Special Issue: DARPA Subterranean Challenge, Advancement and Lessons Learned from the Finals (DARPA SubT Final)

Systems Article

Exploring the Most Sectors at the DARPA Subterranean Challenge Finals

Chao Cao[1], Lucas Nogueira[1], Hongbiao Zhu[1], John Keller[1], Graeme Best[2,3], Rohit Garg[1], David Kohanbash[1], Jay Maier[1], Shibo Zhao[1], Fan Yang[1], Katarina Cujic[1], Ryan Darnley[1], Robert DeBortoli[2], Bill Drozd[1], Peigen Sun[1], Ian Higgins[1], Steven Willits[1], Greg Armstrong[1], Ji Zhang[1], Geoffrey A. Hollinger[2], Matthew Travers[1] and Sebastian Scherer[1]

[1]Carnegie Mellon University
[2]Oregon State University
[3]University of Technology Sydney

Abstract: Autonomous robot navigation in austere environments is critical to missions like "search and rescue", yet it remains difficult to achieve. The recent DARPA Subterranean Challenge (SubT) highlights prominent challenges including GPS-denied navigation through rough terrains, rapid exploration in large-scale three-dimensional (3D) space, and interrobot coordination over unreliable communication. Solving these challenges requires both mechanical resilience and algorithmic intelligence. Here, we present our approach that leverages a fleet of custom-built heterogeneous robots and an autonomy stack for robust navigation in challenging environments. Our approach has demonstrated superior navigation performance in the SubT Final Event, resulting in the fastest traversal and most thorough exploration of the environment, which won the "Most Sectors Explored Award." This paper details our approach from two aspects: mechanical designs of a marsupial ground-and-aerial system to overcome mobility challenges and autonomy algorithms enabling collective rapid exploration. We also provide lessons learned in the design, development, and deployment of complex but resilient robotic systems to overcome real-world navigation challenges.

Keywords: exploration, navigation, subterranean robotics, SLAM, planning

1. Introduction

Robotic systems provide opportunities to carry out safety-critical missions in extreme environments such as search and rescue for disaster response. The ability of a robot to navigate effectively and autonomously through complex environments is crucial for such missions, but remains challenging to develop.

Received: 1 July 2022; revised: 24 October 2022; accepted: 19 December 2022; published: 5 June 2023.

Correspondence: Chao Cao, Carnegie Mellon University, Email: ccao1@andrew.cmu.edu

DOI: https://doi.org/10.55417/fr.2023025

The DARPA Subterranean Challenge (SubT) provides a representative scenario in which robots are sent to a combination of tunnel, urban, and cave environments to search for artifacts within a limited time frame. Challenges to robotic autonomy include GPS-denied navigation, traversing rough terrains, degraded sensing, rapid exploration in large-scale 3D space, and coordination over unreliable communications. A recent example of this was the collapse of the Surfside Condo in Miami (Murphy, 2021). Overcoming these challenges requires novel systems that combine physical resilience to withstand austere conditions and algorithmic intelligence to efficiently perform missions.

Early work in autonomous exploration of subterranean environments 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). The solutions designed by other teams for previous rounds of the SubT challenge involving Tunnel and Urban environments show the open research questions that still exist. (Agha et al., 2021; Hudson et al., 2021; Tranzatto et al., 2022; Rouček et al., 2019; Ohradzansky et al., 2021). The capabilities developed are typically divided into the following categories: Autonomy, Mapping, Mobility, Communications, and Object Detection. Across these categories, there were varying levels of diversity in the solutions deployed at the SubT Final Event. For example, in terms of mobility, we saw wheeled, tracked, and legged ground robots, in addition to custom-designed aerial vehicles. In the mapping space, most teams relied heavily on LIDAR-based SLAM solutions, with variations in the number of sensors deployed in each robot (Palieri et al., 2021); if a rotating mechanism was present (Hudson et al., 2021); and if interrobot mapping constraints were used (Ebadi et al., 2020).

Our previous work (Scherer et al., 2021) covers our approach comprehensively in addressing the challenges of the Tunnel and Urban Circuits. Our work features modular mechanical designs with a unified autonomy stack run on a fleet of heterogeneous robots with ground- and aerial-based mobility. In particular, wheeled and aerial robots are custom-built and designed for subterranean navigation. Despite the differences in mobility, we strive to unify and modularize the sensing payload and autonomy software for mapping and navigation across different robots. Doing so enables us to reduce development cycles while expanding the capabilities of each type of robot.

The focus of this article is on presenting the major changes of our system after the Urban circuit as well as our performance in the SubT Final Event, which took place at the Megacavern in Louisville, Kentucky, in September 2021. Specifically, we detail the major changes since the Urban Circuit in SLAM (Simultaneous Localization and Mapping), navigation autonomy for both aerial and ground vehicles, and communication system. The changes in the artifact detection pipeline, human supervisor user interface, and testing/validation strategies are minor, thus, the discussion of these topics are omitted here for brevity. Readers can refer to (Scherer et al., 2021) for details. In the SubT Final Event, our robots achieved the most efficient traversal and the most thorough exploration among all teams, which won us the "Most Sectors Explored Award." Figure 1 shows the exploration progress, measured by sectors covered, over time for all teams participating in the SubT Final Event. Our robots explored 26 of the 28 sectors in total, more than 80%, of which were explored in the first half of the one-hour competition (map built over time shown in Figure 24).

This paper is divided into two main sections. Section 2 details the methods used in the SubT Final Event, with a focus on the methods developed after the Urban Round. Section 2.1 presents the hardware design process that led us to the use of a mix of custom ground and aerial vehicles. Section 2.2 details ExplorerSLAM, our complete SLAM system that supports multiple heterogeneous robots. The motion planning algorithms that achieved the best exploration results in the Challenge are presented next, with Section 2.3 containing the details of the motion planning of aerial robots, and Section 2.4 the planning algorithms for ground robots. In Section 3, we aim to show all the insights learned from the SubT Final Event, showcasing particular instances where the robots behaved interestingly and what lessons we learned from those moments in Section 4. Lastly, Section 5 presents the conclusion and future work.

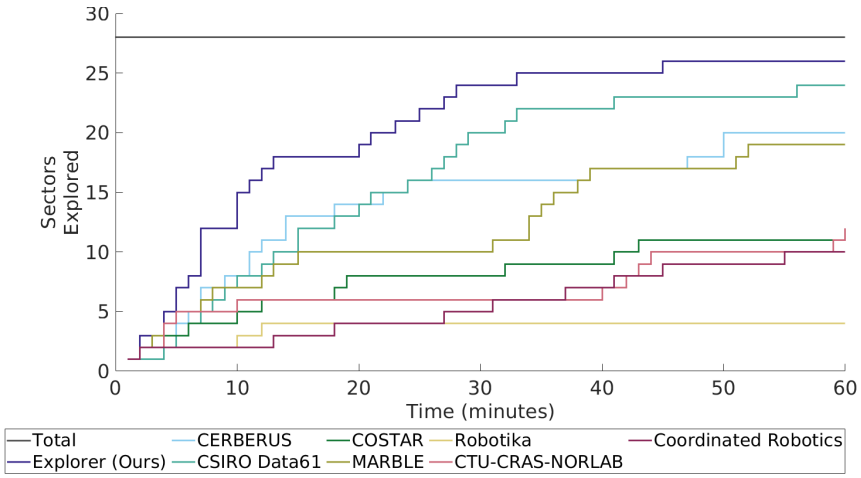


Figure 1. Sectors explored over time for all competing teams in the SubT Final Event. Team explorer achieved the fastest and most complete exploration (26 out of 28 sectors), winning the "Most Sectors Explored Award." The statistics are gathered from the official competition playback from DARPA.

Table 1. A partial illustration of our subsystem requirements table with a few example requirements.

Subsystem	Requirements
Mobility	Robot shall be able to traverse through small passages
Mobility	Robot shall be able to traverse large obstacles
Mobility	Robot shall be able to traverse through vertical shaft
Mobility	Robot shall be able to navigate areas with steps
Mobility	Robot shall be able to navigate areas with inclines
Mobility	Robot shall be able to navigate areas with multiple levels
System	Robot shall operate in wet environments
System	Robot shall operate in mud
Communication	Communication shall work in limited line-of-sight environments
Perception	Robot shall operate in environments with dust, fog, water and smoke
Perception	Robot shall operate in complete darkness
...	...

2. Approach

2.1. Mechanism Design

In this section, we present how we designed and developed the hardware of our robots. After the challenge was announced, our team created a system requirements document (SRD) based on requirements gleamed from the DARPA challenge announcements and based on our thoughts and experience in equivalent domains. Table 1 shows a part of the SRD we used for vehicle designs. Note that the list presented here is not exhaustive and is only for demonstration purpose, where the quantitative specification for each requirement are removed for brevity.

On the basis of the system requirements document, we decided that we needed two ability sets:

- ground exploration,
- aerial exploration.

In essence, ground vehicles have longer endurance and larger capacities for onboard sensors and computation, while aerial vehicles have the agility to explore areas not reachable from ground vehicles, such as tight openings and vertical space.

2.1.1. Ground Exploration Platforms

The highest initial priority was for ground vehicles. This was based on what we knew about mine tunnel systems and the development time required to develop these different systems.

An initial trade study was conducted and the decision was to make three identical four-wheel ground robots with a passive suspension system. However, based on some reasons below, the three ground robots ended up all mechanically different. Some of our design questions included the following.

- **Robot Size?** Based on the initial SRD, the robot needed to fit within an opening of 1×1 meter according to the DARPA documentation for the minimum opening size. The SRD also had a team-self-imposed restriction that the robots needed to fit through a standard "36 inch" 0.9 m door opening. This led us to design the initial vehicles (R1 and R2) to be approximately 0.9 m wide and 1 m tall, as shown in Figure 2. Later in the competition, DARPA specified that openings might be as small as 0.7×0.7 meters; this led us to build the third ground vehicle (R3) to be around 0.7 m wide and 0.7 m tall. This proved to be very valuable to us in the Prize Round, where R3 navigated through tight space easily and explored larger areas compared to R1 and R2.
- **Suspension Method?** Based on previous experience working in underground environments and rough terrain, we selected a simple passive articulation in the center of the robots. Initial design choices included side-to-side or front-to-back differencing, as shown in Figure 3. The

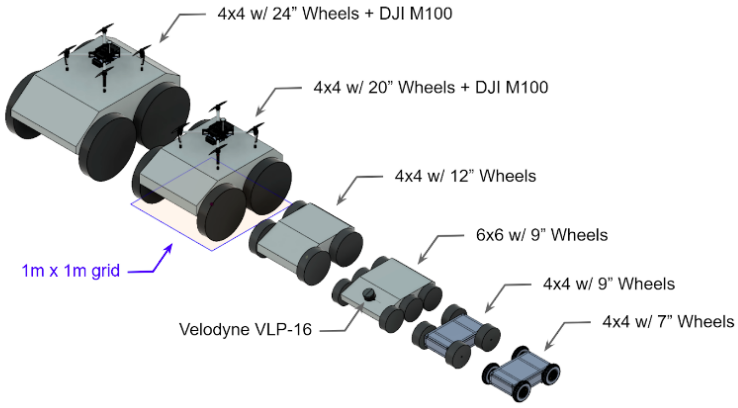


Figure 2. Graphic showing some of the different sizes considered for our ground vehicle platform. The larger sizes were also the sizes that we determined to be feasible to carry a drone on the back for deployment.

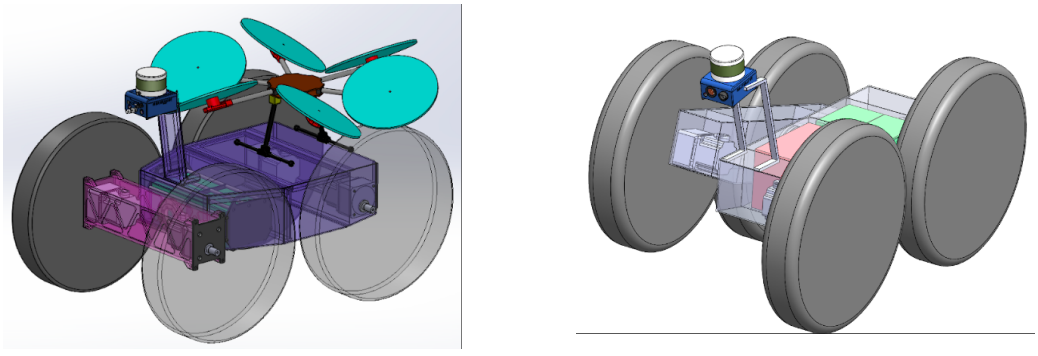


Figure 3. Early computer-aided design (CAD) drawings demonstrating the front-to-back and side-to-side suspension approaches. We eventually adopted the front-to-back suspension approach for R2 and R3 because of the bigger chassis allowed for enclosing more electronics and batteries.

thoughts were that front-to-back differencing would be better for climbing steps head on, easier to mechanically mount components, and easier to carry a drone on the back. A side-to-side differencing solution would be better for climbing steps at an angle, climbing obstacles on a slope, and climbing offset obstacles (where objects under the wheels alternate between sides). We selected the front-to-back differencing for the aforementioned reasons. However, after development and testing we learnt that in practice the wheels slipped while climbing steps so the robots were essentially always climbing at an angle. It is the belief of the authors that side-to-side differencing would have been a better option if the only concern was stair climbing. Due to time constraints R1, the first robot we built, did not have a articulating passive suspensions. However R2 and R3 did have the articulating suspension.

- **Wheel Size?** In order to climb the biggest obstacles, we wanted the largest wheels (that is, 1 m in diameter). The elements of consideration that reduced the size of the wheels were what that would force the vehicle length to be, and minimizing the height of the wheels so that the sensors mounted 1m from the ground had good visibility of the surroundings.
- **Modularity?** Based on the initial plan of having three identical robots, we decided to embrace modularity. To do this, the plan was to have several modules that could be swapped between any of the three robots as needed, and would allow for having common spare modules. The modules were as follows:
  – battery module,
  – electronics box (with power and computing),
  – motor modules (motor with motor controller) (4 per robot),
  – sensor payload.
  This strategy was employed for the first two ground robots designed and built. For the third robot, since it was smaller and would not fit the prior designed modules well, we decided to integrate the electronics into the robot and then have separate battery and motor modules. This allowed for decreased mechanical and electrical complexity. So fewer connectors to increase reliability, faster to fabricate and ease of access for debugging. This third robot was a good balance for what components were integrated into the robot and which components were modules.

During the urban and final challenges, the ability to climb steps was necessary. Since the opening size was reduced to 0.7×0.7 m and we needed to climb and descend stairs, the third vehicle was designed with steps in mind. After many experiments (and several robot flips), we came up with a configuration that worked for descending steps in a limited manner. To climb steps, we tried various wheel configurations with limited/minimal success, but did not have the power and/or traction to climb the steps. To descend steps, we found that added counterweights to the rear of the robot let us descend steps in optimal conditions.

2.1.2. Aerial Exploration Platforms

The design of the aerial platforms in the system was focused on augmenting the ground vehicles in areas where they were weak. In particular, the aerial vehicles were designed to traverse vertical shafts, steeply sloping passageways, staircases, and other obstacles that are intrinsically difficult to traverse for a ground based robot. In the Urban Circuit, we first focused on our fleet of Small Drone platforms (SD) which could be deployed either from the start gate or off the back of a ground robot. For the final event, we expanded the fleet with two large drones built by Canary Aerospace (a Pittsburgh-area company who partnered with our team for the competition). Our SD drone platforms and our Canary drone platforms were similar architecturally, though they are very different sizes. This is because we found that the constraints, including payload weights, sensor types, available computational and battery power, that drove the design of the small drones were very similar to the constraints that drove the design of the large drones.

The SD platforms were designed primarily to be narrow enough to fly through a standard 36 inch doorway. Through testing, we found that we were able to fly much more consistently through narrow openings if we had a robust propeller guard that we could reliably bump into obstacles (such

as the side of an opening). In our design, we dedicated a significant structural effort to supporting these robust propeller guards. This collision tolerant design proved advantageous, both in the way that it enabled more aggressive maneuvering during the competition as well as in the way that it decreased the rate of destructive drone crashes during development (thus enabling an increased pace of development).

