

Systems Article

Resilient and Modular Subterranean Exploration with a Team of Roving and Flying Robots

Sebastian Scherer¹, Vasu Agrawal¹, Graeme Best², Chao Cao¹, Katarina Cujic¹, Ryan Darnley¹, Robert DeBortoli², Eric Dexheimer¹, Bill Drozd¹, Rohit Garg¹, Ian Higgins¹, John Keller¹, David Kohanbash¹, Lucas Nogueira¹, Roshan Pradhan¹, Michael Tatum¹, Vaibhav K. Viswanathan¹, Steven Willits¹, Shibo Zhao¹, Hongbiao Zhu¹, Dan Abad³, Tim Angert¹, Greg Armstrong¹, Ralph Boirum¹, Adwait Dongare¹, Matthew Dworman¹, Shengjie Hu¹, Joshua Jaekel¹, Ran Ji¹, Alice Lai¹, Yu Hsuan Lee², Anh Luong¹, Joshua Mangelson¹, Jay Maier¹, James Picard¹, Kevin Pluckter¹, Andrew Saba¹, Manish Saroya², Emily Scheide², Nathaniel Shoemaker-Trejo¹, Joshua Spisak¹, Jim Teza¹, Fan Yang¹, Andrew Wilson⁴, Henry Zhang¹, Howie Choset¹, Michael Kaess¹, Anthony Rowe¹, Sanjiv Singh¹, Ji Zhang¹, Geoffrey A. Hollinger² and Matthew Travers¹

¹Carnegie Mellon University

²Oregon State University

³Schlumberger

⁴CNH Industrial

Abstract: Subterranean robot exploration is difficult, with many mobility, communications, and navigation challenges that require an approach with a diverse set of systems, and reliable autonomy. While prior work has demonstrated partial successes in addressing the problem, here we convey a comprehensive approach to address the problem of subterranean exploration in a wide range of tunnel, urban, and cave environments. Our approach is driven by the themes of resiliency and modularity, and we show examples of how these themes influence the design of the different modules. In particular, we detail our approach to artifact detection, pose estimation, coordination, planning, control, and autonomy, and we discuss our performance in the tunnel, urban, and self-organized cave circuits of the DARPA Subterranean Challenge. Our approach led to a winning result in the tunnel circuit, and placing second in the urban circuit event. We convey lessons learned in designing and testing a resilient system for subterranean exploration that can generalize to a large range of operating conditions, and potential improvements for the future.

Received: 15 January 2021; revised: 7 May 2021; accepted: 20 June 2021; published: 6 May 2022.

Correspondence: Sebastian Scherer, Carnegie Mellon University, Email: basti@andrew.cmu.edu

This is an open-access article distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Copyright © 2022 Scherer, Agrawal, Best, Cao, Cujic, Darnley, DeBortoli, Dexheimer, Drozd, Garg, Higgins, Keller, Kohanbash, Nogueira, Pradhan, Tatum, Viswanathan, Willits, Zhao, Zhu, Abad, Angert, Armstrong, Boirum, Dongare, Dworman, Hu, Jaekel, Ji, Lai, Lee, Luong, Mangelson, Maier, Picard, Pluckter, Saba, Saroya, Scheide, Shoemaker-Trejo, Spisak, Teza, Yang, Wilson, Zhang, Choset, Kaess, Rowe, Singh, Zhang, Hollinger and Travers

Keywords: aerial robotics, subterranean robotics, extreme environments, exploration, wheeled robots

1. Introduction

Succeeding in the DARPA Subterranean Challenge (SubT Challenge) (DARPA, 2019) requires operation in austere, dynamic terrain with degraded conditions for sensing and communications. In addition to autonomously generating maps and recognizing objects, the tasks need to be completed in limited time. This scenario is representative of many real-life scenarios that are dangerous for humans but where robots can be of immense value, such as in search and rescue, building clearing, and mine disaster situations.

Since by the nature of the challenge the robots will encounter novel situations, it is important for the system to be resilient in the face of significant uncertainty, and for the overall team to be able to adapt quickly to the nature of the terrain. While prior research had demonstrated partial success, it has not addressed all of the challenges that are present in a real-life situation as it is emulated in the SubT challenge.

Autonomous exploration in subterranean environments is a well-studied problem in field robotics. Early successful work in this area includes mapping of mine environments (Baker et al., 2004; Morris et al., 2006; Thrun et al., 2004), planetary exploration (Husain et al., 2013; Agha et al., 2019), and disaster response (Nagatani et al., 2013; Markoff, 1991).

Most research focuses on perception, localization, and exploration strategies for subterranean navigation (Morris et al., 2006; Shaffer et al., 1992). Various locomotion modalities for these environments have been studied, including wheeled locomotion (Löscher et al., 2018; Ferguson et al., 2003), aerial flight (Li et al., 2018; Papachristos et al., 2019), and quadrupedal locomotion (Bouman et al., 2020; Miller et al., 2020). Even wireless communication systems for autonomous navigation in mine environments have been studied (Yarkan et al., 2009).

Research in the field of subterranean robotics has recently been accelerated by the DARPA Subterranean Challenge. Several of our peer teams have published part of their approach to subterranean autonomy challenges. There have been substantial steps forward in localization and mapping, including LiDAR-based methods (Ebadi et al., 2020) and Visual-Inertial methods (Kramer et al., 2019; Khattak et al., 2020). Furthermore, teams have also presented various approaches to exploration of unknown environments (Rouček et al., 2019; Dang et al., 2019). One of the most impactful areas of research from this challenge is the work on long-duration resilient and robust autonomy, including for legged vehicles (Miller et al., 2020; Bouman et al., 2020) and unmanned aerial vehicles (Santamaria-Navarro et al., 2020; Rouček et al., 2019). While this prior work shows valuable component contributions, here we present a holistic approach to address the challenge with a unified approach.

Our unified approach to the challenge is driven by two key themes: modularity and resilience. Since the environments vary greatly, no single robot or sensing system is most successful in all situations. On the other hand, variation in the systems increases system complexity. We apply ideas that we call modular autonomy to manage complexity and increase adaptability. Modular autonomy spans a spectrum from low-level hardware (tracks, limbs, rotors, sensors, and power systems) and software elements (odometry, mapping, and recognition) to high-level algorithms for reasoning and coordination.

The second theme is to push the resilience (Hollnagel, 2011) of the overall system. We strive towards a resilient robotic system by making our robots robust from a mechanical, software engineering, and algorithm perspective. We purposely implement redundant systems consisting of multiple robots, and we use redundant algorithms within the system. We enforce resourcefulness by enabling the robots to communicate and coordinate with each other, and our basestation human interface is designed to leverage the creativity of the operator.

While one can measure the performance of the overall system, it is difficult to specifically attribute the success to our design goals or specific components. Nevertheless, we look for certain traits in

the system: For example, can it respond to different environments, does it know what to monitor to avoid getting itself into trouble, can it adapt if it observes failures, and can it anticipate difficult and untested situations?

We use these two key themes—resiliency and modularity—as guiding principles in the design of the system, robots, software, and algorithms. In the paper, we present a detailed description of our specific approach and system design for subterranean exploration, and we elaborate on our successes, failures, and lessons learned during extensive system testing in a wide range of subterranean environments. Specifically, our system features the following demonstrated capabilities:

1. Efficient mobility with a variety of platforms;
2. Maximized mission success and minimized time to completion using novel methods to coordinate decentralized heterogeneous platforms that collectively adapt their behaviors based on discovered conditions;
3. Accurate 3D mapping in degraded conditions with multimodal sensing in a simultaneous localization and mapping (SLAM) framework designed for long-duration missions;
4. Rapid differentiation of objects of interest with data-efficient segmentation and classification of objects and terrains; and
5. Robust relay of maps in a radio-adverse environment with distributed algorithms that build an ad-hoc communication network.

2. The Challenge

A successful solution of the challenge requires mapping, navigation, and exploration of subterranean environments, from naturally occurring to manmade. To seamlessly work in every possible subterranean environment, a solution needs to simultaneously address all of these difficulties that are all present in the DARPA SubT challenge (DARPA, 2019):

1. Austere navigation: GPS-less and sparsely featured surroundings of varying scales,
2. Degraded sensing: Low-light, obscured, and scattering conditions,
3. Severe communication: Physical impediments to reliable links,
4. Dynamic terrain: Physical changes to the environment, slippery and deformable terrain,
5. Endurance limits: Large spatial scales of several kilometers of traversal, and
6. Terrain Obstacles: Mobility-stressing terrain features, wires, and steps.

DARPA additionally enforces that only a single operator may operate the team of robots, and the mission must be completed in 60 min. Additionally, artifacts must be detected and localized in a DARPA-specified reference frame with an accuracy of at least 5 m. The goal of the challenge is to detect and locate the largest number of artifacts with the correct class and communicate this information to DARPA within the given time limit.

3. Approach

3.1. Overall Architecture

To manage the complexity of a large number and diverse set of robots, our approach is to split the system into a maximum set of common components that are the same for each system, and a minimum of type-specific components that depend on the type of the system. We further reduce complexity in our system by sharing the same or a very similar sensor and computing hardware configuration in each of the autonomy payloads.

To maximize resilience of the overall system, we provide information and interaction ability for our single human operator while minimizing communication requirements. This type of setup requires the system to have the ability to have “sliding autonomy” (Dias et al., 2008; Heger and Singh, 2006), which enables full autonomy but also allows for intervention if required.

Internally, the system interacts within and between robots with a message-subscribe paradigm: ROS 1 (Stanford Artificial Intelligence Laboratory et al., 2018) within a robot and DDS (Pardo-Castellote, 2003) between robots. Communication between robots is tightly regulated and degrades gracefully with reduced link capacity.

On the high level, data flow from a set of visual, thermal, radio, gas, light detection and ranging (LiDAR), and inertial, as well as proprioceptive sensors through a set of processing components to the vehicle-specific actuators and motors. For example, the roving robots have a motor for each wheel as well as release latches for the communication nodes. Data are filtered and accumulated to create and merge several maps that are analyzed in real time to decide traversability and find obstacles. These maps are used by the planning algorithms to decide the best course of action to find objects and maintain communication with the basestation.

Additionally, a mission executive is used on each robot to arbitrate different behaviors. The behavior executive triggers actions, such as explore, return home, recover, and drop a communication node. The executive is implemented as a behavior tree (Colledanchise and Ögren, 2018), which gives us a scalable and quick approach to model, modify, and diagnose the behavior of the robot.

Figure 1 shows our high-level system architecture. In the following, we describe the key components that are critical in achieving resilient exploration and artifact detection in the challenge.

3.2. Perception

3.2.1. Object Detection

While previous perception architectures have been designed for subterranean exploration, such approaches typically do not enable the types of rapid situational awareness required by the SubT Challenge. For example, (Ferguson et al., 2003) developed an early approach that produced high-quality two-dimensional (2D) maps, however such maps were not able to be relayed back to the human operator in real time.

More recently, (Wang et al., 2014b) developed a robotic system that relays video data back to a human operator in real time via a tether. While this provides real-time information, given our multirobot configuration, monitoring and analyzing video feeds from multiple robots simultaneously would be extremely straining cognitively to the single human operator. Additionally, the poor lighting in underground environments often precludes the use of solely an optical camera. Additionally, the SubT Competition environments often include fog machines in certain areas of the course, which further motivate the use of nonoptical sensing modalities.

To solve these challenges and enable rapid situational awareness, our perception system centers around two key principles:

1. Information must be intelligently distilled for a single human operator monitoring a large system of robots. Due to communication constraints and the necessity for real-time analysis by the human operator, processing therefore must be done in real time onboard each robot.
2. Due to the poor lighting and large variety of types of artifacts to detect, our system must be capable of detecting artifacts where RGB (red, green, blue) cameras fail.

To follow these principles, we develop a system that is efficient and scalable to large numbers of robots and artifacts. In this section, we first describe our visual detection approach and then cover detection of nonvisual artifacts. Next, we use the Tunnel Circuit as a case study for our architecture, and finally we finish with a discussion on our improvements since that circuit. Additional details on our hardware and software systems for object detection can be found in (Agrawal, 2019).

Visual detection. For perceiving visual artifacts on both our ground and aerial vehicles, we primarily used RGB cameras due to their high resolution and informative output. On our ground vehicles we also leveraged thermal cameras, which were especially useful during fog or dust-filled scenarios, as illustrated in Figure 2.

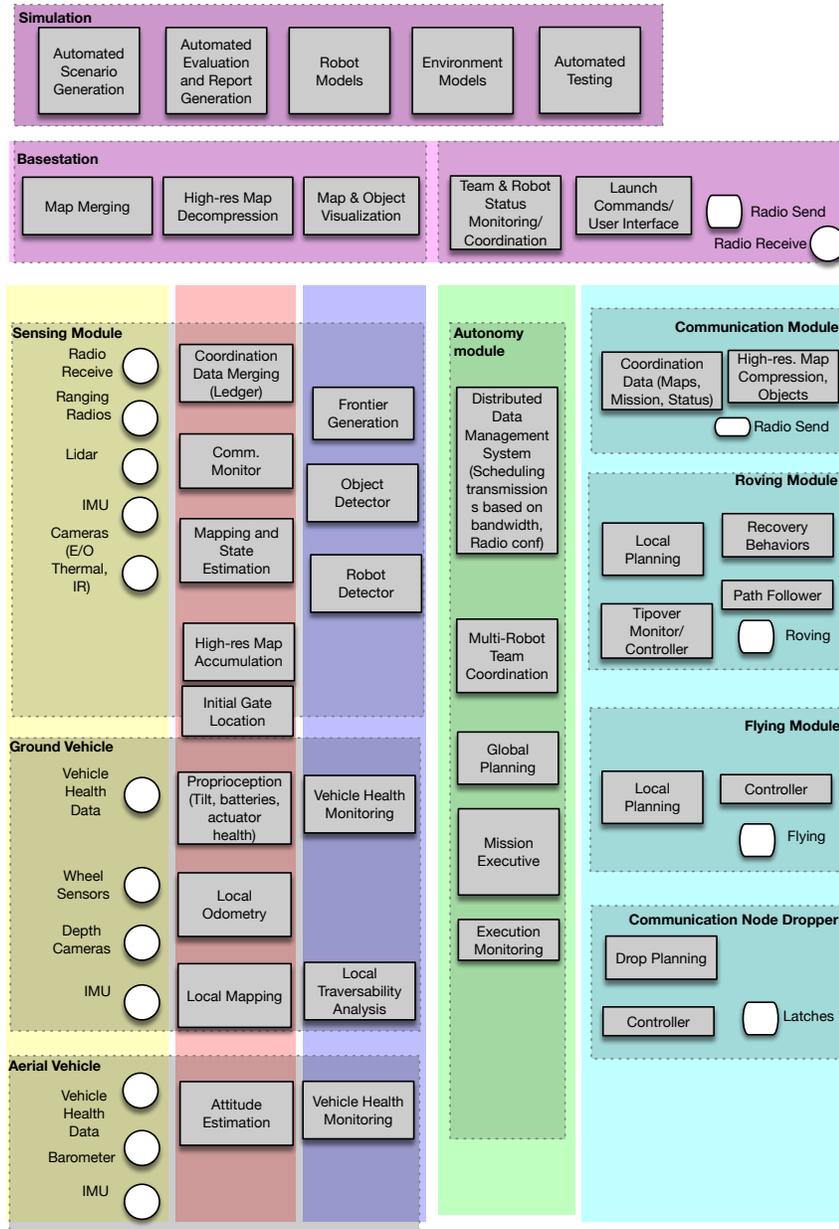


Figure 1. Overall system architecture.

Due to their state-of-the-art performance in terms of speed and detection ability, we used convolutional neural networks (CNNs) to process incoming imagery onboard each vehicle (Abadi et al., 2016). Given our limited onboard compute, the large amount of incoming camera imagery, and the requirement for real-time detection, we evaluated a number of lightweight CNN architectures such as *ssd_inception_v2* (Liu et al., 2016; Szegedy et al., 2016) and *ssd_mobilenet_v1* (Liu et al., 2016; Howard et al., 2017). Final networks were chosen based on their performance on a held-out test set of data gathered by our robot during practice runs. A visual detection of the survivor artifact during the tunnel circuit can be seen in Figure 3.

For the tunnel and urban circuits we utilized *ssd_mobilenet_v1* for both RGB and thermal detections. During our internal cave circuit we utilized the larger and more stable *ssd_inception_v2*



(a) RGB image of the scene. Note the drill is visible in the center of the image, but the human and drone are obfuscated by the fog.

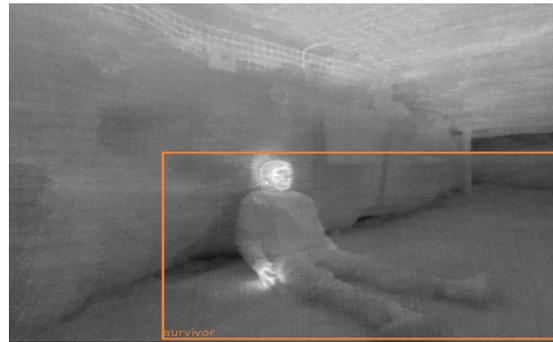


(b) Thermal image of the scene. Note the human and drone are prominently displayed but the drill is no longer visible.

Figure 2. Example of the complementary nature of RGB and thermal cameras in Tour-Ed Mine in Tarentum, PA, USA. The drill is visible in the RGB image (left), the human and drone are visible in the thermal image (right).



(a) Detection of survivor in RGB imagery.



(b) Detection of survivor in thermal imagery.

Figure 3. Multimodal detection of survivor artifact during the Tunnel Circuit. Each type of imagery is processed by a dedicated CNN onboard the vehicle.

architecture for RGB detections. Due to development-time constraints, we treated both RGB and thermal detection similarly, and we developed separate CNNs for each. Additionally, data were gathered and labeled for each modality in a similar manner. In the future, it would be interesting to explore methods for fusing these modalities in a manner such as (Agrawal and Subramanian, 2019). This could potentially enable better leveraging of their complementary capabilities and reduce the number of CNNs required to run on our vehicle.

Eventual deployment in an unknown environment required our CNN to be relatively environment- and appearance-robust, thus motivating the generation of a large training dataset. To generate training data, we used hand-collected data and images from practice runs with our vehicles. Hand-collecting images provided a way to quickly capture dense datasets of artifacts, while the imagery from practice runs allowed us to capture the motion-blur and lighting characteristics for cameras onboard the robot.

At runtime, once artifacts were detected in 2D, we projected these detections into 3D using a custom *depth combiner* module. Due to the limited range of the Realsense depth images (3 m), the depth combiner module combined information from the Realsense depth camera and the LiDAR.

The input to the depth combiner was two preliminary depth images: one produced directly by the Realsense camera (Figure 4b) and one we generated from the LiDAR (Figure 4c). To generate the

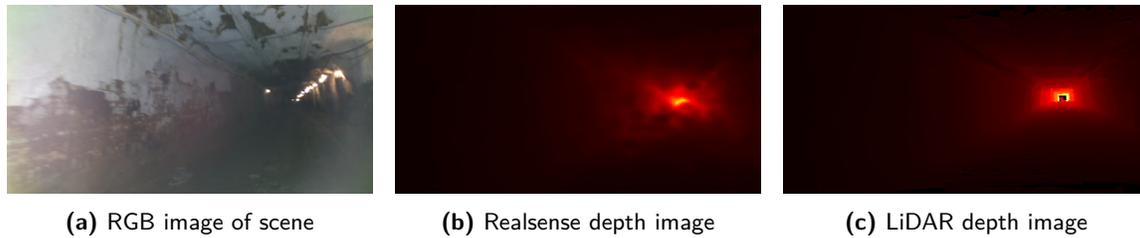


Figure 4. To transform 2D detections into 3D localizations, we combine information from the Realsense depth images (useful at ranges less than 3 m) and the LiDAR point clouds (useful from 3 to 100 m). For long ranges, as can be seen above, the LiDAR point clouds are more accurate.

LiDAR depth map, we used the known transform between the LiDAR and the camera to project the LiDAR points into the field of view of the RGB camera in a pixelwise manner. To generate the output final depth image from the depth combiner, if the corresponding depth pixel from the Realsense was less than 3 m away, we used the Realsense depth image, otherwise we used the range from the LiDAR depth image.

By utilizing depths from both the LiDAR and the Realsense, the depth combiner effectively leveraged the different fields of view for each sensor. For example, due to its narrow vertical field of view, the LiDAR often could not see directly to the side of the vehicle. In such scenarios, the Realsense depth map provided the localization points for the 2D detections. After the depth combiner produced an accurate pixelwise depth map, the 2D detection was projected pixelwise into a 3D point cloud. The centroid of this point cloud was then taken to be the artifact location.

Finally, to leverage information across images at different time steps or from different cameras, we clustered 3D detections within 0.5 m of each other. To reduce false positive detections, for an artifact to be sent to the basestation we required at least three different observations to be made. These observations could come from different modalities (e.g., thermal or RGB), time steps, or cameras.

Nonvisual detection. We classified the cell phone and gas artifacts as primarily “nonvisual” artifacts. While we included the cell phone artifacts as a category in our thermal and RGB CNNs, we found detecting their presence in such imagery to be especially difficult, and therefore we developed approaches for detecting them in a nonvisual manner as well.

To detect the cell phone artifact, we used the *wpa_cli* Linux utility to measure the received signal strength indicator (RSSI) between the phone’s WiFi hotspot and our robot. We found the RSSI values to contain significant noise, which precluded the use of standard log distance models for localization. Therefore, we defined the robot’s position at the maximum received RSSI to be the location of the cell phone. In confined environments, such as those found in the Tunnel Circuit, this worked well, however this approach does require the robot to get close to the cell phone to be accurate. The human operator had the ability to refine these location estimates based on contextual information provided by the communicated maps.

The gas sensor was handled in a similar way: we took point-readings as the robot traversed the environment and considered readings above 2000 parts per million (ppm) to correspond with gas-leak artifacts. This approach worked well on our ground vehicles; however, on our aerial vehicles the turbulence from the propellers prevented this from being an accurate approach. In the future, we will develop more sophisticated adaptive-thresholding approaches, as well as consider hardware changes, to enable accurate gas detection on our aerial vehicles.

Tunnel Circuit case study. To investigate the benefits of our multimodal approach, we analyzed data from a single ground vehicle during our deployment in the Tunnel Circuit Safety Research Mine. We first examined the number of artifacts detected and scored using each individual sensor and compared this with the score of our entire system. The results are shown in Figure 5. The

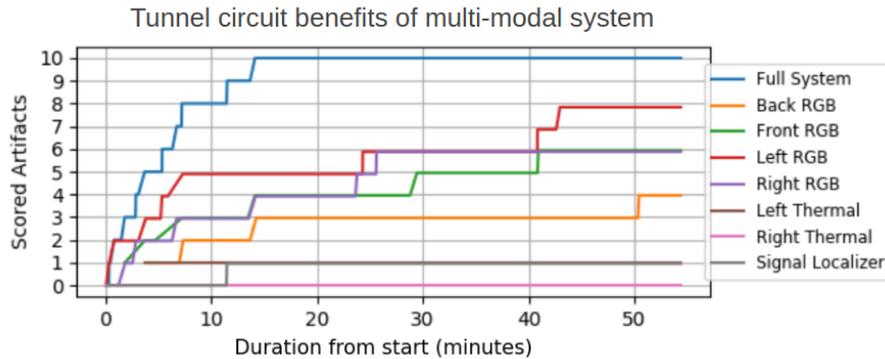


Figure 5. Using a multimodal system (blue) achieves a greater score than any individual sensor, thus motivating the design of our robust hardware/software object detection system. Since most artifacts could be detected with visual sensors during this portion of the run, the RGB cameras contributed most to our score. However, the thermal and WiFi scanner (“Signal Localizer”) also provided complementary observations of artifacts. Due to the placement of artifacts often in areas reachable by a human, the front, left, and right RGB cameras scored more artifacts than the backwards RGB camera, which was tilted upwards and used to mainly view the ceiling of the tunnel.

outcome is intuitive: leveraging the RGB, thermal, and signal detection methods together enabled a higher-scoring system than any single modality.

From this Tunnel Circuit deployment we also noticed two key areas for improvement. First, even though our thermal cameras detected one survivor artifact, it missed another survivor passed by the robot. This could be due to the fact that the resolution of our thermal camera was lower than the RGB camera or the fact that we had limited thermal training data for the thermal CNN. Secondly, we noticed that we received RSSI measurements from three cell phones during this run, but only scored one of them. This could be due to the fact that we did not get close enough to the phones for our robot-centric localization strategy to work.

Urban Circuit. To improve on our Tunnel Circuit results, for the Urban Circuit we focused on data collection and labeling for the new artifacts as well as improved cell phone detection. To rapidly generate a larger amount of training data, we developed a data augmentation pipeline that included flipping, translation, varying image brightness, and scaling images. In addition to training with real images, we also programmatically generated a large synthetic dataset using Microsoft AirSim (Shah et al., 2018).

Before the Urban Circuit, we also explored experimentally validating a physical RSSI-to-range model to make our cell phone localizations more accurate. However, we found similar to previous work (Assayag et al., 2020) that standard models like the original log-loss do not transfer well to real environments due to varying environment geometry and surface roughness. We characterize the most relevant challenges with RSSI localization in the context of subterranean environments as:

- High Signal Variance: RSSI values changed frequently even when a robot is at the same location relative to the cell phone.
- Fading Problem: The cell phone could be nearby but not detected due to multipath signal fading (corners, occlusion, stairwell/elevator, high-ceilings, etc.).
- Hardware Interface Reliability: During field testing, we would notice periodic inconsistencies using the Python ‘rssi’ library and the Linux utility ‘wpa-cli’ to measure RSSI.

