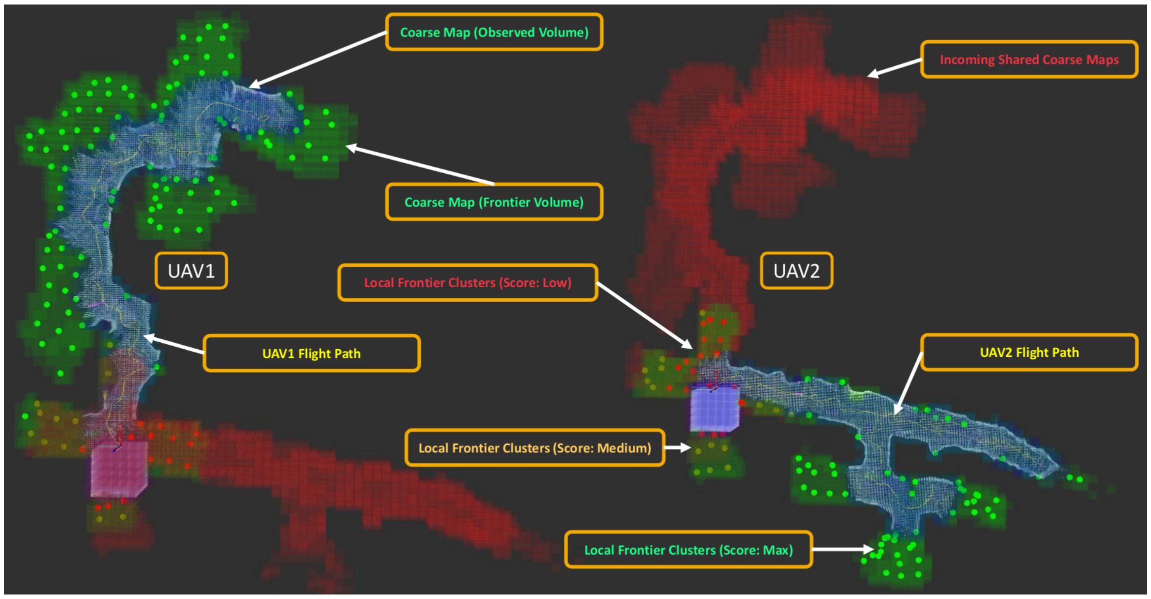

Coarse Map (Observed Volume)
Incoming Shared Coarse Maps
Coarse Map (Frontier Volume)
UAV1
Local Frontier Clusters (Score: Low)
UAV2
UAV1 Flight Path
UAV2 Flight Path
Local Frontier Clusters (Score: Medium)
Local Frontier Clusters (Score: Max)

Figure 4.2: Simultaneous coordinated exploration by two drones at Brady's Bend Mine (Top View). Green voxels represent unexplored volume. Faint blue voxels represent the volume observed by the robot's cameras. Red voxels denote shared coarse sensor observed maps received by the robot. Green points in the map reflect unexplored frontier clusters. The explored map is visualized as white points. The drone flight trajectory can be seen as a continuous line in yellow.

parts that factor into viewpoint scoring and selection as carried out in the RRT-Connect global planning stage.

4.2 Roadmap Planner

Sharing information on what each robot has observed has can have limitations. For instance, what happens if all the locally generated frontier clusters lie in volume that has already been observed by other robots? If we were only sharing coarse observed maps, it would lead to re-exploration of the same volume. In order to overcome this limitation, we added *roadmaps* to supplement the coarse observed maps.

We define a *roadmap* as being a graph $\mathcal{G}$ whose vertices $\mathcal{V}$ comprise of all the key poses visited by the robot as it traverses the environment. The vertices are connected by collision free edges $\mathcal{E}$ along the traveled path and with neighboring vertices.

In addition to tracking key poses, the graph vertices also encode associating edges

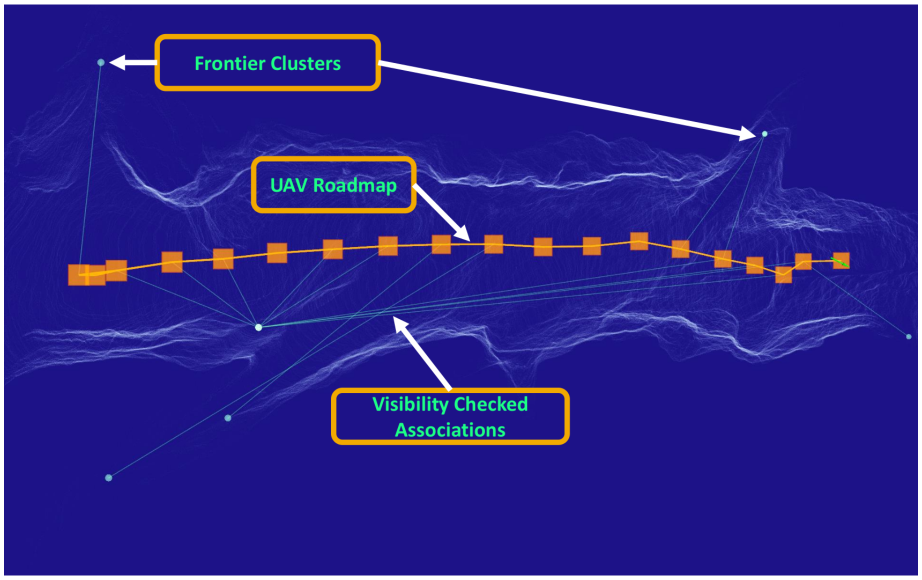


Figure 4.3: Roadmap generated from UAV2 during a flight at Laurel Caverns, PA. The vertices are marked with squares, and collision free edges making connections between them. Frontier clusters are shown as solid green circles, and edges associating clusters with roadmap vertices can be seen in green.

$\mathcal{A}$ with frontier clusters $\mathcal{F}$. These associations are created after conducting a simple line of sight collision check between a cluster and a vertex. The collision check uses a reduced robot model cube dimension so that we can generate enough associations for the clusters. Figure 4.3 shows a roadmap generated during a flight at Laurel Caverns, PA. The graph is highlighted in orange, with vertex-frontier cluster associating edges marked in green.

These roadmaps are then shared in a similar manner to the coarse observed maps between robots over the network. At the receiving end, these shared roadmaps are evaluated for connecting edges with the local roadmap within a threshold distance. The edges that are added are called virtual edges, because they have been created without any free space knowledge of the volume they lie in. Figure 4.4 illustrates this concept. It shows two roadmaps generated from UGV2 and UAV2 during an exploration run at Brady's Bend Mine in PA. UAV2 is launched from the back of UGV2, and on takeoff receives UGV2's roadmap because they are in communications range. Once the shared roadmap is received by UAV2, it generates the purple colored virtual edges that allows the roadmap planner to plan over the combined roadmap of

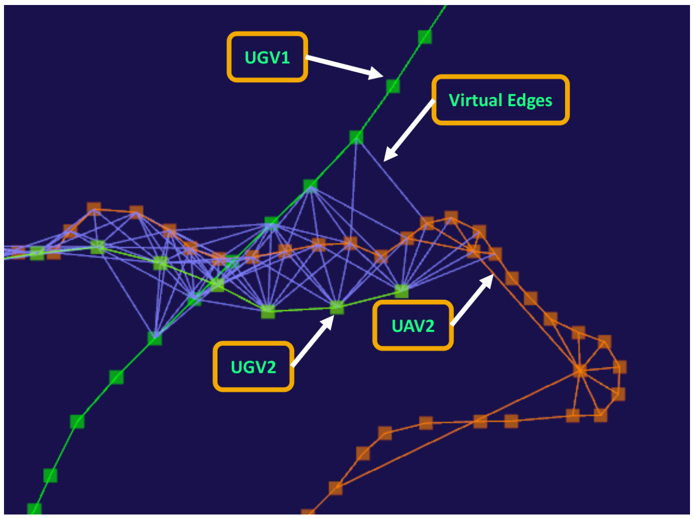


Figure 4.4: Roadmaps from UGV2 and UAV2 during a run at Brady's Bend Mine, PA. The vertices are marked as the squares, with collision free edges making connections between them. Green reflects the roadmap for UGV2 and the orange roadmap belongs to UAV2. Virtual edges are drawn between vertices from the two roadmaps - shown in purple.