For our design, the keys to implementing an effective propeller guard were the following.

- **Placement.** Keep the center of mass (CM) of the vehicle, the propellers, and the propeller guard all on the same plane. This ensures that low-speed collisions result in the vehicle maintaining its attitude through the collision. The goal is to have the collision happen gracefully enough that the attitude controllers running on the vehicle do not react aggressively to the disturbance. If the propeller guard is far above the vehicle CM, a sideways bump into an obstacle will result in the vehicle pivoting about the propeller guard and continuing to move toward the obstacle. This can result in an attitude controller overcorrecting and becoming unstable. If the propellers are not co-planar with the propeller guard, then an obstacle can come into contact with the propellers without hitting the propeller guard first.
- **Robustness.** Build the propeller guard so that it is robust enough to handle collisions at desired flight speeds without failing. It is often tempting to build a propeller guard that is as light as possible. In our experience, this can result in having a propeller guard that does not succeed in protecting the vehicle from collisions.

  In addition to propeller guard design, we also followed a similar approach to propulsion system design between the SD platforms and the Canary drones. When we were designing the SD platform, we did a series of thrust tests with representative motors and propellers in order to let us estimate flight performance for different propulsion architectures. We compared both single level and coaxial propeller configurations. In general, for our platform, we found that the quadcopter configuration had the best combination of small platform size and flight time.

2.1.3. On-board computation

Each ground vehicle carries three computers, including two Intel NUCs with 4.1 GHz i7 CPU for SLAM and Autonomy, and a NVIDIA Jetson AGX Xavier for object detection. Each aerial vehicle carries an Intel NUC with 4.1 GHz i7 CPU for SLAM and autonomy, and an Intel Neural Compute Stick for object detection. The algorithms described in the follows are all run on the on-board computation. The base station has a laptop with 4.1 GHz i7 CPU for running the human supervisor user interfaces and processing the data sent back from robots (details of the base station user interface are provided in (Scherer et al., 2021)).

2.2. SLAM

For the SubT Final Event, we created ExplorerSLAM, a solution for the multiple robot mapping problem that uses different modules developed by our team. It is composed of three modules: (1) a front-end based on SuperOdometry (SO) (Zhao et al., 2021), an inertial measurement unit (IMU)-centric Lidar-Inertial algorithm; (2) a back-end loop-closure method based on the LOAM (Zhang and Singh, 2014) scan registration algorithm; and (3) AutoCalibration, a interrobot alignment algorithm that easily allows multiple (possibly) heterogeneous robots to share a common frame of reference for the mapping, exploration, and object detection capabilities.

Team Explorer used LOAM (Zhang and Singh, 2014) as an odometry front-end in Tunnel and Urban Challenges. For the SubT Final Event, we decided to use SuperOdometry (Zhao et al., 2021). SO fuses the outputs of multiple modules using a probabilistic factor graph optimization. This allows for inherent rejection and/or smoothing of outlier results in any single module. In contrast, LOAM uses a first-order filter to reset the odometry estimate to the most recent laser odometry result.

SuperOdometry was the focus of most of our research effort. It allowed us to better tackle the challenges involved in multimodal sensor fusion. Its first design principle is modularity. It dictates

our goal of being able to add new sensor modalities to the estimation algorithm without needing to modify the existing working framework and with minimal dependencies between them. The second design principle is resiliency. It expresses the idea that failure in a single modality should not impact the whole system as long as a single working modality is available. This is achieved in part by the first principle.

The key insight into SuperOdometry is the focus on the IMU as an environment-independent sensor. This leads to an IMU-centric architecture that combines advantages of loosely and tightly coupled methods. It allows easy integration of different sensor modules that can vary depending on the robot platform.

2.2.1. SuperOdometry Front-End

SuperOdometry itself is composed of two types of modules: the central IMU odometry module and external modules that process different types of exteroceptive sensors such as visual camera and 3D LiDAR. We have developed a lidar-odometry (LO) and a visual-odometry (VO) external module but used only the LO one for the SubT Final Event. In the final preparation stage for the competition, we reasoned that the environment would be too hostile to the visual component, and the reward of running this module in our robots would not be enough to compensate the risk associated with increased CPU usage.

The IMU module receives constraints from the external module that allow it to estimate the IMU biases using a factor graph formulation. Specifically, we obtain relative pose factors from visual ($\mathbf{e}_{ij}^{\mathrm{VO}}$) and/or LiDAR sensors ($\mathbf{e}_{ij}^{\mathrm{LO}}$). The time window $\mathcal{B}$ is the set of pairwise consecutive IMU measurements obtained between the previous scan frame i and the frame $i+1$ to be estimated. We marginalize older variables f, and obtain the marginalization prior $\mathbf{E}_m$. Then, we minimize the following cost function to estimate the imu biases.

$$E = \sum_{(i,j)\in\mathcal{B}} \|\mathbf{e}_{ij}^{\mathrm{LO}}\|^2_{\mathbf{\Sigma}_{LO}} + \sum_{(i,j)\in\mathcal{B}} \|\mathbf{e}_{ij}^{\mathrm{VO}}\|^2_{\mathbf{\Sigma}_{VO}} + \mathbf{E}_{\mathrm{m}}. \tag{1}$$

The pose factor $\mathbf{e}_{ij}^{\mathrm{LO}}$,$\mathbf{e}_{ij}^{\mathrm{VO}}$ has to be weighted with the appropriate information matrix $\mathbf{\Sigma}_{\mathrm{LO,VO}}$, which can be estimated based on the observability of odometry. For instance, in textureless environments, $\mathbf{e}_{ij}^{\mathrm{VO}}$ estimated by visual odometry will have a lower weight. In contrast, in structureless environments, $\mathbf{e}_{ij}^{\mathrm{LO}}$ estimated by LiDAR odometry will have a lower weight. Note that without loss of generality, as long as other sensors can provide pose factor to constrain the bias of IMU, these sensors will be fused into the system successfully.

In our LiDAR-inertial system, the central IMU odometry component provides motion prediction to the external LO module. We obtain the set of point-to-point, point-to-line, and point-to-plane correspondences $\mathbb{F}_i$ between the current scan and the SO local map, from which we derive the residuals $\mathbf{e}_i^{po\rightarrow po,li,pl}$. The optimization solves for the optimal pose $\mathbf{T}^*_{\mathbf{i+1}} \in \mathrm{SE}(3)$, such that

$$\mathbf{T}^*_{\mathbf{i+1}} = \min_{\mathbf{T}_{i+1}} \left\{ \sum_{p\in\mathbb{F}_i} \left\| \mathbf{e}_i^{po\rightarrow po,li,pl} \right\|^2_{\mathbf{W}_l} + \sum_{\mathcal{B}} \|\mathbf{e}_{\mathrm{imu}}\|^2_{\mathbf{W}_{\mathrm{imu}}} + \mathbf{E}^{\mathrm{prior}}_{\mathrm{imuodom}}. \right\} \tag{2}$$

Here, $\mathbf{e}_{\mathrm{imu}}$ is a residual that represents the predicted pose prior and $\mathbf{E}^{\mathrm{prior}}_{\mathrm{imuodom}}$ is the IMU preintegration factor (Forster et al., 2015). $\mathbf{W}_l$ and $\mathbf{W}_{\mathrm{imu}}$ are covariance matrices related to the lidar correspondences and the IMU factor, respectively. We also develop key algorithmic innovations that are presented in Section 2.2.4 that enhance the robustness of the SO. More details can be found in the Super Odometry paper (Zhao et al., 2021).

2.2.2. Loop-closing back-end

Although the SO algorithm produces odometry with low drift, it is still important for the SLAM system to be able to correct for long-term drift. This is especially true when the expected traversed distance is high. Considering that our ground robots moved at an *average* speed of 0.5 m/s and had an aggressive exploration style, it was not uncommon to observe traversed distances larger than 1 km during testing.

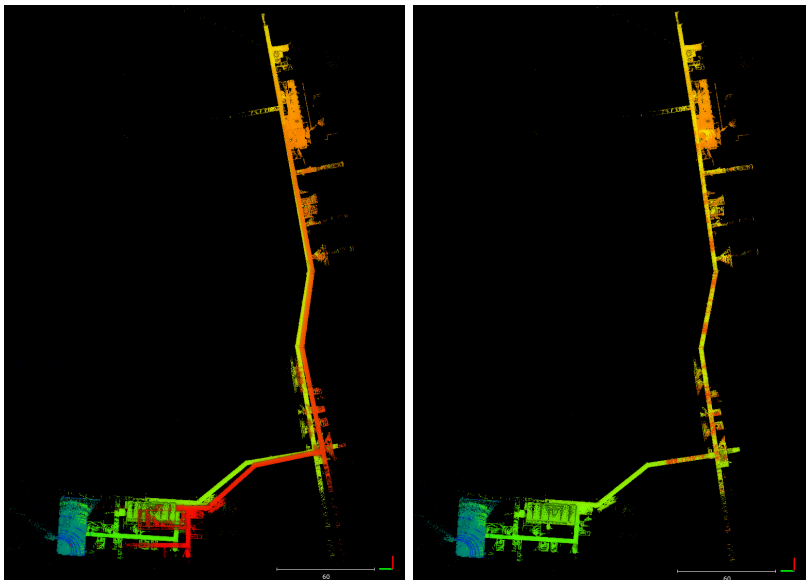


Figure 4. (Left) Output from ExplorerSLAM front-end. The robot started from the lower left corner and came back to the starting point. Color of the map indicates acquisition time. There is a significant misalignment between the starting and end part of the map (red and green). (Right) Output after Loop Closing back-end. The misalignment is corrected.

We formulate this problem with a factor graph that considers the state X as a set of keyposes such that: $X = \{\mathbf{T}_0, \ldots, \mathbf{T}_n\}$, with $\mathbf{T}_i \in \mathrm{SE}(3)$, and we estimate X maximizing the conditional probability $p(X|Z)$, where Z is a set of measurements. The keyposes $\mathbf{T}_i$ are obtained by filtering the SO odometry results so that we have a minimal distance of 0.2 m between them. For each keypose, we accumulate the registered pointclouds obtained since the last one into a *key cloud* P_i. Two consecutive key poses will be connected in the factor graph by an odometric constraint provided by SO. Our loop-closing algorithm also uses a distance-based strategy to detect loops when the robot revisits a location. Considering that we are at the keyframe i, if we find another keyframe j such that the distance between T_i and T_j is less than a given threshold, then we perform a LOAM-style scan matching between P_i and P_j, and add the result as a constraint to the factor graph, using a robust kernel. This factor graph is solved with the GTSAM library (Dellaert and Contributors, 2022) every time we detect a loop.

Figure 4 shows the effect of the loop closing back-end on the map. While the front-end is capable of providing a robust odometry measurement, it is hard to avoid some drift in environments with sharp turns and few geometric features. The loop closure can fix that by creating constraints between nonconsecutive frames.

2.2.3. AutoCalibration

To achieve a shared task, multiple robots must be able to operate in a common frame of reference. In the Subt Challenge, the obvious choice was *DARPA frame*, which was used to express the location of the artifacts. In Tunnel and Urban Circuits, our team used a total station (TS) to obtain a 6 degree-of-freedom transformation $\mathbf{T}_m^d$ that aligned the mapping frame m of each robot with the DARPA frame d. This process was time-consuming, since it required an experienced TS operator to coordinate with the base station operator to obtain two control points for each robot, by moving them as much as possible within the staging area. For reference, Team Explorer launched eight robots in the SubT Final Event.

A key addition to our SLAM pipeline to solve this issue was *Autocalibration*, as show in Figure 5. With it, we only need to use a TS once to obtain the transformation $T_{m_0}^d = \{\mathbf{R}, \mathbf{t}\}$ to the DARPA

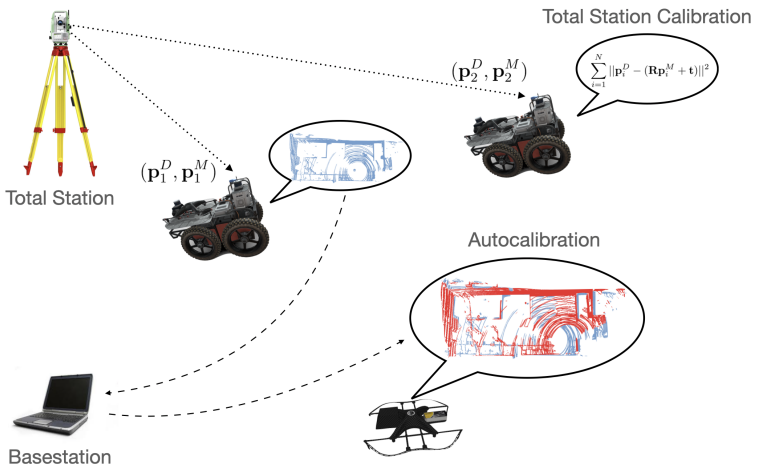


Figure 5. Overview of AutoCalibration. At least one robot is calibrated using the total station via least-squares registration, while also sharing a local map to the basestation. Subsequent robots may request this map from the basestation and perform alignment with respect to its own local map

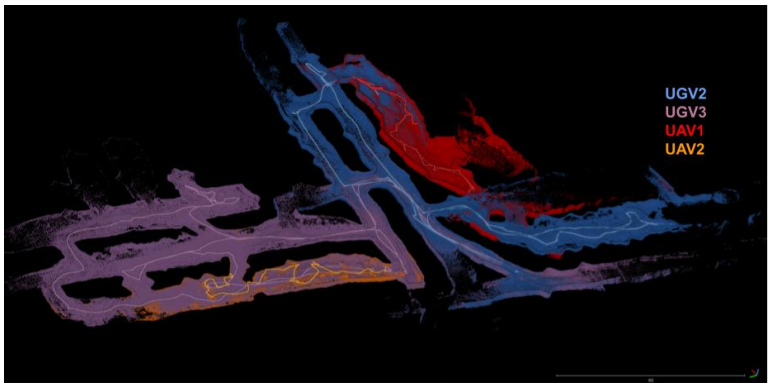


Figure 6. pointcloud maps produced by autonomous marsupial deployment of 2 pairs of ground and aerial robots. The colors correspond to the robot that captured the points.

frame for one robot. This is done by measuring the position of the robot in the map frame and in the world frame in two different locations and solving for $T^d_{m_0}$ using a least-squares registrations. Then, the robot moves to the calibration spot: a position chosen by the operator, usually the center of the staging area. This robot shares its pose with respect to its own map frame $T^{m_0}_{b_0}$ and the last 3 keyframes accumulated in a single point cloud P_0. All subsequent robots will be placed in the same place, receive reference information from the base station, and use Generalized Iterative Closest Point (GICP) (Segal et al., 2009) to align their current keyframes P_i to P_0, obtaining $T^{b_0}_{b_i}$. Specifically, for each robot i, we obtain

$$T^d_{m_i} = T^d_{m_0} T^{m_0}_{b_0} T^{b_0}_{b_i} (T^{m_i}_{b_i})^{-1}. \tag{3}$$

This solution is inherently less accurate than the TS, but its practicality is greatly improved. Since all our robots had a Velodyne VLP-16, we did not need to do any hardware modification to it, whereas with TS, we needed to position an optical prism on each robot manually and remove it before it starts the mission. Additionally, a large amount of time is recovered that would have been spent taking TS measurements for each of the robots.

Figure 6 shows the complete map built by ExplorerSLAM from an autonomous deployment during the testing. Two ground robots leave the staging area, each carrying an aerial vehicle. Their

Table 2. Automatic voxel values.There are three sets of parameters for extracting "corner" and "surface" features in three environment scales: narrow (point-cloud "volume" $V < 25$ m^3), normal ($V > 25$ m^3 and $V < 65$ m^3), and open ($V > 65$ m^3).

	Voxel Size	
Mode	**Corner**	**Surface**
narrow	0.1	0.2
normal	0.2	0.4
open	0.4	0.8

maps are aligned using AutoCalibration. The aerial vehicle becomes active at a chosen location and automatically aligns with the ground robot map. The ground robots return to the start locations and the aerial vehicles land in a safe place.