Due to these challenges, for the urban circuit we maintained our tunnel circuit cell phone detection approach that just used the robot position when the maximum RSSI value was received as the location of the cell phone.

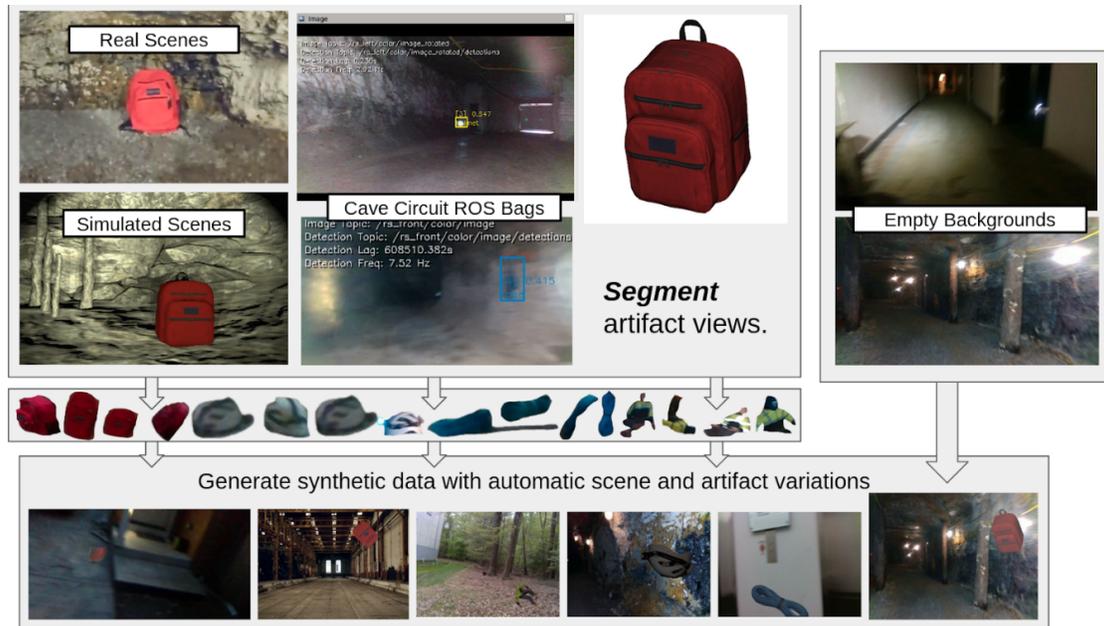


Figure 6. An overview of our segment-and-insert pipeline to promote diversity during data augmentation. Starting in the top left, a set of artifacts from real images are cut out and then inserted into other scenes (bottom row). Before being inserted, data augmentation such as varying brightness or scaling can occur on the object itself.

During the Urban Circuit, we experienced two main challenges. First, our visual CNNs did not perform as well as expected. This was likely due to the increased range required for visual detection from the Tunnel to Urban Circuits and the limited time we had to collect and label new artifact data between the two circuits. In addition, after artifacts were detected on the robot, we had some communication problems relaying this information back to the basestation. These examples demonstrate the importance of system resilience and robustness, which we will continue to improve on leading into the next stages of the SubT Challenge.

Self-Organized Cave Circuit. To address the two main challenges from the Urban Circuit, we focused on developing a more powerful data generation pipeline and a more robust data communication pipeline back to the basestation.

To generate data quickly while also improving the diversity of the data, we first implemented a segment-and-insert approach for data augmentation, as illustrated in Figure 6, inspired by (Dwivedi et al., 2017). We first take full images of artifacts and “cut out” the artifacts from the scene. After amassing a large set of these cutouts, we can insert them into other scenes at random locations, thus reducing location bias in our detection CNNs. Additionally, the cutouts themselves can be augmented by scaling, rotating, blurring, or varying the brightness of the object before insertion. This augmentation enabled CNNs to be trained on objects of various poses and of various appearances.

In addition to boosting our data augmentation pipeline, we also increased the speed at which we could label real images. Prior to the Cave Circuit, we hand-labeled every image of artifacts across a variety of videos which was extremely slow. To speed up this process, we implemented a tracker in our data labeling pipeline. Specifically, after a human labeler identified the artifact in a single image, we used OpenCV (Bradski and Kaehler, 2000) to track objects automatically throughout the remainder of the video being labeled. This resulted in the ability to generate artifact labels from real images at $5\times$ the previous rate. Our code for this tracked version is available here: <https://github.com/debortoli/VideoLabeling>.

To address the communications challenges during the urban circuit, we fully evaluated our artifact detection and communications interface and noticed that even when a specific vehicle did not have connectivity directly to the basestation, it was possible to be connected to another robot. In one instance specifically, our drone was able to connect with all three of our ground vehicles simultaneously (including R1, which was cut off from the basestation due to a communications failure). To enable this functionality, we developed an “artifact ledger” for more reliable communication and data muling between robots. We cover this ledger in more detail in Section 3.3.2.

3.2.2. Super Odometry

State estimation is a fundamental component to establish the relative position to the start location. Until now, we have utilized LOAM (LiDAR Odometry and Mapping) in the challenge, which is detailed in (Zhang and Singh, 2014), and we are now using a new method called Super Odometry as outlined below.

Simultaneous Localization and Mapping (SLAM) systems need to achieve robust performance for long periods of time in various environments. An ideal SLAM system should include fail-safe and failure-aware operation modes. For example, the system needs to predict the risk of imminent failure due to degraded environments and hardware failures. Recently, multimodal sensor fusion such as tightly coupled methods (Ye et al., 2019) and loosely coupled methods (Zhang and Singh, 2018; Zhao et al., 2019) have become the main solution for autonomous mobile robots to overcome challenging environments. However, we argue that both tightly coupled methods such as factor-graph optimisation and loosely coupled methods such as Kalman filters are not the optimal choice for robust performance in challenging environments because there are several inherent limitations. First, most of the loosely coupled fusion methods are not accurate and robust, while tightly coupled fusion methods are typically difficult to extend to other sensors and to recover from imminent failures. Secondly, most of the tightly coupled and loosely coupled methods adopt one modality such as a LiDAR or camera as the dominant sensor, which makes them vulnerable in various environments because they are environment-dependent sensors.

Overview. To overcome these limitations, we propose a general inertial measurement unit (IMU)-based data processing pipeline named Super Odometry (Zhao et al., 2021), which combines the advantages of loosely coupled methods with tightly coupled methods. Since it is environment-independent and robust to outliers, it is significantly more robust than LiDAR (Zhang and Singh, 2014) or visual/thermal (Qin et al., 2018; Khattak et al., 2019; Zhao et al., 2020)-based SLAM methods.

The proposed Super Odometry data processing pipeline is shown in Figure 7. The system design follows a key insight: An IMU produces smooth measurements with noise but few outliers, which can make its estimate very accurate as long as the bias drift can be well-constrained by other sensors. Therefore, Super Odometry is designed around the IMU as the primary sensor. However, it is not an IMU-only odometry. Instead, it makes use of constraints provided by other methods such as visual/thermal odometry and LiDAR odometry. However, naively using all these constraints before fusion will lead to unreliable state estimation. Therefore, Super Odometry adopts a novel selective fusion strategy that automatically selects the most informative constraints and rejects incorrect data association based on the reliability of measurements. Meanwhile, the constrained IMU odometry provides a prediction to the visual/thermal and laser odometry, which recovers motion in a coarse-to-fine manner and significantly increases real-time performance. Most importantly, this pipeline can easily recover even if one of the sensors is suffering imminent failure such as intermittent data.

Methodology. Since Super Odometry does not combine all sensor data into a full-blown factor graph, instead it divided the big factor graph into several “subfactor-graphs.” For example, IMU Odometry, Visual Odometry, and Laser Odometry are organized as “subfactor-graphs.” When we examine the “subfactor-graph” itself, it is a tightly coupled method since it processes the sensor raw

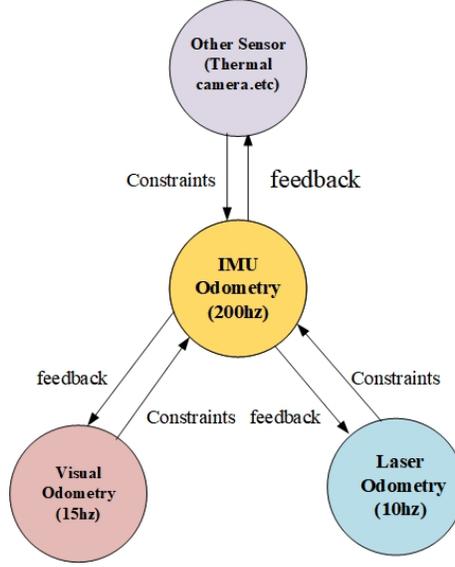


Figure 7. The high-level structure of the Super Odometry approach to SLAM.

measurements. However, when we examine the relation between “subfactor-graph,” it is a loosely coupled method since it only transmits the pose constraints to each other. Here we will briefly introduce three types of sub-odometry-factor-graph, namely (a) IMU; (b) LiDAR-inertial; and (c) Visual-inertial odometry factors. For more details, please refer to (Zhao et al., 2021).

For IMU Odometry Factor Graph, the states from visual-inertial odometry and laser-inertial odometry are added to the graph and get connected with Combined IMU Factors and Pose Between Factors (Dellaert and Kaess, 2017). The nonlinear optimization of IMU Odometry is given by

$$E = \sum_{(i,j) \in \mathcal{B}} \mathbf{e}_{ij}^{LO\top} W_{ij}^{-1} \mathbf{e}_{ij}^{LO} + \sum_{(i,j) \in \mathcal{B}} \mathbf{e}_{ij}^{IMU\top} W_{ij}^{-1} \mathbf{e}_{ij}^{IMU} + \sum_{(i,j) \in \mathcal{B}} \mathbf{e}_{ij}^{VO\top} W_{ij}^{-1} \mathbf{e}_{ij}^{VO} + E_m, \quad (1)$$

where the terms E_m , \mathbf{e}_{ij}^{IMU} , \mathbf{e}_{ij}^{LO} , and \mathbf{e}_{ij}^{VO} represent the residuals of marginalization prior factor, IMU preintegration factors, and a pose between factors from laser odometry and visual odometry. Meanwhile, W_{ij} represents the covariance matrix between frame i and frame j .

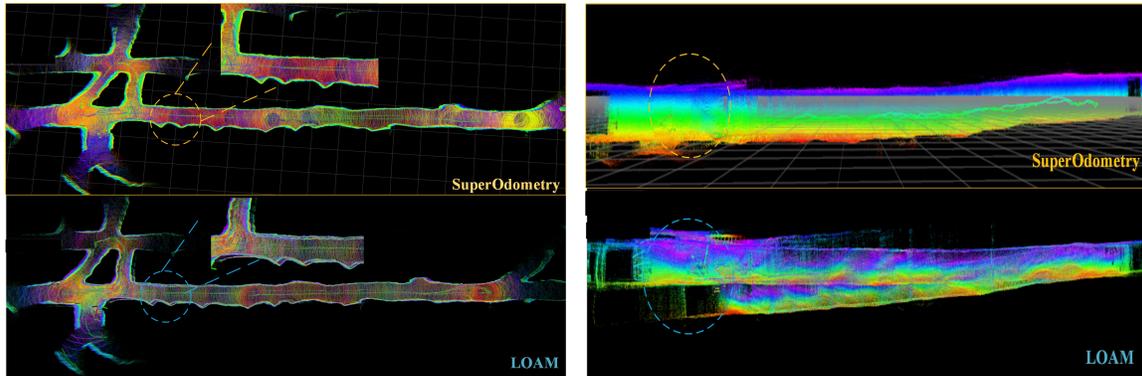
For Laser-inertial Odometry Factor Graph, we use IMU preintegration factor as motion prediction for current scan-map matching, and we adopt the Levenberg-Marquardt method to solve for the optimal transformation by minimising:

$$\min_{\mathbf{T}_{i+1}} \left\{ \sum_{\mathbf{p} \in \mathbb{F}_{i+1}^{po}} w_i \mathbf{e}_{i+1}^{po \rightarrow po} + \sum_{\mathbf{p} \in \mathbb{F}_{i+1}^{li}} w_i \mathbf{e}_{i+1}^{po \rightarrow li} + \sum_{\mathbf{p} \in \mathbb{F}_{i+1}^{pl}} w_i \mathbf{e}_{i+1}^{po \rightarrow pl} + \sum_{(i,i+1) \in \mathcal{B}} w_i \mathbf{e}^{imu} + \mathbf{e}_{imuodom}^{prior} \right\}, \quad (2)$$

where $\mathbf{e}_{i+1}^{po \rightarrow po}$, $\mathbf{e}_{i+1}^{po \rightarrow li}$, and $\mathbf{e}_{i+1}^{po \rightarrow pl}$ are the point-to-point (line, plane) distance, and their correspondence sets are represented as $\mathbb{F}_{i+1} = \{F_{i+1}^{po}, F_{i+1}^{li}, F_{i+1}^{pl}\}$. To overcome geometrically degraded environments, the predicted pose constraints $\mathbf{e}_{imuodom}^{prior}$ and IMU preintegration factor \mathbf{e}^{imu} from IMU odometry are used in the optimization. w_i represents the information matrix, and $\mathbf{T}_{i+1} = \{\mathbf{R}, \mathbf{t}\}$ is the optimal transformation we expect to estimate.

For Visual-inertial Odometry Factor Graph, we add visual reprojection factor e_{reproj} , IMU preintegration factor e_{imu} , marginalization factor e_m , and a pose prior form IMU odometry $\mathbf{e}_{imuodom}^{prior}$ in the subfactor-graph,

$$\min_{\mathbf{T}_{i+1}} \left\{ \sum_{i \in \text{obs}(i)} \mathbf{e}_{reproj}^T W_{reproj}^{-1} \mathbf{e}_{reproj} + \sum_{(a,b) \in \mathcal{C}} \mathbf{e}_{imu}^T W_{imu}^{-1} \mathbf{e}_{imu} + e_m + \mathbf{e}_{imuodom}^{prior} \right\}, \quad (3)$$



(a) Top-down map comparison of Super Odometry and LOAM. (b) Side-view map comparison of Super Odometry and LOAM.

Figure 8. Comparing Super Odometry and LOAM in a cavelike environment. The dashed circles highlight major differences in the map.

Table 1. Accuracy evaluation of LOAM and Super Odometry.

Datasets	ATE (in m) Transl. RMSE		ATE (in m) Transl. MEAN		ATE (in m) Transl. MAX	
	LOAM	Super Odometry	LOAM	Super Odometry	LOAM	Super Odometry
Constrained environments	0.36	0.18	0.31	0.29	0.77	0.62
Aggressive motion	0.29	0.19	0.24	0.20	1.01	0.97
Long corridor	0.74	0.53	0.59	0.22	1.73	1.14

where $\mathbf{obs}(\mathbf{i})$ is a set that contains tracked features, and \mathbf{C} is a set that contains pairs of visual nodes (a, b) connected by IMU factors. W_{reproj} and W_{imu} are covariance matrices for visual reprojection factor and IMU preintegration factor.

Performance of Super Odometry. We have compared Super Odometry with the LOAM algorithm in cavelike environments. Figure 8a shows the mapping result of the Super Odometry and the LOAM algorithm in the X and Y directions. It can be seen that Super Odometry can provide a more accurate map, which indicates that the pose estimation from Super Odometry is more accurate. Figure 8b shows the mapping result of Super Odometry and the LOAM algorithm in the Z direction. It can be seen that the mapping result from the LOAM algorithm has a big drift, while Super Odometry can provide a more precise mapping result. Table 1 lists the accuracy evaluation of LOAM and Super Odometry in some challenging scenarios including constrained environments, aggressive flight, and a long corridor. These results indicate that Super Odometry overall performs better than LOAM for a SubT-like environment. More experimental results can be found at <https://sites.google.com/view/superodometry>.

3.2.3. Gate Calibration

To correctly report artifacts and map updates, each robot must calibrate its map frame to the DARPA-defined reference frame. An optimal strategy for exploring an unknown environment is implemented during a competition run, so all robots are not released simultaneously. Naively calibrating every robot before releasing one requires significant amounts of time and resources. Furthermore, a decision on the type of robot to release at a given time must be flexible. The basestation operator's time is an essential resource to conserve throughout the exploration mission, so ideally, the operator's effort on calibration should be minimized. Thus, we focus on providing

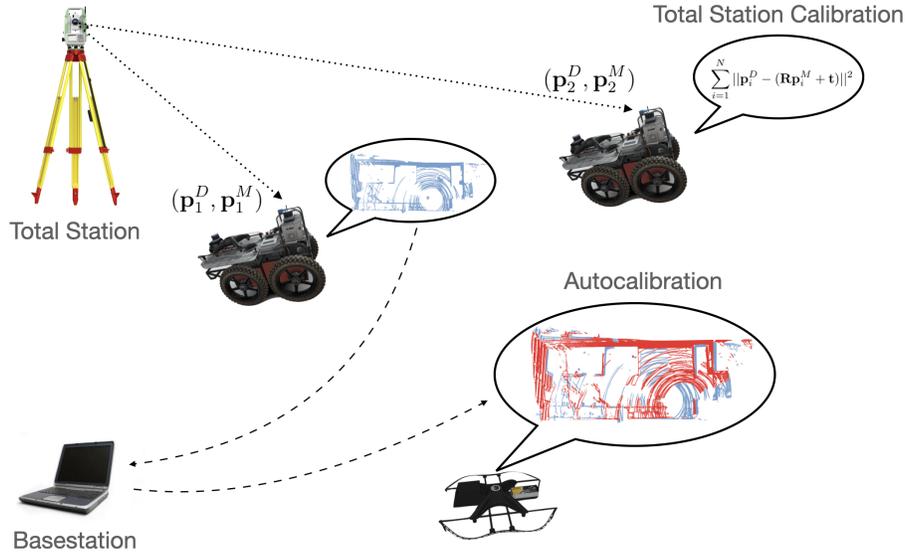


Figure 9. Overview of gate calibration methods. At least one robot is calibrated using the total station via least-squares registration, while also sharing a local map to the basestation. Subsequent robots may request this map from the basestation and perform alignment with respect to its own local map.

algorithms with minimal intervention that scale to large-scale teams. Our procedure consists of two components: total station calibration and an automatic calibration procedure. An overview of this is shown in Figure 9.

Total Station Calibration. Total Stations (TS) are a high-precision optical instrument commonly used for surveying applications. In the robotics community, TS are typically used to collect ground truth trajectories for evaluating state estimation (Burri et al., 2016; Wisth et al., 2020). With regard to multirobot exploration in unknown environments, (Ebadi et al., 2020) integrates TS measurements directly into a pose graph. While this can improve global trajectory estimates, this section is focused on first aligning each robot’s map frame to a global frame, which is required to start sharing information in a common frame. Therefore, the approaches are complementary.

When a robot is turned on, it anchors its map frame and starts the localization and mapping modules. As mentioned previously, the relative pose between the map frame, M , and the world frame, D , must be known for global information to be synchronized between robots. To find this transformation for the first robot, we utilize a Leica TS15 Robotic Total Station and survey prisms on the starting gate. To monitor a robot’s trajectory, an additional prism is rigidly attached to the robot’s sensor payload so that correspondence between localization in the map frame and the prism’s position in the world frame can be generated. The basestation operator collaborates with the total station operator to track the robot and records correspondences.

To reduce the minimal set size required to solve for the pose, additional structure of the problem can be leveraged. The z-axis of the DARPA-defined reference frame is aligned with gravity, which enables us to directly use the robot’s own roll and pitch angle since they are equipped with IMUs. This simplifies the 3D alignment problem by removing some degrees of freedom. After N correspondences are generated, the transformation can be formulated as minimizing a least-squares problem,

$$\mathbf{R}_{DM}, \mathbf{t}_{DM} = \arg \min_{\mathbf{R} \in \text{SO}(2), \mathbf{t} \in \mathbb{R}^2} \sum_{i=1}^N \|\mathbf{p}_i^D - (\mathbf{R}\mathbf{p}_i^M + \mathbf{t})\|^2. \quad (4)$$

The solution can be decomposed into first solving for rotation via singular value decomposition, and then calculating the translation, as in (Arun et al., 1987). The minimum number of points required

is two. While more points can be recorded to provide an average, LOAM (Zhang and Singh, 2014) has minor drift locally, so two point correspondences are sufficient, and accuracy depends on the baseline between the two points. We have eliminated two degrees of freedom, roll and pitch, and solved for three others: yaw and in-plane translation. The z-translation can then be solved directly from the total station and odometry positions.

Autocalibration. Although the total station calibration only requires two points, each robot must be stopped more than once, and the basestation operator must interact with the total station operator. In addition, either the number of prisms must scale with the number of robots, or other team members must attach and remove them for each robot to be calibrated. Lastly, drones either require an additional crew member to physically provide a baseline for calibration, or they will waste valuable flight time to hold steady during the procedure. For these reasons, we move towards an automatic calibration procedure that requires even less operator interaction.

First, one robot is calibrated using the total station, so that map updates from this robot can be registered globally. The basestation operator may position the robot at a designated physical location, at which a local map and odometry information are requested and transmitted to the basestation via the ledger, as described in Section 3.3.2. To reduce bandwidth while providing an informative map, a point cloud is stitched from a fixed number of previous sets of scans, and then down-sampled. Now that this calibration reference exists on the basestation, the first robot is free to explore.

Subsequent robots may be calibrated by being moved to or placed at the designated spot. The operator can then remotely request the reference from the basestation. Once it arrives, the robot will attempt to align its own local map to the reference via GICP (Generalized Iterative Closest Point) (Segal et al., 2009). The initial transform also compensates for prior knowledge of each robot's height so that the registration will converge to a global minimum. The output provides a relative transform between the reference robot's and the current robot's map frame. Then, the reference robot's total station calibration can be used to infer the current robot's calibration. While GICP is very accurate, it is best to avoid concatenating multiple alignments, so all reference robots are typically calibrated using the total station.

3.3. Communications

3.3.1. Comms Node Planning

To facilitate coordination and message relaying between robots, an ad-hoc communications network is created. Given the environment's negligible infrastructure and dynamic terrain, the network must be easily deployed and robust to both natural and manmade perturbations. Furthermore, as some large messages, including maps or artifact detections, are aperiodic, this network must accommodate high bandwidth bursts with stable steady-state traffic.

Given the need to deploy a robust, modular communication environment in difficult, unmapped terrain, the ad-hoc wireless mesh network was chosen for the communications infrastructure. Incrementally built as the ground robots explore away from the basestation, this mobile ad-hoc network (MANET) accommodates the exchange of data, including but not limited to mapping, odometry, multirobot coordination, and artifact detections. Fundamentally, the entire hardware and software pipeline of the communications platform can be partitioned into the following hardware and software subsections: communication nodes, communication node dropper, autonomous node dropping logic, and connectivity middleware.

The communication nodes fitted to each robot and additionally dropped by all three ground vehicles are 5GHz Rajant DX2 radios, each containing a CPU and a radio card. Acting as repeaters under the IEEE 802.11n wireless networking standard, the radios create an OSI layer 2 topological mesh that can relay data over several hops. The radios transmit using OFDM (Orthogonal Frequency-Division Modulation) and have been configured to maximize performance over ranges of 50 m. Furthermore, to maximize the coverage of the mesh, each node is additionally fitted with



Figure 10. Node dropper with loaded nodes.

two omnidirectional antennas of 5dBi gain. To monitor network strength between peers in the mesh, a link-to-link cost function iPerf3 performance test, and a topological mesh overlay, can be captured from the Rajant hardware. While such functionality is used to supplement other network status information, the robots rely mainly on ping time and the reliability of odometry data to monitor network connectivity. First, the ability for a robot to ping another member over several seconds indicates at least minimal connectivity between participants. Second, the ability for a robot to receive a high frequency, medium-sized, and synchronous message like odometry indicates a connection that is functional.

With the Rajant nodes acting as “breadcrumbs” in the mesh network, the next step in the communications pipeline is the containment and deployment of the radios. A graphic of the communication node dropper can be seen in Figure 10. Capable of holding nine nodes on ground robots “R1” and “R2,” and six nodes on “R3,” the node dropper utilizes a solenoid actuated latch triggered to drop individual nodes when desired. To interface the DX2 radio shell to the manufactured node dropper, a node shell is used for each radio. These shells can additionally be seen in Figure 10 as the numbered green fiberglass containers.

Using the Rajant nodes and the node dropper, autonomous dropping logic deploys the nodes during competition runs. The dropping algorithm relies on a robot odometry, local point clouds, ray casting, and previously dropped nodes. Using the robot’s local point cloud, a ray cast is created from the robot to any previously dropped node. If the ray cast does not intersect a set threshold of point cloud points, then the communication node is deemed “line-of-sight,” otherwise it is deemed “non-line-of-sight.” Under either condition, the robot will then autonomously drop a new node if the robot exceeds a set distance to all previously deployed nodes. As “line-of-sight” conditions promote stronger communication links, the minimum distance threshold to drop a node is ultimately higher for “line-of-sight” conditions. Such logic has created a node dropper that is both rugged and predictable. Such performance traits make this autonomous dropping algorithm the solution most typically used in field testing and competition. Figure 11 illustrates the algorithm in operation by showing a top-down plot of “R2” driving around a garage. The previously dropped nodes are represented using blue circles. Specifically, there are three blue circles forming a triangle around “R2.” On each of the circles is a red ray cast that propagates from the ground robot. The presence of each line indicates that the ground vehicle is currently “line-of-sight” with each of the three nodes.