UAV2 and UGV2.

This way, if UAV2 encounters a situation in which it no longer has its own local frontiers to explore, or if they have already been observed by other robots, then it can plan over the combined roadmap to navigate to UGV2's unexplored frontiers.

The roadmap planner's algorithm pseudocode has been outlined in Algorithm 2. It is always planning in the background, and waits for a signal that enables it to publish the path. This published path is then used by the local planner to navigate the robot to the next unexplored frontier cluster.

Figure 4.5 shows the Roadmap Planner in operation during two sequential drone flights held at an urban location containing corridors with rooms. UAV1 takes off first and flies down the corridor - effectively observing the entire main corridor. This shows up as the red voxels within the Shared Coarse Sensor Observed Maps received by UAV2 from UAV1. So when UAV2 takes off, it has no local frontiers to explore, since they were already observed by UAV1. At this point, the Roadmap Planner gets activated, and uses UAV1's shared roadmap to plan a path to frontiers that were left unexplored by UAV1. This path is shown in blue. Once the local planner executes this path and the drone reaches the green circle at the intersection of 4 corridors, the Roadmap Planner at that point hands over control to the RRT-Connect Exploration Planner to continue from thereon. The two green arrows reflect the two directions in which UAV2 can fly to start exploring areas left unexplored by UAV1 (as shown by the yellow voxels).

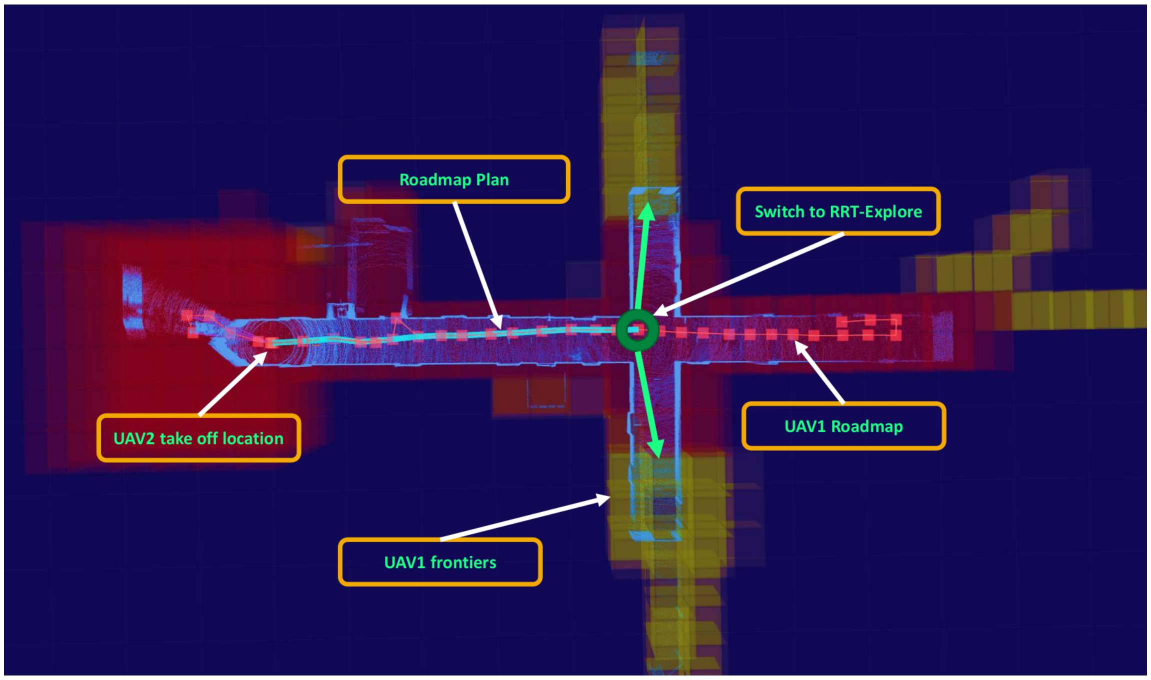


Figure 4.5: Top view of UAV2's map at takeoff inside a corridor at an urban test site. Roadmap in red is generated by UAV1 that flew first. Path generated by the Roadmap Planner for UAV2 to follow is shown in blue. Shared Coarse Sensor Observed Maps received by UAV2 from UAV1 are shown in Red and Yellow. Red voxels denote observed volume, and yellow voxels denote frontier volume. The green circle at the end of the path denotes the point at which the Roadmap Planner would hand over control to the RRT-Connect exploration planner so that it can go explore the frontiers lying to the left and the right of the intersection (denoted by the two green arrows).

Algorithm 2 Overview of the roadmap exploration planner. The implementation runs in a single thread as part of one ROS Node.

```
1: ▷ Initialize Maps
2: local_roadmap, shared_roadmaps, virtual_edges ← InitRoadmapDatabase
3: ▷ Update Plan at a Fixed Frequency
4: for each timestep do
5:     ▷ Receive Data
6:     odometry, lidar_map ← ReadSensors
7:     shared_roadmaps ← ReceiveSharedRoadmaps
8:     frontier_clusters ← ReceiveFrontierClusters
9:     global_map ← DeserializeOccGridFromSharedMemory
10:    ▷ Update local roadmap and the virtual edge between local and shared roadmaps
11:    local_roadmap ← AddNewVertexInRoadmap(odometry, lidar_map)
12:    virtual_edges ← UpdateVirtualEdges(local_roadmap, shared_roadmaps)
13:    ▷ Path Planning
14:    goal_cluster ← ClosestCluster(local_roadmap, shared_roadmaps, virtual_edges)
15:    path ← ShortestPath(goal_cluster)
16:    ▷ Command Callback to publish plan
17:    request_new_path ← RoadmapPublishPathCommandCallback
18:    if request_new_path && path then
19:        ▷ Publish Path
20:        PublishPlannedPath(path)
```

Chapter 5

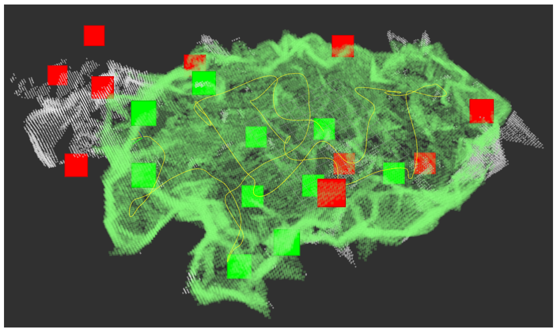
Results

We have performed extensive testing of our aerial vehicle exploration and coordination pipeline. Here we present a representative set of results from both simulated and hardware experiments.

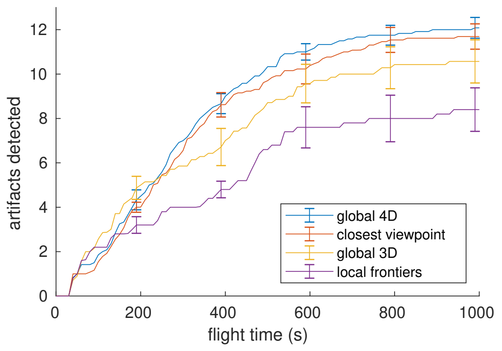
5.1 Simulation runs

Simulated experimental results are presented in Figure 5.1. These simulations were performed within a custom procedurally-generated Gazebo world (Figure 5.1) that consists of a random polyhedron-shaped surface and 20 artificial artifacts distributed over the surface. These simulation worlds are a new variant of the random tunnel worlds presented earlier in [24]. In these experiments, the aerial vehicle is assumed to observe artifacts that are within the field of view of the artifact detection cameras up to 5.0 m away, and with an angle of incidence to the surface of less than 60°. We compare the following exploration policy variants to evaluate our proposed method:

- *Global 4D* is our proposed method with 4D viewpoints and a heuristic viewpoint selection policy as described above;
- *Closest viewpoint* is the same except with a closest-first viewpoint selection policy that uses Euclidean distance only;
- *Global 3D* does not consider the orientation (yaw) of the viewpoint when planning; and,



(a) A partially-explored procedurally-generated random 3D cave world with artificial artifacts placed on the walls. The yellow line shows a partial exploration trajectory. White points illustrate the global map observed by Lidar, while green are predicted to have been observed by the artifact detection cameras. The green squares specify artifacts observed according to the sensor model, while the red squares have not yet been observed.