2.2.4. Technical Innovations

In the last testing phases leading up to the SubT Final Event, most of the work of the SLAM team focused on increasing the reliability and the ability of the systems to generalize to the very diverse range of scenarios that we were expecting to encounter. Some of these efforts are presented in this section, as they became crucial to the ability of our SLAM system of handling the diverse environments in the Prize Round.

AutoResolution

The most important parameter tuned during testing was related to the downsampling of the pointclouds prior to the scan-to-map registration step in SO. It affected the number and quality of features available to properly constrain the laser mapping optimization process. To control the downsampling, we set the voxel size in the PCL library (Rusu and Cousins, 2011) voxel grid filter that is applied to the set of line and plane features. When the robot traverses a narrow urban corridor, it is desirable to use a smaller voxel size to avoid decimating important details in the pointcloud. In contrast, in a large open cave area we want a larger voxel size, otherwise the system slows down processing so many features available. In the SubT Final Event, we expected to face a variety of environments in an unpredictable way, so we could not set the voxel size beforehand. To solve this problem, we created an heuristic method to toggle between different voxel sizes in real time.

The method consisted of calculating the average absolute value of the points in the current pointcloud for each 3D axis separately, in the robot-centered frame. Then we multiply the 3 values together to obtain an "average volume." This volume was thresholded to create 3 different modes, which had predefined voxel sizes, as shown in Table 2. The method will also affect the pointclouds already stored in the map that are in the neighborhood of the current position. This means that when we transition from a small area to an open area, it is possible to destroy some detail on the smaller area on the map. This was shown to be not a problem during testing. The system also used a distance-based sliding window to forget old parts of the method that have not been visited recently. Finally, note that this downsampling affects only the SLAM component of the system. Other components, such as obstacle avoidance, receive the full resolution pointcloud and the SLAM estimates. This is particularly important to maintain the ability of avoiding difficult obstacles such as hanging wires, that would be otherwise decimated by downsampling the pointcloud.

Dust Filters

Team Explorer had a strong focus on aerial vehicles. These platforms bring their own unique challenges to the SLAM problem. One that was particularly important for the Subt Challenge is being able to handle the amount of dust that arises from the robot's propellers. The dust particles are captured by the lidar sensor and become outliers in the registration step and impedes it from capturing good geometric features, potentially causing catastrophic failure when dust covers the

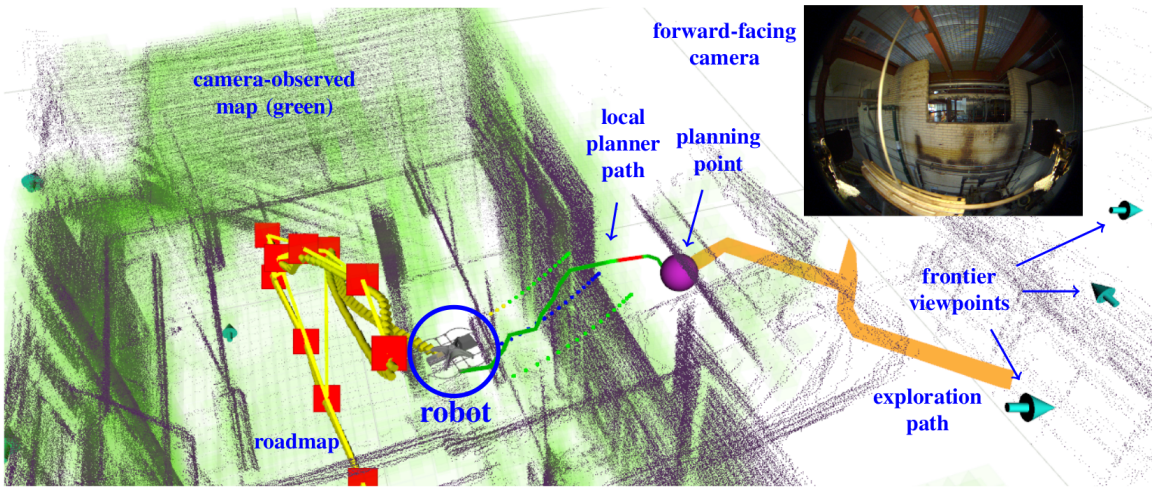


Figure 7. Our aerial robot exploring a boiler plant in Pittsburgh, PA. The robot has finished observing the room on the left (green points), so it plans a path between beams hanging from the ceiling and a small hole in the wall (visible in camera field of view) to reach the frontier viewpoints in the unexplored room on the right. All computation is performed online and onboard the robot. The point cloud is generated with our onboard LiDAR sensor and SuperOdometry.

robot from all sides. The implemented solution was to test if there was a minimum number of features farther than 3 m from the robot. If there were, all the other points inside this radius would be ignored during pose estimation. This solution was based on the assumption that dust would usually accumulate in a circular manner around the robot, but there would still exist other features available in the environment that allow the optimization to be solved correctly. In most cases, the robot finds a way out of the dust area and resumes its exploration. Otherwise, the dust will eventually cover the robot from all sides and no geometric features can be extracted at all, causing complete SLAM failure. More details of the dust filter is provided in (Best et al., 2022).

Robot-specific computational budget

As noted above, SO was deployed throughout our robot fleet, regardless of the computing power available. As indicated in 2.1.3, a ground robot has 3 computers onbroad while a drone has only one computer. Because of this heterogeneity, we designed a parameter that can be adjusted on a per-robot basis to limit SO cpu usage. We found that one of the key indicators for the processing time of a scan frame was the number of surface features. Furthermore, we determined that after a certain number of features, having more features to process did not necessarily lead to further gains in accuracy. Therefore we set a maximum limit on the number of surface features that we process per scan frame. If the current scan frame contains more than this number, we will uniformly sample the list of features to achieve this maximum. Note that this is done after deciding on the voxel size and performing the downsampling, which we believe allows us to keep the appropriate level of detail necessary to solve the optimization problem.

2.3. Motion and Path Planning for Aerial Robots

Our approach to planning and coordination for our aerial robots consists of a hierarchy of planners and a set of map processing modules. An annotated example of the approach is presented in Figure 7. We provide a summary of our approach in this section; full details are presented in (Best et al., 2022) and our software has been released.[1] Central to the decision making is the

[1] https://theairlab.org/research/2022/05/02/subt_code/

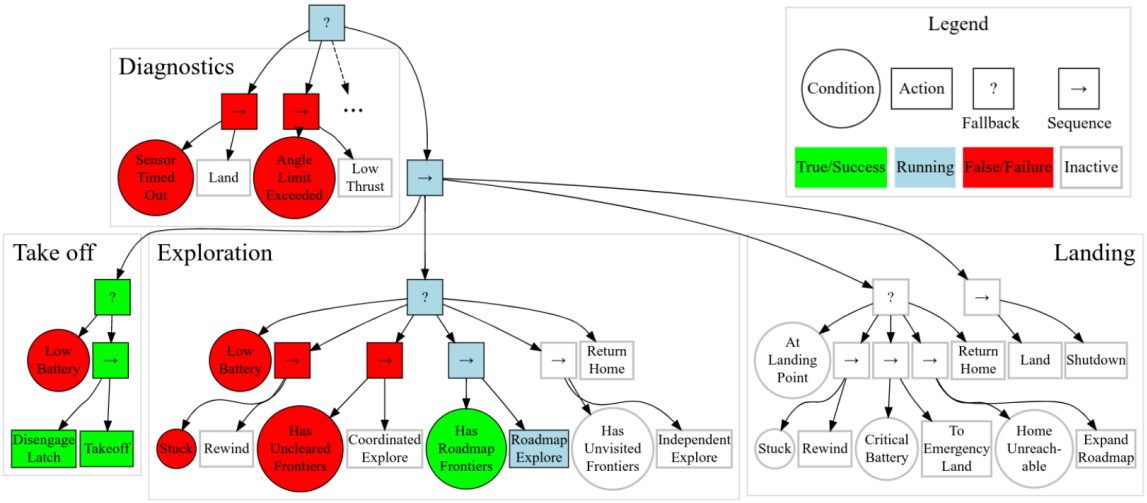


Figure 8. Our aerial planning behavior tree is responsible for adaptively switching between autonomous behaviors (actions) as the conditions change. A behavior tree is interpreted from left to right, as introduced in Section 2.3.1. In the pictured state (colors), there are no diagnostics errors, the robot has successfully taken off, the battery is not low, and so it is currently exploring. The "Roadmap Explore" behavior is selected because all locally computed frontiers have been cleared by other robots, but other robots have shared a roadmap containing paths to frontiers beyond the sensor range. A full description of each of the behaviors is provided in Section 2.3.3.

behavior executive: it employs a behavior tree logic structure to reason over various signals to decide which behavior to execute, such as a particular exploration method, backtrack out of trouble, or a landing sequence (detailed below in Section 2.3.1). The *map processor* generates different types of world representations relevant to the different planning behaviors (Section 2.3.2). The *global planner* selects goals based on the currently selected behavior, and plans feasible paths that achieve these goals (Section 2.3.3). The *local planner* takes intermediate goals from the global planner, plans a path while ensuring a higher degree of safety, adapts the speed, and controls the yaw during tight maneuvers (Section 2.3.4). Trajectory commands are output to an onboard controller.

2.3.1. Behavior Executive

Our behavior tree, depicted in Figure 8, commands the high-level decisions for the aerial robots. A behavior tree models plan execution with a directed tree (Colledanchise and Ögren, 2018). *Condition* nodes are evaluated as true or false based on the system state. An *action* node describes a behavior that is executed when active, and evaluates as success, running, or failure. The tree is evaluated from left to right, with the internal nodes recursively controlling which leaf nodes are active. *Fallback* nodes are true if any of its children are true, while *sequence* nodes are false if any of its children are false. The left-most children of an active *fallback* node are active up to the first evaluated as true/success or running. Similarly, the children of an active *sequence* node are active up to the first false/failure or running. Behavior tree instantiations are typically hand-designed by humans, though in other we have also explored the automatic generation of these structures (Scheide et al., 2021).

Our behavior tree has four components: diagnostics, take-off, exploration, and landing. The diagnostics has highest priority, and has the purpose of reacting to critical events, such as landing if a sensor fails, or disengaging the motors after a crash. The take-off subtree initiates a sequence involving disengaging the latch on the ground vehicle, booting the sensors, then launching. Once the take-off is complete, the robot enters the main exploration mode until the battery is low.

The behavior tree specifies several different exploration behaviors. If the robot is stuck, then it rewinds by backtracking blindly over the previous trajectory. If it has frontier viewpoints not observed by other robots, then it executes coordinated exploration. Otherwise, it travels

to viewpoints communicated by other robots. If neither of these are possible, then independent exploration is performed. As a last resort, the robot returns home. If a higher-priority condition becomes true, then the associated behavior is immediately activated. These different behaviors are discussed further in Section 2.3.3.

Once the battery life is less than the estimated travel time to the take-off location, the robot follows the roadmap back and lands. For improved resiliency, we also have fallback behaviors: backtrack if stuck, emergency land at a safe landing zone, or repair the roadmap by removing vertices in a region and adding more edges in other regions of free space if the path is blocked.

2.3.2. Map processing

Our multisensor autonomy pipeline uses maps for several purposes: remembering what has been observed, discovering what has not been observed, collision avoidance, coordination, and returning home. As such, we require processing the data from the range and vision sensors to produce mapping products that are specialized for each purpose. These maps are generated by and shared among both aerial and ground vehicles, but the maps introduced in this section only influence the aerial robot decision making. *Roadmaps* provide a topological representation of where the robots have been and how the environment is connected. *Collision checking maps* are required by global and local planners, each with different considerations. *Exploration maps* inform the global planner of the value of future observations.

Exploration maps. The exploration maps are represented as occupancy grids with the OpenVDB data structure (Museth, 2013). OpenVDB is a hierarchical grid that resembles a B+ tree, and provides an average constant-time random access to voxel data. This property enables fast ray-casting operations for probabilistic occupancy updates, collision checking, and frontier detection. Therefore OpenVDB is ideal for constructing large-scale maps for exploration in real time.

Collision checking. The *global map* used for collision checking is a probabilistic 3D occupancy grid generated by ray casting with range-sensor scans. We write these maps in shared memory to enable low-latency sharing to the global planning modules. Since the global planner often repeats collision-check queries, we cache the results in another OpenVDB map to reduce the planning time.

Frontier maps. There are three maps relevant to generating and scoring viewpoints: *vision-observed map*, *vision-frontier map*, and *range-frontier map*. The *vision-observed map* is a subset of the occupied cells in the *global map* that are estimated to have been observed by the vision sensors. An example of this map is shown with the green shading in Figure 7.

The *vision-frontier map* is a subset of the occupied cells in the *global map* that are estimated to have not yet been observed by the vision sensors. The map leverages the geometry discovered by the range sensors to direct the planning towards surfaces unobserved by the vision sensors.

The *range-frontier map* represents the boundary of the range-sensor observations and follows the classical frontier definition (Yamauchi, 1997). This map contains all free cells in the *global map* that neighbor several unknown cells.

Coordination maps. Our coordination strategy for the aerial robots relies on sharing this mapping information between robots (both ground and aerial). Thus, we define the *communicated map*, which is a map that merges a lower resolution *vision-observed map* and *vision-frontier map*, such that each cell is marked as *vision-observed* or *vision-frontier*. The *communicated map* is merged with maps from other robots to form the *coordination map*. Directed by the *coordination map*, aerial vehicles avoid exploring regions that have already been explored by ground robots or other aerial robots.

Roadmaps. In order to facilitate traveling to areas that robots have been, we generate roadmaps from each robot's odometry. The roadmaps are graphs with vertices placed at 1 m intervals along the traversed path. Edges are connected between consecutive vertices. Additionally, edges are added between vertices that are up to 5 m apart and are line-of-sight collision free. Each robot has two roadmaps: *single-robot roadmap* and *coordination roadmap*. The *single-robot roadmap* is used for searching paths to return home. The *coordination roadmap* merges all communicated single-robot roadmaps and associated frontier clusters of the *vision-frontier map*. With the *coordination*

roadmap, one aerial vehicle can compute paths to explore frontiers that are discovered by other vehicles.

Local planner map. Since the local planner plans at a higher frequency (10 Hz), over shorter distances, and with greater safety requirements, we opt for a different representation for local planner collision checking. We employ a Euclidean distance transform (EDT) map as it enables efficient querying of distances to closest obstacles (Scherer et al., 2009). This map is maintained only for a local region around the current location. It is built using the output point cloud from the dust filter. Ray casting and distance updates are executed on the graphics processing unit (GPU).

2.3.3. Global planning

The global planning seeks to generate paths that result in sufficiently exploring the environment with the range sensors and maximally covering the surfaces with the fields of view of the vision sensors. This is achieved in several steps: candidate viewpoints are generated, these viewpoints are scored, then paths are planned to viewpoints. Figure 7 illustrates these planning components in an example mission: when the local planner locks the *planning point*, the global planner plans from this position. There are several variants of exploration behaviors, which are selected by the behavior tree based on information available from other robots. The global planner also plans for a resilient landing behavior.

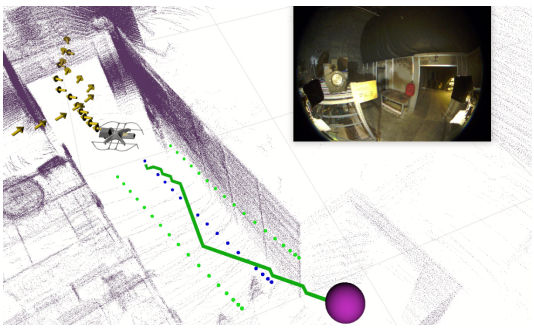
A set of *vision viewpoints* are generated by processing the *vision-frontier map*. These viewpoints are poses where it is predicted the robot will observe surfaces that have yet to have been observed. The *vision-frontier map* is first segmented into fixed-size clusters. Next, candidate viewpoints are generated on a cylinder around the cluster centroids with orientations set in the direction of each centroid. A viewpoint is kept if the viewpoint is in free space and has line-of-sight visibility to the centroid. We also generate *range viewpoints* that encourage discovering the geometry of the environment. These viewpoints are generated by segmenting the *range-frontier map* into connected components and placing a collision-free viewpoint in each cluster.