The final piece of the communications pipeline is the connectivity middleware. Acting as the software infrastructure to the mesh network, this piece creates and monitors the pathways through which robots and the basestation can exchange data. Designed using DDS (data distribution service) (Pardo-Castellote, 2003), the communication manager leverages a publish-subscribe network topology through which all network participants (i.e., robots and basestation) can publish and



Figure 11. Autonomous Dropping Solution

subscribe to the appropriate topics of interest. Furthermore, as the robot platforms experience both ROS (robot operating system) networking and DDS networking, the communication manager helps coordinate these two local and global networks. For ROS topics that should be sent over the network, the communication manager running on that robot will have the appropriate ROS subscribers. The subscriber callback functions will then trigger the ROS message to be converted into an IDL (interface definition language) format and transmitted over the network by a DDS publisher. Once received by the corresponding DDS subscriber, the locally running communication manager will have a DDS callback that will convert the IDL message back to its ROS format and publish the message with the appropriate ROS publisher.

The communication manager encompasses several layers of the OSI model to create a highly abstracted network representation which is highly modular. Fundamentally, the communication manager relies on UDP (user datagram protocol) transport with modified QoS (quality of service) parameters to account for varying degrees of data importance. To maximize the quantity of data that can travel over the network, the communication manager also uses Zlib compression. Furthermore, as the strength of the network can deteriorate as a robot increases its number of hops from the basestation, the communication manager performs dynamic message dropping. Used to supplement every message's QoS profile, dynamic message dropping temporarily ignores low-importance messages until the robot returns to a stronger communications link.

In an attempt to improve and further optimize the layout of the MANET, a second more theoretically sound node dropping algorithm was also created. Specifically, this algorithm tries to optimize the layout of a MANET using the underlying physics of the radio transceiver signal, the geometry of the environment, and predictive mapping that enables the robot to reason about unknown portions of the environment (Tatum, 2020). The objective of the algorithm is to maximize communications coverage over the unknown environment.

Prediction Approach. This autonomous MANET creation algorithm to maximize communications coverage in unknown environments is the Prediction Approach. As the agent explores the unknown environment, it uses what it knows of the environment from exploration to predict the rest of the map. Every time the agent reaches a location that is on the edge of being out of communications of the MANET built thus far, the agent solves for sensor coverage for each of its remaining available nodes using the greedy connected coverage algorithm on the predicted map. The greedy connected coverage algorithm implements the greedy maximum coverage algorithm (Feige, 1998). This polynomial time greedy solution to the maximum coverage problem constructively approximates the optimal maximum coverage within a minimum ratio of $1 - \frac{1}{e} \simeq 0.632$ (Feige, 1998).

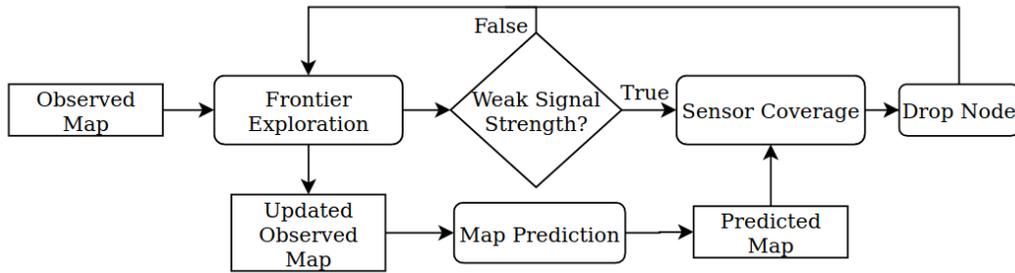


Figure 12. Autonomous MANET creation flowchart. An initial Observed Map of the agent’s environment is used to calculate the agent’s next goal destination with Frontier Exploration. As the agent moves with Frontier Exploration, the Observed Map is updated, which is used for Map Prediction. If the agent’s signal strength to all dropped nodes is weak, based on our fixed sensor range R , sensor coverage is solved for potential node locations using the Predicted Map. A node is dropped at the calculated location nearest to the agent. If there is not weak signal strength, the agent continues exploring with Frontier Exploration.

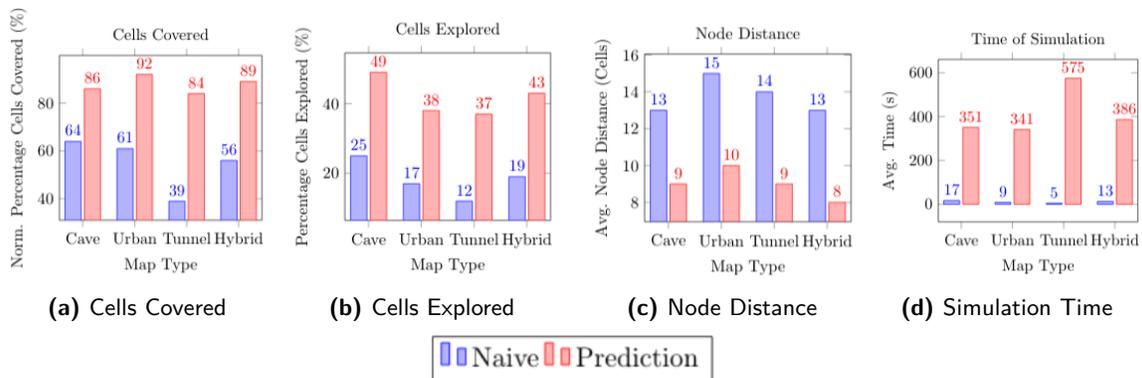


Figure 13. Results in the simulation for 100 maps of each type using the Naive and Prediction Approaches showing the percentage of free cells covered in communications range normalized over the percentage of free cells covered by the Greedy Connected Solution (a), the percentage of free cells explored by the Greedy Connected Solution (b), the average node distance to the Greedy Connected Solution (c), and the average run-time (d).

The greedy *connected* coverage algorithm deviates from the greedy *maximum* coverage algorithm because it ensures that each set (e.g., potential node location) is within communications range of a dropped node $n_D \in N_D$. This constraint ensures that a fully connected network is created. The greedy connected coverage algorithm runs N iterations, one for each node to be dropped. The Prediction Approach was evaluated by comparing it to our previous approach, the Naive Approach, in which nodes were dropped solely based on being line-of-sight (LoS) to other nodes and the robot’s signal strength. Figure 12 demonstrates a high-level system overview of the Prediction Approach.

Simulation Results. The following results (Tatum, 2020) come from running the Naive and Prediction Approaches and the Greedy Connected Solution on 100 32×32 gridworld maps from each map type: cave, urban, tunnel, and hybrid. Figure 13a demonstrates the significant increase in cell coverage when using the Prediction Approach over the Naive Approach. This is expected, given the Prediction Approach’s goal of placing nodes in free areas, whereas the Naive Approach drops a node as soon as the edge of communications range with a dropped node is reached or line-of-sight to a dropped node is lost.

Figure 13b shows that in every environment, the Prediction Approach explores nearly, if not more than, double the percentage of free cells explored by the Naive Approach. This behavior will benefit deployments that seek to maximize exploration with limited resources.

Table 2. Results from the field test in a subterranean urban environment. Each test ended when the sixth node was dropped.

	Cells covered	Node distance (cells)	Time (s)
Naive	52%	26.99	418.59
Predicted	78%	19.03	1142.73

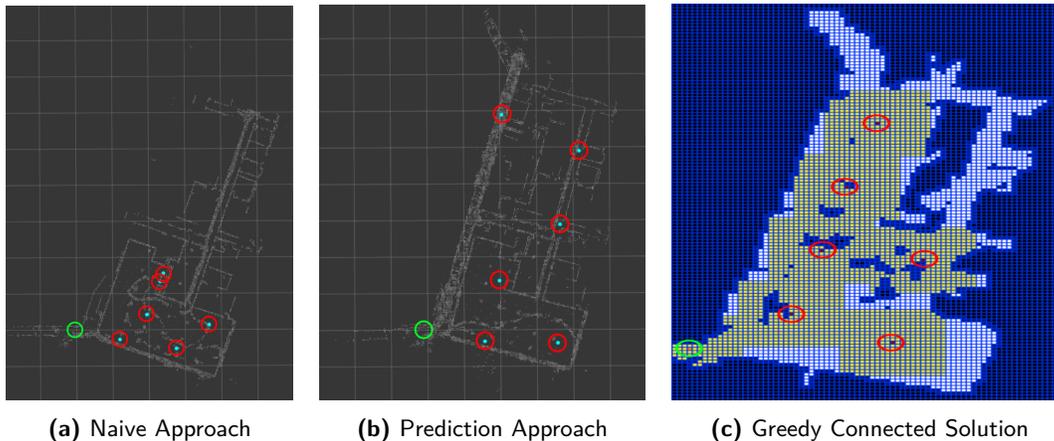


Figure 14. Field test results when dropping six nodes (cyan dots with red circles) in the subterranean urban environment. The green circle is the robot's initial location. Figure 14c is the simulated Greedy Connected Solution.

Figure 13c shows the average node distance between nodes dropped at the same order in the dropping queue from the Prediction and Naive Approaches compared to the Greedy Connected Solution. The distance is an average over the 100 map simulation tests run of the average node location distances. When compared to the Naive Approach, the Prediction Approach is at most a 50% decrease in node distance from the Greedy Connected Solution, shown in the urban maps, and at least a 31% decrease, shown in the cave maps.

Figure 13d demonstrates the large discrepancy in simulation run-times between the Prediction and Naive Approaches. The Prediction Approach's longer run-time can be attributed to its increased exploration as well as the time to run map prediction and the greedy connected coverage algorithm. This metric is one of the main tradeoffs that must be considered when using this method. Measures were taken to decrease this run-time, such as decreasing the number of times the greedy connected coverage algorithm was run, and only performing map prediction every three steps the agent took, but more measures need to be taken if faster times are desired.

Field Results. Testing of the Prediction and Naive Approaches was performed in real-world and simulated environments (Tatum, 2020). The real-world tests were performed in a complex subterranean environment, representative of the environment in the Urban Circuit of the DARPA Subterranean Challenge (DARPA, 2019). For both approaches, a mobile robot was sent into the unknown urban environment without an existing communications network. The robot traversed the environment and autonomously dropped six nodes to create a fully connected MANET. The resulting network topology for both approaches is shown in Figure 14. We compare these results to the simulated Greedy Connected Solution, shown in Figure 14c, in which nodes are optimally placed if the environment is fully known before exploration.

Table 2 shows that the Prediction Approach results in greater communications coverage and more accurate placement to the Greedy Connected Solution in a real-world environment, rather than the Naive Approach. In addition, the Prediction Approach promotes greater exploration of the environment while maintaining communications, as shown by the larger and more detailed point

cloud results in Figure 14b compared to Figure 14a. The time in which the robot was exploring and dropping nodes was shorter for the Naive Approach, but this is mostly due to it not exploring as much of the environment before its final node was dropped and the test was ended. Based on these experiments, we can see that our autonomous MANET creation approach (the Prediction Approach) improves communications coverage and exploration within communications over the suboptimal human-centric approach (the Naive Approach).

3.3.2. Ledger

The team of robots generates various map types while exploring the environment. These different map types are made up of artifact detections, explored regions, and point clouds for autocalibration. A common trait that these map types share is that they can be broken up into smaller submaps. For example, one artifact detection submap contains the location of one artifact and several images of it with respect to a particular transform. Different entities on the network have different requirements about which map types are relevant to them. For example, the basestation computer may only care about artifact detections and point clouds for autocalibration and not explored regions. Additionally, we want to be able to achieve a data muling behavior for object detections so that objects detected by one robot that meets another robot can be synchronized in case one of the robots gets stuck and never returns to the basestation.

To share these data efficiently across the communication network without taking up too much bandwidth and determine which data will be shared with which robots, we use a data structure called the ledger.

The ledger is a record of submaps that gets broadcast over the network. A submap is uniquely identified by its map type, the entity that created it, and a map ID number which is unique given the map type and entity that created it. As an entity creates submaps, it adds them to its ledger and periodically sends the ledger to other robots. When other robots receive the ledger, they merge its contents into their own ledger. In this way, over time, each robot will build a record of all submaps in existence. An entity may know about the existence of a submap, but not hold the submap's data locally, so this is also stored in the ledger. With this information, when a robot needs access to a certain submap for a task it is performing, it can choose from which robot to request the data. Submaps are sent across the network in response to requests, and acknowledgments are sent in return. The time it takes from sending a submap to receiving an acknowledgment allows the ledger to estimate the bandwidth between pairs of robots. The bandwidth estimate is used so that the network is not overloaded. When there are pending requests for multiple map types, the ledger tries to send an equal amount of data from each map type across the network so that the sending of one map type does not exclude others. Figure 15 shows an example of how the ledger functions.

3.4. Ground Robots

3.4.1. Vehicle Design

We have developed roving robots for this challenge: “R1,” “R2,” and “R3,” as pictured in Figure 16.

All of the ground robots are a custom design, use the same general methods of construction, and share the same electronics. The chassis is constructed of a welded steel frame. Within the frame are four motor/wheel modules, a battery module, and an electronics module. On the exterior of the frame, various payloads can be mounted. On all three ground robots, a perception payload and communications node dropper is mounted. R1 and R2 also have a platform that allows drones to be carried and secured.

There are some differences between the three ground robots. In particular, R1 is a fixed chassis with no suspension. R2 and R3 were constructed with a centerline pivot to allow the UGVs (unmanned ground vehicles) to traverse over obstacles and maintain better ground contact. In Figure 17, the split (pivot-point) in the chassis can be seen. R1 and R2 are close in size; R3 was designed to have the same pivot as R2, but be smaller and lighter. R3 is also designed to use different types of wheels/tracks for different environments.

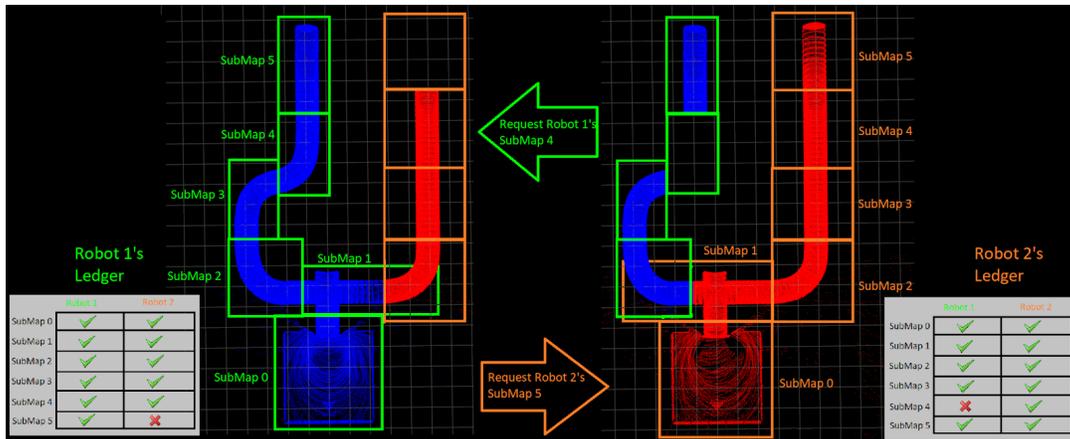


Figure 15. This shows an example of how the ledger works. The left side shows robot 1's map. It has created the green submaps itself, and received the orange submaps from robot 2. The right side shows robot 2's map. It has created the orange submaps itself, and received the green submaps from robot 1. Each robot knows about the existence of all submaps because they have shared ledgers with each other. Robot 1 is missing data for robot 2's submap 5. Robot 2 is missing data for robot 1's submap 4. As a result, they sent the appropriate requests to each other for the missing submap data.

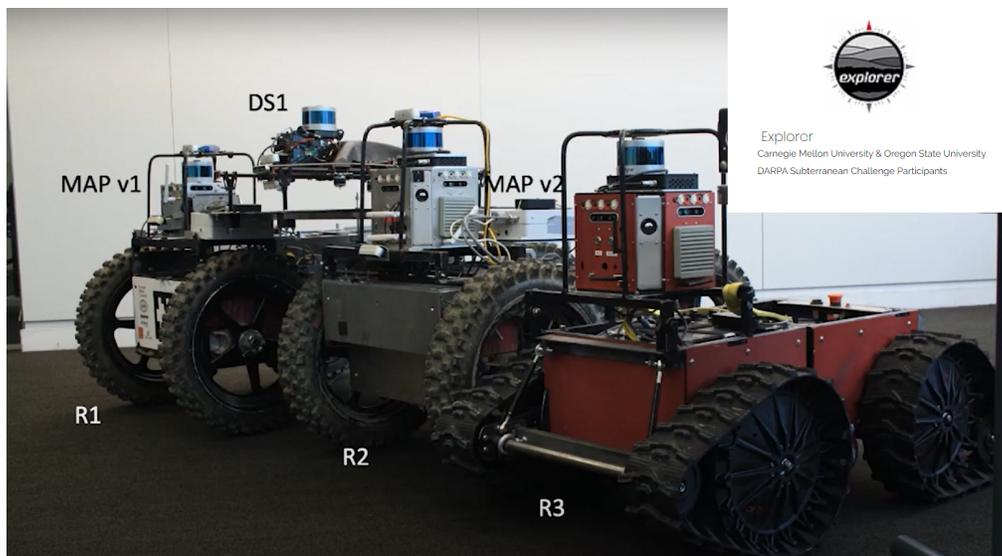


Figure 16. The three roving robots, with their payloads. Note “R1” has a “DS” drone mounted to the rear.

The decision to build custom ground robots was due to the nature of subterranean environments, particularly mines and caves. Based on prior experience, the team wanted to have large diameter wheels and body differencing (pivoting) for mobility. When reviewing mobile platforms on the market, we were not able to find any that had a good balance of large wheels, the ability to operate on difficult terrain, a rugged design, and would allow us to stay in the 1 m square opening that the initial DARPA documents suggested the robots needed to be able to pass through. In designing these vehicles, the top of the sensor payload roll cage was designed to be just under 1 m tall. R1 was built with no pivot due to time constraints; R2 and R3 have exhibited better mobility performance in rough terrain relative to R1. After building R1 and R2, the decision was made to build the next robot to be able to fit in smaller openings, hence the smaller R3 robot.

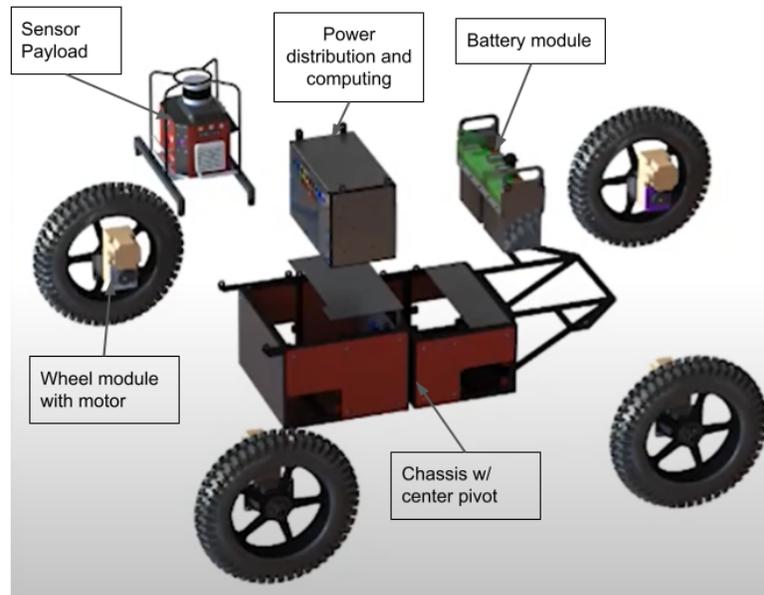


Figure 17. An exploded view of “R2” showing its core modules. All three robots share similar modules.

Table 3. Specifications of ground robots. Note that these are typical specifications and can change based on robot configuration. Example: configurations to carry drones or tires/tracks for better rough terrain or stair climbing ability.

	R1	R2	R3
Dimensions (m)	1.2 (L) × 0.8 (W)	1.2 (L) × 0.8 (W)	1.0 (L) × 0.6 (W)
Mass (kg)	187	218	110
Speed (m/s)	2	2	2.5
Wheel diameter (m)	0.55	0.55	0.38
Ground clearance (m)	0.20	0.20	0.12
Nodes in dropper	9	9	6

3.4.2. Exploration and Multirobot Coordination

Efficient Exploration for large and complex environments. Autonomous exploration in large-scale and complex environments is crucial for an increasing set of applications in both civilian and military domains, yet the problem still remains challenging. Traditional methods formulate the problem using frontiers, i.e., the boundary between known and unknown space. As the robot moves toward the frontiers, the unknown space covered by the sensor field-of-view (FOV) becomes part of the mapped area, thus the boundary is pushed forward generating new frontiers. The process is repeated until the entire environment is covered. Such formulation works well in 2D but can suffer from noisy frontiers in 3D due to occlusion and sensor noise. Furthermore, most of the existing approaches rely on greedy strategy, which suffers from being myopic and produces inefficient exploration patterns such as going back-and-forth and unnecessary revisiting of the same places.

Here we developed a hierarchical exploration planner to improve both the exploration efficiency and runtime efficiency for large-scale cavelike environments. In particular, the planner operates at two levels:

1. At the global level, the planner maintains sparse information to plan a coarse tour for the robot as high-level guidance, and
2. At the local level, the planner maintains dense information to plan a detailed path to ensure full sensor coverage of the local area.

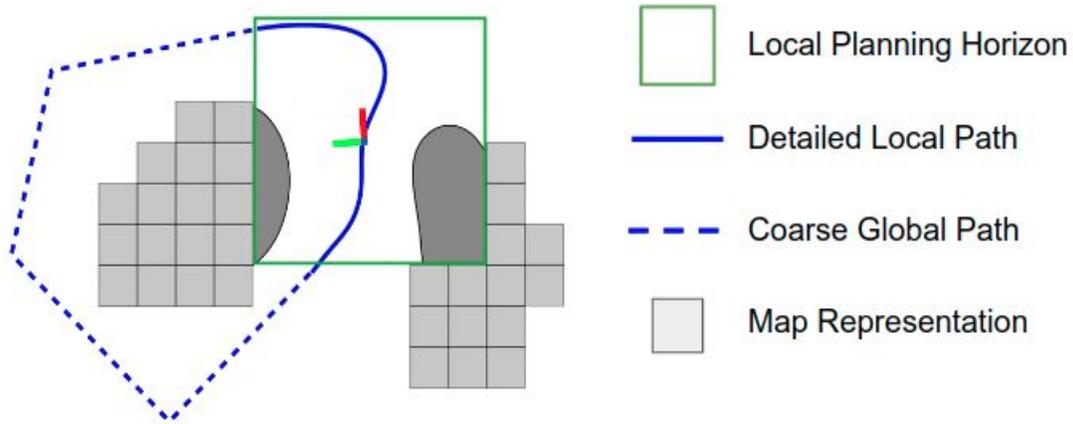


Figure 18. Illustration of the hierarchical framework. The coordinate frame represents the vehicle. The green box represents the local planning horizon. The grey blocks represent the global world representation with low resolution. The darker area shows the local area represented with higher resolution. The blue curve is the planned path, of which the solid part is locally planned and the dotted part is globally planned.

Figure 18 illustrates the concept. The green rectangle shows the local planning horizon that the detailed path (solid blue curve) is planned within. The dotted blue lines show the coarse global path planned upon a lower-resolution map (grey blocks).

Unlike the traditional exploration planners that are driven by eliminating frontiers (unmapped areas), our planner focuses on sensor coverage of structures in the environments. This is particularly important in search-and-rescue scenarios, where a thorough inspection of the area of interest is desirable. In particular, a surface is considered “covered” if it is perceived by a sensor within a distance range and an angle range. Our robots are equipped with a long-range (about 100 m) laser scanner and multiple short-range depth cameras (about 5 m). To accurately detect and localize artifacts, the robots must drive close to the artifacts for the depth cameras to obtain enough pixels of the object. For this purpose, traditional exploration approaches driven by building a complete map using solely the laser scanners are not capable of performing such a thorough search.

We conduct experiments using a vehicle platform equipped with a Velodyne Puck LiDAR and a MEMS-based IMU. The system uses our prior method for state estimation and mapping explored areas (Zhang and Singh, 2018). Collision avoidance uses a trajectory library-based method (Zhang et al., 2020). Our exploration algorithm runs on a laptop computer with a 4.1GHz i7 CPU and uses a single CPU thread for processing. Here, we compare our method with two existing methods that are considered state-of-the-art:

- *NBVP* (Bircher et al., 2016): A method using a Rapidly-exploring Random Tree (RRT) (LaValle, 1998) to span the space. It finds the most informative branch in the RRT as the path to the next viewpoint. We evaluate using the open-source code.
- *GBP* (Dang et al., 2019): An extension of NBVP where the method builds a roadmap through the traversable space and searches the roadmap for a route to relocate the vehicle. The method explicitly switches between exploration mode and relocation mode. We use an open-source code that was tuned and adapted to the testing environments.