(b) Simulation results showing the number of artifacts detected by the comparison methods. *Global 4D* is our proposed method, and the other three exploration policy variants are described in the body text. Error bars indicate standard error over 10 trials.

Figure 5.1: Comparison of different exploration strategies in simulation.

- *Local frontiers* only uses frontiers at the boundary of the sensor observed map.

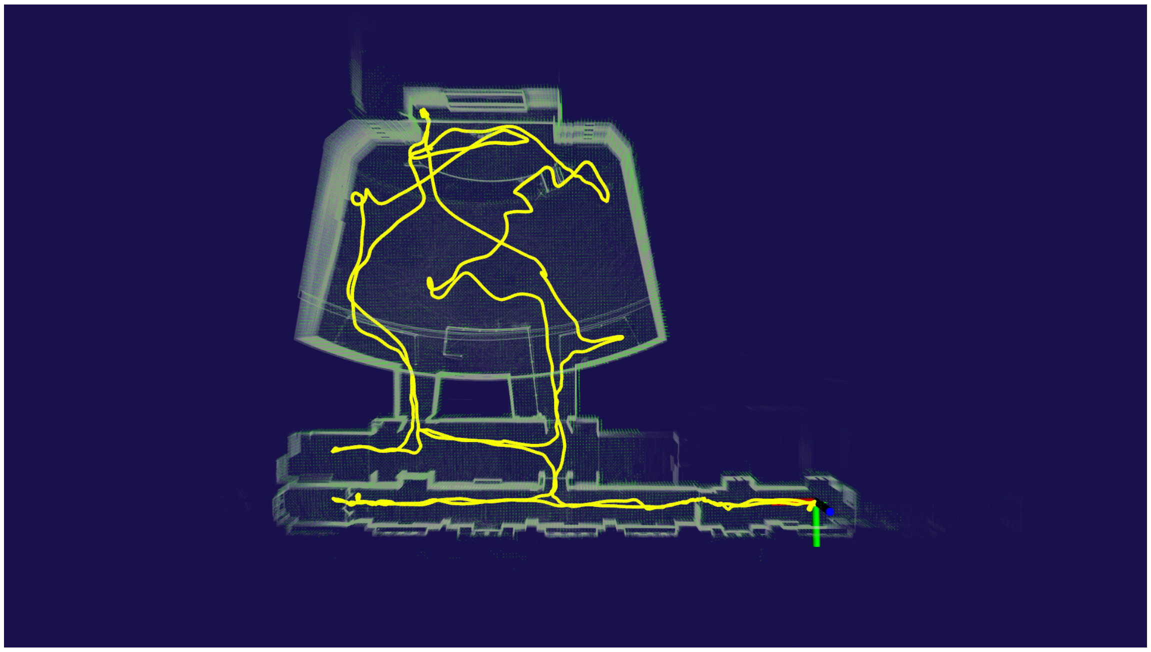
Our results presented in Figure 5.1b compare these four exploration policy variants. The highest performance was achieved when using our full method with the 4D viewpoints and frontiers extracted from the *frontier map*. The comparison method that defines frontiers only at the boundary of the *observed map* performed poorly as this boundary is often noisy, resulting in regions being missed. The comparison method that uses 3D viewpoints (i.e., does not specify the yaw) performed similar to our method; however, it misses some of the harder to observe artifacts that are hidden in the corners and are only in the camera field of view for a small range of yaw values. The comparison method that does not use the momentum term of the viewpoint selection heuristic performed almost as well as our full method in this environment; however, we note that in other environments with more sharp corners in the surfaces, the performance was often observed to be poorer when not using the momentum term due to excessive back-and-forth motions slowing down the robot.

5.2 Real World Tests

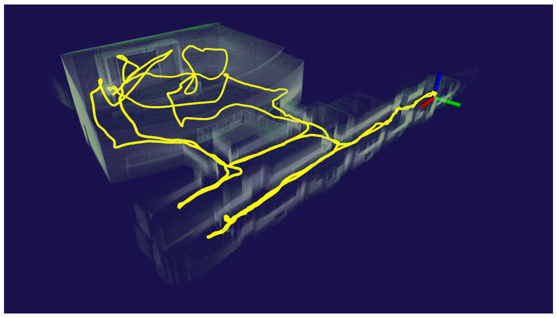
We carried out field tests of our systems at a number of locations similar to the environments that are expected in the DARPA SubT Challenge. The following list provides a brief description of the test sites.

- Brady's Bend Underground Storage Facility: A now inactive limestone mine in Armstrong County, Pennsylvania. It started being used as an underground storage facility in the 1970s. This site is a good representation of the Megacaverns at Louisville, Kentucky, where the final SubT Challenge Circuit is being held. It is the largest and most open environments of all the sites we have been testing at.
- Hawkins Auditorium: Hawkins is a codename for an abandoned urban facility consisting of several building in a fairly large complex. It is located within the city of Pittsburgh. The auditorium is one of the larger open spaces at this site. We used it to test the vertical exploration abilities of the UAVs.
- Hawkins Qualification Course: In one of the buildings at Hawkins, we created a robot qualification course that consists of corridors and rooms with artifacts placed inside them. We used it for regression testing and validation of our systems.
- Laurel Caverns: The largest cave system in Pennsylvania by volume and area. It consists of two sections, the upper section is a network of interconnecting grid-like passages. The lower section consists of subterranean watercourses into a dendritic system of passages [30].
- Tour-Ed Mine: A now unused coal mine, it was converted into a museum for tours in 1970. It is located about 20 minutes from downtown Pittsburgh. This mine has been a good practice ground for exploring tunnel like environments.

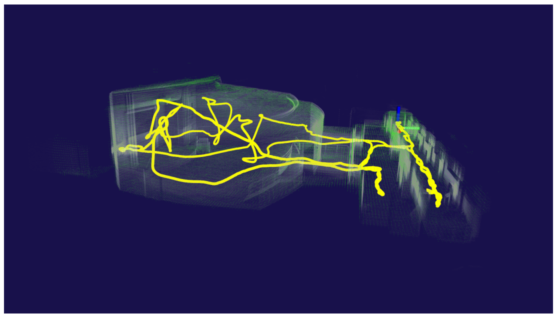
Single Robot Exploration Runs Figures 5.2, 5.3, 5.4 show maps and flight trajectories from single robot exploration runs at Hawkins and Tour-Ed Mine. The sensor observed map (in green) has been overlaid over the map built from registered Lidar point clouds (in grey) to show coverage by the UAVs camera. Flight trajectory is visible in solid yellow. One point to note here is that these flights have been conducted without changing parameters betwen environments. During the final DARPA SubT Circuit Event, we expect to encounter all three types of environment geometries during the same run, which means we won't have the ability to change exploration parameters with changes in the environment. In the Hawkins environment, we see fairly reasonable exploration behavior. There are instances wherein the UAV exhibits some back and forth behavior by making large changes in directions to cover a left out volume. In bounded spaces such as these, having good coverage is desirable because it means the robot will visit rooms as it flies through the corridor. This does come at the cost of flying at a relatively lower speed because of large momentum shifts required to do coverage. The flight through Tour-Ed Mine looks fairly straightforward owing to the uniform nature of the main tunnel through which the UAV flies. The robot flies a relatively straight trajectory down the tunnel before turning around to return home.