The candidate viewpoints are then sorted into a priority queue in order of their predicted value. High-value viewpoints are likely to: have a small flight time from the current state, be sufficiently far away to encourage high speeds, and have not been observed by other robots. We employ a computationally efficient heuristic function that balances these objectives (Best et al., 2022).

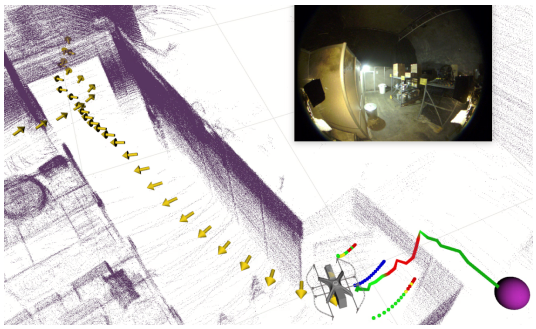
A path, like the orange path in Figure 7, is planned to the viewpoints in order of their scores. We employ RRT-Connect (Kuffner and LaValle, 2000) for path planning in 3D over the *global map*. For each viewpoint, we trial a large then a small collision radius. The viewpoints are trialled in order until a feasible path is found. The yaw along the path is a free variable for the controller up until near the viewpoint where it is specified as the viewpoint heading.

The exploration behavior described above is referred to as "Coordinated Explore" in the behavior tree, which is the default exploration behavior. If all of the locally generated viewpoints have already been observed by other robots, then the behavior tree switches to "Roadmap Explore." This behavior instead plans to frontiers shared by other robots. We also need to rely on the *coordination roadmap* to find paths to these frontiers. A path is planned to the frontier that has not been observed by any robot and has the shortest path distance. As the robot approaches the selected frontier, new locally discovered viewpoints are typically generated, and the behavior tree switches back to "Coordinated Explore." "Independent Explore" is rarely triggered, but is a backup exploration behavior for when the previous conditions are false; in this behavior, the robot explores locally generated viewpoints that have already been observed by other robots.

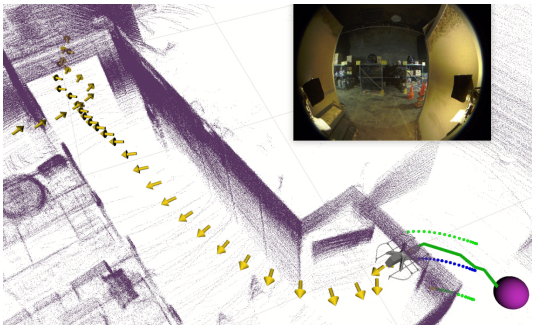
The "Return Home" behavior is triggered if the exploration is failing or battery life is less than the estimated return-home time. The return home path is found over the *single-robot roadmap*. If the battery is critically low, then the path is redirected to a safe landing location. If the return home path is blocked, the local planner will fail to find a path, and this is communicated to the global planner. If this occurs three times, then the roadmap is repaired and extended in an attempt to find an alternate path home (a more detailed description of this behavior is provided in (Best et al., 2022)).



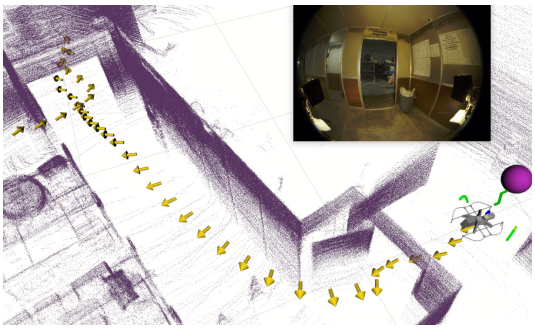
(a) Time 0 s. Robot increases speed in the wider corridor.



(b) Time 6 s. Robot cannot fit sideways through this doorway.



(c) Time 13 s. Robot rotates to fly backwards through doorway.



(d) Time 17 s. Doorway maneuver successfully completed.

Figure 9. A time-lapse sequence of an aerial robot exploring the office warehouse in the final competition. The local planner adjusts the speed and yaw of the robot as the robot maneuvers through the doorway. The yellow arrows are a history of poses; inset is the view from the forward facing camera; the purple sphere is the current local planner goal; the solid path is the local planner path (with red indicating a narrow passage, and green otherwise); the dots and coloring near the future trajectory represent clearance checking.

2.3.4. Local planning

The local planner receives a sequence of waypoints from the global planner and plans a collision free trajectory to them while trying to fly as fast as possible. A time-lapse of an example local planner maneuver through a doorway is presented in Figure 9.

The local planner proceeds in several steps. First, the robot's distance to obstacles, computed using the EDT, is used to determine its maximum allowable velocity. If it is open space it is allowed to travel faster, while in narrower spaces it slows down. In order to have a safety cushion for control error at higher velocities, the radius of the robot used for collision checking in the EDT is increased as velocity increases. Next, the local planner algorithm performs an A* search through the EDT, then it follows this path using a trajectory library. The A* algorithm is modified to place a penalty on getting too close to obstacles so that it tends to fly in more open space when possible. When a segment of the A* path is below a threshold distance to obstacle, it is treated as a narrow passage. Since the robot is longer than it is wide, it aligns its heading, either forwards or backwards, with its direction of travel while traveling through narrow passages.

The A* path is matched to a trajectory library; an example trajectory library is illustrated in Figure 10. The trajectories are formed by snaps in different directions which are integrated to get the jerk, acceleration, velocity and position of each point on the trajectory. Each trajectory also has an end segment which goes to zero velocity to ensure that the robot chooses trajectories which give it enough time to stop before a collision. The selected trajectory is collision-free and minimizes the sum of distances to the A* path.

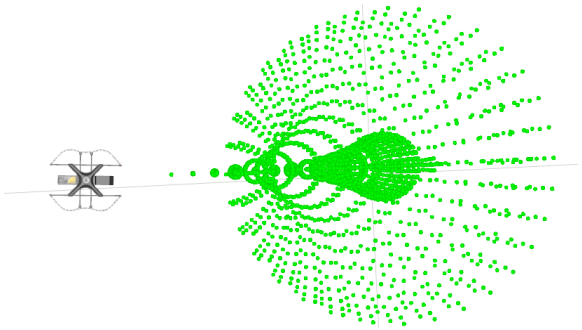


Figure 10. A trajectory library (green) of dynamically feasible trajectories as the robot flies to the right at high speed. These trajectories are matched to the local planner A* path. There are many of these trajectory libraries, which are selected at any time instant according to the current dynamics of the robot in order to maintain the current momentum of the robot.

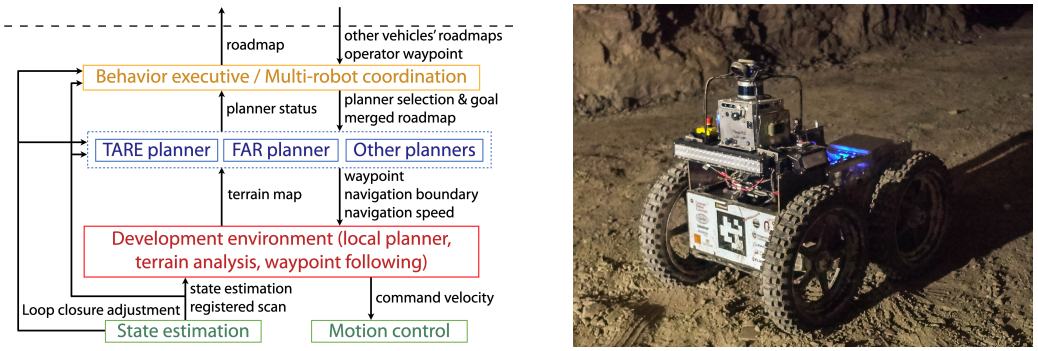
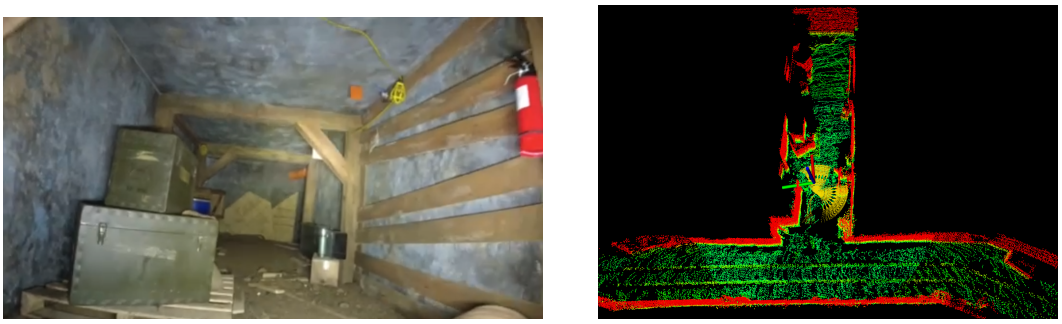


Figure 11. (a) System diagram for the ground vehicles used by Team Explorer in DARPA Subterranean Challenge. The system has four layers for multirobot coordination (yellow), task-specific high-level planning (blue), low-level navigation modules (red), and modules for state estimation and motor control (green). (b) One of our ground vehicles (R1) in a tunnel environment.

2.4. Motion and Path Planning for Ground Robots

Figure 11a shows the system diagram for the planning architecture of our ground robots. The state estimation and motion control modules at the bottom layer interface with hardware, including sensors and actuators. The state estimation module outputs position information generated in our case by ExplorerSLAM (Section 2.2). The state estimation module can also detect and introduce loop closures to correct accumulated drift. When loop closure is triggered, it outputs loop closure adjustments to globally relax the vehicle trajectory and the corresponding maps. Loop closure adjustments are used by high-level planners, that act on a global scale. The local planner and terrain analysis modules (Section 2.4.1) only process the local environment surrounding the vehicle and work in the odometry frame. The local planner and terrain analysis modules are extended to handle complex terrains that include negative obstacles such as cliffs, pits, and water puddles with a downward-looking depth sensor. The high-level planners such as the TARE exploration planner, the FAR route planner (Section 2.4.2), and other planners (for stuck recovery, etc.) are run in parallel for tasks such as exploration, going to waypoints, and returning home. The following sections provide details for each individual planner. The reader can refer to (Cao et al., 2022) and our open source repository[2] for a detailed review of the system.

[2] www.cmu-exploration.com



(a) (b)

Figure 12. (a) Camera view of the environment. (b) The terrain map: green points are traversable areas and red points are nontraversable areas. The coordinate frame indicates the robot's current location. The yellow curves are collision-free paths computed by the local planner.

2.4.1. Local motion planning

The local planner(Zhang et al., 2020) ensures safety when reaching the waypoints sent by high-level planners. It computes and follows collision-free paths that lead to the waypoint. The module precomputes a motion primitive library and associates the motion primitives to 3D locations in the vicinity of the vehicle. The motion primitives are modeled as Monte Carlo samples and organized in groups. In real time, when a location is occupied by obstacles, the module can determine which motion primitives are collided with the obstacle within milliseconds. The module then selects the group of motion primitives with the highest likelihood of reaching the waypoint. In Figure 12(b), the yellow paths represent the collision-free motion primitives. For ground vehicles, the traversability of the vehicle is determined by the characteristics of the terrain. The local planner takes the terrain map from the terrain analysis module, which is detailed in the next section. The module also has an interface to take in additional range data for collision avoidance as an extension option.

The terrain analysis module examines the traversability of the local terrain surrounding the vehicle. The module builds a cost map where each point on the map is associated with a traversal cost. The cost is determined by the roughness of the terrain. We use a voxel grid to represent the environment and analyze the distributions of data points in adjacent voxels to estimate the ground height. The points are associated with higher traversal costs if they are further away from the ground. Figure 12 gives an example terrain map obtained from the SubT Final Event, covering a $40\,\text{m} \times 40\,\text{m}$ area with the vehicle in the center. The green points are traversable, and the red points are nontraversable. For safety, the threshold for identifying nontraversable areas are set to be more conservative than what the robot can physically handle. Additionally, the terrain analysis module can handle negative obstacles that often result in empty areas with no data points on the terrain map. When negative obstacle handling is turned on, the module treats areas with no LiDAR points as nontraversable. Specifically, terrain analysis module creates "virtual" obstacle points around the boundary of areas lacking LiDAR points. The "virtual" obstacles are marked with high traversal cost and will be taken by the local planner to block off motion primitives that intersect with them.

2.4.2. Global path planning

Two main tasks for the ground robot autonomy are exploration and relocation to distant goal points for coordination or upon the operator's request. The two tasks are performed by the TARE planner for autonomous exploration and the FAR planner for route planning.

TARE planner (Cao et al., 2021) is a hierarchical framework for autonomous exploration, which computes paths that can be followed by the vehicle to thoroughly sweep its onboard sensors in environments unknown *a priori*. It uses a two-layer representation of the environment to plan the exploration path in a multiresolution way. As illustrated in Figure 13a, the planner uses low-resolution information to plan a coarse path at the global level. In the local area surrounding the

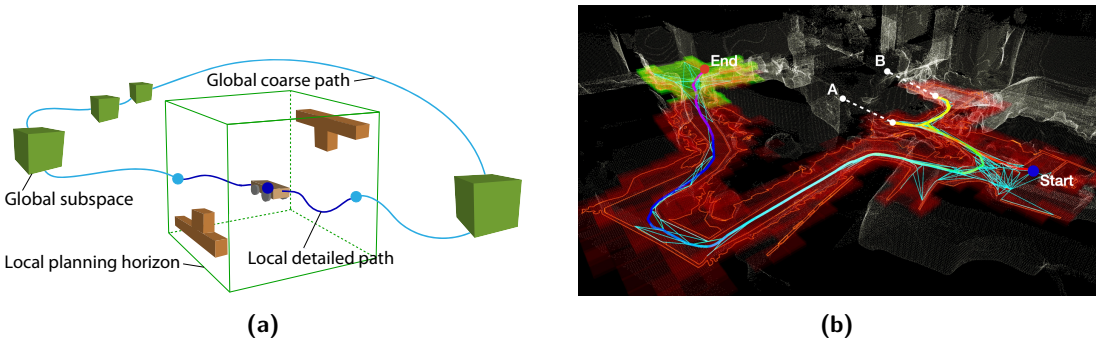


Figure 13. (a) Illustration of TARE planner's framework. In areas surrounding the vehicle (green box), dataare densely maintained and a local detailed path is computed (dark-blue curve). At the global scale, data are sparsely maintained in distant subspaces (solid green cubes) and a global coarse path is computed (light-blue curve). The local path and global paths are connected to form the exploration path. (b) Illustration of the FAR planner, where the robot is tasked with navigating through an unknown environment in simulation. The colorful curve is the vehicle trajectory staring at the blue dot and ending at the red dot. The planner attempts to guide the vehicle to the goal by registering obstacles in the environment (red polygons) and building a visibility graph (cyan lines) along with the navigation. The red area shows the global environment observed, and the yellow area represents the local environment used to update the local graph. A and B are dead ends. The vehicle first attempts the path that leads to the dead-ends, but re-plans to guide the vehicle out, eventually reaching the goal.

vehicle, the planner plans a detailed path using high-resolution information. The method optimizes the overall path by solving a traveling salesman problem at both the global and local levels. Compared to existing methods that rely on greedy strategies, the planner can effectively adapt to the structural environment and produce an approximately optimal exploration path that avoids redundant revisiting. TARE planner is evaluated in several large and complex environments both in simulation and in the real world. It is compared to state-of-the-art methods, namely, NBVP (Bircher et al., 2016), GBP (Dang et al., 2020), and MBP (Dharmadhikari et al., 2020). The results indicate that the TARE planner produces significantly higher efficiency in exploration and computation (Cao et al., 2021).

FAR planner (Yang et al., 2022) plans long-term routes for guiding the vehicle to reach goal points. It is a visibility graph-based planner that dynamically builds and maintains a reduced visibility graph along with navigation. The planner can handle both known and unknown environments. In a known environment, it uses a prior map to plan the route. However, in an unknown environment, it attempts multiple ways to guide the vehicle to the goal and identifies the layout of the environment during navigation. FAR planner models obstacles in the environment as polygons. It extracts edge points around the obstacles and convert the edge points into a set of polygons. The polygons are then merged into sensor data frames, from which the visibility graph is developed. FAR planner is evaluated in large and convoluted environments. It is compared to RRT* (Karaman and Frazzoli, 2011), RRT-connect (Kuffner and LaValle, 2000), A* (Hart et al., 1968), and D* Lite (Koenig and Likhachev, 2002) planners and demonstrates its strength in fast replaning over long distances. The planner uses ∼15% of a single CPU thread on an i7 computer to expand the visibility graph and a second CPU thread for path search. A path is found within 0.3ms in all of our experiments.