The methods are evaluated in a large indoor environment consisting of lobbies and dining areas connected by long corridors, as shown in Figure 19. Note that the map is decluttered for visualization purposes. The space expands on both sides of the corridors, and windows and glass walls add additional difficulty to the environment. NBVP and GBP are not able to cover the space completely and leave a large portion uncovered. NBVP is set to explore the maximum area (80 m×80 m) without leading to a performance drop. GBP produces endless back-and-forth motion to the end of

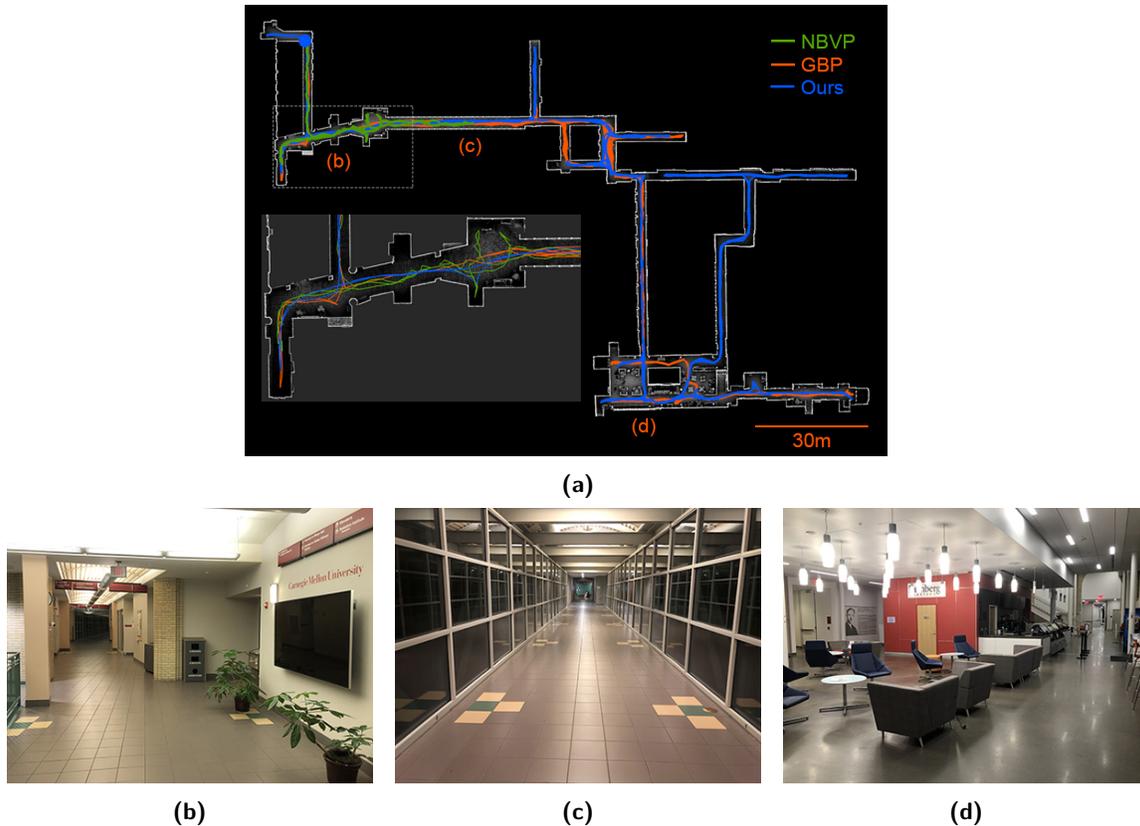


Figure 19. Results of the exploration test in an urban environment. Part (a) shows the resulting map of our method and the trajectories of NBVP, GBP, and our method overlaid on the map. The blue dot indicates the start point of all three trajectories. Parts (b)–(d) are three photos taken from the locations as labeled in (a).

the run and is stopped. Our method completes the exploration after traveling over 988 m in 1167 s. In particular, we can see in the closeup view in Figure 19a that the green trajectory (NBVP) is repetitive and inefficient. Figure 20a presents the explored volumes. As the method terminates, NBVP takes 1228 s to cover 687 m³ while the same amount of space is covered by our method in 206 s. GBP takes 1801 s to cover 1388 m³ and that amount of space is covered by our method in 550 s. In both cases, our method is able to explore the same volume six times faster than NBVP and three times faster than GBP. In Figure 20b, NBVP uses 1228 s to travel over 231 m and the same distance is completed by our method in 265 s. GBP travels over 958 m overall, almost the same as our method. Since the vehicle often needs to navigate through explored areas before relocating to a different area to explore further, we consider the explored volume as a better measure of the efficiency. In Figure 20c, GBP travels over 958 m to explore 1388 m³, whereas our method travels 450 m to explore the same amount. In experiments, GBP and NBVP are prone to revisiting the explored area, which reduces the overall efficiency of exploring the whole space. Figure 20d presents the runtime. The average runtime for NBVP is 2.5 s, for GBP it is 1.8 s, and for our method it is 0.13 s. Again, our method is more than five times faster than NBVP and GBP in terms of processing. For more details, please refer to (Chao et al., 2021).

Coordination. Roadmap-sharing is a term we use to identify where each of our robots are at a given time and their path history. The roadmap built by each robot is shared with all the other robots on a course through submap sharing described below. A roadmap is a graphical representation of a robot’s traversable space and travel history. Figure 21 shows an example for the roadmap built

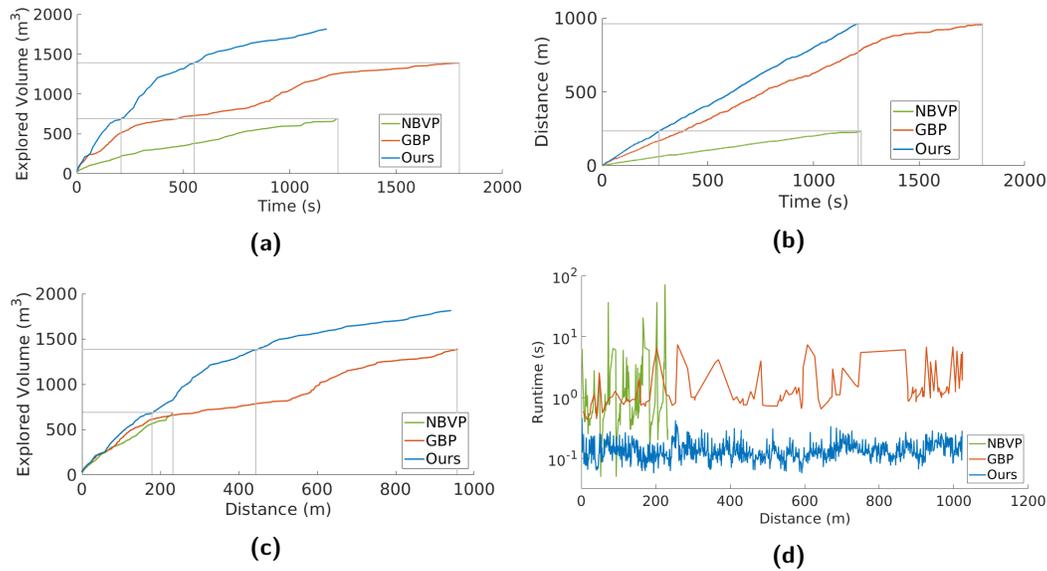


Figure 20. Part (a) is the explored volumes vs time, (b) is the traveling distances vs time, (c) is the explored volumes vs traveling distance, and (d) is the runtime vs traveling distance. Neither NBVP nor GBP is able to explore the space completely. Both leave a large portion of the space unexplored. Our method completes after traveling over 988 m in 1167 s.

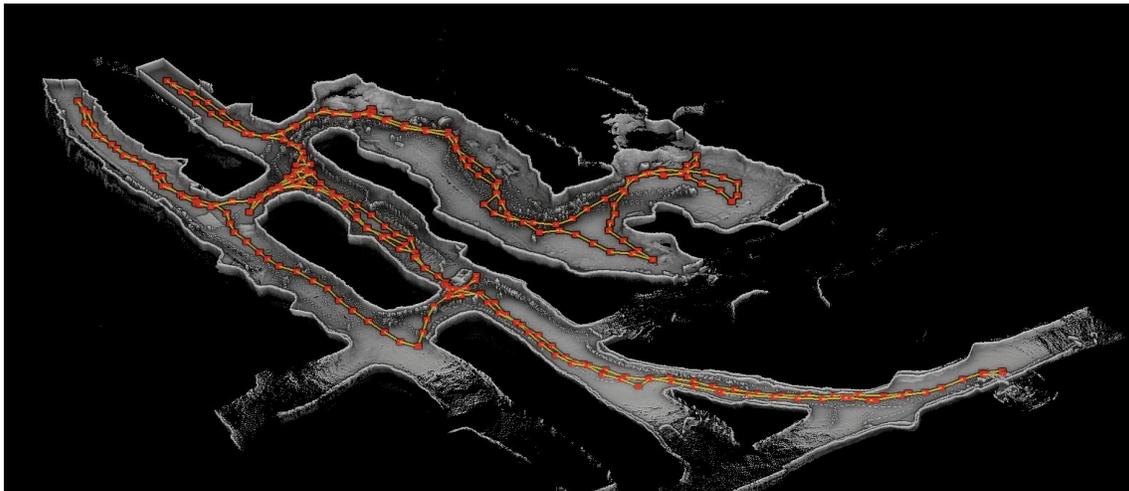


Figure 21. Resulting map built by a ground robot and the roadmap with red dots showing the key poses and yellow edges representing their connectivity.

by one of the ground robots at Brady’s Bend underground after exploring some areas. The red dots show the position visited by the robot, and yellow lines show their connectivity. With shared roadmaps from other robots, one robot can plan a path to places that were never explored by other robots. We provide a video¹ to show the process of sharing and merging roadmaps from different robots. Our first robot is initially exploring an area to the left side of the map. A second robot is then launched and receives information from the first robot to take a divergent path to the right.

¹ The video file can be found in supplemental file 1.

This method minimizes redundancy in path exploration and maximizes the areas covered by both robots.

Submap sharing. We extended the single-robot exploration planner for multiple ground robots to explore a large area in a coordinated manner. In particular, we discretize the area of interest into submaps, which are evenly spaced cubical areas. During the exploration, a submap is considered “covered” if the robot has fully explored it and no new information can be gained from it. Otherwise, a submap has the state of “uncovered” or “covering” depending on whether the robot has already gathered information from the submap. For coordination, robots share both the “covered” and “covering” submaps with each other. In this manner, one robot will not reexplore a place that has already been explored by other robots. In addition, if a robot cannot find new places to explore based on its own knowledge, it will utilize the shared submap to determine the next place to go and plan a path to get there using the shared roadmaps. In other words, the shared “covering” submap tells a robot where to go for further exploration, while the shared roadmap tells the robot how to get there. Note that the “uncovered” submaps are not shared because no robots have entered those areas, where no reachability information can be provided from the shared roadmap. The following video shows an example of coordinated exploration in simulation by two robots: see <https://youtu.be/HKEHpoEZDvs²>

3.4.3. Terrain Evaluation and Local Planner

Due to its complex terrain, the subterranean environment is challenging for ground robots to navigate. In addition to detecting positive obstacles, which is the most common terrain, we need to distinguish other terrain, such as slopes, holes, and negative obstacles, to ensure the robots travel safely in the environment. Most methods that use cost maps for terrain analysis can only detect obstacles and free space but are unable to distinguish different terrain features to assist the robot in making an appropriate decision through the corresponding terrain.

We developed an elevation-map-based method that evaluates traversability of the terrain for the ground vehicles. Our method maintains a $5\text{ m} \times 5\text{ m}$ local area map around the robot. The local map consists of a set of geometric points associated with a score revealing the difficulty of navigating over that point. Points with higher scores are considered more difficult to navigate over. The score of a point is computed based on the relative elevation of that point with respect to its neighbors. For example, the score of a point on a flat ground is zero and the score of a point on an obstacle is its height difference to the ground plane. In this manner, the local planner can find a feasible path towards the navigation goal while avoiding rough terrains with frequent elevation changes. Our method also locates negative obstacles, which correspond to drop-off edges or holes in the ground. Similar to positive obstacles, the score of points on the negative obstacles is determined by its height difference from the ground plane. We consider areas in the sensor’s field of view that have no points to be holes, and we manually add virtual obstacles in the corresponding area to the final terrain map. A basic version of our algorithm implementation is open-sourced at https://github.com/jizhang-cmu/ground_based_autonomy_basic. Our method actively detects openings and slopes in the robot’s surroundings. If the navigation waypoint is on the other side of a narrow opening or slope, the terrain analysis informs the local planner about the width of the opening and the height of the slope. With that, the local planner is able to reduce the speed and orientate the robot accordingly for passing through the narrow opening and avoiding larger slopes. Compared to our previous approach where the local planner blindly searched paths that pass through the opening or slope, the current approach significantly improves the efficiency of handling such cases.

We adopted an approach similar to (Zhang et al., 2019) to develop the local planner for the ground vehicles, which is responsible for low-level autonomy including path following and collision avoidance. The key challenge in developing such a local planner lies in the need to efficiently perform

² The video file can be found in supplemental file 2.

collision checking and generate kinodynamically feasible trajectories with limited computational resources onboard the robots. To this end, we utilize a trajectory library and precompute the collision correspondences between each trajectory and a set of geometric locations in the vicinity of the robot. In this manner, the computational cost of real-time collision checking is avoided. Further, the local planner takes in the results from terrain analysis to evaluate the feasibility of the trajectories. Instead of searching for the shortest path leading to the navigation goal, our planner finds the path that maximizes the likelihood of reaching the goal while minimizing the navigation cost over rough terrains.

3.4.4. Recovery Behaviors

All ground-based robots can encounter terrain that causes mobility issues. For example, our ground robots had a tendency to become trapped between railroad tracks while exploring Tour Ed mine. Turning on the tracks caused the wheel sidewalls to jam against the steel track and prevented recovery. In these situations, the robot is trapped physically while no sensor is able to observe and sense it, making it impossible for the robot to move if it continues to try the usual commands. Therefore, reasonable recovery behavior is essential for ground robots to overcome these difficult cases.

In our system, recovery behaviors include two parts, namely stuck recovery behavior and tip-over recovery behavior. Stuck recovery behavior is designed to help the robot escape instances where it is unable to move as intended. Tip-over recovery behavior is used to prevent the robot from exceeding predefined dangerous roll and pitch attitudes. Roll and pitch attitudes are easily computed in real time using information from the IMU, and once tip-over behavior is detected, the robot will move in the opposite direction immediately. Climbing a wall will trigger a stop command followed by a reverse command to back away from the obstacle and recompute a new trajectory path. This also works when traversing up and down forward slopes or stairs. When a roll attitude limit is exceeded, the robot will be commanded to stop and turn either up or down the slope, depending on the local and global planning goals.

Stuck recovery behavior is more complex than tip-over recovery behavior. By integrating the commanded velocity over a short time horizon, including linear and rotating velocity, we obtain the expected robot motion. We decide if a stuck behavior is occurring by comparing the actual motion obtained from SLAM with the expected motion. Depending on the difference between the true and expected motion, all stuck behaviors are grouped into four categories: forward-stuck, backward-stuck, rightward-stuck, and leftward-stuck. In all cases, for example, a stuck behavior occurs when the expected (fake) motion is ahead of the actual (real) motion, which means the robot is unable to move as expected. We use four stuck recovery behaviors: forward recovery, backward recovery, rightward recovery, and leftward recovery, to attempt to free the robot from the four stuck situations. The directional words in these four recovery behaviors indicate the directions the robot should move to. Figure 22 shows the relationship between the expected odometry and the real odometry in the four stuck status cases, including the priority of each recovery behavior for all cases.

The priority in Figure 22 means the sequence that all these four recovery behaviors will be executed in. For instance, in Figure 22(a), when the robot is in a forward stuck state, it will first move backward, then rotate rightward, leftward, and finally move forward. Forward and backward recovery behaviors always have higher priority over leftward and rightward because in most situations we have experienced, the robot becomes stuck when rotating around the gravity axis more often than moving linearly.

3.4.5. Behavior Executive

The behavior executive observes the state of the robot and decides which state to be in and transitions between different behaviors. We use a behavior tree representation (Colledanchise and Ögren, 2018) to transition between the different states. While the actual behavior tree is quite large, we show an extracted simplified behavior tree in Figure 23. The tree has four modes: explore, return home, go-to waypoint, and recovery.

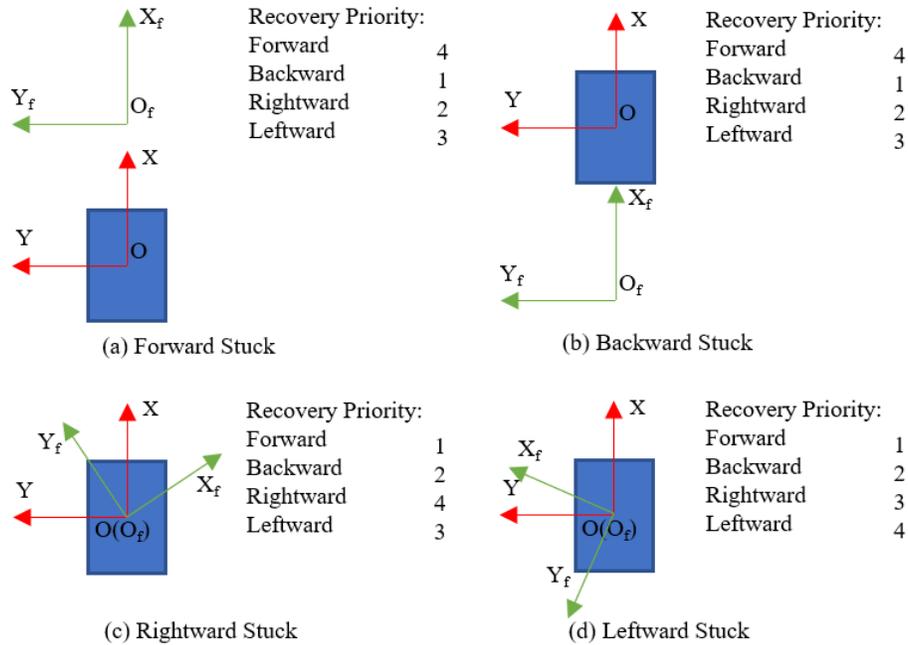


Figure 22. Four stuck status and priorities of these four recovery behaviors in each status. (a) Forward Stuck. The fake odometry $X_f O_f Y_f$ is in front of the real odometry $X O Y$. (b) Backward Stuck. The fake odometry $X_f O_f Y_f$ is behind the real odometry $X O Y$. (c) Rightward Stuck. The fake odometry $X_f O_f Y_f$ has an angular displacement along clockwise relative to the real odometry $X O Y$. (d) Leftward Stuck. The fake odometry $X_f O_f Y_f$ has an angular displacement along counterclockwise relative to the real odometry $X O Y$.

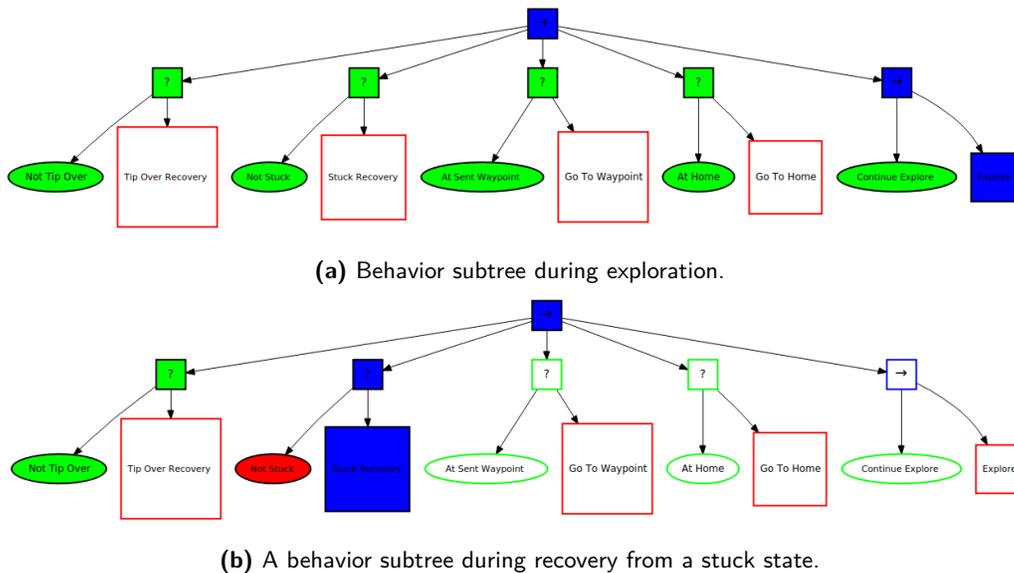


Figure 23. Two behavior subtrees for the roving robot. Actions are denoted by large squares, conditions by ovals, and control flow (logic) nodes by small squares. Solid squares and ovals are currently being evaluated (due to the control flow logic). Blue indicates a “running” status, green indicates “success,” and red indicates “failure.”

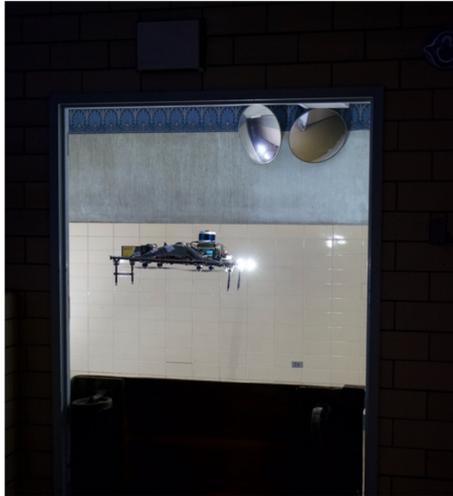


Figure 24. “DS” custom air vehicle optimized for flight in indoor environments.

The behavior executive collects information about all states, updates the relevant states (ovals), and finally defines all actions (rectangles) that should be performed by the behavior tree. When the states are set correctly, the behavior tree can automatically activate corresponding actions. If no command is issued, the behavior tree triggers the lowest priority exploration mode. If the behavior executive receives any command, regardless of the current command such as return home or go-to waypoint, it will update the corresponding states and trigger the correct mode. Recovery mode has the highest priority of all these modes. Whenever the robot is stuck or is in danger of a tip-over, the behavior executive will receive the state information and then switch the robot to recovery mode. To demonstrate the process of switching modes with behavior tree, we show two example experiments where tip-over and stuck behavior happen. Figures 23(a) and 23(b) show the changes when the robot switches from exploration to stuck recovery. The behavior tree first detects the stuck status and triggers the stuck recovery behavior automatically.

3.5. Air Vehicles

3.5.1. Vehicle Design

Flying indoors and underground in visually degraded situations and austere environments requires a robust and collision-tolerant drone with a heavy mission payload of sensors used to navigate through small openings, vertical shafts, and stairways. The gross weight of the drone determines the motor and propeller size, while the endurance and range determine the battery size. Commercially available drones did not meet our needs primarily due to the required sensor package and integration with other robotic systems. Hence, we designed a custom drone that is collision-tolerant, robust, easy to repair, can fit through doors, and has sufficient flight time to perform its mission. The design maximizes off-the-shelf components with a few custom carbon-fiber frame pieces. A picture of our custom drone called “DS” is in Figure 24 and detailed specifications are provided in Table 4.

The propellers are in a pusher configuration with the rotor axis below the frame to maximize protection and open the top side for the battery and sensor payload. The payload is mounted on top to minimize width while maximizing rotor area below the frame. Our goal was to enable the system to be able to fly through a standard-sized door (1 m openings).

3.5.2. Exploration and Multirobot Coordination

The exploration behavior of our aerial robots is planned and coordinated through the use of frontier-based exploration and the sharing of coarse observed maps. The goal is to plan and

Table 4. “DS” air vehicle specifications.

Flight time	13 min
Size	81 (L) × 68 (W) × 27 (H) cm
Gross Weight	5.2 kg
Battery	LiPo 4 s, 16 Ah
Motor	KDE2814XF-775
Propeller	T-Motor P12x4

execute trajectories that maximize the chances of observing artifacts within the field of view of the object detection cameras. While the exploration objectives are similar to that of the ground robot exploration described in Section 3.4.2, we opted to develop a different exploration pipeline for the aerial robots so that we could directly address the unique challenges faced by our aerial robots. These challenges include the limited sensor field of view, exploiting the ability to move in three dimensions, planning with reduced computational resources, and making the best use of a relatively short flight time.

We note that this aerial exploration pipeline has evolved significantly over the three circuits of the SubT Challenge. In this section, we focus on describing our current system at the time of the Cave Circuit. Earlier for the Tunnel Circuit, we developed a corridor-following exploration algorithm, but our aerial robots played a relatively minor role in this event. In the Urban Circuit, the exploration pipeline was similar to what is presented here, except we have since made improvements to how we consider the limited field of view of the cameras on the aerial robots, and we added a map sharing component between aerial robots.

Sufficient literature already exists describing a wide variety of approaches for robot exploration and coordination. We have found it difficult to find an approach with sufficient robustness that can handle the unique challenges presented by each environment in the competition, in addition to handling a multitude of corner cases arising from extensive field tests. To satisfy robustness requirements and other constraints mentioned above, we had to come up with a customized solution that uses a select set of existing approaches as building blocks.