(a)

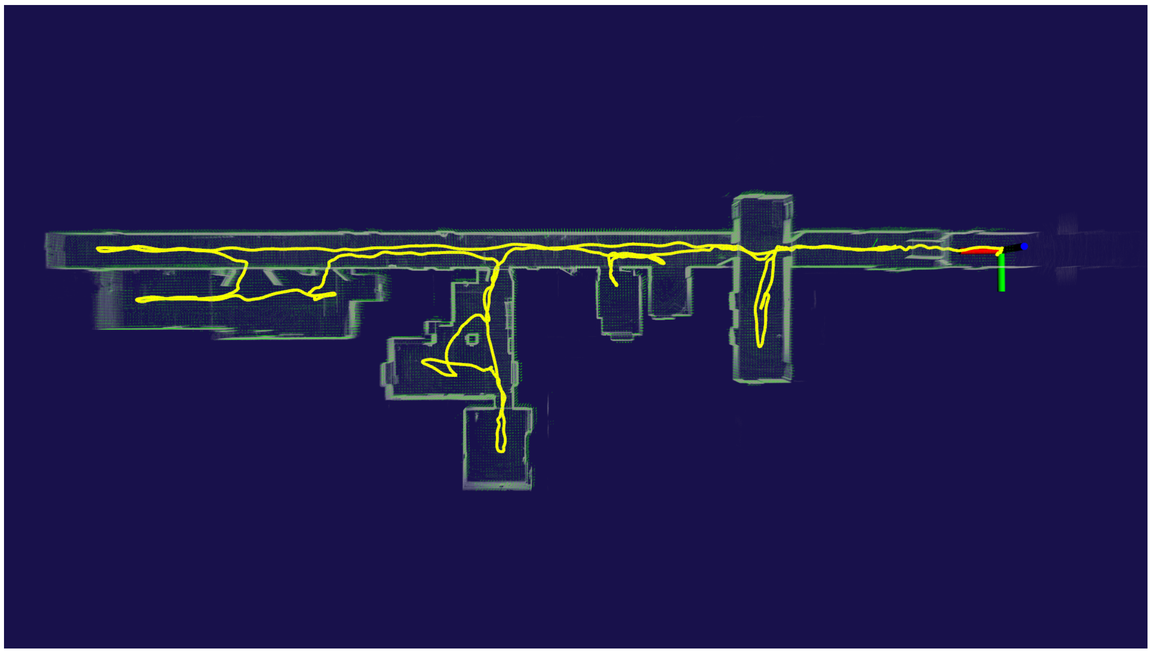


(b)

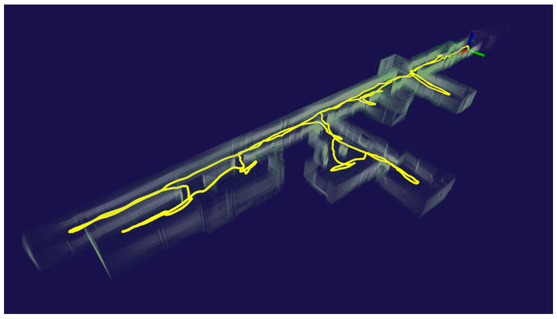


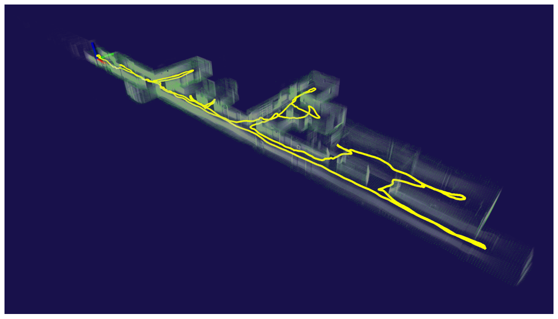
(c)

Figure 5.2: Hawkins Auditorium



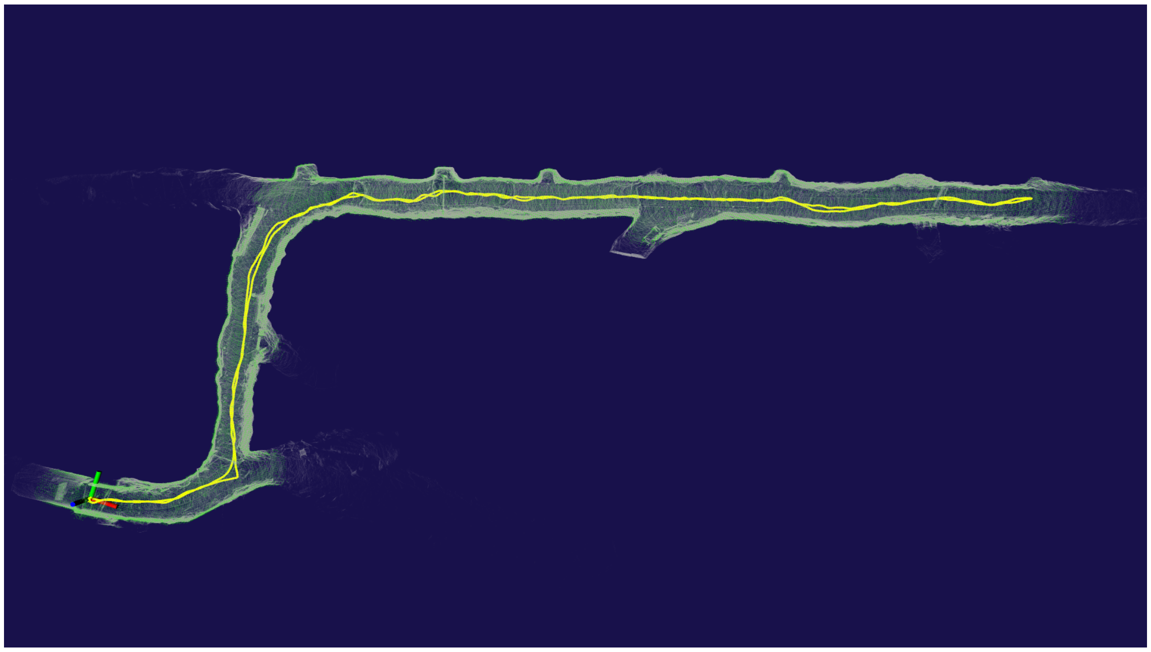
(a)

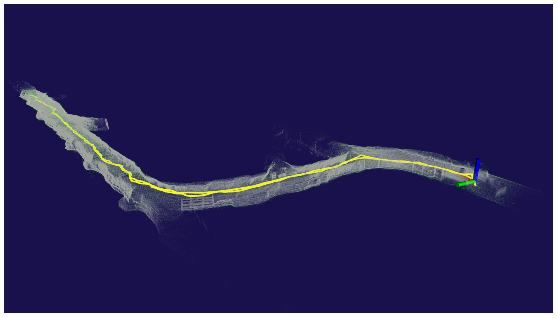




(b) (c)

Figure 5.3: Hawkins Qualifying Circuit


(a)


(b)

