

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

Field Report

UAVs Beneath the Surface: Cooperative Autonomy for Subterranean Search and Rescue in DARPA SubT

Matěj Petrlík[✉], Pavel Petráček[✉], Vít Krátký[✉], Tomáš Musil[✉], Yurii Stasinchuk[✉], Matouš Vrba[✉], Tomáš Báča[✉], Daniel Heřt[✉], Martin Pecka[✉], Tomáš Svoboda[✉] and Martin Saska[✉]

Faculty of Electrical Engineering, Czech Technical University in Prague, Czech Republic

Abstract: This paper presents a novel approach for autonomous cooperating UAVs in search and rescue operations in subterranean domains with complex topology. The proposed system was ranked second in the Virtual Track of the DARPA SubT Finals as part of the team CTU-CRAS-NORLAB. In contrast to the winning solution that was developed specifically for the Virtual Track, the proposed solution also proved to be a robust system for deployment onboard physical UAVs flying in the extremely harsh and confined environment of the real-world competition. The proposed approach enables fully autonomous and decentralized deployment of a UAV team with seamless simulation-to-world transfer, and proves its advantage over less mobile UGV teams in the flyable space of diverse environments. The main contributions of the paper are present in the mapping and navigation pipelines. The mapping approach employs novel map representations—SphereMap for efficient risk-aware long-distance planning, FacetMap for surface coverage, and the compressed topological-volumetric LTVMap for allowing multirobot cooperation under low-bandwidth communication. These representations are used in navigation together with novel methods for visibility-constrained informed search in a general 3D environment with no assumptions about the environment structure, while balancing deep exploration with sensor-coverage exploitation. The proposed solution also includes a visual-perception pipeline for on-board detection and localization of objects of interest in four RGB stream at 5 Hz each without a dedicated GPU. Apart from participation in the DARPA SubT, the performance of the UAV system is supported by extensive experimental verification in diverse environments with both qualitative and quantitative evaluation.

Keywords: Unmanned Aerial Vehicles, Search and Rescue, DARPA, SubT, autonomy, exploration, navigation, deployment, subterranean environment, degraded sensing

Support materials

The paper is supported by the multimedia materials available at mrs.felk.cvut.cz/fr2022darpa. Open-source implementation of the core of the UAV system is available at github.com/ctu-mrs/

Received: 6 June 2022; revised: 7 October 2022; accepted: 29 November 2022; published: 2 January 2023.

Correspondence: Matěj Petrlík, Faculty of Electrical Engineering, Czech Technical University in Prague, Czech Republic, Email: matej.petrlik@fel.cvut.cz

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

Copyright © 2023 Petrlík, Petráček, Krátký, Musil, Stasinchuk, Vrba, Báča, Heřt, Pecka, Svoboda and Saska

DOI: <https://doi.org/10.55417/fr.2023001>

[mrs_uav_system](https://github.com/ctu-mrs/slam_datasets). The SLAM datasets are available at github.com/ctu-mrs/slam_datasets. The visual detection datasets are available at github.com/ctu-mrs/vision_datasets.

1. Introduction

The research of new robotic technologies and solutions is accelerating at an unprecedented rate mainly in case of aerial robotics. Technological development is improving many areas of our lives and, hopefully, even the future of humanity. The authors of (Shakhatreh et al., 2019) reviewed current research trends and future insights on potential Unmanned Aerial Vehicle (UAV) use for reducing risks and costs in civil infrastructure. The survey of UAV applications is accompanied by a discussion of arising research challenges and possible ways to approach them.

This paper focuses on a robotic system developed to autonomously search subterranean environments. The motivation behind searching subterranean environments is to gain situational awareness and assist specialized personnel in specific missions. Such missions may include: assessing the structural integrity of collapsed buildings, tunnels, or mines; exploration of a newly discovered branch in a cave network; or searching for lost persons. These tasks can often be life-threatening to human workers as many hazards are present in subterranean environments. In order to reach survivors quickly in unstable caves or partially collapsed burning buildings, first responders, such as emergency rescuers and firefighters, may potentially put their lives at risk. In firefighting tasks, fires can be either localized and reported to personnel by robots or the robots can even directly begin extinguishing flames if the presence of human firefighters is too risky (Spurny et al., 2021; Pritzl et al., 2021; Martinez-Rozas et al., 2022). In such scenarios, ceilings can suddenly collapse, toxic gas can appear in a mine, flames can extend to an escape corridor, or a cave cavity can flood with water. In distress situations, it is essential to swiftly coordinate the rescue operation as the survivors of a catastrophe might need acute medical assistance or have a limited amount of resources available, namely oxygen and water. However, without conducting a proper reconnaissance of the environment and assessing the potential risks prior to the rescue mission, the involved rescuers are exposed to a much higher probability of injury.

To reduce the possibility of bodily harm or to avoid risks altogether, a robotic system can be sent on-site before the rescuers in order to either quickly scout the environment and report any hazards detected by the onboard sensors, or directly search for the survivors. The rescue mission can be further sped up by deploying a team of robots capable of covering larger areas and offer redundancy in case of losses of some robot units in harsh environments. Multirobot teams can also consist of heterogeneous agents with unique locomotion modalities to ensure traversability of various terrains, including muddy ground, stairs, and windows, which is discussed in the overview of collaborative Search and Rescue (S&R) systems (Queralta et al., 2020). Similarly, sensing modalities can be distributed among individual robots to detect various signs of hazards, such as increased methane levels or the potential presence of survivors deduced from visual or audio cues. Mounting all sensors on a single platform would negatively affect its dimensions and, consequently, its terrain traversability as it may not be able to fit into narrow passages, such as crawlspace-sized tunnels or doorways. It would also mean a single point of failure for the rescue operation. On the other hand, the operation of a single robot can be managed by just one person, while commanding a robot team may be unfeasible for a single operator. Assigning each robot to an individual operator would also be an ineffective allocation of resources. Moreover, the range of the robot would be limited by the communication link to the operator. To provide a valuable tool for the rescue team, the robots must be able to move through the environment on their own and infer about the environment using their sensor data. The rescuer can then also act as an operator, providing only high-level inputs to the robotic system to bias their behavior based on *a priori* information (e.g., someone was last seen on the east side of the third floor). The research and development of such autonomous systems for assisting first responders is the primary focus of the S&R robotics, and also the motivation for the S&R UAV system presented in this paper.