Overview and Rationale. An overview of our exploration and coordination software architecture is provided in Algorithm 1. We briefly summarize the pipeline here and provide further details about each submodule below. The map representation used in the pipeline is a custom occupancy grid mapping library that is built using an open source data structure optimized for volumetric data named OpenVDB (Museth, 2013). This enables fast and computationally efficient generation and processing of large-scale maps, which is a critical requirement for a compute-constrained aerial platform. Built on top of the core mapping base 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. 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 (Choudhury et al., 2019). There are three planners that the robot can choose from, namely the RRT-Connect planner, the Random Walk Planner, and the Graph 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.

Mapping. The exploration pipeline utilizes a set of maps, with each map containing information that serves a different purpose. Each map is stored as an OpenVDB grid (Museth, 2013). This data structure allows us to scale our maps to very large environments such as tunnels and cave systems while maintaining efficient random access and fast iteration. Each map consists of a grid of voxels, with the data type stored at each voxel varying between the maps.

The *global map* is a fine resolution map that uses the point cloud observations. The 3D points are raycast to update each voxel from origin to the point’s location. This is done using an additive

Algorithm 1. Overview of the aerial exploration and coordination pipeline described in Section 3.5.2. Our implementation executes many of the listed functions in parallel within separate ROS nodes.

```

1: ▷ Initialize Maps
2: global_map, observed_map, frontier_map, shared_observed_submaps ← OPENVDBMAPS({})
3: ▷ Update Plan at a Fixed Frequency
4: for each timestep do
5:   ▷ Receive Data
6:   odometry, lidar ← READSENSORS
7:   shared_observed_submaps ← RECEIVESHAREDMAPS
8:   ▷ Mapping
9:   global_map ← BUILD OCCUPANCYMAP(lidar, odometry, global_map)
10:  observed_map ← RAYCASTING(global_map, odometry, observed_map)
11:  frontier_map ← EXTRACTFRONTIERS(observed_map, odometry)
12:  ▷ Frontier Generation and Viewpoint Sampling
13:  clusters ← CLUSTERING(frontier_map)
14:  viewpoints ← GENERATEVIEWPOINTS(clusters)
15:  coordination_scores ← EVALUATESUBMAPS(viewpoints, clusters, shared_observed_submaps)
16:  ▷ Viewpoint Selection
17:  sorted_viewpoints ← SCOREANDSORT(viewpoints, odometry, coordination_scores)
18:  ▷ Progress Monitor
19:  request_new_path ← CHECKPROGRESS(path, odometry)
20:  if request_new_path then
21:    ▷ Path Planning
22:    for each viewpoint ∈ sorted_viewpoints do
23:      path ← BI-RRT-CONNECT(odometry, viewpoint, global_map)
24:      if PATHFOUND(path) then
25:        break
26:    ▷ Publish Incremental Goals to Local Planner
27:    waypoint ← EXECUTEPATH(odometry, path)
28:    PUBLISHWAYPOINT(waypoint)
29:    ▷ Share Coarse Submaps with Other Robots
30:    outgoing_shared_observed_submaps ← GENERATESUBMAP(observed_map, frontier_map)
31:    COMMUNICATESHAREDMAP(outgoing_shared_observed_submaps)

```

log-odds probability update similar to Octomap (Hornung et al., 2013). The *observed map* is a subset of the *global map* that only contains voxels that are predicted to have been observed within the field of view of the object detection cameras. The *frontier map* is also a subset of the *global map* but only contains voxels that are not within a distance threshold of any voxels of the *observed map*. The *shared observed submaps* is a coarse representation of the *observed map* that is split into submaps and shared with other robots via the communication ledger (see Section 3.3.2). The *global map* contains floating point data while the other maps are Boolean.

We note that our frontier definition extends the standard definition of exploration frontiers (Yamauchi, 1997) to also include information beyond the boundary using observations from another sensor. Similar techniques have been proposed for other multisensor exploration planners (Vidal et al., 2020; Best et al., 2018). 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. We also note that we opted for a frontier-based approach [e.g., (Yamauchi, 1997; Burgard et al., 2005; Saroya et al., 2020; Zhou et al., 2021)] rather than an information-theoretic approach [e.g., (Charrow et al., 2015; Tabib et al., 2016; Lauri and Ritala, 2016; Corah and Michael, 2019; Sukkar et al., 2019)], as frontier-based approaches are generally more computationally efficient while yielding similar exploration behavior.

Frontier Generation and Viewpoint Sampling. The *frontier map* is then further processed for generating informative viewpoints. The occupied voxels of the *frontier map* are clustered into groups of neighboring voxels using a simple and efficient clustering algorithm. Small clusters are removed to reduce noise. Then the centroid of these clusters is extracted, which represent a set of points that we aim to observe by the object detection cameras.

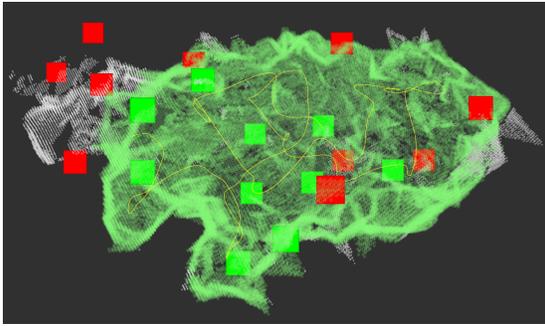
To generate a set of viewpoints for each cluster, a small, discrete set of positions and directions lying on and within a cylinder centered at each centroid is trialled. Each pose is a 3D position, and a yaw angle that will point the front of the aerial vehicle towards the centroid. Each candidate pose is determined to be valid if there is unoccupied line of sight to the centroid and that most of the voxels within a neighborhood of the pose are unoccupied and known. These candidate poses are passed to the planning module instead of the frontier centroids because planning directly to the frontier centroids would result in the aerial robots navigating unnecessarily close to the walls. This frontier clustering and viewpoint generation approach closely follows the method proposed in (Zhou et al., 2021), while also doing line-of-sight checking in a similar way to (Vidal et al., 2020).

Viewpoint Selection. The next module in the pipeline prioritizes the viewpoints according to a scoring function. Ideally, this scoring function should estimate the time to reach each viewpoint. However, it is infeasible to compute these time estimates while considering vehicle dynamics and collision avoidance to all viewpoints within the time constraints of our online planning pipeline. Therefore, we propose a heuristic scoring function, and these scores are then used to prioritize which viewpoints should be considered for explicit path planning.

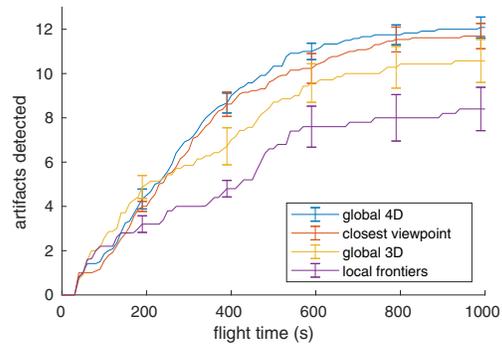
Our proposed scoring function is a weighted sum of (1) the Euclidean distance to the viewpoint, (2) a momentum reward, and (3) a penalty for observing regions that have already been observed by other aerial robots. The momentum reward favors selecting viewpoints that are in the current direction of movement 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. This heuristic is similar to that proposed in (Vidal et al., 2020) for underwater exploration, but uses velocity to estimate momentum rather than relying on the vehicle orientation as our aerial robot local planner allows for omnidirectional motion. 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; adding additional heuristic terms to the sum had a negative affect on the behavior as they diluted the intentions of the three main heuristic terms.

Path Planning and Execution. Path planning is performed to each of the viewpoints in order of their priority, and the first feasible path that is found is selected to be executed. This path planning is performed using a standard bidirectional RRT-Connect (Rapidly Exploring Random Tree) planner, as described in Section 3.3 of (Jordan and Perez, 2013). Each path is planned as a sequence of 3D positions using a Euclidean distance metric. The yaw of each point is considered free (to be determined later by the local planner in Section 3.5.4), up until it moves near the viewpoint, where it is then specified as the yaw of the viewpoint.

The selected feasible path is then locked in and followed by the local planner until a switching criteria is met. Switching occurs when either the frontier associated with the goal viewpoint is cleared out of the *frontier map*, a viewpoint with a significantly higher score appears, the drone is making insufficient progress along the trajectory, or the viewpoint pose is reached. In the case of insufficient progress along the trajectory, a backtracking or random walk behavior is briefly executed to resolve the issue, then a new viewpoint is selected. If the battery is low or no frontiers are detected, then the aerial vehicle follows a generated roadmap back to the takeoff location, in a similar way to the ground robots. In all cases, the relevant planner follows a trajectory with intermediate waypoints that are passed to and executed by the local planner (see Section 3.5.4). This behavior switching is achieved through the use of a behavior tree architecture (see Section 3.4.5). While our current



(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 points 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 of the text. Error bars indicate standard error over 10 trials.

Figure 25. Comparison of different exploration strategies in simulation.

system uses a hand-designed behavior tree structure, we are also investigating ways to learn the best behavior tree structure for improved system performance (Scheide et al., 2021).

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

Simulated experimental results are presented in Figure 25. These simulations were performed within a custom procedurally generated Gazebo world (Figure 25a) 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 we presented in (Saroya et al., 2020). 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,
- *Local frontiers* only uses frontiers at the boundary of the sensor observed map.

Our results presented in Figure 25b 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 similarly 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

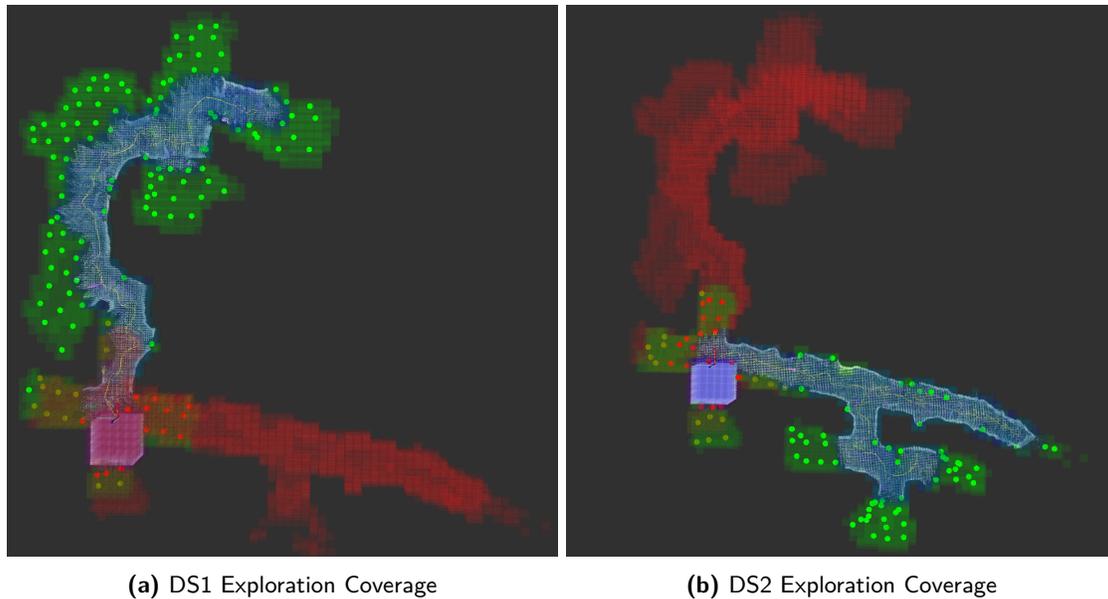


Figure 26. Simultaneous coordinated exploration by two drones at Brady's Bend Mine (Top View). Green voxels represent unexplored volume. Faint blue voxels represent the volume observed by the robot's cameras. Red voxels denote shared submaps received by the robot. Green points in the map reflect unexplored frontier clusters. The explored map is visualized as white points. The drone flight trajectory can be seen as a continuous line in yellow.

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.

For our real-world experiments, Figure 26 illustrates a coordinated exploration run carried out with two aerial robots in simultaneous flight. This experiment was performed inside the mine at Brady's Bend, PA. Each flight lasted approximately 8 min. In Figure 26a, the green and faint blue voxels represent two voxel states transmitted in the outgoing coarse submaps generated by DS1 during its exploration run. These submaps are shared over the network and are received by DS2 (depicted as red voxels in Figure 26b). DS2 is launched a minute after DS1 starts its exploration run. We can observe that DS2 is able to successfully explore regions that were not explored by DS1 and avoid regions previously explored by DS1.

Figure 27 shows the quantitative results from DS1 and DS2 during this experiment. Both figures show the monotonically increasing explored volume as observed by the object detection cameras. The fact that volume explored shows this increasing trend for both time (Figure 27a) and distance (Figure 27b) means that the robots are exhibiting efficient exploration behavior both with respect to time and distance.

The following video shows an example of a single robot exploration run at Brady's Bend mine while the robot is searching for artifacts in the environment.³

3.5.3. Dust Filtering

All of the subterranean environments we have experienced have significant dust accumulation that creates dust clouds when a drone flies through it. The prop wash will blow this dust into the air, which obscures the LiDAR returns. If returns from dust are not filtered from the point cloud, it can cause the local planner to incorrectly identify regions of the environment as untraversable. To address this issue, we have developed a method to quickly filter dust returns.

³ Supplemental file 1.

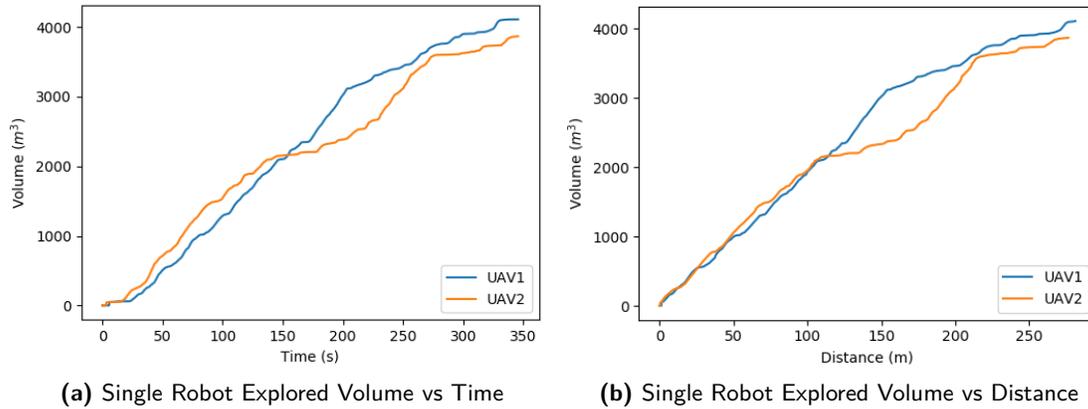


Figure 27. Volume explored by air vehicles DS1 and DS2 individually during the exploration experiment at Brady's Bend illustrated in Figure 26.

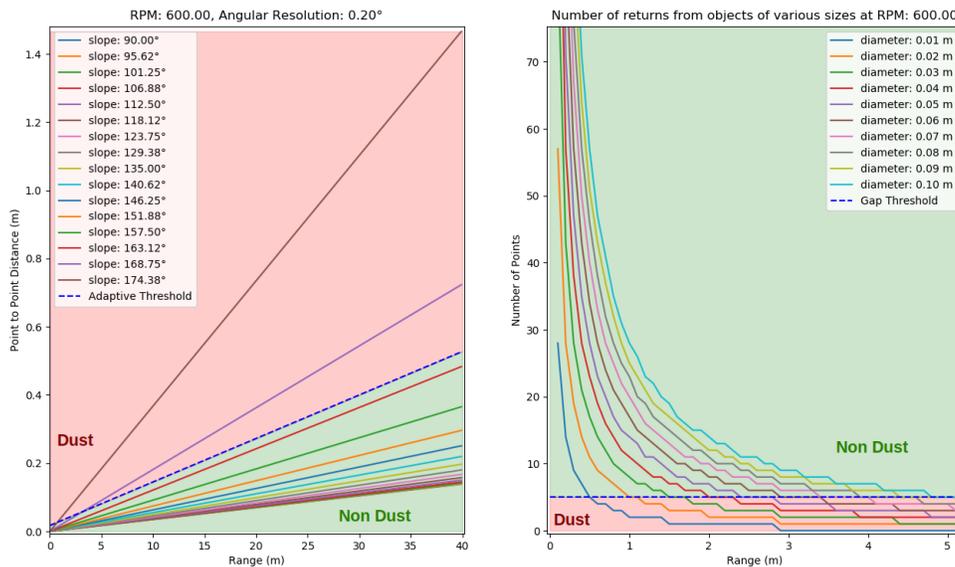


Figure 28. The graph on the left shows the distance between adjacent LiDAR returns from a Velodyne VLP-16 spinning at 600 rpm vs the distance to walls with various slopes. The area below the adaptive threshold line is slopes that will not be filtered out as dust. The graph on the right shows the number of points that will be returned in one scan line vs the distance to objects with various diameters. This is used to predict how close the LiDAR needs to be to an object before it will be detected as an obstacle and not filtered out as dust.

Dust usually appears in the point cloud as a small number of points near the drone that stand out from static obstacles in the environment. Dust is filtered by iterating along each ring of the 16 rings of the LiDAR scanner and checking the distance between consecutive points. If this distance is above some threshold, the pair of points is labeled as dust. This threshold is adaptive based on the distance from the LiDAR since there is naturally more distance between neighboring points from faraway obstacles. If there is some small threshold number of points in between dust points, the in between points are also labeled as dust. This helps to label small clusters of points as dust, but it can incorrectly cause thin obstacles like a rope or wire to be labeled as dust. Figure 28 shows how these parameters effect what is considered dust. This dust filtering is only done from points within 4 m of the drone, since outside of this range the drone does not generate significant dust.

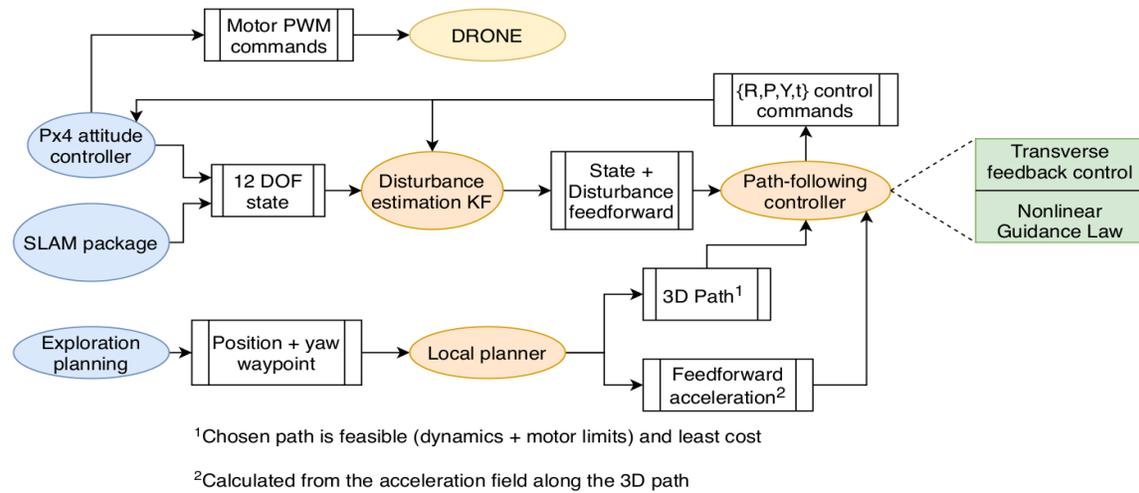


Figure 29. Local planning and control schematic.

3.5.4. Local Planning and Control

The local planning and control layer for the drones is tasked with safely and efficiently visiting the position and yaw viewpoints generated by the exploration planner. The main components are a Snap-continuous Local Planner, a Path-following Controller, and an Online Disturbance Estimator (ODE). A schematic is illustrated in Figure 29.

Snap-continuous Local Planner. The local planner maintains a trajectory library of desired motions, a combination of uniformly distributed and goal-directed primitives, that are sampled at every time step. The library is designed by defining a set of desired motion directions to sample; some are uniformly distributed, and some are aimed at the goal waypoint. These directions (represented by 3D velocities) are passed as set points to a low-level trajectory controller that outputs feasible trajectories that best match the set point, while satisfying kinodynamic constraints and saturation. The trajectory controller outputs a 3D snap vector given a set-point velocity, which is forward-integrated to obtain the trajectory. We design this controller by applying an infinite horizon LQR (linear quadratic regulator) on the triple integrator system from snap to velocity, such that snap limits are obeyed at all times. Since snap is algebraically related to the motor inputs (Mellinger and Kumar, 2011), this is equivalent to enforcing saturation constraints. These primitives are checked for sensor observability and collisions (using a Euclidean distance transform) and the lowest cost primitive is stitched onto the existing path.

Path-following Controller. The key objective behind implementing geometric path-following instead of trajectory tracking was to increase robustness to shocks and delays in tracking. The goal of the controller is to provide smooth convergence to the 3D geometric path obtained from the local planner, instead of tracking a time-parametrized trajectory. We implemented a 3D Nonlinear Guidance Law (Park et al., 2004) algorithm for path-following, which calculates the intersection of a spherical bounding box (centered around the drone) with the 3D path to obtain a virtual tracking point (VTP). Based on this VTP, cross-track error is calculated to be converted to control commands using a PD loop. These control commands are then passed to the PX4 firmware (Meier et al., 2015), as shown in the schematic. Additionally, the acceleration field along the local planner trajectory is used as a feed-forward term, thus improving tracking performance on sharp turns.

Online Disturbance Estimator (ODE). A common problem experienced by heavy drones flying in constrained environments is significant aerodynamic downwash and recirculation generated by the

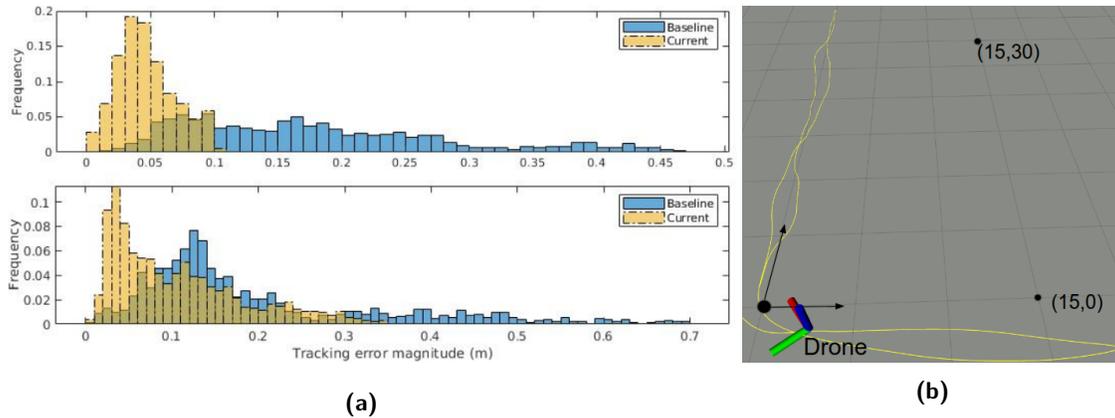


Figure 30. (a) Histogram of tracking error. Top: constrained environments at 0.75 m/s, bottom: navigating a set of waypoints using a Local Planner at 1.5 m/s. (b) Trajectory followed by Local Planner during benchmarking tests.

props, which makes accurate tracking very difficult. Our approach to tackle this problem is to run a disturbance estimator online that learns external disturbances and internal model biases and actively rejects them in real time. This is implemented using an augmented-state Kalman filter (Hentzen et al., 2019), which fuses the internal motion model with pose measurements from SLAM to infer filtered states (linear position and velocity) and disturbances (roll, pitch, and thrust offsets). This Kalman filter runs at 120 Hz and is able to converge rapidly to changes of environment / wind field without any need for integrator terms in the controller. In practice, having good information about the drone model is crucial to the filter estimating and reacting fast enough, which entails explicitly modeling and identifying actuator dynamics and battery voltage dependencies.

The baseline used for comparison and performance evaluation is a previous version of the controls stack that we deployed in the Urban Circuit of the competition. It relies on a time-parametrized tracking law and requires integral action for correcting unmodeled disturbances. We observe significant improvements in robustness and tracking accuracy when we implement the ideas presented in this section, as can be seen from Figure 30.

3.5.5. Takeoff Procedure from Roving Robots

To conserve the drone’s battery life, the onboard computer is powered off until the ground robot is ready to initiate a takeoff. The drone is securely latched onto the ground robot to protect it from being jolted loose while the ground robot is moving. The takeoff procedure begins when the basestation operator commands the ground robot to launch a drone. The ground robot sends a wake-on-lan ethernet packet to the drone’s onboard computer to turn it on. When the drone boots up it launches all of its autonomy processes using a startup script. When the autonomy code is running, it repeatedly sends an “alive” message to the ground robot. When the ground robot receives this, it begins sending an accumulated point cloud of its nearby surroundings. The drone uses its current LiDAR scan to align to the ground robot’s accumulated scan so that it can register to the DARPA coordinate frame. After receiving the accumulated point cloud and performing the registration, the drone sends an acknowledgment to the ground robot and takes off. When the ground robot receives the acknowledgment, it waits 30 s for the drone to take off and travel away. To take off, the drone disengages the latch holding it down, turns on its lights, and starts exploring.

In our current system, the human operator makes the decision to initiate an air robot launch by considering the received maps and the context of the mission. We are also currently investigating ways to automate these launch decisions by reasoning over the exploration value of launch locations and the remaining mission time (Lee et al., 2021).