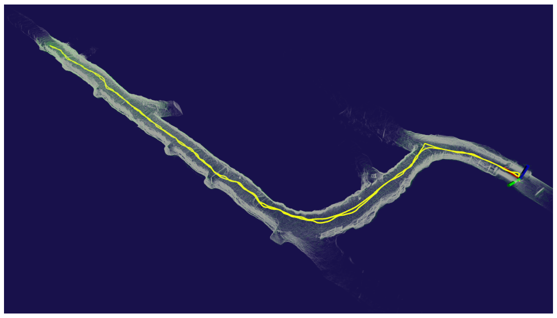
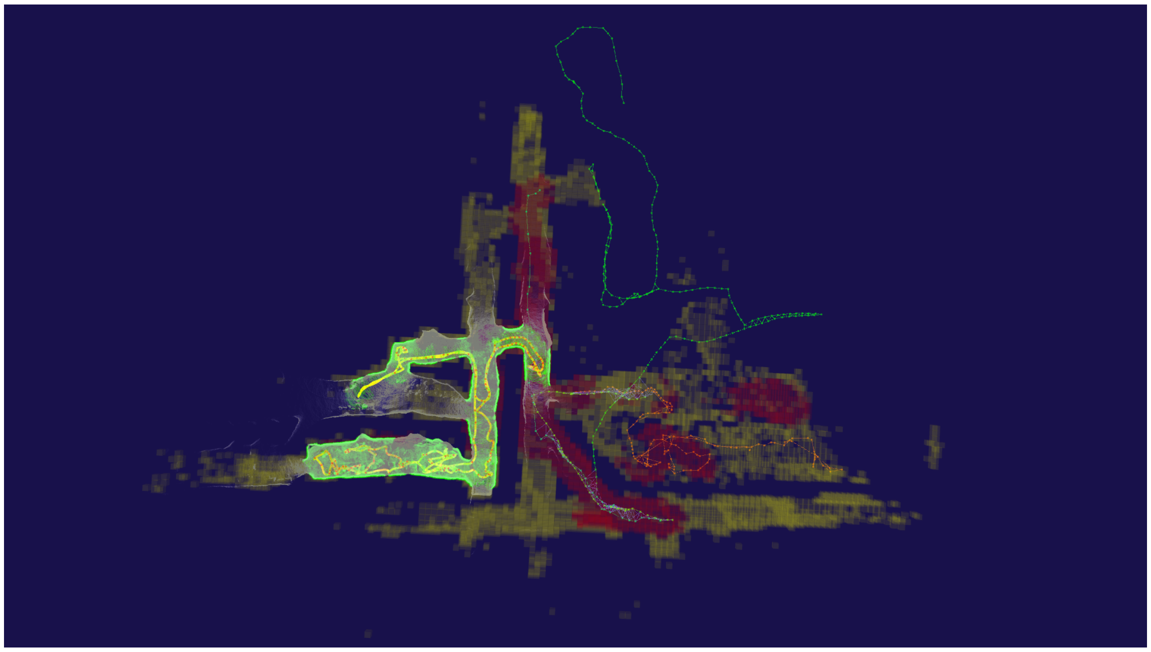

(c)

Figure 5.4: Tour-Ed Mine

Coordinated Exploration Runs Figures 5.5, 5.6 show two flights with coordination enabled between all the robots. Fig. 5.5 shows UAV1's maps. Coarse sensor observed maps and roadmaps were shared by other aerial and ground platforms. Fig. 5.5a details the shared maps. The coarse observed maps are marked with yellow and red voxels. Yellow voxels signify volume that is a frontier for the robot and red voxels signify volume that has been fully observed. Roadmaps from all the different robots are also visualized as graphs in distinct colors. Virtual edges are shown in purple. UAV1's flight trajectory is marked out in solid yellow and shown in more detail in Fig. 5.5b. The observed volume is marked out in green and the lidar map is shown in grey. The roadmap planner was never really triggered during this run because there were plenty of globally unobserved frontiers for UAV1 during its flight. But one can see the effect of the coarse observed maps. They prevent the drone from re-exploring the main tunnel which had already been traversed by UGV1 and it proceeds to explore the tunnel to the left. A similar effect is seen in UAV2's flight in Fig. 5.6a. The coarse observed maps prevent it from flying back into the main tunnel, and it chooses to explore the tunnel to the right of the main tunnel.

Figures 5.7, 5.8 show two flights with coordination enabled between the drones at Laurel Caverns. The coarse sensor observed maps are shown in Fig. 5.8a. In UAV2's exploration flight, one can see how much further down the cave it ends up flying because the initial part of the cave had already been explored by UAV1.


(a) UAV1 Coordination Maps

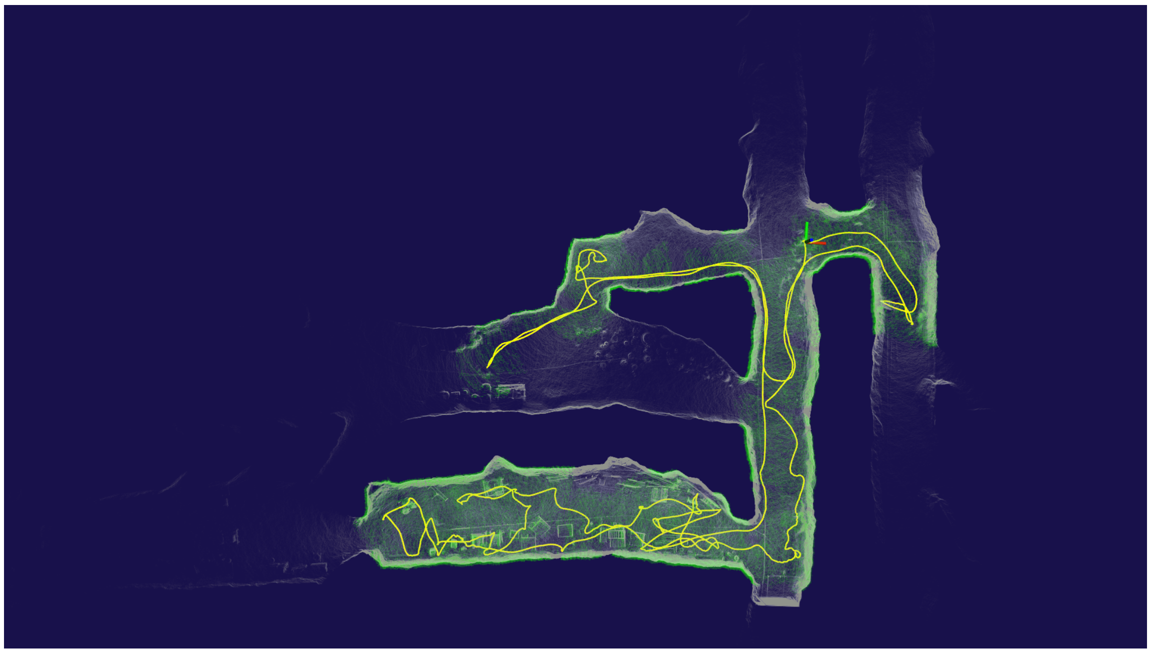
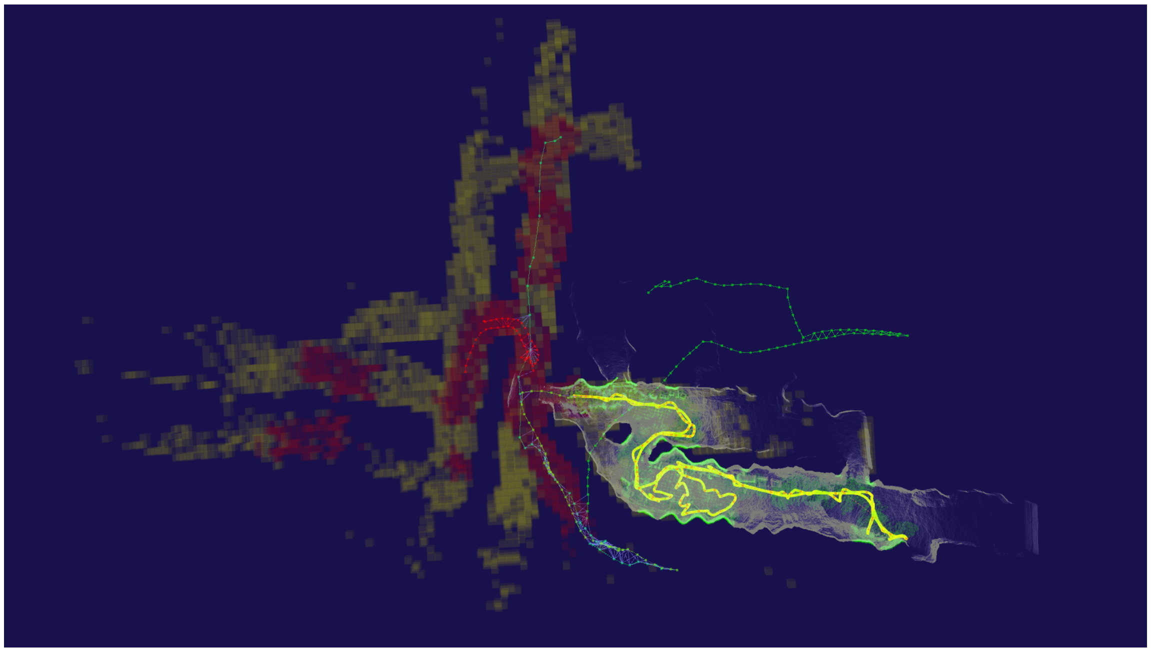