In addition to the TARE and FAR planners, behavior executive and multirobot coordination modules are developed for task switching and coordinating multiple robots, as shown in Figure 11a. The modules share explored and unexplored areas among different robots and invoke TARE and FAR planners cooperatively to achieve collaborative exploration. In particular, when a long-distance transition is determined due to new areas containing artifacts being discovered or operator waypoints being received, the behavior executive switches to FAR planner for relocating the vehicle. During the relocation, FAR planner uses a sparse roadmap merged from multirobots for high-level guidance. After the relocation is complete, the TARE planner becomes active for the exploration phase.

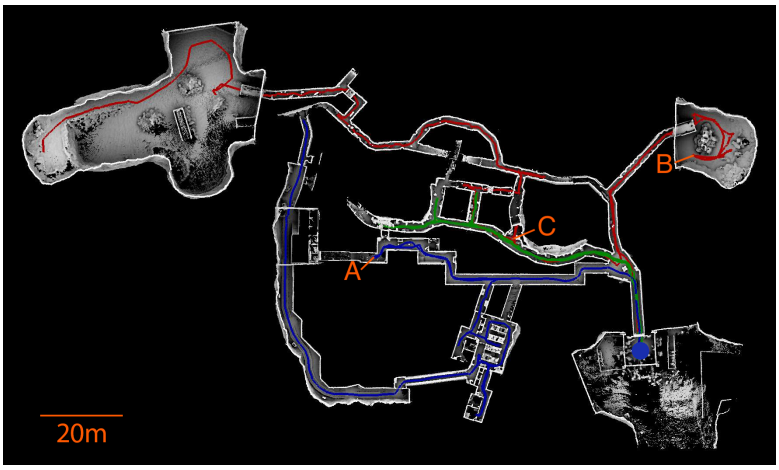


Figure 14. Result from SubT Final Event in Louisville Mega Cavern, KY. Three vehicles with red, green, and blue trajectories are deployed running TARE and FAR planners together. The segments between A and B on the red trajectory and between the start point (blue dot) and C on the blue trajectory use FAR planer to transit. Other segments on the trajectories use TARE planner to explore.

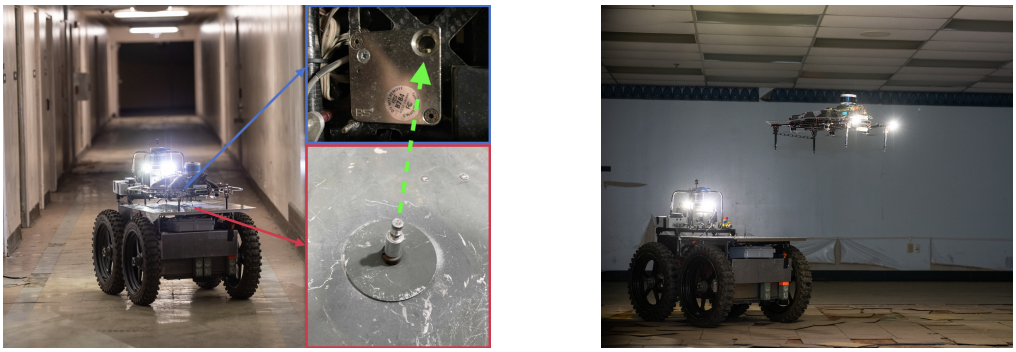
Figure 14 shows the result of the SubT Final Event in Louisville Mega Cavern, KY. The course combines tunnel, urban, and cave settings with complex topology. Our three vehicles use TARE and FAR planners together to traverse the environment. On the red trajectory, the vehicle first uses the TARE planner to explore B. Then, a request is received to transport to C using the FAR planner. At C, the vehicle resumes exploration using the TARE planner for the rest of the run. On the green trajectory, the vehicle uses only the TARE planner. On the blue trajectory, the vehicle passes to A using the FAR planner and switches to the TARE planner to explore thereafter. The three vehicles travel over 596.6 499.8, and 445.2 meters, respectively, over a time span of 2259 seconds.

2.5. Marsupial Ground-and-aerial System for Collaborative Exploration

Marsupial robotics systems have long been developed as collective navigation strategies to address mobility challenges. Such systems are able to exploit and combine the mobility strength of each robotic platform, including surface, aerial, wheeled, legged vehicles, in navigating over complex terrains which cannot be fully reached by each platform individually (Marques et al., 2015; Kalaitzakis et al., 2021). In particular, ground-and-aerial systems can lead to more thorough mapping or inspection of a 3D space by taking advantage of both the long endurance of ground vehicles and the agile maneuverability of aerial vehicles (Lee et al., 2021a; Lee et al., 2021b; De Petris et al., 2022).

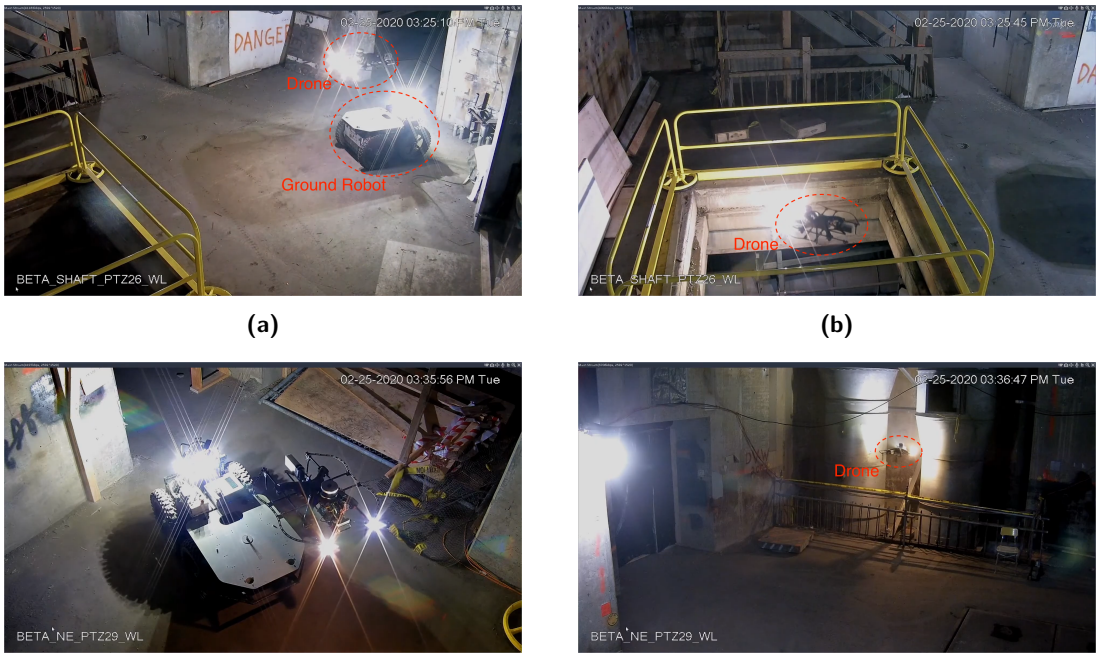
Team Explorer has developed a ground-and-aerial marsupial system in preparation for the Subterranean Challenge Urban Circuit in February 2020. The system involves a ground carrying a launch pad on which a quad-copter drone resides. Figure 15a shows the marsupial configuration where the drone is carried on top of the ground robot. The subfigures on the right of Figure 15a show the latching mechanism to secure the drone in place. When the ground robot is moving, the pin on the launch pad (lower red rectangle) is locked in the hole of the latch using a servo motor at the bottom of the drone (upper blue rectangle), indicated by the green arrow. Right before the launch, the latch releases and the pin is pulled back by a spring attached underneath to detach the drone from the launch pad. Figure 15b shows the instance where the drone hovers above the ground robot after being launched.

The system was successfully deployed in the SubT Urban Circuit and achieved collaborative exploration in complex environments. Figure 16 shows two instances in which ground robots launched drones to explore spaces that were not accessible by ground robots. Figures 16a and 16b



(a) (b)

Figure 15. (a) The configuration of our wheeled-aerial marsupial system, where a drone is carried at the top of a ground robot. Figures on the right show the latching mechanism used to secure the drone on the launch pad. (b) The instance after the drone is launched. The lights on the drone indicate its heading.

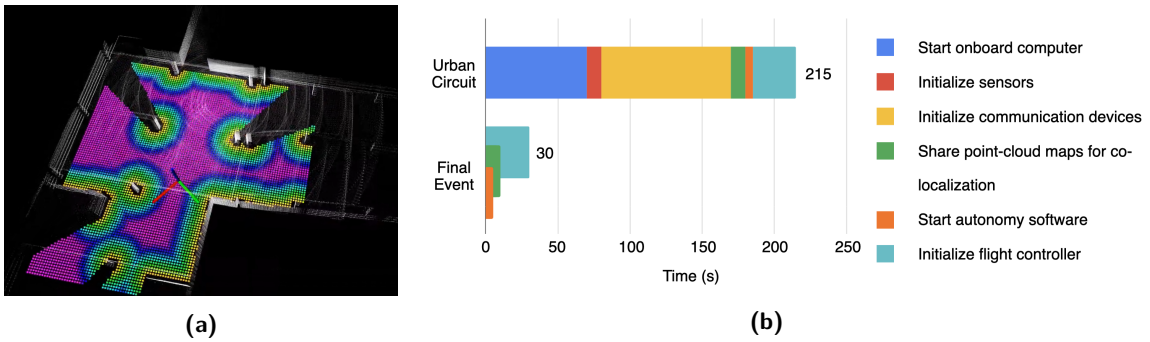


(a) (b)

(c) (d)

Figure 16. (a) The ground robot launched a drone next to a vertical shaft in the SubT Urban Circuit, 2020. (b) After the launch, the drone flew through the vertical shaft to explore the level underneath. (c) Another instance at the SubT Urban Circuit where a drone was launched. (d) After the launch from (c), the drone flew in between and scouted two pipelike structures, the lower part of which was blocked by barriers thus was not accessible by ground robots.

show the case when the drone was launched next to a vertical shaft, through which the drone was able to reach and explore another level of the building. Figures 16c and 16d show another instance where the drone was launched to scout the tight space between two pipelike structures. In both cases, the ground robot continued to explore the remaining space after launching the drone. The two examples demonstrate the use case where a marsupial aerial-and-ground system can be deployed to map and explore complex environments involving 3D structures and tight spaces, where either aerial or ground vehicles only cannot fully traverse.



(a) (b)

Figure 17. (a) A 2D slice of the Euclidean Distance Field for safety check before deploying the drone from the ground robot. The euclidean distance to the nearest obstacles is color coded with purple being the farthest and red being the nearest. (b) Comparison of time profiling for the deployment time in the Urban Circuit and the SubT Final Event.

Automated safety and utility check for choosing launch locations. In the Urban Circuit, the operator was involved in the launch procedure to manually choose a location that is safe and beneficial for the deployment of the drone. A preferable location should have sufficient vertical and horizontal clearance for the drone to takeoff and can lead to unexplored areas only accessible by drones. Otherwise, the drone deployment is likely to fail by crashing into obstacles or wastefully re-exploring the same areas as the ground robot. Determining such locations and putting the ground robot in place manually requires considerably time and mental bandwidth of the operator, and uninterrupted communication between the ground robot and the base station. To automate the process, we adopted a method similar to (Han et al., 2019) to compute an Euclidean Distance Field (EDF) around the ground robot for safety check. For a given location, the EDF gives the euclidean distance and the position to the nearest obstacle. When the ground robot navigates in the environment, it continuously computes the EDF with the accumulated LiDAR scan data and estimates the distance from the drone carried to the nearest obstacles. A 2D slice of the EDF at the height of the ground robot is shown in Figure 17a. A location is deemed safe with sufficient clearance both vertically and horizontally. In addition to the safety check, the ground robot also constantly checks the utility of launching the drone at different locations. This is done by running the exploration planner for the drone on the ground robot. The exploration planner extracts frontiers that are of interest to the drone, for example, frontiers that are higher up in the space or at narrow openings. With the heavier payload, thus more computational power carried by the ground robot, running an additional exploration planner is practical. When the ground robot has determined a safe location and enough space for the drone to explore, it enters the deployment procedure and the frontiers, roadmaps, and other information is shared with the drone.

Parallelized initialization procedures for software on the drone. We use "deployment time" to refer to the time it takes from the moment the ground robot initiates the deployment procedure to when the drone is hovering in the air. In the Urban Circuit, the drone was powered off before launch to preserve battery life. The deployment time was about 215 seconds, which consists of the time needed to start the onboard computer, share the point-cloud maps with the ground robot for co-localization in the global world frame (Section 2.2.3), and initialize the sensors, communication devices, and the flight controller. In the SubT Final Event, the deployment time was reduced to about 30 seconds. This is achieved by placing the drone in "hibernation" mode before launch rather than turning the drone off. In the "hibernation" mode, the onboard computer is powered on and communication devices are initialized, yet the computationally expensive, thus power-consuming, autonomy software is not running to minimize power consumption. Empirically, the "hibernation" mode has a negligible affect on the flight time of the aerial vehicles. Furthermore, the initialization of different software modules is parallelized to minimize the overall time, as shown in Figure 17b. Other than the "hibernation" mode and parallelized initialization procedure, the overall pipeline

Figure 18. (a) The place where the ground robot launched a drone in the SubT Final Event. The red rectangle at the lower left corner shows the obstacles that the ground robot cannot traverse through: a set of stairs and a pole at the top. (b) After the launch, the drone flew beyond the obstacles and explored the subway station. (c) The drone flew up to explore the top of the scaffolding, where it detected and localized a backpack artifact. (d) When the drone ran out of battery, it landed at the place where it was launched.

for launching aerial vehicles from ground vehicles remain the same as in the Urban Circuit, which has been proven effective in collaborative exploration. Readers can refer to (Scherer et al., 2021) for detailed description of the overall launching procedure.

Our marsupial ground-and-aerial system was successfully deployed in the SubT Final Event. In the Urban environment, one of the wheeled robots (R1) deployed a drone (DS4) in a hallway. The hallway was blocked by a set of stairs and a pole in the middle (Figure 18a). The ground robot was able to identify a safe location to launch the drone. After launch, the drone flew over obstacles to explore a subway station (Figure 18b) and scout the scaffolding structures at height (Figure 18c). When it almost reached the flight-time limit, the drone flew back and landed where it was launched for other robots to pick up the artifacts it had detected. The point-cloud map built after the exploration is shown in Figure 27f.

2.6. Communications

Communications was one of the principal challenges of SubT. Not only was communications the backbone for optimized robot exploration and artifact transmission to the base station, the physical infrastructure of the communication network had to be autonomously deployed in an unknown, comms-degraded world. As a whole, we break our communication solution down into a sequence of layers: physical layer, middleware or mesh layer, and an application layer.

2.6.1. Physical Layer

For the final competition, we used DX2-50 radios supplied by Rajant Radio. These were 5.0 GHz modules that we used as breadcrumbs for our *adhoc* wireless mesh network (WMN). Running at a power of 1 watt and IEEE 802.11a, these radios were capable of handling upwards of 15 mbps of robot traffic.

Figure 19. A Rajant DX2-50 radio (Rajant, 2023) that we used for communication between robots.

Figure 20. The communication node dropper onboard the R2 ground robot with 9 fiberglass node casings.

As can be seen in Figure 19, the DX2-50 radios had a blue magnesium shell. Despite the compact form-factor, the dimensions of the shells were not conducive to the node dropping mechanism we had designed. Instead, we created new fiberglass shells, which consisted of the electronics board and radio card of the radios.