3.6. Basestation Interaction

Efficiently utilizing the basestation operator maximizes the resilience of the system by allowing the operator to input their creativity, high-level decision making, and robust adaptability. However, it is easy for the cognitive load of the operator to become too high with a large number of robots if each of the robots needs to be carefully supervised. Therefore, we design our system with a sliding autonomy paradigm where the robots are typically fully autonomous; however, the operator can intervene if necessary.

Related Works. There are several paradigms for user interfaces for multiple robots, and a central visual representation of the different robot entities is essential in most multirobot applications, especially those such as the SubT Challenge, where only one human is available for interfacing. CERN has developed a user interface for multirobot remote control for environments hazardous to humans. They allow for user input such as a keyboard and mouse, a haptic robotic arm manipulator, or voice commands (Lunghi et al., 2019). Our method does not pursue haptic interfaces because of the lack of necessary high-precision commands. Team PLUTO’s user interface for the Subterranean Tunnel circuit gives the user eight commands for the robots. Robot paths and artifact locations are viewable in a top-down, 2D view (Miller et al., 2020). Team CTU-CRAS’s user interface from the Tunnel circuit offers UGV teleoperation and control, but no UAV input (Rouček et al., 2019). Our method expands on the latter two methods with three-dimensional visualization and interaction for all robots. This increases user situational awareness as well as user control.

Our Approach. The interface is modular, allowing for a variable number of robots. Calibration and artifact verification are streamlined for the essentials. Robot control is designed to maximise user awareness, relying on autonomy most of the time, while allowing the user to interrupt autonomy at will and manually command robots at both a high level and a low level. This results in maximum user impact and supports our sliding autonomy style of control. The user interface has four main functions: calibration/setup, robot control, user situational awareness, and artifact verification/submission.

Calibration and Setup. After a Total Station survey instrument is calibrated into DARPA’s coordinate system, our Calibration GUI (graphical user interface) is used to correspond points between the Total Station and our robots’ odometries. Once a robot is calibrated, other robots’ point clouds can be aligned to it. In this way, we have a method to automatically calibrate additional robots. Figure 31 shows this GUI. Its simplicity makes it easy to use. It is closed after all robots are calibrated.

Robot	Aerial	Points	Save	Reset	Calibrate	Share Map	Request Map	Lock TF	Has TF	Last Save
1 Ground1	<input type="checkbox"/>	0							<input type="checkbox"/>	
2 Ground2	<input type="checkbox"/>	2							<input type="checkbox"/>	2020-12-18 10:55:08.240114
3 Ground3	<input type="checkbox"/>	0							<input type="checkbox"/>	
4 Aerial1	<input checked="" type="checkbox"/>	0							<input type="checkbox"/>	
5 Aerial2	<input checked="" type="checkbox"/>	0							<input type="checkbox"/>	
6 Aerial3	<input checked="" type="checkbox"/>	0							<input type="checkbox"/>	
7 Aerial4	<input checked="" type="checkbox"/>	0							<input type="checkbox"/>	

Figure 31. Calibration GUI.

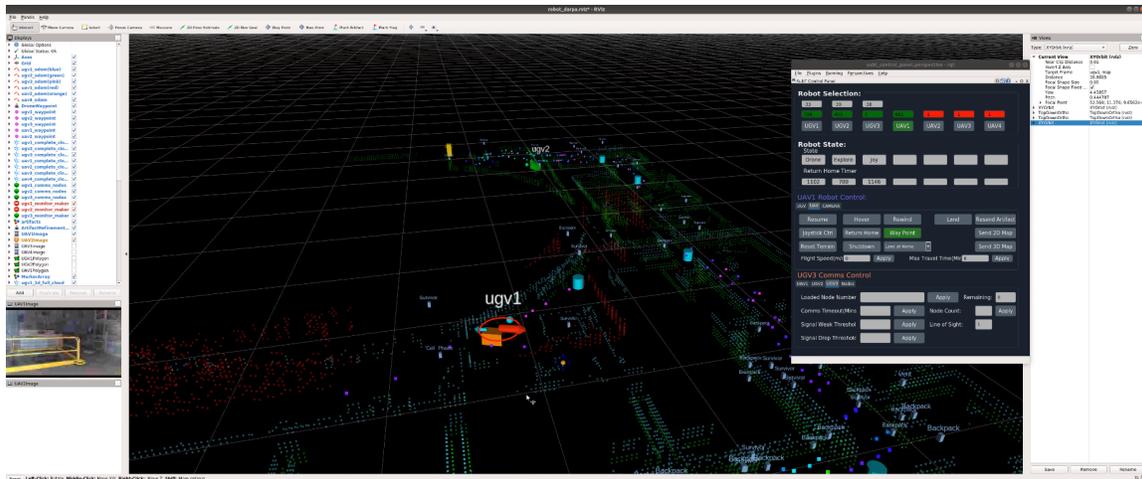


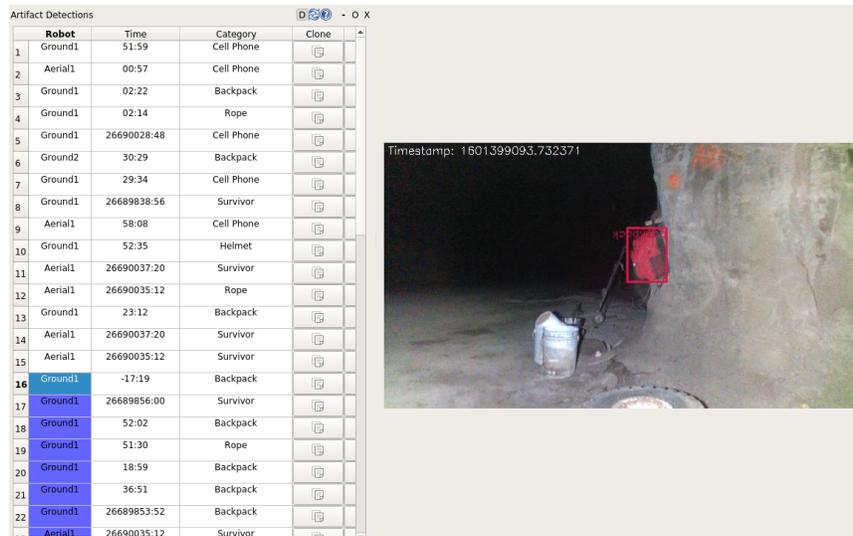
Figure 32. Example screenshot of the 3D visualization and robot control panel during the Urban Circuit event.

User Situational Awareness. The Control Panel provides some basic data such as each robot's current behavior state and communication status. ROS's RViz visualization is the main interface for user situational awareness (see Figure 32); it shows point clouds, odometries, artifact locations, communication node locations, and other spatial data. Its 3D visualization lets the user see the layout of the environment to identify areas of interest, assist robot coordination and exploration, and extract artifact locations. Another GUI is also used that streams back live video from any of the robots' cameras on request from the user, up to nine streams at a time, each with a bandwidth allocation of 10 kbps. In addition to enhancing general environmental awareness, camera streams augment manual joystick teleoperation and have the potential for spurious artifact detections by the user. Therefore, while the robots emphasize robust autonomous operation, the basestation and communication infrastructure create additional system resilience by accommodating human intervention as needed.

Robot Control. The two main user interfaces of robot control are RViz and the Control Panel. RViz is used to send way points, and the Control Panel is used to send additional commands, such as return home, e-stop, toggle joystick teleoperation, launch drone from UGV, etc. RViz and the Control Panel are used in tandem to inject the benefits of user situation awareness into every decision.

Artifact Verification and Submission. Figure 33 shows the Artifact GUI. Potential artifacts are sent from the robots to the basestation with location, type, and images by the communication manager and ledger. As these message types are larger in size than most other messages, and the most important, the communications infrastructure prioritizes successful artifact transmission in all situations, including degraded communications. To prevent potential saturation of the network due to any spurious detections, the communication infrastructure limits artifact detection transmission to 500 kbps per robot. The images are used by the user to verify the artifact, and RViz can be used to precisely refine the position in the event of an imprecise localization. These features maximize the user's power over submissions as their number is limited.

Discussion. We have noticed that the performance of our operator degrades rapidly once their workload becomes too great. If our operator must closely supervise more than two robots and verify artifacts, then we observe that their strategic thinking becomes impaired, and low-level mental tasks, such as monitoring time, following checklists, and directing other teammates, become burdensome. To mitigate this issue, we use two additional strategies beyond user interface improvements. First, we attempt to increase the autonomy and reliability of the robots, as detailed in the above sections,



Robot	Time	Category	Clone
1	Ground1	51:59	Cell Phone
2	Aerial1	00:57	Cell Phone
3	Ground1	02:22	Backpack
4	Ground1	02:14	Rope
5	Ground1	26690028:48	Cell Phone
6	Ground2	30:29	Backpack
7	Ground1	29:34	Cell Phone
8	Ground1	26689838:56	Survivor
9	Aerial1	58:08	Cell Phone
10	Ground1	52:35	Helmet
11	Aerial1	26690037:20	Survivor
12	Aerial1	26690035:12	Rope
13	Ground1	23:12	Backpack
14	Aerial1	26690037:20	Survivor
15	Aerial1	26690035:12	Survivor
16	Ground1	-17:19	Backpack
17	Ground1	26689856:00	Survivor
18	Ground1	52:02	Backpack
19	Ground1	51:30	Rope
20	Ground1	18:59	Backpack
21	Ground1	36:51	Backpack
22	Ground1	26689853:52	Backpack
23	Aerial1	26690035:12	Survivor

Figure 33. Artifact visualization GUI. Left panels list all artifacts from all robots, while right panels show images of potential artifact.

which reduce necessary interactions and supervision. Secondly, we have an additional human playing the role of a “director,” who cannot see or interact with the user interface, but who maintains a strategy guide checklist and confirms that every necessary action from our operator is performed. Human input into the system is a scarce resource as only one person can interact with the basestation. Competent autonomy and the role of the director help conserve the operator’s actions for only the most high-impact tasks possible.

3.7. Software Infrastructure

The large distributed team of developers, the rapid pace, and the large number of robots required for the competition mean that we require a software infrastructure that adapts to the requirements of the challenge. As the circuit events have been progressing, we have been pursuing a *development operations (DevOps)* process of maintaining the stability of the core software deployed to different remote or local systems. Deployment systems span a variety of environment types, such as robot hardware, local development, cluster servers, and cloud computing. While each environment type facilitates its own configuration setup, the constant requirement among all these systems is that we can always run the most recent code changes on said environment choice. Furthermore, as systems scale rapidly, the operations of provisioning, configuration, and updates should maintain the state of the system with simplicity.

In 2010, the term “Cloud Robotics” (Kuffner, 2010) was coined in the field of robotics. Cloud Robot and Automation introduces the ability to leverage vast amounts of highly available, centralized cloud resources over the network, such as computing, storage, and other IoT (Internet of Things), to support robotic operations (Kehoe et al., 2015). Advantages to cloud computing can be found in off-loading computational-intensive tasks, processing big data, and the ability to share databases of information. To centralize the process of integrating robotics systems with the cloud, cohesive frameworks were created such as RoboEarth (Waibel et al., 2011), RoboBrain (Saxena et al., 2015), DAVinCi (Arumugam et al., 2010), and Rapyuta (Mohanarajah et al., 2015a). However, cloud robotics, especially real-time cloud robotics, suffers from key limitations of network QoS (Wang et al., 2015) and privacy and security concerns (Kerns et al., 2014). Nonetheless, cloud potential has been expanding to push the boundaries of real-time robotics problems, such as localization (Zhu et al., 2017), collaborative 3D mapping (Mohanarajah et al., 2015b), and resource retrieval (Wang et al.,

2014a). Our benefit of cloud robotics has been found in terms of utilizing cloud resources for offline multirobot simulations.

3.7.1. Initial Design Choices and Challenges

Initially, our infrastructure leveraged third-party operational tools such as git, submodules, rosinstalls, and docker as independent components. The organization, structure, and maintenance of these tools was left up to the developer leads. The challenge we found was that it quickly became increasingly difficult for users to manage, access, and recreate the same infrastructure on their own systems and similarly on the robot systems. Example problems that arose range from software versioning to docker and direct system dependency conflicts, all of which led to inconsistent systems between users and robots.

Therefore, with these and more issues arising, there was a need to monitor, maintain, and organize a user's organizational setup. The term *organization* refers to Infrastructure as Code (IaC), particularly to the idea of using configuration files as a means of managing and provisioning an environment state for different systems. With more users working remotely during the Cave circuit, we added Microsoft Azure cloud resources, where the goal pertains to creating a multirobot simulation setup on the cloud, configured similarly to the hardware robot setup.

3.7.2. Automating Infrastructure: Pathway to Infrastructure As Code (IaC)

We investigated various solutions, and we found that the following design choices were the most advantageous to our organizational setup:

- Centralized, multilevel meta repo as git submodules for versioning snapshots.
- Docker, docker-compose, docker network setups, docker context for simple swarm management, custom docker accessibility scripts, custom exported environment scenarios.
- Catkin profiles, for grouping different workspace build types.
- Terraform, for management of Microsoft Azure cloud resources.
- Ansible, for automating the installation and configuration of base packages directly on system hosts.
- Deployer (*custom python scripts with yaml as configs*), for automating the operations and deployment process to the various systems.
- Command line, tab-completed user interface layer on top of the *deployer*, for the discovery of operational tools.

The localhost and robot infrastructure deployment workflow can be summarized by Figure 34.

3.7.3. Cloud Simulation Infrastructure

Our goal was to be able to provide the tools and means to simulate competition style runs during any instance of development. Automating system management for cloud resources perceives added complexity for users (Burns et al., 2016). As we found, our local simulation setup would be prone to additional complexity once we layer cloud resources. Our approach was to structure the cloud IaC to match, *as similarly as possible*, the IaC hardware testing procedures. The cloud IaC procedure, using Microsoft Azure cloud resources, is described as follows. Multiple virtual machines are created, initialized, and workspaces are kept in *rsync* between the localhost and remote systems. The VMs are preconfigured with the same network configurations as found on the hardware robots, allowing the communication manager to leverage the setup, for message passing between the virtual machines. Each virtual machine defines its role as a different part of the simulation, such as the ugv or uav robot, basestation GUI controls, or perception. Therefore, we are able to simulate competition runs, easily varying the number of robots used: We perceive the perception VMs relay detected artifacts, found in its respective robot's Gazebo instances, back to the basestation VM, while the operator uses the basestation GUI to monitor the robot's navigation.

The cloud software infrastructure deployment workflow can be summarized in Figure 35.

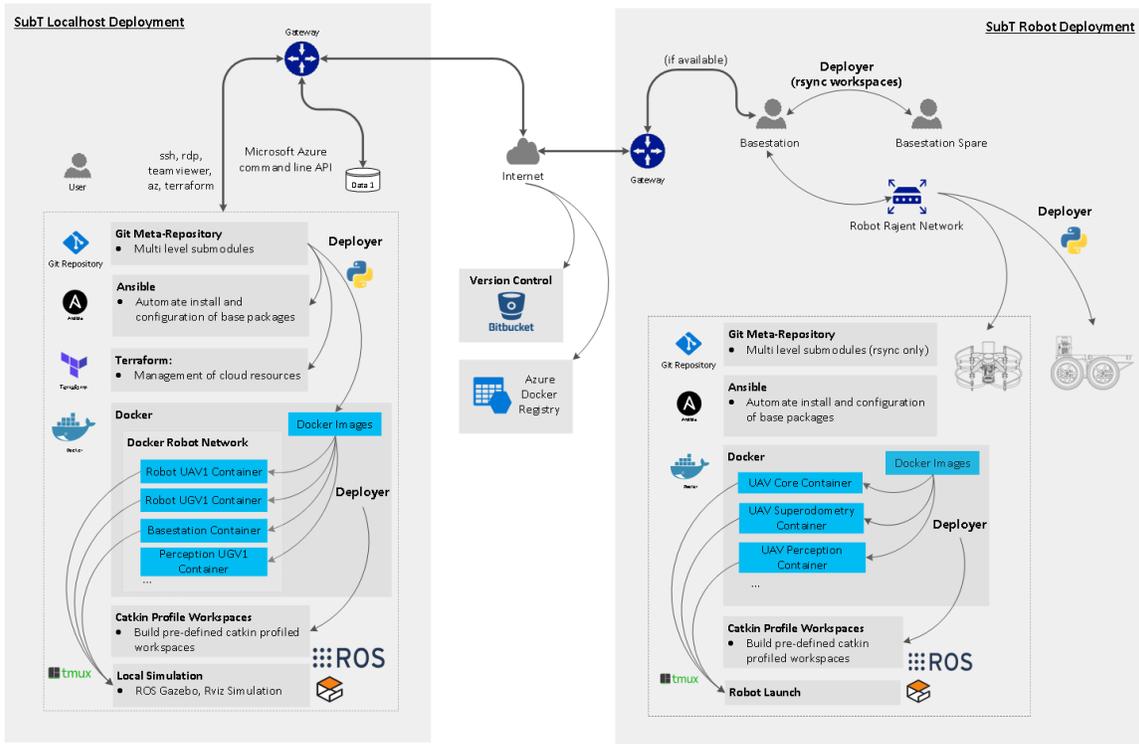


Figure 34. Software localhost and robot deployment flowchart.

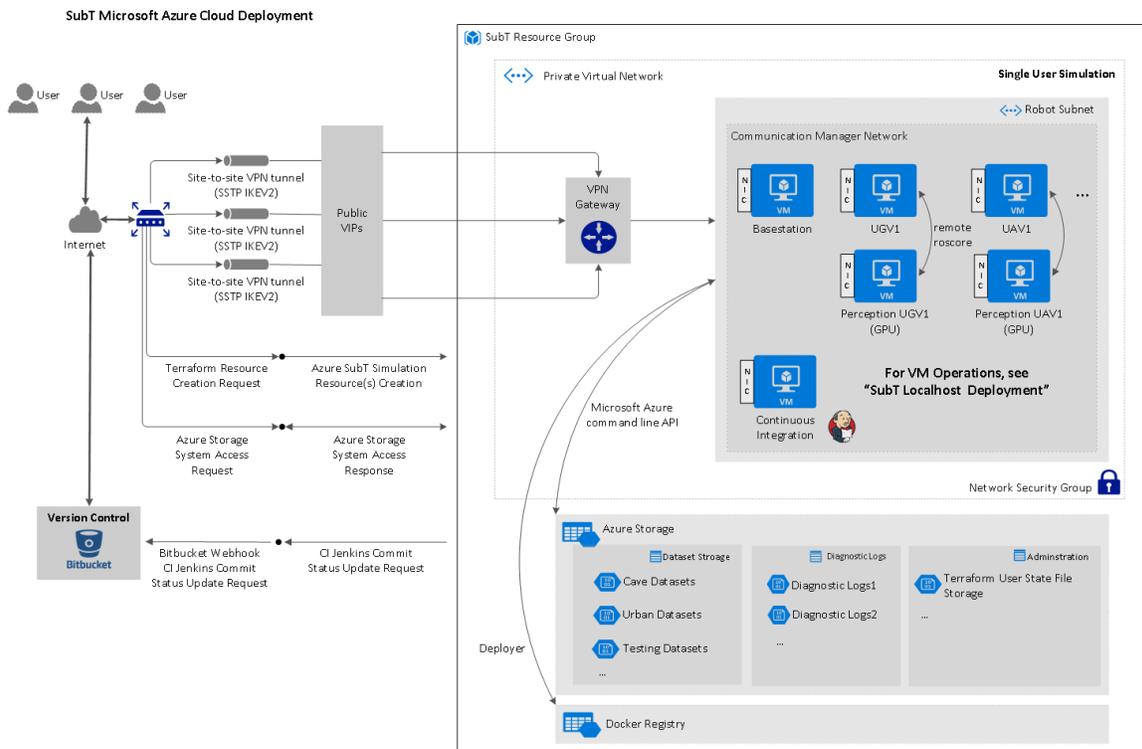


Figure 35. Cloud infrastructure setup.

4. Evaluation and Performance in Circuit Events

Below we outline our performance at the two DARPA-organized circuit events as well as our internally organized Cave Circuit event.

4.1. Performance at the Tunnel Circuit Event

This section summarizes our performance at the Tunnel Circuit event at NIOSH research mine in Pittsburgh, PA. In this circuit, we used a manual coordination strategy sending points that would bias exploration in one direction and manual deployment of communication nodes instead of the algorithms outlined above.

4.1.1. Day 1, August 17, 2019 (*Experimental Research Course*)

Our strategy was to be conservative in our exploration by sending in our R1 ground robot with a full stack of communication nodes to lay out an initial network and to follow with our R2 ground robot. Seeing the tunnel for the first time, we were concerned with the long, smooth passage walls at the start. Our SLAM system and state estimation needs geometric definition, particularly in the early stage, to avoid drift errors in the feedback loop. Our initial calibrations seemed valid with small errors, but those errors quickly accumulated on our initial run into the mine with both robots. We discovered that the maps between both robots had drifted apart. Although we detected artifacts on our initial run, we did not receive scored points, further indicating a significant drift in our state estimates. We brought both robots to the staging area via remote control and started over. This time, we aligned several crew members at the gate to provide more definition for our SLAM software to rigidly define the state and pose. This cost us 20 min of time into our 60 min run window but proved to be the correct decision as our second runs found the initial artifacts and scored points. With an adjusted return to home time for each robot, we explored roughly 1 km of terrain with the robots, resulting in a run score of 10 artifacts. The paths taken by the robots during this run are shown in Figure 36.

After analyzing our first run, we decided that our strategy to lay out an initial communication network using our R1 ground robot prior to sending in our (faster and more agile) R2 ground robot

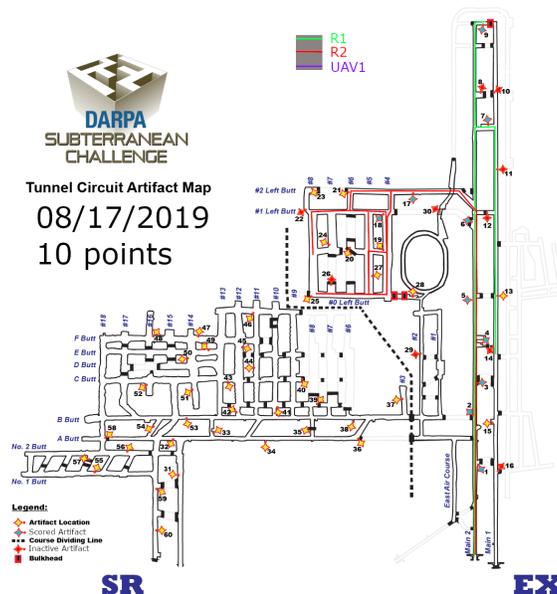


Figure 36. Tunnel Circuit Day 1: Map and robot paths (R1: green; R2: red) for Experimental Research Course run 1.

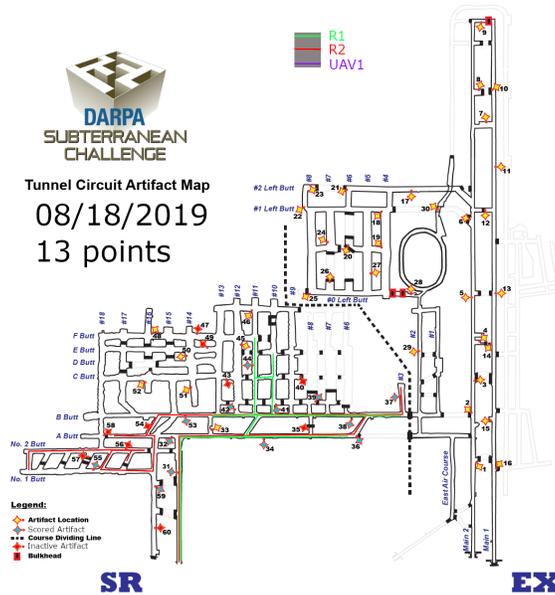


Figure 37. Tunnel Circuit Day 2: Map and robot paths (R1: green; R2: red) for Safety Research Course run 1.

was costing valuable exploration time on the Circuit. We changed our strategy to send in R2 first, exploring deeper into the mine and relying on our DFS (depth first search) planner and return to home function after a period of time, and then to send R1 behind R2 to establish a local area WiFi network before exploring. In this way, we maximized our time on course and still established communications for the latter part of the run. The risk we took assumed we could rely on the exploration and return home ability from the robots. We understood that this risk meant we may not score our points until toward the end of the run, and if our robots became stuck outside of communication range, we may forfeit all points.

4.1.2. Day 2, August 18, 2019 (Safety Research Course)

Our strategy on this day was to follow the “deep exploration first” principle using our R2 ground robot and to follow with our R1 ground robot to lay out a communications network. The discrete geometric features at the entrance to the Safety Research Course eliminated the issues we had the prior day with SLAM drift. We locked into an aligned map for both robots at the start and were able to proceed using the full 60 min run interval after our R2 robot correctly identified the positions of its initial artifacts for early scored points. Both ground robots explored for roughly 40 min before returning home. We did stage our drone for flight after the ground robots but determined that we would gain no additional value in sending in the drone based on the real-time feedback in the first 30 min of the run. We saw no multilevel, obstructed or unexplored areas that would make the drone useful. We abandoned the drone run and relied upon the ground robot performance. We were able to explore roughly 2 km of distance using both ground robots and scored 13 correct artifacts. The robot paths in this run are shown in Figure 37.