(b) UAV1 Exploration Close up

Figure 5.5: Brady's Bend Mine (UAV1)


(a) UAV2 Coordination Maps

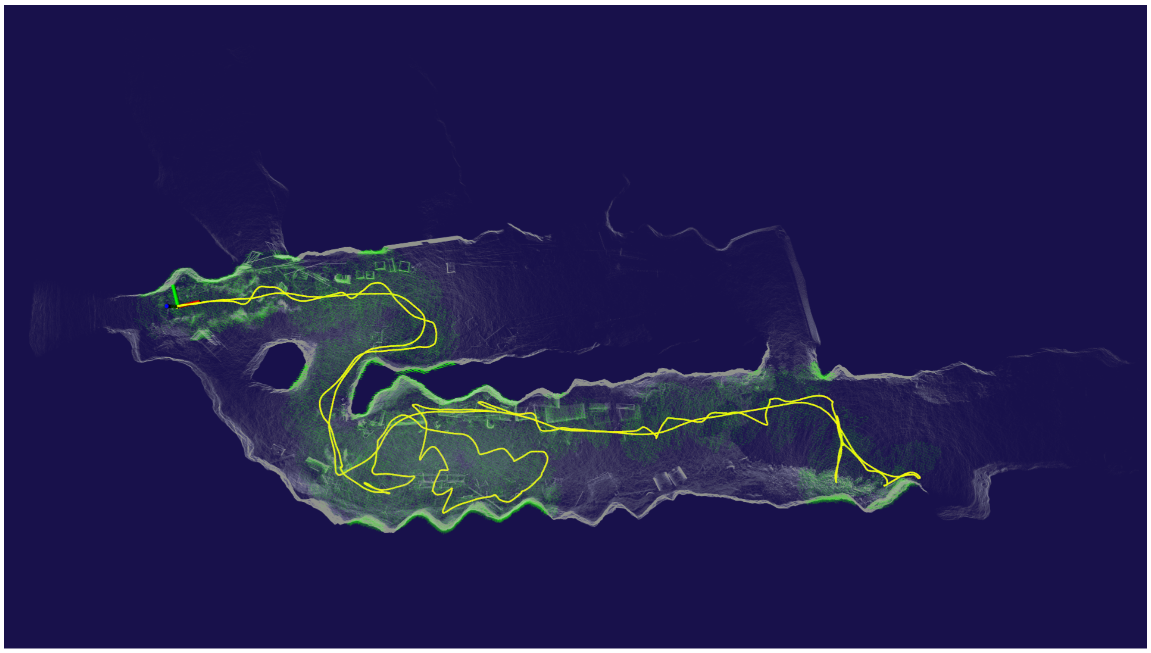
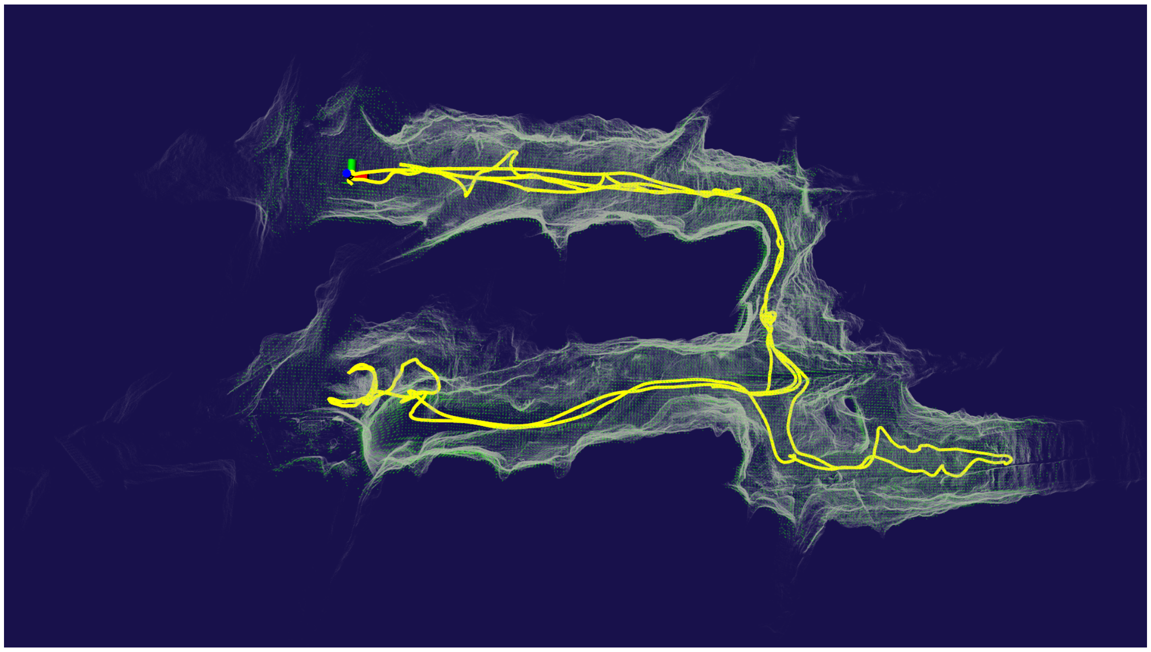
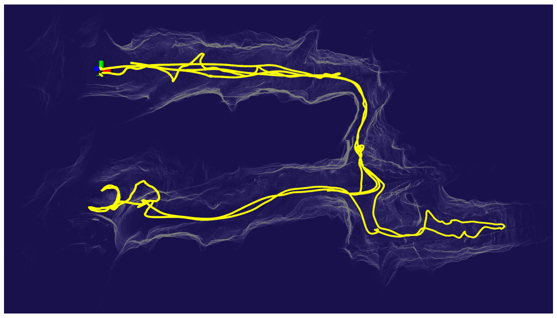

(b) UAV2 Exploration Close up

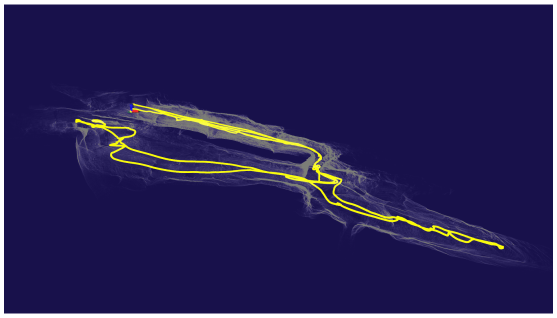
Figure 5.6: Brady's Bend Mine (UAV2)



(a)

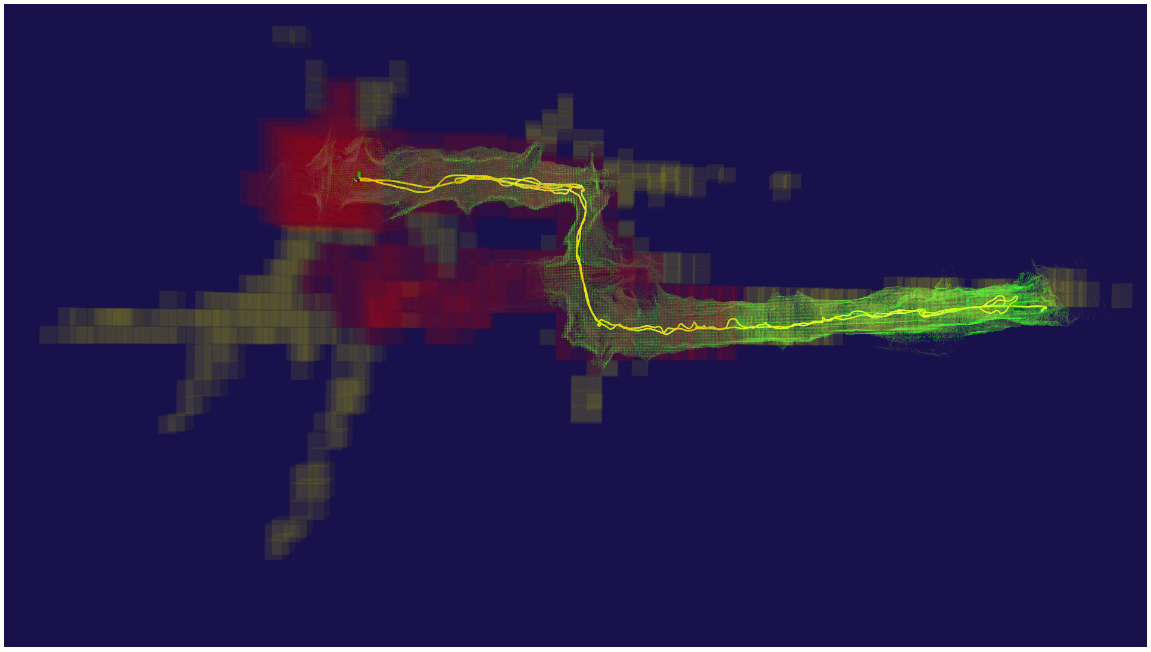


(b)