In order to deploy the nodes in the field, we manufactured a device called the node dropper. This mechanism, as well as the fiberglass nodes, can be seen on the front of R2 in Figure 20. The mechanism used solenoid actuated furniture locks to hold onto the top piece of the fiberglass nodes. When a ROS message was commanded to drop a radio, the digital signal would be relayed to a LabJack, which would convert the signal to a 3.3 V analog out signal that drove a 12 V relay to the solenoid actuator. The transmission of this signal would then release the actuator, thus dropping the node to the ground.

As a whole, our three ground robots (i.e., R1, R2, R3) could hold 9, 9, and 6 nodes, respectively. This provided us the capability to deploy 24 repeater nodes in the field, in addition to the radios fitted to each endpoint (e.g., all robots and basestation).

2.6.2. Middlware (Mesh) Layer

At the middleware (mesh) layer, there were two core services used. Closer to the physical layer was the Rajant InstaMesh (Rajant, 2023). This was Rajant's proprietary layer 2 meshing software used with the DX2-50 radios. While the algorithms of the mesh were untouched, we did adjust many QoS parameters including message reliability, number of retries, attempted throughput, and the like. Appropriate adjustment of queue sizes and number of retries was particularly important considering that we had several layers within the OSI model all performing varying forms of message reliability and resend. Instances of this could be seen in the Rajant InstaMesh, networking middleware, and ROS nodes run on vehicles. As multiple layers of reliability and resend protocols can create an unbounded number of queued messages, we had to ensure consistency across our different abstractions of the communication stack.

The second core service we used for our middleware (mesh) layer was Data Distribution Service, also known as, DDS (Pardo-Castellote, 2003). This service was closer to our application or logical level logic. While there are many instances of DDS available on the market, we procured a license from Real Time Innovations (RTI). This DDS instance provided several core functions out-of-the-box including autodiscovery, flow control, quality of service (QoS), multicast, and many others. While we did some configuration adjustment to bootstrap the DDS auto-discovery process, much of the low-level networking configuration was done automatically from DDS. This included processes such as endpoint discovery, topic discovery, and TCP handshaking.

2.6.3. Application Layer

At the application layer there were three main processes running. The first and most important was the communication manager. Overall, the communication manager served as the ROS1 bridge to DDS. An example of the communication manager can be seen in Figure 21. As a whole, the purpose of it was to spin up instances of DDS participants per endpoint (e.g., robots and basestation). Each DDS participant would then publish and subscribe to a series of relevant DDS topics. The job of the communication manager was to convert local ROS messages of interest into DDS compliant messages (IDL format) and send them over the wire to other robots of interest. Once received, the DDS participant on the receiver side would convert the message back from IDL to ROS format and publish that message locally on the new robot.

While the main objective of the communication manager was to serve as a ROS-DDS bridge, it also included a network watchdog which would monitor the general health of the wireless mesh network. Phenomena such as increased package drops, large message latency, or too many robots on the network would cause the network health watchdog to throttle the frequency and priority of specific DDS messages. This was done both by throttling messages at the ROS layer, as well as, leveraging the QoS and flow control parameters offered by DDS to maximize network health.

The second main application process was the communication node planner. This ROS1 application ran only on the three ground robots and its objective was to determine appropriate places

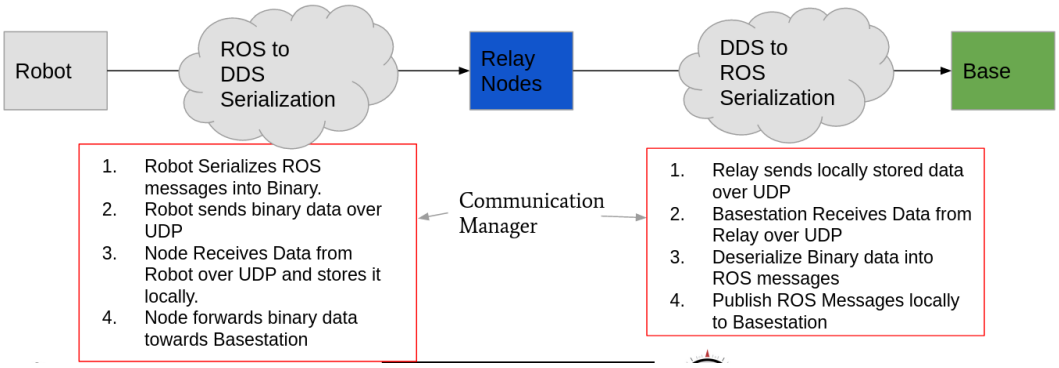


Figure 21. An overview of our communication manager pipeline.

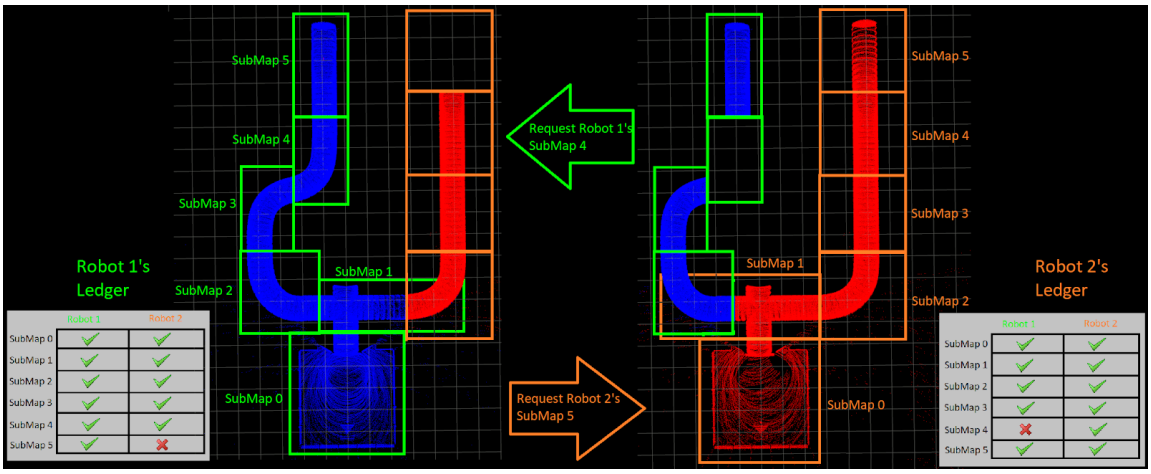


Figure 22. An illustrated example of our communication ledger used to store and share information between robots.

for the ground robots to drop communication nodes. While experimenting with the node planning logic, many different algorithmic approaches, such as the "Art Gallery Problem" (Chesnokov, 2018), were used to try to determine the optimal node locations. Given the large uncertainty of the spaces, as well as, the need to create a robust, redundant network architecture, we selected a much more simple distance and line-of-sight algorithm. More specifically, the robot would drop communication nodes somewhere between a minimum and a maximum distance metric to other communication nodes based on the quality of the robot's line-of-sight to previously deployed communication nodes. As a whole, this algorithm did not ensure maximum potential network coverage, however, it was very robust and maximized topology redundancy.

The third main application process was the ledger. This ROS1 application was tightly coupled to the communication manager serving as the ROS1-DDS bridge. The ledger, as per its standard definition, tried to create a consensus of knowledge between all robots and base stations. An example of this can be seen in Figure 22.

Important information that was of interest to the entire robot team, such as pointclouds or artifacts, would be encapsulated in a message called a submap. While each robot would explore, expand its local map, and identify artifacts of interest, it would make submaps of these events. In an attempt to share that useful information across all robots, the ledger would poll each robot to determine what submaps it did and did not have. Omissions from a robot's personal ledger would induce submap requests from other robot's possessing the information of interest. This mechanism of requesting and replying with submaps in order to create a consensus of total information across all vehicles, worked very similarly to the standard definition of a ledger. As the ledger was a ROS1 node, it did not explicitly have the ability to send messages over the wire. Using the communication manager though, its ROS1 submap messages were frequently sent between robots in the DDS compliant IDL format.

3. Evaluation and Performance in the Final Event

In total, eight robots were deployed in the competition, including three custom-built wheeled robots (R1, R2, and R3 detailed in Section 2.1.1), one commercially available legged robot (Spot from Boston Dynamics), three custom-built quadcopters (DS1, DS3, DS4 detailed in Section 2.1.2) and one commercially available quadcopter (Canary). Figure 23 shows all types of robots deployed in the competition. Due to hardware and integration issues of the commercially available platforms (Spot and Canary), they had limited deployment time and role in the competition. In the following discussion, we focus on the performance of our custom-built robot platforms.

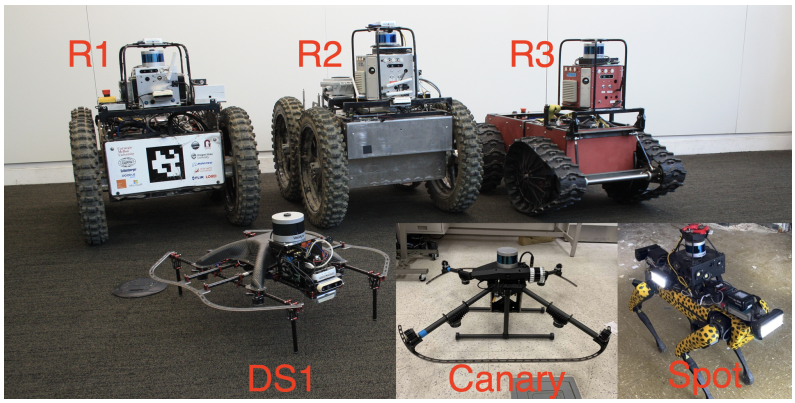


Figure 23. The six types of robots that we used in the Final Event.

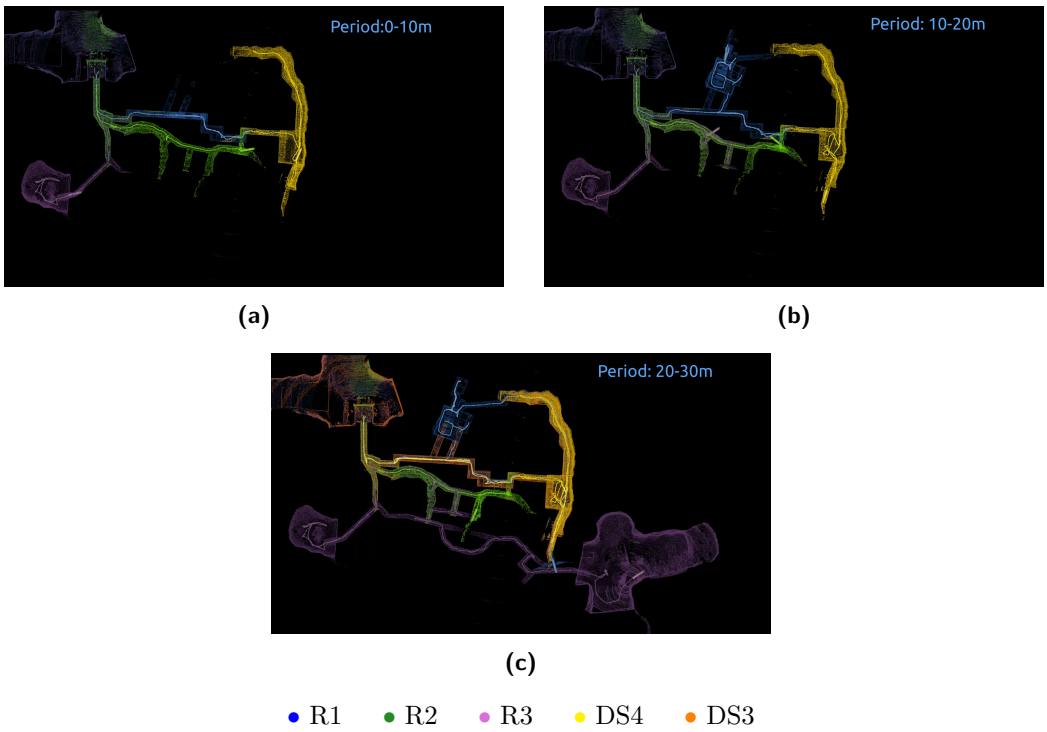


Figure 24. Exploration progress in the first 30 minutes. (a) In the first 10 minutes, there were four robots deployed which covered more than half of the environment (about 15 out of 28 sectors). [(b) and (c)] In the first 30 minutes, three ground robots and two aerial robots had already covered more than 80% of the environment (24 out of 28 sectors).

Team Explorer scored 17 out of 40 points and achieved fourth place in the Final Event of the Subterranean Challenge. In addition, we received the "Most Sectors Explored" award, having reached 26 of the 28 sectors, as defined by DARPA. The 26th sector was reached in less than 30 minutes. Figure 1 shows the number of sectors explored over time by all participating teams. Our system not only explored the most sectors but also explored the fastest among all teams. The exploration progress visualized by the point-cloud maps built over time is shown in Figure 24.

Figure 25 summarizes how our pipeline handled each of the artifacts on the course. Out of the 40 artifacts, our robots were able to visit 32 of them. Then, the object detection system correctly

Table 3. Portion of the full map sensed by each robot (map coverage) and the number of points in their pointcloud map that are further than 1 meter from ground truth (map deviation) for each robot.

Robot	Coverage (%)	Map Deviation (%)	Visited Sectors	Scores
R1	36.3	0.07	10	7
R2	23.8	0.07	4	4
R3	52.8	14.96	18	8
DS1	28.7	0.71	5	5
DS3	21	15.56	5	4
DS4	13.3	11.4	4	3
Canary	13	0.09	2	1
All	84.7	11.6	30	17

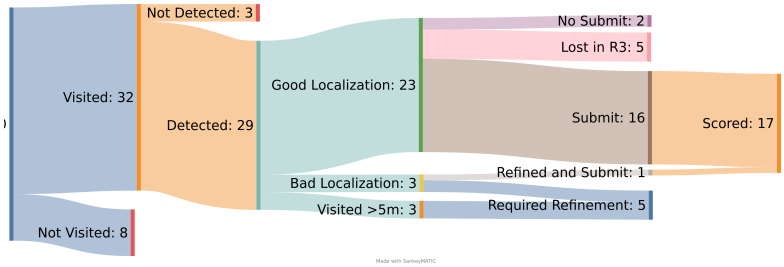


Figure 25. A diagnosis of the scored and unscored artifacts in the Final Event. Team Explorer scored 17 points in total. The team of robots physically visited 32 our of the 40 artifacts during the one-hour competition run. However, 3 artifacts were not detected by the artifact-detection pipeline, 6 artifacts had large localization error and 7 artifacts were not communicated back to the base station or not submitted in time by the operator.

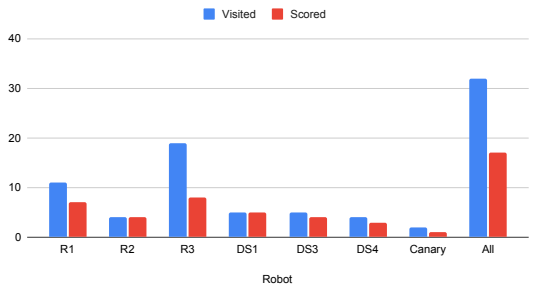


Figure 26. Number of artifacts visited and scored by each robot (including duplicates). There was a fairly even distribution of artifacts between the robots. R3 visited the most, but many of these were not scored due to the cliff fall outside of communication range (see Section 3.1.3).

detected 29. Most of these detections (23) were correctly localized to the map. Another 3 artifacts were detected when the robot was farther than 5 m from the artifact, and therefore the localization was unable to pinpoint them correctly. These were signal-emitting artifacts such as gas, WiFi, and the cube. Three artifacts were badly localized even when we visited them close enough. One of these was refined by our operator and scored. The others were on the base station and would require time for the human operator to refine their positions based on the geometry of the environment.

Out of the artifacts with good localization, 5 were lost when R3 fell down a cliff in the farthest area of the course. Two other artifacts were at the base station and were not submitted before time ran out. The other 16 were correctly submitted and scored, making a total of 17 points. Figure 26 shows how many artifacts each robot visited and how many of those were scored. Note that some artifacts were visited by multiple robots; therefore, the sum of individual robots is greater than the final score. Table 3 summarizes these metrics.