4.1.3. Day 3, August 20, 2019 (Safety Research Course)

Our strategy on this day was similar to that on Day 2, but with a few changes. First, we increased the run speed of our R1 robot to match the speed of our R2 robot. The risk we took was that the increased speed on R1 would make it more difficult for our base station human operator to make decisions about when and where to lay out the communication nodes. Our experience shows that the communication nodes we are using need to maintain line-of-site to avoid latency or failure to connect back to the basestation. A faster robot ground speed, combined with inherent latency in

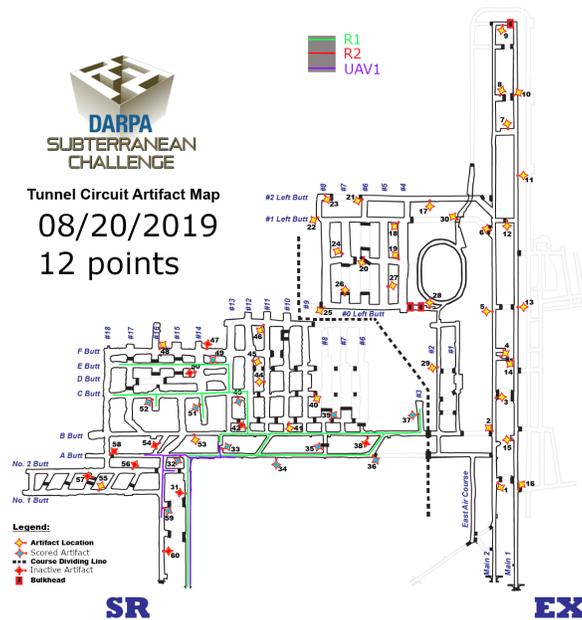


Figure 38. Tunnel Circuit Day 3: Map and robot paths (R1: green; UAV1: purple) for Safety Research Course run 2.

data rates, introduced additional uncertainty for our ability to maintain a coherent communication network due to placement of nodes.

Although our focus was on the risk related to R1 speed, we encountered an initialization problem today with R2 related to FKIE node connections on ROS. We tried restarting our R2 computers and other initialization steps several times before and after the start time but were not successful in bringing R2 status online. Our R1 robot was exploring and we adjusted the return home time while still in communications to extend to 50 min after start. In place of our R2 ground robot, we staged our D1.5 drone and launched it for a 20 min flight time. The drone was coordinated through our basestation to take a separate area of exploration from the R1 ground robot. Our drone successfully flew 500 m into the mine and returned home, scoring us 2 points in addition to R1. Our R1 ground robot successfully explored roughly 1.5 km and returned home, scoring us 10 points. Our combined score for the day was 12 points. The paths are illustrated in Figure 38.

After more evening work and analysis on our R2 robot, we determined that the FKIE failure was a result of a bad DHCP network configuration. We were able to fix this issue for Day 4. We also discovered that R1 on course had roamed into an area with significant fog/smoke and spent several minutes trying to navigate through. Our prior work with dust filtering and testing at Tour Ed Mine in fog/smoke probably saved this run for us. Our path planners were successful in filtering and identifying a safe passage through the occluded areas.

4.1.4. Day 4, August 21, 2019 (Experimental Research Course)

Our strategy today was to focus on eliminating the SLAM drift we had initially on Day 1. Instead of using humans as discrete geometric features, we set up two foldout tables at the edges of the Start gate, placed in an upright position, to add stationary features at the start. We successfully calibrated R1 and R2, both loaded with communication nodes, and both with maximum speed settings to the motors. We launched both robots at the beginning of the 60 min run-time interval. Once inside, we staged our D1.5 drone to fly into the mine. Our basestation manager was successful in managing all three robots simultaneously, sending R1 to the right northeast area, R2 to the left northwest area, and our drone to the northwest area.

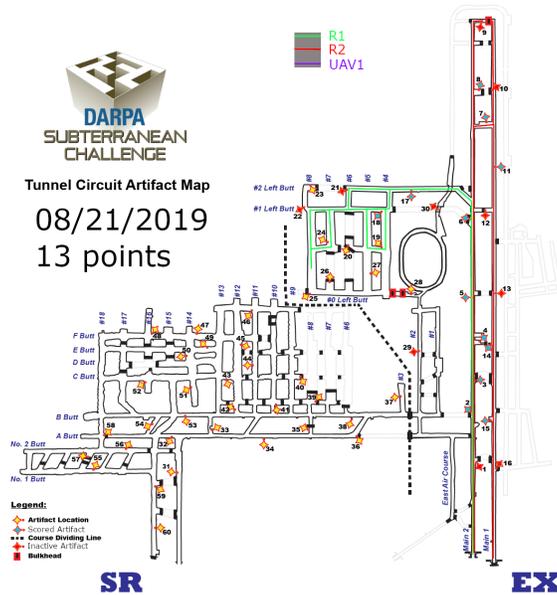


Figure 39. Tunnel Circuit Day 4: Map and robot paths (R1: green; R2: red) for Experimental Research Course run 2.

Our R1 robot made the loop back to the secondary entrance of the mine but became stuck in this area. Therefore, our R1 robot did not return into communication range to report artifacts. Our R2 robot did explore approximately 1.5 km and returned within communication range. Our D1.5 drone did not explore deep into the mine but remained within communication range and was commanded to return home after about 5 min of exploration time. Our R2 ground robot reported 13 successful artifact detections for our final day score of 13 points. The robot paths from this run are shown in Figure 39.

4.2. Performance at Urban Circuit Event

This section summarizes our performance at the Urban Circuit event in Satsop, WA. In this circuit, we added robotic platforms R3 and DS drones to our fleet, as described in the previous section. We used an automated coordination strategy sending points that would bias exploration in one direction and manual deployment of communication nodes instead of the algorithms outlined above.

4.2.1. Day 1, February 21, 2020 (Beta Course)

Our strategy on this day was to gain early situational awareness of the new environment by sending in our R2/DS2 ground/air robots first using a full stack of 9 WiFi nodes to lay out an initial network and to follow with our R1/DS1 ground/air robots. This gave us 4 robots on course within the first 15 min. We had no intention of using our R3 ground robot until any staircases leading towards another level could be identified.

We completed our Start Gate calibrations within the 30 min setup time and were able to launch our R2/DS2 robots at the exact start time. After our R2/DS2 robots provided the initial maps, we had situational awareness of the course before making a decision to send in more robots. After the first 10 min we experienced some communication issues with R2/DS2. The robot was programmed to return home when it lost communication with the basestation. Because of this, we did regain communication with R2/DS2, and our operator command it to go into exploration mode again, but without return to home enabled. After R2/DS2 was out of the communication range, we deployed R1/DS1 on course. Based on the last point-cloud map sent from R2/DS2, our operator identified

a large open area for R1/DS1 to explore and guided R1/DS1 to the area by sending waypoints. After reaching the last waypoint, R1/DS1 was set to autonomous exploration mode. The streaming cameras from R1/DS1 showed the operator two sets of staircases for descending/ascending. At that time, our team calibrated R3 and our operator guided R3 to the first set of stairs by sending waypoints.

After 35 min into the run, R2/DS2 returned to communication range and was returning to the start gate. From the point-cloud map sent back, the operator could see that we had explored a significant area, yet there were no artifacts detected and reported back to the base station. Given the remaining time, the operator sent R2/DS2 to follow R1/DS1, which had become stuck at a far corner of the map. The operator tried to free R1/DS1 using joystick control from the base station but the communication lagged and the operator had to abort to focus on getting R3 to ascend the stairs with 20 min left in the run. The operator again tried to use a joystick controller to guide R3 going up the stairs. However, the communications still lagged and we could not get R3 to follow the commands. After trying for 5 min, the operator had to give up on R3 to focus on going through the artifact list. We were seemingly having issues with our visual detections. Visual artifacts that we could clearly see from our camera streams were not being detected through our object detection software. However, we were able to score 5 points for the run, which tied us with CoStar for the overall lead on Day 1.

Figure 40 shows the point-cloud map and robot trajectories from this run. The green curve shows R2's trajectory while the blue one shows R1's trajectory. The trajectory of R3 is not shown here. The DS1 and DS2 drones were not launched during this run.

4.2.2. Day 2, February 22, 2020 (Alpha Course)

Again, our strategy on this day was to gain early situational awareness of the new environment by sending in our R2/DS2 ground/air robots first and to follow with our R1/DS1 robots. At the Start gate there was a clear staircase visible that descended to a lower level. We chose to send R2/DS2 to explore the main level and attempt to descend the stairs using our R1/DS1 robot.

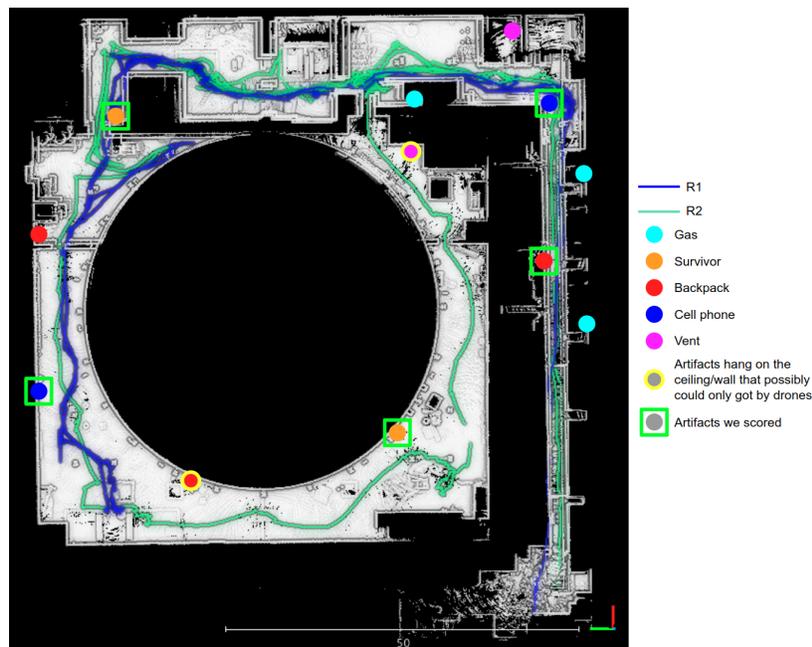


Figure 40. Urban Circuit Day 1: Resulting map and paths from the Beta course. R1's path is shown in blue and R2's is shown in green. The legend shows our performance in scoring artifacts.

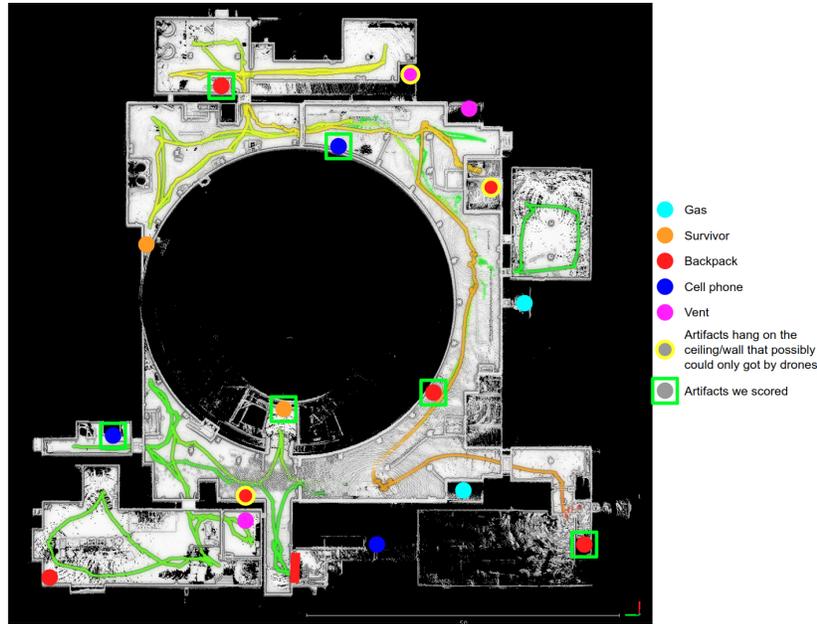


Figure 41. Urban Circuit Day 2: Resulting map from the Alpha course. Only R2's trajectory is shown. The legend shows our performance in scoring artifacts.

While our R2/DS2 robot was exploring in autonomy mode, we focused on getting our R1/DS1 robot down the stairs to another level. We spent over 20 min trying to descend the stairs with R1/DS1. Our operator needed to pause R1/DS1's descent several times to manage R2/DS2 from the basestation to either get it unstuck or to explore a different direction. The difficulty of this course had surprised us relative to prior circuits. After much effort on stair descent and almost reaching the bottom, we eventually flipped our robot forward down the last set of stairs and had to abandon R1/DS1 to focus on R2/DS2.

R2/DS2 did well in exploring on its own. Our operator was able to score a few artifacts based on the camera streams. About 30 min into the run, we deployed R3 into the field to increase ground coverage. However, it later appeared that R3 did not explore much more than R2/DS2 as R2/DS2 had already covered most of the space. We were still having issues with our visual detections but were able to score 6 points for the run, putting us in 1st place overall after Day 2.

Figure 41 shows the point-cloud map and R2's trajectory from the run. The trajectories from R1/DS1/DS2 and R3 are not shown. The DS1 and DS2 drones were not launched during this run.

4.2.3. Day 3, February 25, 2020 (Beta Course)

Our strategy on this day was similar to that on Day 1 but more aggressive in that we planned to utilise our R1/DS1 and R2/DS2 platforms to explore deeper initially toward areas where the drones could be used. The drones would explore lower levels through stairwells or openings, and our R3 platform would ascend any staircases to an upper level. This ideally would give us five robots to explore three different vertical levels of the course.

During the 30 min setup time, we experienced a technical issue with our total station, which had been working for more than a decade without issues. Fortunately, we did have a procedure in case this might happen. Because we were not able to calibrate the robots as usual with our total station, we manually positioned the robots to be at the origin of the Start gate. Doing that for two robots took more than 15 min away from the competition time. Despite that, the operation plan was executed fairly well. R1/DS1 and R2/DS2 explored much of the main level. We launched our DS2 drone from the back of the R2 robot to fly down through a vertical opening in the floor and explore

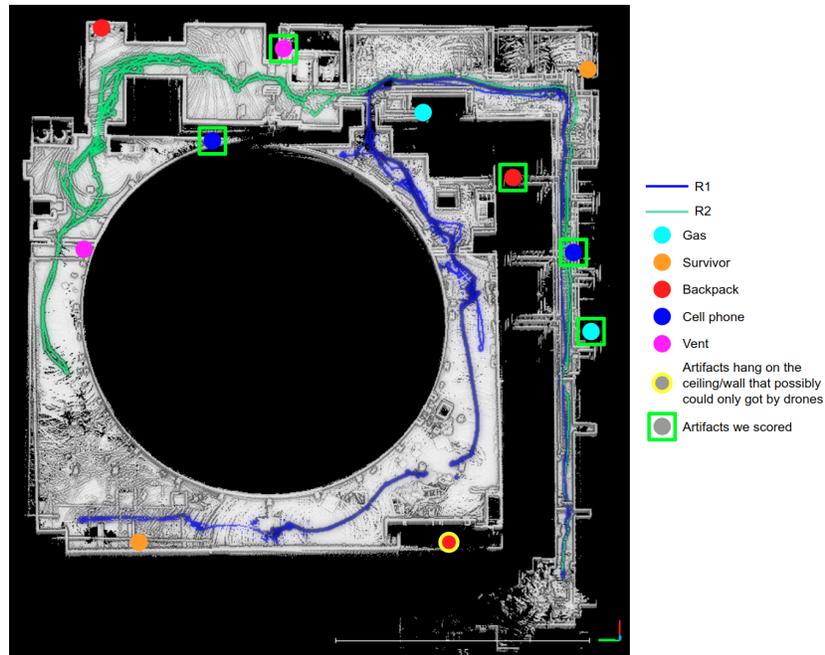


Figure 42. Urban Circuit Day 3: Resulting map and paths on the Beta course. R1's path is shown in blue and R2's is shown in green. The legend shows our performance in scoring artifacts.

the level below. Also, the operator was able to manually drive R3 up a staircase to the mezzanine level above. However, we did not score any points from the two vertical levels. All the points we received were from the main Start gate level.

Post-run analysis showed that we missed a few artifacts during the run. The artifacts were detected but never sent back to the basestation. Further examination of the data showed that the Prism we use for start gate calibration on top of the velodyne was unintentionally left attached and being continuously detected as a false positive by one of the cameras, which flooded the communication network so the true positives were not sent back in time. In addition to all of these issues, the gas sensor and two of the cameras stopped working during the run, causing the artifact detection pipeline to fail at certain points. These failures and mishaps emphasize the need for resilient and robust system operation, and they are a good example of what failure modes can occur under real-time operational scenarios.

Figure 42 shows the point-cloud map and robot trajectories from this run. The green curve shows R2's trajectory, while the blue one shows R1's trajectory. The trajectory of R3 and our drones DS1 and DS2 are not shown here.

4.2.4. Day 4, February 26, 2020 (Alpha Course)

We spent most of the night before our last run attempting to fix the artifact detection issues from Day 3. We were also able to rent a total station from a company in Seattle that was configured as desired. Our strategy on this day was similar to that of Day 2, but this time we decided to send R3 down the stairs as it has a smaller footprint, which could make it easier to maneuver at the stair landings.

During the run we followed the plan. We launched two drones from both the R1 and R2 ground robots on the main floor. One of the drones was able to fly above and over some barriers set up for ground robots. The other one flew for 15 min and covered a large area of the course. While our four robots (R1, R2, DS1, and DS2) were exploring the same floor, the operator tried to go downstairs with our R3 robot. The descent was smooth for the first three sets of stairs and R3 was able to easily

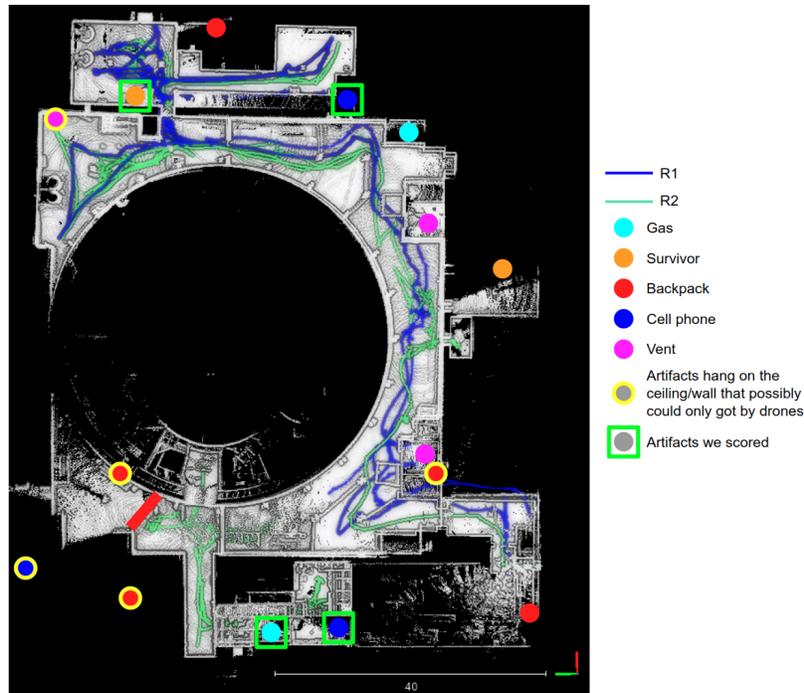


Figure 43. Urban Circuit Day 4: Resulting map and paths on the Alpha course. R1's path is shown in blue and R2's is shown in green. The legend shows our performance in scoring artifacts.

turn at the landings. During the last set of stairs to the lower level, we had some issues with the internal network on R3. The cameras were not streamed back as frequently as usual. Our operator had to rely on the odometry and point-cloud map to determine the pose of R3 after we lost camera visuals. At this point, R3 became stuck on the stair railing. One of the motors of R3 was spinning idle and not listening to commands. The operator then had to give up on getting R3 down and focus on the other robots.

Unfortunately, our fixes to the artifact detection did not seem to work during the run. All of the artifacts we did score were from the camera streaming, identified by the operator. The ground robots did not report true positive detections by themselves. We only scored 4 positive detections on this last day, which placed us in 2nd place overall for the competition.

Additionally, DS1 did not return home to land and instead executed an emergency landing due to a bug in the time logic. Since DS1 did not return home, it was never able to communicate maps and other information back to the operator.

Figure 43 shows the point-cloud map and trajectories of the ground robots from this run, while Figure 44 shows the map and trajectories of the two air vehicles.

4.3. Performance at Self-Organized Cave Circuit Testing

We self-organized a Cave Circuit event at a former limestone mine called Brady's Bend Storage, where we performed testing during the months of July and August, 2020. We established a Start gate and had access to several branches of cavelike terrain at this facility where we were able to survey artifact locations and set up a competition-like course. While Brady's Bend is a limestone mine, it has many cavelike features, such as uneven terrain, open cavernous spaces, as well as narrow openings.

Coordinated and improved exploration of multiple robots was a big focus for our Cave Circuit. The map view shown in Figure 45 was taken from a test at Brady's Bend where both R1 and



(a) DS1's path and generated map.



(b) DS2's path and generated map.

Figure 44. Urban Circuit Day 4 air vehicle flights: Map and paths for (a) DS1 and (b) DS2 on the Alpha course. The legend shows our performance in scoring artifacts. The red bar in (b) shows a terrain barrier for ground robots.

R2 ground robots were started for a competition run. In this case, both robots began at the start gate and were in communication with one another. The first robot chose to take the left sector of the course (shown in green) while the other (shown in blue) received this information and chose accordingly to explore the right sector of the course. Post test analysis confirmed that this behavior was an accurate result of our submap sharing as the human operator from the base station was not commanding this behavior. This gives us confidence that our new ledger and coordination algorithms are working to automate maximum exploration scenarios from the start gate with minimum human intervention.

We conducted several days of competition testing at Brady's Bend in September, 2020. These tests were conducted over a period of two weeks. Our focus during these tests was to emulate a competition environment similar to the anticipated Cave Circuit. We established a calibrated start gate location and a command center located nearby. We had 20 surveyed artifact locations within the course that were used with our DARPA command-post computer to score our runs. We used all of the artifacts that were to be used in the Cave Circuit, including survivors, backpacks,

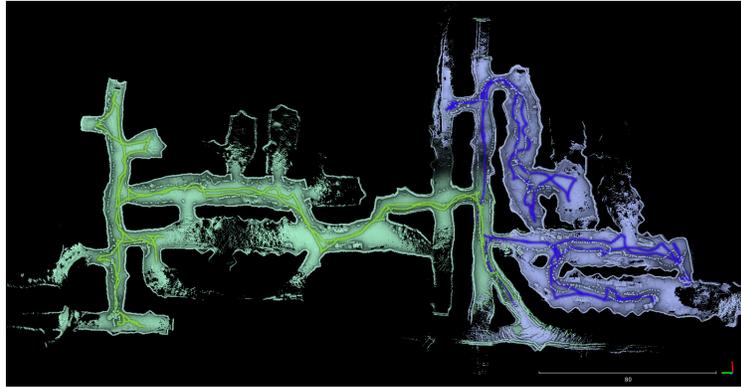


Figure 45. Self-organized Cave Circuit: Resulting map and paths from a test at Brady's Bend. R1's map and path are in blue, while R2's are in green.

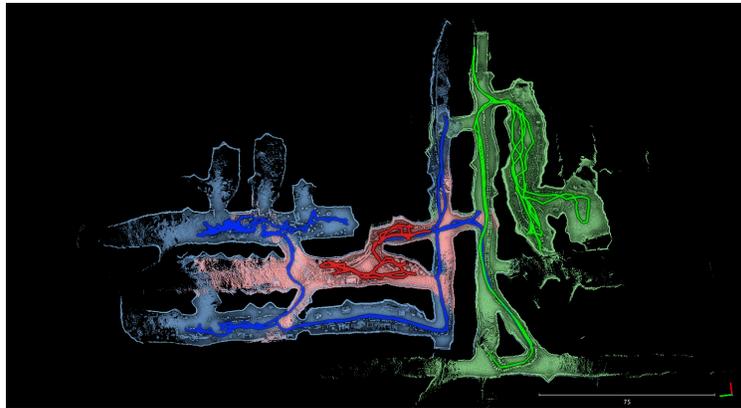


Figure 46. Self-organized Cave Circuit: Resulting map and paths of the competition run at Brady's Bend. R1's map and paths are in blue, while R2's are in green and DS1's are in red.

ropes, helmets with lights, and cell phones. These artifacts were randomly placed at various survey locations throughout the course.

Figure 46 shows the point clouds/map of one of our competition runs at Brady's Bend. During this run, we launched two ground robots (both carrying drones) from the start gate location. This run again highlights the coordination we have developed between the robots such that each robot shares its location with other robots to explore multiple frontiers. Artifacts located along the course are accurately being detected by all of the robots. A video of a similar flight can be seen in supplemental file 3.

5. Lessons Learned

Carefully manage system complexity vs potential performance improvements: The large number of robots, interaction between systems, the autonomy of the robots, and human interaction increase the complexity, which will reduce reliability and also make it difficult to modify the system. As more advanced capability is added, the overall reliability of the system can decrease even though the performance when it does work may be higher.