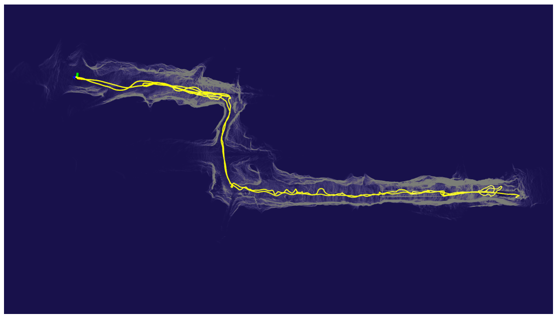


(c)

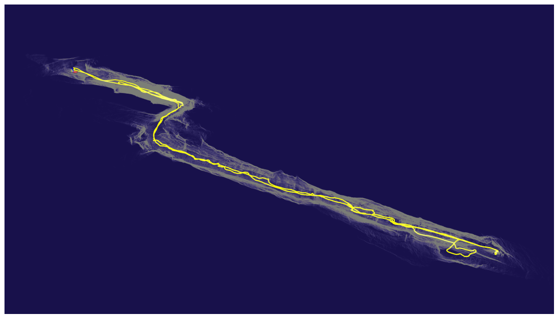
Figure 5.7: Laurel Caves (UAV1)



(a)



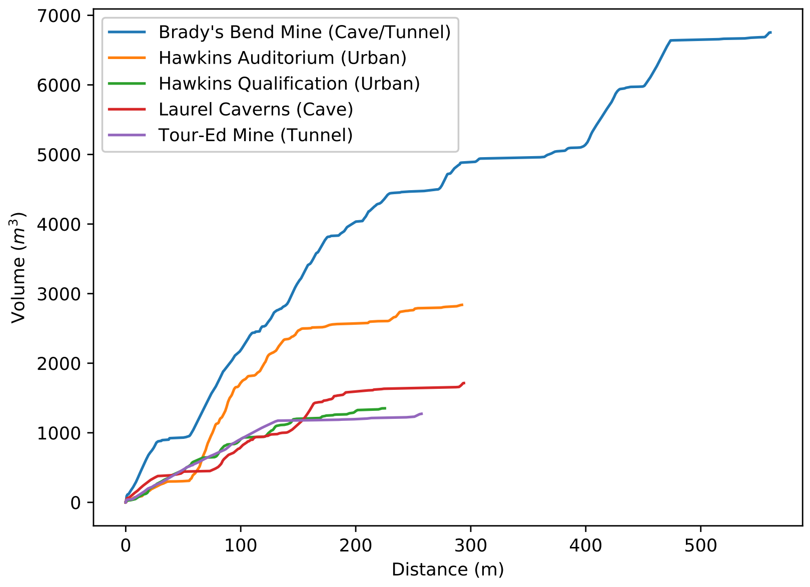
(b)



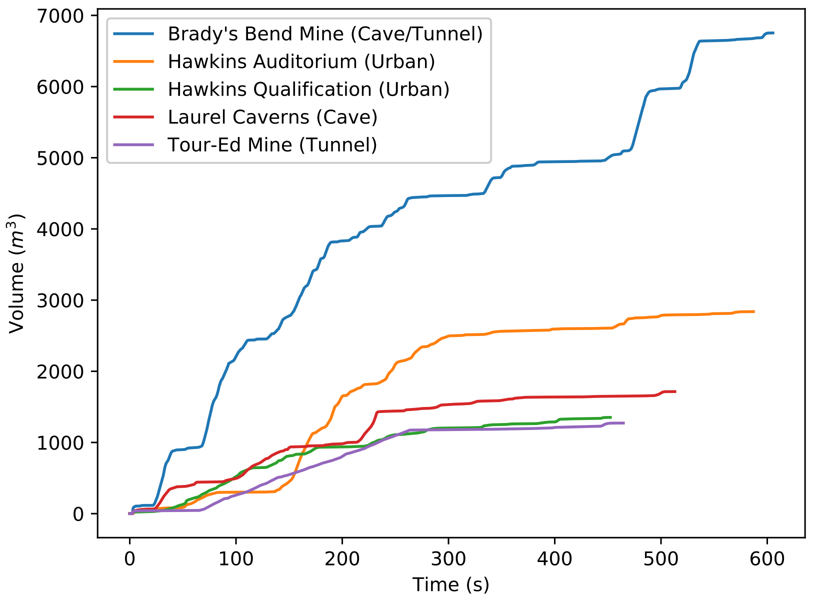
(c)

Figure 5.8: Laurel Caves (UAV2)

Quantitative Metrics Figure 5.9 shows the relative rates of exploration by one UAV across all the five maps. Both figures show the monotonically increasing explored volume as observed by the object detection cameras. The fact that volume explored shows this increasing trend in (Figure 5.9b) and (Figure 5.9a) means that the robot is exhibiting efficient exploration behavior both with respect to time and distance. It's no surprise that Brady's Bend's flight shows the steepest curves. It is the most open and relatively larger space compared to the other environments. The auditorium at Hawkins comes second based on the same argument. Hawkins qualification circuit and the Tour-Ed Mine show the lowest exploration rates. This can be attributed to the narrow passageways that the drone flies through in these environments. We're not able to fully utilize the entire camera field of view in narrow spaces, which leads to relatively lower volumetric exploration rates.



(a) Volume vs. Distance



(b) Volume vs. Time

Figure 5.9: Volume explored during single robot exploration in different environments

Chapter 6

Conclusions

This thesis proposes an integrated exploration and coordination pipeline designed to generalize across a variety of subterranean environments. It comprises of a single robot exploration approach that leverages a longer range sensor to inform the search for frontiers in 3D space. An ensemble of planners is employed to build a resilient system designed to keep the robot in perpetual motion during the exploration phase of the flight. The coordination aspect of the pipeline employs two kinds of maps, one that tracks observed/unobserved volumes, and the second that keeps a skeletal graph (roadmap) tracking globally unexplored frontiers. This ensures the robots minimize the overlap in explored volumes when they have the ability to communicate with each other.

Experiments conducted in simulation, and extensive field tests reflect efficient and robust exploration behavior across environments. It is important to note that there is no parameter tuning done between environments. Additionally, the experiments also show the utility of sharing coordination maps between the agents. They end up exploring different parts of the environment and avoid getting too close to each other.

Having said that, this approach isn't without limitations. The visibility assumption made while sampling viewpoints to each frontier cluster can mean that even though the UAV tracks the viewpoints, there are no guarantees on coverage. The viewpoints by the way they are sampled, can sometimes be close to surfaces, which reduces exploration efficiency. The mapping part of the pipeline does not account for drift or accommodate loop closure adjustments, so severe SLAM drift can pose problems.

These points lead into the further work part of this Chapter, to discuss improvements that can be made to make this system even better.