3.1. SLAM

The ExplorerSLAM system was used on all robots. The aerial vehicles did not use the loop closure module due to the limited computing capabilities and lower runtime. The individual maps obtained are shown in Figure 27. Figure 28 shows the map deviation metric calculated for each robot using the DARPA's own scoring system (Schang et al.,). Figure 29 shows the overall map and deviation aggregated from all robots. We also plot the coverage metric, which indicates which proportion of the points in the ground-truth pointcloud were mapped within 1m. Note that 4 of the 7 robots had a map deviation less than 1%, indicating a near perfect run. Overall, our combined system achieved the deviation of 11.6% of the map while covering 84.7% of the map. All of our visual artifact detections were correctly localized within the DARPA scoring tolerance, indicating that SLAM was not a major issue in accounting for missing artifacts. The 11.6% map deviation was mainly due to the SLAM drift of 3 robots, which is explained in the following.

3.1.1. DS4

This UAV was deployed in a marsupial setup on the back of R1. It explored the subway station and went to the rail section. There, it encountered heavy fog. It almost completely encircled the robot and caused drift in the SLAM module. In turn, this caused problems for the local planner, leading the robot to push against a wall, as seen in Figure 30. It was able to recover and resume its exploration. It is worth noting that the dust points accumulated in the point-cloud map are also counted as mapping deviation by the DARPA system, since they have no corresponding point in the ground truth cloud.

3.1.2. DS3

This robot started from the basestation area, explored the train tracks, and landed back at the starting gate. The problem with DS3 was that the auto-calibration process resulted in an erroneous transformation between the robot origin and the DARPA-defined frame. In particular, it added a roll and pitch offset that increases the deviation of the map (Figure 27e). This also prevented the drone from properly coordinating with other drones that had previously visited the area, preventing it from finding more unexplored areas. Finally, it also encountered heavy fog on the train tracks, but was able to handle it robustly due to the dust filter.

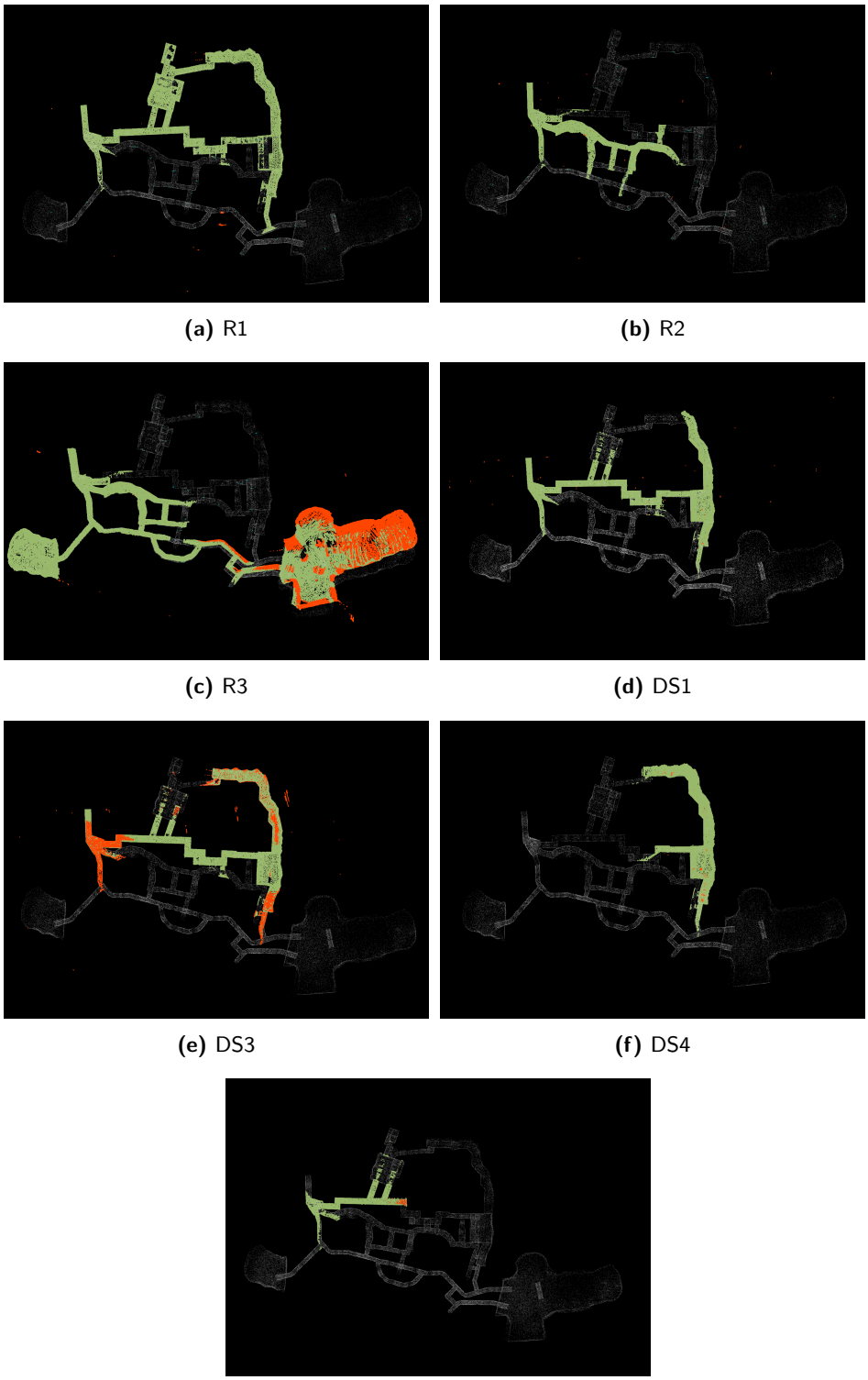
3.1.3. R3

This ground robot covered the most area among all robots. This is because of its smaller size which enables it to easily go through narrow passages and maneuver in tight space in the environment. It explored by itself more than half of the course. Unfortunately, it fell off a cliff and did not return to the staging area. Before the competition, we had determined that R3 consistently had higher levels of SLAM drift than our other ground robots due to the LiDAR having higher sensor noise, and we made an effort to correct that on the loop closure level. However, because it had the longest trajectory, the drift eventually grows large enough that parts of the cave area at the end of its trajectory are farther than 1 m from the ground truth (Figure 27c). However, the maximum deviation was still below the 5-m tolerance necessary to score a point.

3.2. Mobility Challenges for Ground Robots

Our robots collectively achieved the farthest traversal and most thorough exploration, yet there were cases when they were unable to handle or recover from perception and mobility challenges. In the following scenarios, ground robots were unable to plan paths, which eventually led to immobilized robots that would need the operator to intervene.

- *Narrow openings.* R1 was stuck in a narrow opening that led to the cave environment, as shown in Figure 31. The local planner has been tuned to allow the robot to enter openings that are barely wider than its width. However, the protruding rocks on the sides of the entrance were in the sensor blind spot when the robot moved too close, which were wedged between the



(a) R1 **(b)** R2

(c) R3 **(d)** DS1

(e) DS3 **(f)** DS4

(g) Canary

Figure 27. Maps obtained by each robot in the Final Round, compared against ground-truth (white) provided by DARPA (Schang et al.,). Green points are registered within a 1 meter mapping tolerance to the ground-truth, while orange points are registered with error greater than the tolerance.

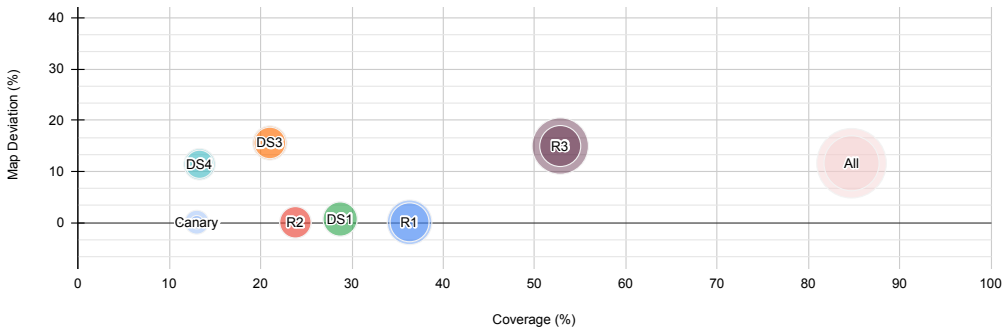
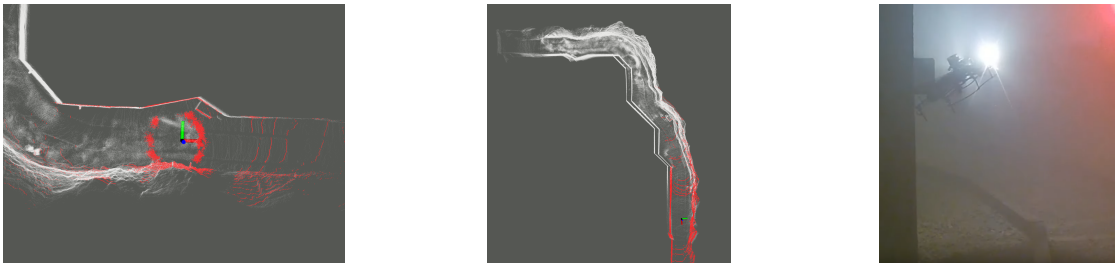


Figure 28. Impact Analysis per robot. The figure shows how much of the ground truth pointcloud was covered by each robot, and the mapping deviation accrued, both as defined by DARPA guidelines. The inner bubbles indicate how many artifacts were scored and the outter bubbles indicate how many were visited.

Figure 29. Aggregated map deviation of complete map collected by Team Explorer. Green points are within 1 meter tolerance to the ground-truth and orange points are out of the tolerance.

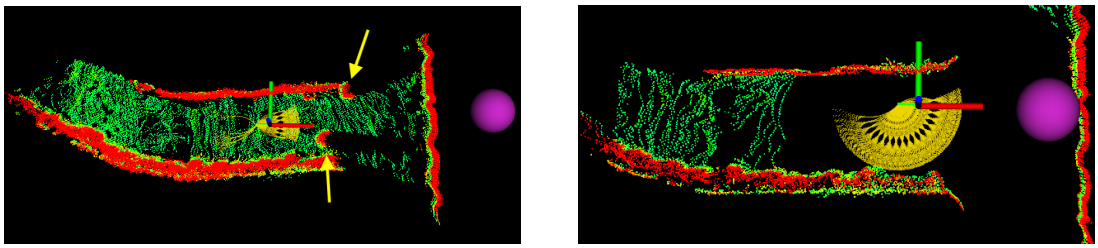


(a) **(b)** **(c)**

Figure 30. DS4 SLAM Issues: It encountered heavy dust in the rail section, which caused SLAM drift and led to collision with the wall. (a) The drone, indicated by the coordinate frame, was surrounded by noise points caused by heavy dust. (b) Misaligned point-cloud map due to the SLAM drift. (c) Camera view of the drone hitting a wall.

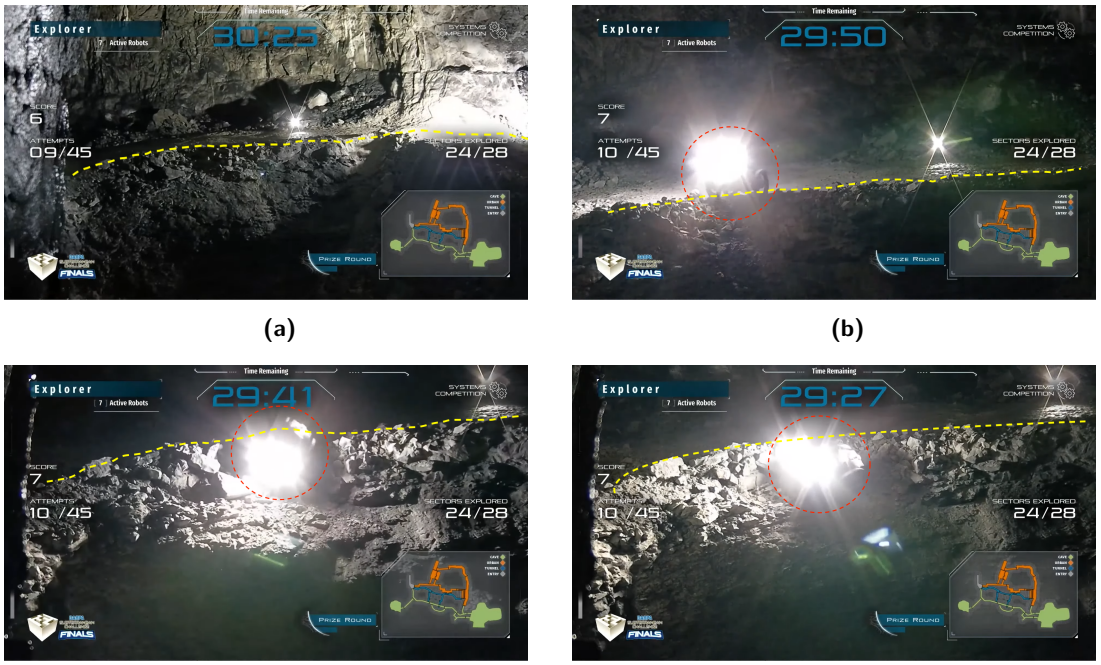
front and rear wheels of the robot when it passed by, as shown in Figure 31(d). The robot was immobilized by the obstacles hereafter. Since the obstacles were in the blind spots of the sensors, the planners for recovery were not able to plan paths to free the robot.

- *Negative obstacles.* R3 fell off a cliff at the farthest end of the environment, as shown in Figure 32. The robot uses a downward facing depth camera for detecting drop-offs. Due to the limited sensor range (2 m), the robot can only detect drop-offs when it is close enough to



(a) (b)

Figure 31. R1 stuck at a narrow opening to the cave environment. The coordinate frame indicates the robot's position and orientation (facing the red axis). Yellow curves are collision-free paths computed by the local planner. Green and red points indicate traversable and nontraversable areas output by the terrain analysis. Two obstacles [indicated by the yellow arrows in (a)] on the sides were wedged in between the front and rear wheels of the robot when it tried to pass the opening, causing the robot to stuck. (b) As the robot moved close, the obstacles fell in the blind spot of the LiDAR sensor, causing the local planner fail to account for them. The purple points indicate waypoints the robot was trying to reach.



(a) (b)

(c) (d)

Figure 32. R3 fell off a cliff. Dotted circles indicate locations of the robot. The dotted yellow curve indicates the edge of the drop-off. (a) The drop-off, the robot was to the right of the camera view. (b) The robot approached to a rock pile before the drop-off. (c) The robot climbed over the rock pile and detected the drop-off. (d) The robot tried to turn back but slid on loose dirt. It fell down the cliff afterwards.

the edge of the drop-off. In the competition, the robot did not detect the rounded and rocky drop-off edge from far away. When it approached close enough to detect the drop-off and tried to turn back, the robot slid on the loose dirt and fell down the cliff.