The challenge requires constant decision making on what advanced features to deploy and when they will add or detract from the overall system. For example, we want the robots to coordinate; however, initial implementations of the coordination communication messages were too large, which

significantly degrades the performance of the communication network and has follow on effects to other autonomy components of the system. In another example, terrain evaluation can make the robot safer and help to prevent it from falling off stairs or other overhangs. On the other hand, false positives in the terrain evaluation will cause the robot to unnecessarily avoid areas that are reachable. This complexity increases operator cognitive load and can result in reduced overall performance.

Resolve your weakest links: A failure of a single component can degrade or prevent the system from working as desired, and we have experienced some of these failures during the circuit events. For example, if communication is not possible, the robots cannot report artifacts, or if the camera connections are unreliable, then no artifacts can be detected. One of the ways these issues can be prevented is with a strong and rigorous system engineering and testing program.

Be mindful of the basestation operator’s cognitive ability: One of the difficulties of the SubT challenge is that only a single operator can control the entire system, and it is easy to exceed the cognitive capability of the operator. There is a distinct threshold between an operator that is well-tasked and an operator that is overloaded. An overloaded operator makes mistakes and starts to focus only on the most urgent task, forgets about other robots, and cannot consider the larger strategy of the challenge. The way we overcome this is by having a “director” that communicates with the operator to monitor the operator’s performance to ensure that the operator does not become overloaded.

No single system or algorithm can address all challenges: Subterranean environments are extremely varied, from large open spaces to narrow openings, vertical shafts, and long traverses. Every mode of locomotion has its advantages and disadvantages. Our roving robots have a long range, speed, and can traverse larger obstacles; however, they have a limited ability on slopes and steps. Flying robots, on the other hand, can overcome any mobility challenge, but they cannot enter very narrow spaces, and they have a limited flight time. Walking robots perform suboptimally on slippery ground, such as in muddy mines, but they are optimized for steps and narrow spaces.

Combined exploration and coverage problem: While often treated as a pure exploration problem, the SubT challenge is actually a simultaneous exploration and coverage problem. Our range sensors reveal the geometry at a much longer (~ 100 m) distance than the artifacts can be detected (~ 5 m). Additionally, artifacts can only be detected if a robot looks at them. For example, at the urban circuit, our drone exploration planner resulted in good discovery of the overall topology of the environment; however, artifacts were missed due to insufficient close-range observations of all surfaces by our artifact detection sensors. We failed to detect some artifacts that were clearly along our flight path but located at a distance, orientation, or elevation that was not in the field of view of our object detection cameras. This presents an exploration and exploitation (coverage) tradeoff where the designer needs to decide how much to prioritize coverage versus exploring new areas.

6. Conclusions and Future Work

Our approach to the system architecture, hardware, and algorithms focuses on embodying the themes of resilience and modularity to maximize success. These themes permeate throughout the design of our algorithms, hardware, as well as infrastructure.

Around these themes we have built an approach to address the DARPA SubT Challenge, and we have realized a team of coordinating roving and flying vehicles that can explore and cover a wide range of spaces while detecting objects of interest. With this approach, we were able to win the Tunnel Circuit event and place second in the Urban Circuit event.

Our system continues to evolve based on the demands of the challenge, and a driving factor in increasing the performance has been the goal of reducing the cognitive load on the single operator by increasing the autonomy of the robots and automating tasks that currently need to be performed by the operator.

In future work, we intend to extend our approach to legged systems, more capable drones, more robust SLAM, and to increase the autonomy and deconfliction ability of our robots. The challenge has shown us that many unsolved longer-term research problems exist. For example, one future

research direction is that the strategy for optimal tasking of a heterogeneous fleet of robots with a map and an exploration and coverage problem that depend on the environment is currently solved only heuristically. Other research directions are an even more robust state estimation system that utilizes more sensor modalities, predictive communication coverage, more accurate Wi-Fi and gas artifact localization, and tighter coordination between heterogeneous robots.

Acknowledgments

This work would not have been possible without the dedicated efforts of Team Explorer and the generous support of our sponsors. Approved for public release; distribution is unlimited. This research was sponsored by DARPA (#HR00111820044). Content is not endorsed by and does not necessarily reflect the position or policy of the government or our sponsors.

ORCID

Sebastian Scherer  <https://orcid.org/0000-0002-8373-4688>
 Vasu Agrawal  <https://orcid.org/0000-0003-3077-1212>
 Graeme Best  <https://orcid.org/0000-0003-0443-8248>
 Chao Cao  <https://orcid.org/0000-0002-4192-7641>
 Ryan Darnley  <https://orcid.org/0000-0002-3677-0627>
 Robert DeBortoli  <https://orcid.org/0000-0003-1460-1635>
 Eric Dexheimer  <https://orcid.org/0000-0003-2402-6742>
 Bill Drozd  <https://orcid.org/0000-0001-9994-8990>
 Rohit Garg  <https://orcid.org/0000-0001-9646-224X>
 Ian Higgins  <https://orcid.org/0000-0003-4921-4344>
 John Keller  <https://orcid.org/0000-0002-3672-6212>
 David Kohanbash  <https://orcid.org/0000-0002-9526-7458>
 Lucas Nogueira  <https://orcid.org/0000-0001-7220-2937>
 Vaibhav K. Viswanathan  <https://orcid.org/0000-0002-8722-8096>
 Shibo Zhao  <https://orcid.org/0000-0003-1801-8156>
 Greg Armstrong  <https://orcid.org/0000-0002-3393-7954>
 Kevin Pluckter  <https://orcid.org/0000-0002-1958-9367>
 Andrew Saba  <https://orcid.org/0000-0003-2462-2050>
 Manish Saroya  <https://orcid.org/0000-0002-0807-4173>
 Emily Scheide  <https://orcid.org/0000-0002-5682-6245>
 Nathaniel Shoemaker-Trejo  <https://orcid.org/0000-0001-9769-9038>
 Fan Yang  <https://orcid.org/0000-0002-4452-4979>
 Howie Choset  <https://orcid.org/0000-0002-2266-8744>
 Anthony Rowe  <https://orcid.org/0000-0003-2332-9450>
 Sanjiv Singh  <https://orcid.org/0000-0001-5412-2888>
 Geoffrey A. Hollinger  <https://orcid.org/0000-0001-6502-8950>

References

- Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al. (2016). Tensorflow: A system for large-scale machine learning. In *Proc. USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 265–283.
- Agha, A., Mitchell, K. L., and Boston, P. (2019). Robotic exploration of planetary subsurface voids in search for life. In *Proc. AGU Fall Meeting Abstracts*, volume 2019, pages P41C–3463.
- Agrawal, K. and Subramanian, A. (2019). Enhancing object detection in adverse conditions using thermal imaging. *arXiv preprint arXiv:1909.13551*.
- Agrawal, V. (2019). Ground up design of a multi-modal object detection system. Master’s thesis, Carnegie Mellon University, Pittsburgh, PA.

- Arumugam, R., Enti, V. R., Bingbing, L., Xiaojun, W., Baskaran, K., Kong, F. F., Kumar, A. S., Meng, K. D., and Kit, G. W. (2010). DAVinCi: A cloud computing framework for service robots. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 3084–3089.
- Arun, K. S., Huang, T. S., and Blostein, S. D. (1987). Least-squares fitting of two 3-D point sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-9(5):698–700.
- Assayag, Y., Oliveira, H., Souto, E., Barreto, R., and Pazzi, R. (2020). Indoor positioning system using dynamic model estimation. *Sensors*, 20(24):7003.
- Baker, C. R., Morris, A. C., Ferguson, D., Thayer, S., Whittaker, C., Omohundro, Z., Reverte, C., Whittaker, W. R. L., Haehnel, D., and Thrun, S. (2004). A campaign in autonomous mine mapping. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, volume 2, pages 2004–2009.
- Best, G., Faigl, J., and Fitch, R. (2018). Online planning for multi-robot active perception with self-organising maps. *Autonomous Robots*, 42(4):715–738.
- Bircher, A., Kamel, M., Alexis, K., Oleynikova, H., and Siegwart, R. (2016). Receding horizon “next-best-view” planner for 3D exploration. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*.
- Bouman, A., Ginting, M. F., Alatur, N., Palieri, M., Fan, D. D., Touma, T., Pailevanian, T., Kim, S.-K., Otsu, K., Burdick, J., et al. (2020). Autonomous spot: Long-range autonomous exploration of extreme environments with legged locomotion. *arXiv preprint arXiv:2010.09259*.
- Bradski, G. and Kaehler, A. (2000). OpenCV. *Dr. Dobb’s journal of software tools*, 3.
- Burgard, W., Moors, M., Stachniss, C., and Schneider, F. E. (2005). Coordinated multi-robot exploration. *IEEE Transactions on Robotics*, 21(3):376–386.
- Burns, B., Grant, B., Oppenheimer, D., Brewer, E., and Wilkes, J. (2016). Borg, Omega, and Kubernetes. *ACM Queue*, 14:70–93.
- Burri, M., Nikolic, J., Gohl, P., Schneider, T., Rehder, J., Omari, S., Achtelik, M. W., and Siegwart, R. (2016). The EuRoC micro aerial vehicle datasets. *The International Journal of Robotics Research*, 35(10):1157–1163.
- Chao, C., Hongbiao, Z., Howie, C., and Ji, Z. (2021). Exploring large and complex environments fast and efficiently. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*.
- Charrow, B., Kahn, G., Patil, S., Liu, S., Goldberg, K., Abbeel, P., Michael, N., and Kumar, V. (2015). Information-theoretic planning with trajectory optimization for dense 3D mapping. In *Proc. Robotics: Science and Systems*, volume 11.
- Choudhury, S., Dugar, V., Maeta, S., MacAllister, B., Arora, S., Althoff, D., and Scherer, S. (2019). 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.
- Colledanchise, M. and Ögren, P. (2018). *Behavior trees in robotics and AI: An introduction*. CRC Press.
- Corah, M. and Michael, N. (2019). Distributed matroid-constrained submodular maximization for multi-robot exploration: Theory and practice. *Autonomous Robots*, 43(2):485–501.
- Dang, T., Khattak, S., Mascarich, F., and Alexis, K. (2019). Explore locally, plan globally: A path planning framework for autonomous robotic exploration in subterranean environments. In *Proc. International Conference on Advanced Robotics (ICAR)*, pages 9–16.
- Dang, T., Mascarich, F., Khattak, S., Papachristos, C., and Alexis, K. (2019). Graph-based path planning for autonomous robotic exploration in subterranean environments. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.
- DARPA (2019). DARPA subterranean (SubT) challenge. <https://www.darpa.mil/program/darpa-subterranean-challenge>.
- Dellaert, F. and Kaess, M. (2017). *Factor Graphs for Robot Perception*. Now Publishers Inc.
- Dias, M. B., Kannan, B., Browning, B., Jones, E., Argall, B., Dias, M. F., Zinck, M. B., Veloso, M., and Stentz, A. T. (2008). Sliding autonomy for peer-to-peer human-robot teams. Technical Report CMU-RI-TR-08-16, Carnegie Mellon University, Pittsburgh, PA.
- Dwivedi, D., Misra, I., and Hebert, M. (2017). Cut, paste and learn: Surprisingly easy synthesis for instance detection. In *Proc. IEEE International Conference on Computer Vision*, pages 1301–1310.
- Ebadi, K., Chang, Y., Palieri, M., Stephens, A., Hatteland, A., Heiden, E., Thakur, A., Funabiki, N., Morrell, B., Wood, S., et al. (2020). LAMP: Large-scale autonomous mapping and positioning for exploration of perceptually-degraded subterranean environments. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 80–86.
- Feige, U. (1998). A threshold of $\ln n$ for approximating set cover. *Journal of the ACM*, 45:314–318.

- Ferguson, D., Morris, A., Haehnel, D., Baker, C., Omohundro, Z., Reverte, C., Thayer, S., Whittaker, C., Whittaker, W., Burgard, W., et al. (2003). An autonomous robotic system for mapping abandoned mines. *Advances in Neural Information Processing Systems*, 16:587–594.
- Heger, F. W. and Singh, S. (2006). Sliding autonomy for complex coordinated multi-robot tasks: Analysis & experiments. In *Proc. Robotics: Science and Systems*.
- Hentzen, D., Stastny, T., Siegwart, R., and Brockers, R. (2019). Disturbance estimation and rejection for high-precision multirotor position control. *arXiv CoRR*, abs/1908.03166.
- Hollnagel, E. (2011). RAG-the resilience analysis grid. *Resilience engineering in practice. A guidebook*. Farnham, UK: Ashgate, pages 275–296.
- Hornung, A., Wurm, K. M., Bennewitz, M., Stachniss, C., and Burgard, W. (2013). OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*, 34(3):189–206.
- Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.
- Husain, A., Jones, H., Kannan, B., Wong, U., Pimentel, T., Tang, S., Daftry, S., Huber, S., and Whittaker, W. L. (2013). Mapping planetary caves with an autonomous, heterogeneous robot team. In *Proc. IEEE Aerospace Conference*, pages 1–13.
- Jordan, M. and Perez, A. (2013). Optimal bidirectional rapidly-exploring random trees. Technical Report MIT-CSAIL-TR-2013-021, MIT Computer Science and Artificial Intelligence Laboratory.
- Kehoe, B., Patil, S., Abbeel, P., and Goldberg, K. (2015). A survey of research on cloud robotics and automation. *IEEE Transactions on Automation Science and Engineering*, 12(2):398–409.
- Kerns, A., Shepard, D., Bhatti, J., and Humphreys, T. (2014). Unmanned aircraft capture and control via GPS spoofing. *Journal of Field Robotics*, 31.
- Khattak, S., Nguyen, H., Mascarich, F., Dang, T., and Alexis, K. (2020). Complementary multi-modal sensor fusion for resilient robot pose estimation in subterranean environments. In *Proc. IEEE International Conference on Unmanned Aircraft Systems (ICUAS)*.
- Khattak, S., Papachristos, C., and Alexis, K. (2019). Keyframe-based direct thermal-inertial odometry. *Proc. IEEE International Conference on Robotics and Automation (ICRA)*.
- Kramer, A., Kasper, M., and Heckman, C. (2019). VI-SLAM for subterranean environments. In *Proc. Field and Service Robotics*, pages 159–172.
- Kuffner, J. (2010). Cloud-enabled humanoid robots. In *Proc. IEEE-RAS International Conference on Humanoid Robots*.
- Lauri, M. and Ritala, R. (2016). Planning for robotic exploration based on forward simulation. *Robotics and Autonomous Systems*, 83:15–31.
- LaValle, S. M. (1998). Rapidly-exploring random trees: A new tool for path planning. Technical report.
- Lee, C., Best, G., and Hollinger, G. A. (2021). Optimal sequential stochastic deployment of multiple passenger robots. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*.
- Li, H., Savkin, A. V., and Vucetic, B. (2018). Collision free navigation of a flying robot for underground mine search and mapping. In *Proc. IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 1102–1106.
- Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. (2016). SSD: Single shot multibox detector. In *Proc. European Conference on Computer Vision*, pages 21–37.
- Lunghi, G., Marin, R., Castro, M. D., Masi, A., and Sanz, P. (2019). Multimodal human-robot interface for accessible remote robotic interventions in hazardous environments. *IEEE Access*, 7:127290–127319.
- Lösch, R., Grehl, S., Donner, M., Buhl, C., and Jung, B. (2018). Design of an autonomous robot for mapping, navigation, and manipulation in underground mines. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1407–1412.
- Markoff, J. (1991). The creature that lives in Pittsburgh. *The New York Times*.
- Meier, L., Honegger, D., and Pollefeys, M. (2015). PX4: A node-based multithreaded open source robotics framework for deeply embedded platforms. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 6235–6240.
- Mellinger, D. and Kumar, V. (2011). Minimum snap trajectory generation and control for quadrotors. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 2520–2525.
- Miller, I. D., Cladera, F., Cowley, A., Shivakumar, S. S., Lee, E. S., Jarin-Lipschitz, L., Bhat, A., Rodrigues, N., Zhou, A., Cohen, A., Kulkarni, A., Laney, J., Taylor, C. J., and Kumar, V. (2020). Mine tunnel exploration using multiple quadrupedal robots. *IEEE Robotics and Automation Letters*, 5(2):2840–2847.

- Mohanarajah, G., Hunziker, D., D'Andrea, R., and Waibel, M. (2015a). Rapyuta: A cloud robotics platform. *IEEE Transactions on Automation Science and Engineering*, 12(2):481–493.
- Mohanarajah, G., Usenko, V., Singh, M., D'Andrea, R., and Waibel, M. (2015b). Cloud-based collaborative 3D mapping in real-time with low-cost robots. *IEEE Transactions on Automation Science and Engineering*, 12(2):423–431.
- Morris, A. C., Ferguson, D., Omohundro, Z., Bradley, D., Silver, D., Baker, C. R., Thayer, S., Whittaker, C., and Whittaker, W. R. L. (2006). Recent developments in subterranean robotics. *Journal of Field Robotics*, 23(1):35–57.
- Museth, K. (2013). VDB: High-resolution sparse volumes with dynamic topology. *ACM Transactions on Graphics (TOG)*, 32(3):1–22.
- Nagatani, K., Kiribayashi, S., Okada, Y., Otake, K., Yoshida, K., Tadokoro, S., Nishimura, T., Yoshida, T., Koyanagi, E., Fukushima, M., et al. (2013). Emergency response to the nuclear accident at the Fukushima daiichi nuclear power plants using mobile rescue robots. *Journal of Field Robotics*, 30(1):44–63.
- Papachristos, C., Khattak, S., Mascarich, F., and Alexis, K. (2019). Autonomous navigation and mapping in underground mines using aerial robots. In *Proc. IEEE Aerospace Conference*, pages 1–8.
- Pardo-Castellote, G. (2003). OMG data-distribution service: Architectural overview. In *Proc. 23rd International Conference on Distributed Computing Systems Workshops*, pages 200–206.
- Park, S., Deyst, J., and How, J. (2004). A new nonlinear guidance logic for trajectory tracking. In *Proc. AIAA Guidance, Navigation, and Control Conference and Exhibit*, page 4900.
- Qin, T., Li, P., and Shen, S. (2018). Vins-mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Transactions on Robotics*, 34(4):1004–1020.
- Rouček, T., Pecka, M., Čížek, P., Petříček, T., Bayer, J., Šalanský, V., Heřt, D., Petrлік, M., Báča, T., Spurný, V., et al. (2019). DARPA subterranean challenge: Multi-robotic exploration of underground environments. In *Proc. International Conference on Modelling and Simulation for Autonomous Systems*, pages 274–290.
- Santamaria-Navarro, A., Thakker, R., Fan, D. D., Morrell, B., and Agha-mohammadi, A.-a. (2020). Towards resilient autonomous navigation of drones. *arXiv preprint arXiv:2008.09679*.
- Saroya, M., Best, G., and Hollinger, G. A. (2020). Online exploration of tunnel networks leveraging topological CNN-based world predictions. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.
- Saxena, A., Jain, A., Sener, O., Jami, A., Misra, D. K., and Koppula, H. S. (2015). Robobrain: Large-scale knowledge engine for robots. *arXiv: cs.AI*.
- Scheide, E., Best, G., and Hollinger, G. A. (2021). Behavior tree learning for robotic task planning through Monte Carlo DAG search over a formal grammar. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*.
- Segal, A., Haehnel, D., and Thrun, S. (2009). Generalized-ICP. In *Proc. Robotics: Science and Systems*, volume 2, page 435.
- Shaffer, G. K., Stentz, A., Whittaker, W. L., and Fitzpatrick, K. W. (1992). Position estimator for underground mine equipment. *IEEE Transactions on Industry Applications*, 28(5):1131–1140.
- Shah, S., Dey, D., Lovett, C., and Kapoor, A. (2018). Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In *Proc. Field and Service Robotics*, pages 621–635.
- Stanford Artificial Intelligence Laboratory et al. (2018). Robotic operating system (melodic morenia). <https://www.ros.org>.
- Sukkar, F., Best, G., Yoo, C., and Fitch, R. (2019). Multi-robot region-of-interest reconstruction with DecMCTS. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 9101–9107.
- Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826.
- Tabib, W., Corah, M., Michael, N., and Whittaker, R. (2016). Computationally efficient information-theoretic exploration of pits and caves. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3722–3727.
- Tatum, M. (2020). Communications coverage in unknown underground environments. *Masters's thesis, The Robotics Institute, Carnegie Mellon University, USA*.
- Thrun, S., Thayer, S., Whittaker, W., Baker, C., Burgard, W., Ferguson, D., Hahnel, D., Montemerlo, D., Morris, A., Omohundro, Z., et al. (2004). Autonomous exploration and mapping of abandoned mines. *IEEE Robotics & Automation Magazine*, 11(4):79–91.

- Vidal, E., Palomeras, N., Istenič, K., Gracias, N., and Carreras, M. (2020). Multisensor online 3D view planning for autonomous underwater exploration. *Journal of Field Robotics*, 37(6):1123–1147.
- Waibel, M., Beetz, M., Civera, J., D’Andrea, R., Elfving, J., Galvez-Lopez, D., Häussermann, K., Janssen, R., Montiel, J., Perzylo, A., Schieffle, B., Tenorth, M., Zweigle, O., and Molengraaf, M. (2011). Roboearth—A world wide web for robots. *IEEE Robotics & Automation Magazine*, 18:69–82.
- Wang, L., Liu, M., and Meng, M. (2014a). Hierarchical auction-based mechanism for real-time resource retrieval in cloud mobile robotic system. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 2164–2169.
- Wang, L., Liu, M., and Meng, M. Q.-H. (2015). Real-time multisensor data retrieval for cloud robotic systems. *IEEE Transactions on Automation Science and Engineering*, 12(2):507–518.
- Wang, W., Dong, W., Su, Y., Wu, D., and Du, Z. (2014b). Development of search-and-rescue robots for underground coal mine applications. *Journal of Field Robotics*, 31(3):386–407.
- Wisth, D., Camurri, M., and Fallon, M. (2020). Preintegrated velocity bias estimation to overcome contact nonlinearities in legged robot odometry. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 392–398.
- Yamauchi, B. (1997). A frontier-based approach for autonomous exploration. In *Proc. IEEE International Symposium on Computational Intelligence in Robotics and Automation*, pages 146–151.
- Yarkan, S., Guzelgoz, S., Arslan, H., and Murphy, R. R. (2009). Underground mine communications: A survey. *IEEE Communications Surveys & Tutorials*, 11(3):125–142.
- Ye, H., Chen, Y., and Liu, M. (2019). Tightly coupled 3D lidar inertial odometry and mapping. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*. IEEE.
- Zhang, J., Hu, C., Chadha, R. G., and Singh, S. (2019). Maximum likelihood path planning for fast aerial maneuvers and collision avoidance. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.
- Zhang, J., Hu, C., Chadha, R. G., and Singh, S. (2020). Falco: Fast likelihood-based collision avoidance with extension to human-guided navigation. *Journal of Field Robotics*, 37(8):1300–1313.
- Zhang, J. and Singh, S. (2014). LOAM: Lidar odometry and mapping in real-time. In *Proc. Robotics: Science and Systems*, volume 2.
- Zhang, J. and Singh, S. (2018). Laser-visual-inertial odometry and mapping with high robustness and low drift. *Journal of Field Robotics*, 35(8):1242–1264.
- Zhao, S., Fang, Z., Li, H., and Scherer, S. (2019). A robust laser-inertial odometry and mapping method for large-scale highway environments. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1285–1292.
- Zhao, S., Wang, P., Zhang, H., Fang, Z., and Scherer, S. (2020). TP-TIO: A robust thermal-inertial odometry with deep ThermalPoint. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4505–4512.
- Zhao, S., Zhang, H., Wang, P., Nogueira, L., and Scherer, S. (2021). Super Odometry: IMU-centric LiDAR-visual-inertial estimator for challenging environments. <https://ieeexplore.ieee.org/document/9635862>.
- Zhou, B., Zhang, Y., Chen, X., and Shen, S. (2021). FUEL: Fast UAV exploration using incremental frontier structure and hierarchical planning. *IEEE Robotics and Automation Letters*, 6(2):779–786.
- Zhu, X., Qiu, C., Deng, F., Pang, S., and Ou, Y. (2017). Cloud-based real-time outsourcing localization for a ground mobile robot in large-scale outdoor environments: Cloud-based real-time outsourcing localization. *Journal of Field Robotics*, 34(7):1313–1331.

How to cite this article: Scherer, S., Agrawal, V., Best, G., Cao, C., Cujic, K., Darnley, R., DeBortoli, R., Dexheimer, E., Drozd, B., Garg, R., Higgins, I., Keller, J., Kohanbash, D., Nogueira, L., Pradhan, R., Tatum, M., Viswanathan, V. K., Willits, S., Zhao, S., Zhu, H., Abad, D., Angert, T., Armstrong, G., Boirum, R., Dongare, A., Dworman, M., Hu, S., Jaekel, J., Ji, R., Lai, A., Lee, Y. H., Luong, A., Mangelson, J., Maier, J., Picard, J., Pluckter, K., Saba, A., Saroya, M., Scheide, E., Shoemaker-Trejo, N., Spisak, J., Teza, J., Yang, F., Wilson, A., Zhang, H., Choset, H., Kaess, M., Rowe, A., Singh, S., Zhang, J., Hollinger, G. A., & Travers, M. (2022). Resilient and modular subterranean exploration with a team of roving and flying robots. *Field Robotics*, 2, 678–734.

Publisher’s Note: Field Robotics does not accept any legal responsibility for errors, omissions or claims and does not provide any warranty, express or implied, with respect to information published in this article.