6.1 Further Work

1. Using a distance map to sample viewpoints. This would ensure the viewpoints are always a threshold distance away from surfaces.
2. Using submaps within the mapping pipeline that would allow for adjustments to each submap during a loop closure event, reducing the effects of SLAM drift.
3. Communications aware planning. A return to home behavior that takes into account the last known good communications location. This way the return to home path can be reduced since the robot does not have to fly all the way back to the initial launch point. Which would allow for a longer exploration flight time.
4. Post processing global plans to maximize information gain. Computing additional yaw angles for the robot to track when flying towards a viewpoint could enhance the volume exploration rate during flight.

Bibliography

[1] Hannah Beech, Richard C. Paddock, and Muktita Suhartono. 'still can't believe it worked': The story of the thailand cave rescue. *The New York Times*, Jul 2018. URL `https://www.nytimes.com/2018/07/12/world/asia/thailand-cave-rescue-seals.html`. (document), 1.1, 1.1

[2] JJ Belwood and RJ Waugh. Bats and mines: Abandoned does not always mean empty, 1991. 1.1

[3] Graeme Best, Jan Faigl, and Robert Fitch. Online planning for multi-robot active perception with self-organising maps. *Autonomous Robots*, 42(4):715–738, 2018. 1.2

[4] Frederic Bourgault, Alexei A Makarenko, Stefan B Williams, Ben Grocholsky, and Hugh F Durrant-Whyte. Information based adaptive robotic exploration. In *IEEE/RSJ international conference on intelligent robots and systems*, volume 1, pages 540–545. IEEE, 2002. 1.2

[5] Wolfram Burgard, Mark Moors, Cyrill Stachniss, and Frank E Schneider. Coordinated multi-robot exploration. *IEEE Transactions on robotics*, 21(3):376–386, 2005. 1.2

[6] Benjamin Charrow, Gregory Kahn, Sachin Patil, Sikang Liu, Ken Goldberg, Pieter Abbeel, Nathan Michael, and Vijay Kumar. Information-theoretic planning with trajectory optimization for dense 3d mapping. In *Robotics: Science and Systems*, volume 11, pages 3–12. Rome, 2015. 1.2

[7] Benjamin Charrow, Sikang Liu, Vijay Kumar, and Nathan Michael. Information-theoretic mapping using cauchy-schwarz quadratic mutual information. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4791–4798. IEEE, 2015. 1.2

[8] Helier Cheung and Tessa Wong. The full story of thailand's extraordinary cave rescue. *BBC*, Jul 2018. URL `https://www.bbc.com/news/world-asia-44791998`. 1.1

[9] Jian Ming Chew. *Assessment of Mesh-Network Performance for Small Aerial Drones in the Urban Environment*. PhD thesis, Monterey, CA; Naval Postgradu-

ate School, 2019. 4

[10] Sanjiban Choudhury, Vishal Dugar, Silvio Maeta, Brian MacAllister, Sankalp Arora, Daniel Althoff, and Sebastian Scherer. High performance and safe flight of full-scale helicopters from takeoff to landing with an ensemble of planners. *Journal of Field Robotics*, 36(8):1275–1332, 2019. 2.1

[11] Timothy Chung. Darpa subterranean (subt) challenge. *DARPA,[Online]. Available: https://www. darpa. mil/program/darpa-subterranean-challenge.[Accessed 27 September 2019]*, 2019. (document), 1.1, 1.3

[12] Micah Corah. Sensor planning for large numbers of robots. *arXiv preprint arXiv:2102.04054*, 2021. 1.2

[13] Dieter Fox, Jonathan Ko, Kurt Konolige, Benson Limketkai, Dirk Schulz, and Benjamin Stewart. Distributed multirobot exploration and mapping. *Proceedings of the IEEE*, 94(7):1325–1339, 2006. 1.2

[14] Kshitij Goel, Wennie Tabib, and Nathan Michael. Rapid and high-fidelity subsurface exploration with multiple aerial robots. *arXiv preprint arXiv:2012.10788*, 2020. 1.2

[15] Kshitij Goel, Micah Corah, Curtis Boirum, and Nathan Michael. Fast exploration using multirotors: Analysis, planning, and experimentation. In *Field and Service Robotics*, pages 291–305. Springer, 2021. 1.2

[16] Gabriel M Hoffmann and Claire J Tomlin. Mobile sensor network control using mutual information methods and particle filters. *IEEE Transactions on Automatic Control*, 55(1):32–47, 2009. 1.2

[17] 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. 2.1

[18] Brian J Julian, Sertac Karaman, and Daniela Rus. On mutual information-based control of range sensing robots for mapping applications. *The International Journal of Robotics Research*, 33(10):1375–1392, 2014. 1.2

[19] Thomas Kollar and Nicholas Roy. Efficient optimization of information-theoretic exploration in slam. In *AAAI*, volume 8, pages 1369–1375, 2008. 1.2

[20] James J Kuffner and Steven M LaValle. Rrt-connect: An efficient approach to single-query path planning. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, volume 2, pages 995–1001. IEEE, 2000. 3.4

[21] Aaron Morris, Dave Ferguson, Zachary Omohundro, David Bradley, David Silver, Chris Baker, Scott Thayer, Chuck Whittaker, and William Whittaker. Recent developments in subterranean robotics. *Journal of Field Robotics*, 23(1):35–57,

2006. 1.1

[22] Ken Museth. Vdb: High-resolution sparse volumes with dynamic topology. *ACM transactions on graphics (TOG)*, 32(3):1–22, 2013. (document), 2.1, 2.1, 2.2

[23] Narcís Palomeras, Natalia Hurtós, Eduard Vidal, and Marc Carreras. Autonomous exploration of complex underwater environments using a probabilistic next-best-view planner. *IEEE Robotics and Automation Letters*, 4(2):1619–1625, 2019. 1.2

[24] Manish Saroya, Graeme Best, and Geoffrey A Hollinger. Online exploration of tunnel networks leveraging topological cnn-based world predictions. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6038–6045. IEEE, 2020. 5.1

[25] Cyrill Stachniss, Oscar Martinez Mozos, and Wolfram Burgard. Efficient exploration of unknown indoor environments using a team of mobile robots. *Annals of Mathematics and Artificial Intelligence*, 52(2-4):205–227, 2008. 1.2

[26] Wennie Tabib, Micah Corah, Nathan Michael, and Red Whittaker. Computationally efficient information-theoretic exploration of pits and caves. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3722–3727. IEEE, 2016. 1.2

[27] Wennie Tabib, Kshitij Goel, John Yao, Mosam Dabhi, Curtis Boirum, and Nathan Michael. Real-time information-theoretic exploration with gaussian mixture model maps. In *Robotics: Science and Systems*, 2019. 1.2

[28] Eduard Vidal, Narcís Palomeras, and Marc Carreras. Online 3d underwater exploration and coverage. In *2018 IEEE/OES Autonomous Underwater Vehicle Workshop (AUV)*, pages 1–5. IEEE, 2018. 1.2

[29] Eduard Vidal, Narcís Palomeras, Klemen Istenič, Nuno Gracias, and Marc Carreras. Multisensor online 3d view planning for autonomous underwater exploration. *Journal of Field Robotics*, 37(6):1123–1147, 2020. 1.2

[30] Wikipedia. Laurel Caverns — Wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=Laurel%20Caverns&oldid=1012427495`, 2021. [Online; accessed 30-July-2021]. 5.2

[31] Brian Yamauchi. A frontier-based approach for autonomous exploration. In *Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97.'Towards New Computational Principles for Robotics and Automation'*, pages 146–151. IEEE, 1997. 1.2, 3.1

[32] Brian Yamauchi. Frontier-based exploration using multiple robots. In *Proceedings of the second international conference on Autonomous agents*, pages 47–53, 1998. 1.2

[33] Luke Yoder and Sebastian Scherer. Autonomous exploration for infrastructure modeling with a micro aerial vehicle. In *Field and service robotics*, pages 427–440. Springer, 2016. (document), 1.2, 3.1, 3.3

[34] Delong Zhu, Chaoqun Wang, Wenshan Wang, Rohit Garg, Sebastian Scherer, and Max Q-H Meng. Vdb-edt: An efficient euclidean distance transform algorithm based on vdb data structure. *arXiv preprint arXiv:2105.04419*, 2021. 2.1