- *Heavy fog.* In the tunnel environment, heavy fog appeared as false positive obstacles to the depth camera mounted in front of the robot. The red trapezoid in Figure 33(a) shows the traversability map using the data from the depth camera. In Figure 33(b), the camera was partially blocked by heavy fog, causing part of the terrain to be considered as obstacles thus

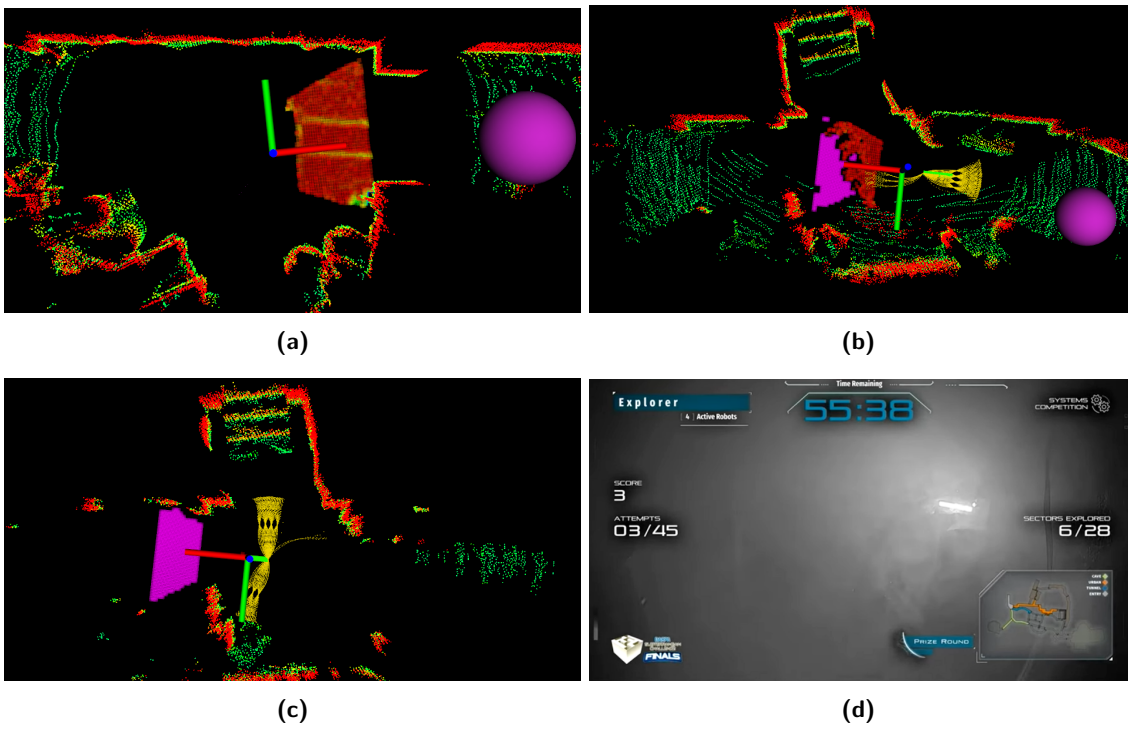


Figure 33. (a) The traversability map (red trapezoid) computed with the data from the depth camera when there was no fog. Purple point indicates the waypoint that the robot tried to reach. (b) The depth camera was partially blocked by the fog, causing part of the terrain to be considered obstacles (purple). (c) The depth camera was completely blocked thus the terrain was deemed nontraversable, causing the local planner fail to find collision-free paths (yellow curves) through the area. (d) Camera view of the robot stuck in the heavy fog.

nontraversable (colored purple). In Figure 33(c), the entire field of view of the depth camera was blocked, causing the local planner to fail to find collision-free paths through the area.

4. Lessons Learned

In this session, we present the lessons learned through our participation in the DARPA Subterranean Challenge and our broader testing. We hope that they can provide valuable insights to help the future development and deployment of a complex robotic system for time-critical missions.

1. Minimize changes and avoid adding new systems close to performance deadlines. last-minute changes are high risk. It is human nature to seek perfection, but it can come at the risk of losing everything. Sometimes, a robust system that can reliably perform a small number of tasks is more desirable than one that is versatile but reliable. Later in the program, we spent too much time preparing for things that were not in the challenge, e.g., (a) traversing multifloor staircase, (b) detecting objects far from robots, and (c) SLAM for very long and featureless corridors. They may have complicated the system and taken time away from practicing with a proven functional system.
2. Cave environments are challenging for mobility. The man-made cave environment in the final competition is more challenging than we have expected. It disabled 2 of our ground robots loaded with artifacts. It is not hard to imagine that a natural cave can only be more difficult to deal with for robots.
3. Deployment strategy matters. Being aggressive in deploying robots into the field by going fast and deep may be risky. Keeping things steady and slow can be more efficient when dealing

with unknown environments, by ensuring that the robots are able to successfully return to the base station and report findings about the environment. In particular, an iterative deepening strategy might be more successful in balancing the risk of deep traversal and gathering more information.

4. Communication devices should be mounted in a safe place on the robot to avoid damage in rollover/crash situations. Communication loss in the event of mobility issues negates the entire effort. Situational awareness for operators is sometimes more important than a functional robot.
5. Unit tests should be conducted in more complex and realistic scenarios. For example, fog and drops caused us to pay the price in the final competition. We only found that fog was a problem for our L515 cameras after the second round. We purposely never tested the ground robots' capability of avoiding large drops to avoid damaging the robots before the competition.
6. Full system tests are valuable and should have been done more frequently. It is important for the operator to learn how to handle all the complexity when the full system is running: We spent too much time later in the program trying to perfect the individual subsystems, and less time focused on the operator interactions with the full system.
7. The importance of issue tracking, follow-up, and closing issues. Complex robotic systems are cross-correlated, every change may affect other parts of the system. Tracking these takes time and thorough testing.
8. Focus on the metrics that matter for the mission. It is important to ensure that the robots can be resilient and fast for autonomous navigation in general. However, it is also critical to develop the mission-specific capabilities, such as artifact detection for a search and rescue mission.
9. User interfaces are important. In the final run our robots produced an enormous amount of potential artifacts in a very short time since we had so many systems covering such a large amount of the course and since the environment had a larger range of potential artifact-like textures such as graffiti, jackets, and colorful walls. This caused our operator to be overwhelmed with a large list of potential artifacts. Even simple user interface improvements such as sorting objects by detection confidence could have helped prioritize submissions since we ended the run with potential submissions remaining.

5. Conclusions and Future Work

In this work, we presented our approach to developing and deploying a resilient robotic system for navigation in challenging environments. Our system consists of a team of heterogeneous robots that leverage both aerial and ground mobility, whose superior navigation performance won the "Most Sectors Explored Award" in the DARPA Subterranean Challenge Final Event. We detail the mechanical designs and autonomy algorithms that enable fast traversal and efficient collaborative exploration in complex environments. Lastly, we provide our lessons learned during the participation in the competition as well as develop a complicated yet practical robotic system.

In future work, we intend to address the underlying challenges that we have encountered through the competition and those that can be anticipated in real-world deployment. In particular, we have learned from competition the importance of reliable detection and robust recovery strategies for subsystem failures, especially for perception and mobility modules. The ability to deal with unprecedented situations intelligently is crucial for real-world deployment, of which scenarios can only be more challenging than in competition. One potential research direction is to incorporate semantic information into the traversability analysis for both ground and aerial vehicles, which can foster long-term reasoning to avoid or escape adversarial situations. Another direction is to develop communication-aware coordination strategies for multirobot exploration, which can potentially both improve exploration efficiency and alleviate the negative impact of any failing robots. Finally, the identification and recovery of mapping failure in multirobot coordination is worth investigating.

Acknowledgments

The content is approved for public release and the distribution is unlimited. This work is partially sponsored by DARPA under agreement No. HR00111820044. The content is not endorsed by and does not necessarily reflect the position or policy of the government or our sponsors.

This work would not have been possible without the dedicated efforts of Team Explorer and the generous support of our sponsors. In particular, we would like to thank Team Explorer and advisors: Joshua Spisak, Hengrui Zhang, Andrew Saba, Nathaniel Shoemaker-Trejo, Ralph Boirum, Warren Whittaker, Tim Angert, Jim Teza, Howie Choset, Michael Kaess, Anthony Rowe, Sanjiv Singh for their great contributions to the project. We would like to thank our sponsors: Richard King Mellon Foundation, Schlumberger, Microsoft, Boeing, CNH Industrial, AWS Robomaker, Near Earth Autonomy, Honeywell, Real-Time Innovations (RTI), CanaryAero, Epson, Parker Hannifin Corporation, FLIR, IDS Imaging, and Doodle Labs.

ORCID

Chao Cao https://orcid.org/0000-0002-4192-7641
Lucas Nogueira https://orcid.org/0000-0001-7220-2937
Hongbiao Zhu https://orcid.org/0000-0002-1227-4047
John Keller https://orcid.org/0000-0002-3672-6212
Graeme Best https://orcid.org/0000-0003-0443-8248
Rohit Garg https://orcid.org/0000-0001-9646-224X
David Kohanbash https://orcid.org/0000-0002-9526-7458
Shibo Zhao https://orcid.org/0000-0003-1801-8156
Fan Yang https://orcid.org/0000-0002-4452-4979
Ryan Darnley https://orcid.org/0000-0002-3677-0627
Robert DeBortoli https://orcid.org/0000-0003-1460-1635
Bill Drozd https://orcid.org/0000-0001-9994-8990
Ian Higgins https://orcid.org/0000-0003-4921-4344
Greg Armstrong https://orcid.org/0000-0002-3393-7954
Ji Zhang https://orcid.org/0000-0002-4692-5645
Geoffrey A. Hollinger https://orcid.org/0000-0001-6502-8950
Sebastian Scherer https://orcid.org/0000-0002-8373-4688

References

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.

Agha, A., Otsu, K., Morrell, B., Fan, D. D., Thakker, R., Santamaria-Navarro, A., Kim, S.-K., Bouman, A., Lei, X., Edlund, J., et al. (2021). NeBula: Quest for robotic autonomy in challenging environments; team CoSTAR at the DARPA subterranean challenge. *arXiv preprint arXiv:2103.11470*.

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., Garg, R., Keller, J., Hollinger, G. A., and Scherer, S. (2022). Resilient multi-sensor exploration of multifarious environments with a team of aerial robots. In *Robotics: Science and Systems*.

Bircher, A., Kamel, M., Alexis, K., Oleynikova, H., and Siegwart, R. (2016). Receding horizon "next-best-view" planner for 3D exploration. In *International Conference on Robotics and Automation (ICRA)*, Stockholm, Sweden.

Cao, C., Zhu, H., Choset, H., and Zhang, J. (2021). TARE: A hierarchical framework for efficiently exploring complex 3D environments. In *Robotics: Science and Systems Conference (RSS)*, Virtual.

Cao, C., Zhu, H., Yang, F., Xia, Y., Choset, H., Oh, J., and Zhang, J. (2022). Autonomous exploration development environment and the planning algorithms. In *IEEE International Conference on Robotics and Automation (ICRA)*, Philadelphia, PA, USA.

Chesnokov, N. (2018). The art gallery problem: An overview and extension to chromatic coloring and mobile guards.

Colledanchise, M. and Ögren, P. (2018). *Behavior trees in robotics and AI: An introduction*. CRC Press.

Dang, T., Tranzatto, M., Khattak, S., Mascarich, F., Alexis, K., and Hutter, M. (2020). Graph-based subterranean exploration path planning using aerial and legged robots. *Journal of Field Robotics*, 37(8):1363–1388.

De Petris, P., Khattak, S., Dharmadhikari, M., Waibel, G., Nguyen, H., Montenegro, M., Khedekar, N., Alexis, K., and Hutter, M. (2022). Marsupial walking-and-flying robotic deployment for collaborative exploration of unknown environments. *arXiv:2205.05477*.

Dellaert, F. and Contributors, G. (2022). borglab/gtsam.

Dharmadhikari, M., Dang, T., Solanka, L., Loje, J., Nguyen, H., Khedekar, N., and Alexis, K. (2020). Motion primitives-based path planning for fast and agile exploration using aerial robots. In *International Conference on Robotics and Automation (ICRA)*, Paris, France.

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.

Forster, C., Carlone, L., Dellaert, F., and Scaramuzza, D. (2015). Imu preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation. In *Proc. of the Robotics: Science and Systems (RSS)*, Rome, Italy.

Han, L., Gao, F., Zhou, B., and Shen, S. (2019). Fiesta: Fast incremental euclidean distance fields for online motion planning of aerial robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Macau, China.

Hart, P. E., Nilsson, N. J., and Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2):100–107.

Hudson, N., Talbot, F., Cox, M., Williams, J., Hines, T., Pitt, A., Wood, B., Frousheger, D., Surdo, K. L., Molnar, T., et al. (2021). Heterogeneous ground and air platforms, homogeneous sensing: Team CSIRO Data61's approach to the DARPA subterranean challenge. *arXiv preprint arXiv:2104.09053*.

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.

Kalaitzakis, M., Cain, B., Vitzilaios, N., Rekleitis, I., and Moulton, J. (2021). A marsupial robotic system for surveying and inspection of freshwater ecosystems. *Journal of Field Robotics*, 38(1):121–138.

Karaman, S. and Frazzoli, E. (2011). Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research (IJRR)*, 30(7):846–894.

Koenig, S. and Likhachev, M. (2002). D* lite. In *AAAI Conference on Artificial Intelligence*, Alberta, Canada.

Kuffner, J. J. and LaValle, S. M. (2000). RRT-connect: An efficient approach to single-query path planning. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*.

Lee, C., Best, G., and Hollinger, G. A. (2021a). Optimal sequential stochastic deployment of multiple passenger robots. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*.

Lee, C. Y. H., Best, G., and Hollinger, G. A. (2021b). Stochastic assignment for deploying multiple marsupial robots. In *International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*. IEEE.

Markoff, J. (1991). The creature that lives in Pittsburgh. *The New York Times*.

Marques, F., Lourenço, A., Mendonça, R., Pinto, E., Rodrigues, P., Santana, P., and Barata, J. (2015). A critical survey on marsupial robotic teams for environmental monitoring of water bodies. In *OCEANS 2015-Genova*.

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.

Murphy, R. R. (2021). How robots helped out after the surfside condo collapse. *IEEE Spectrum*.

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.

Ohradzansky, M. T., Rush, E. R., Riley, D. G., Mills, A. B., Ahmad, S., McGuire, S., Biggie, H., Harlow, K., Miles, M. J., Frew, E. W., et al. (2021). Multi-agent autonomy: Advancements and challenges in subterranean exploration. *arXiv preprint arXiv:2110.04390*.

Palieri, M., Morrell, B., Thakur, A., Ebadi, K., Nash, J., Chatterjee, A., Kanellakis, C., Carlone, L., Guaragnella, C., and Agha-mohammadi, A.-a. (2021). Locus: A multi-sensor lidar-centric solution for high-precision odometry and 3d mapping in real-time. *IEEE Robotics and Automation Letters*, 6(2):421–428.

Pardo-Castellote, G. (2003). Omg data-distribution service: Architectural overview. In *23rd International Conference on Distributed Computing Systems Workshops, 2003.*, pages 200–206. IEEE.

Rajant (2023). Rajant kinetic mesh network. https://rajant.com.

Rouček, T., Pecka, M., Čížek, P., Petříček, T., Bayer, J., Šalanskỳ, V., Heřt, D., Petrlík, M., Báča, T., Spurnỳ, 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.

Rusu, R. B. and Cousins, S. (2011). 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China.

Schang, A., Rogers, J., and Maio, A. Map analysis (version 2). https://github.com/subtchallenge/map_analysis.

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)*.

Scherer, S., Agrawal, V., Best, G., Cao, C., Cujic, K., Darnley, R., DeBortoli, R., Dexheimer, E., Drozd, B., Garg, R., et al. (2021). Resilient and modular subterranean exploration with a team of roving and flying robots. *Journal of Field Robotics*.

Scherer, S., Ferguson, D., and Singh, S. (2009). Efficient C-space and cost function updates in 3D for unmanned aerial vehicles. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 2049–2054.

Segal, A., Haehnel, D., and Thrun, S. (2009). Generalized-ICP. In *Proc. Robotics: Science and Systems*, volume 2, page 435.

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.

Tranzatto, M., Mascarich, F., Bernreiter, L., Godinho, C., Camurri, M., Khattak, S., Dang, T., Reijgwart, V., Loeje, J., Wisth, D., et al. (2022). Cerberus: Autonomous legged and aerial robotic exploration in the tunnel and urban circuits of the DARPA subterranean challenge. *arXiv preprint arXiv:2201.07067*.

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.

Yang, F., Cao, C., Zhu, H., Oh, J., and Zhang, J. (2022). Far planner: Fast, attemptable route planner using dynamic visibility update. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, Kyoto, Japan.

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.

Zhao, S., Zhang, H., Wang, P., Nogueira, L., and Scherer, S. (2021). Super Odometry: IMU-centric LiDAR-visual-inertial estimator for challenging environments.

How to cite this article: Cao, C., Nogueira, L., Zhu, H., Keller, J., Best, G., Garg, R., Kohanbash, D., Maier, J., Zhao, S., Yang, F., Cujic, K., Darnley, R., DeBortoli, R., Drozd, B., Sun, P., Higgins, I., Willits, S., Armstrong, G., Zhang, J., Hollinger, G. A., Travers, M., & Scherer, S. (2023). Exploring the most sectors at the DARPA Subterranean Challenge finals. *Field Robotics, 3*, 801–836.