The robotic platforms typically considered for S&R tasks are categorized into wheeled, tracked, legged, marine, and aerial platforms (Delmerico et al., 2019). Among these locomotive modalities,

aerial robots are considered to have the highest traversal capabilities since they can fly over most obstacles which are untraversable by other platforms. One example of an autonomous aerial research platform for S&R is found in (Tomic et al., 2012). The mobility of UAVs also surpasses other robot types thanks to its dynamic flight which can achieve large velocities and accelerations. These qualities make UAVs ideal for swift environmental scouting for gaining initial knowledge about a situation. As such, the aerial platform is predetermined to be deployed as the first robot during the first minutes of the rescue operation. A team deployed in an outdoor multi-UAV disaster response task (Alotaibi et al., 2019) can effectively cover a large search area and minimize the time to find and reach survivors. On the other hand, UAVs cannot operate for extended periods of time due to their limited flight time, and the sensory equipment is limited by the maximum payload of the UAV. Some sensing modalities might even be unsuitable for the use on aerial robots due to their propulsion system, e.g., detecting gas due to the aerodynamic effects of the propellers, or sound detection due to noisy operation. Due to the aforementioned pros and cons of UAV platforms, it is convenient to combine the capabilities of other robot types to form a heterogeneous robotic team.

This manuscript proposes an autonomous cooperative UAV approach for S&R. The approach used by Unmanned Ground Vehicles (UGVs) is not presented here because it is vastly different from the UAV system and as such would not fit into the scope of this article, which is already moderately extensive as we did not want to omit any details about the deployed system. The UGV solution was developed by our colleagues who are acknowledged at the end of this article. The proposed UAV together with legged, wheeled, and tracked UGVs formed the CTU-CRAS-NORLAB team, which participated in the Defense Advanced Research Projects Agency (DARPA) Subterranean Challenge (SubT). The team consisted of Czech Technical University in Prague (CTU) and Laval University.

1.1. DARPA SubT challenge

After major success in accelerating the development of self-driving cars in the Grand Challenges of 2004 and 2005 and the Urban Challenge in 2007, DARPA announced the Subterranean Challenge (SubT) (Orekhov and Chung, 2022) for the years 2017-2021 to advance the state of the art of S&R robotics. Participants had to develop robotic solutions for searching subterranean environments for specific objects that would yield points if reported with sufficient accuracy. To achieve the task at hand, the competitors had to develop complex multirobot systems spanning nearly all research areas of mobile robotics, from design of the robotic platforms to high-level mission planning and decision-making.

The rules of the competition can be summarized in a few points. Each team has a dedicated time slot, or *run*, to send their robots into a previously unvisited course and search for specific objects, referred to as artifacts (Figure 1). Each run starts at a predefined time and ends exactly one hour later. A single team is present on the course at a time during which they can deploy an unconstrained number of robots of arbitrary size. The movement of team personnel and their handling of robots is allowed only in the area in front of the entrance to the course, as shown in Figure 2. Only robots can enter the course and only one human operator/supervisor can command the robots and access



Figure 1. All 10 artifacts searched for in the Final Event of DARPA SubT (image courtesy of DARPA). The operator had to submit the position of the identified artifact with accuracy better than 5 m. While the first three artifacts (survivor, cellphone, and backpack) were present in all circuits, the drill and the fire extinguisher were tunnel-specific. Similarly, the gas and vent were located in the urban environment, and the helmet with rope could be found in the caves. The last artifact (the cube) was introduced only for the Final Event.

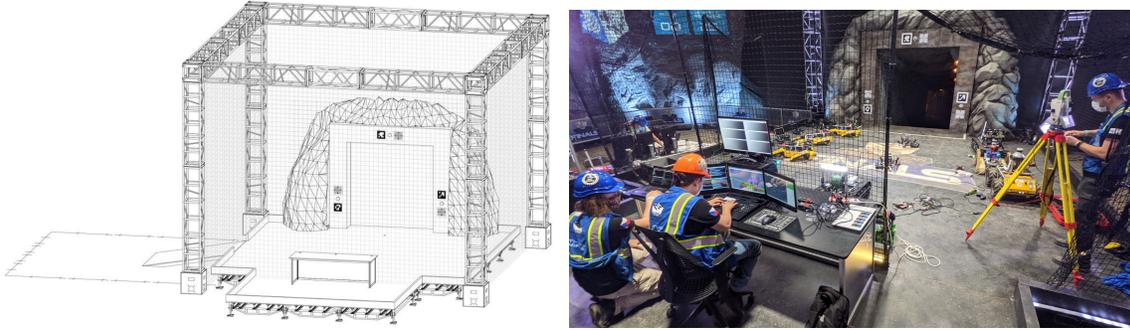


Figure 2. The bounded staging area (image courtesy of DARPA) is the only place where the human crew members can handle the robots. The person sitting behind the displays is the operator who is the only one allowed to issue commands to the team of robots, and also to view and interpret mission data.

Table 1. The prize money awarded for achieving the first three places in the Final Event.

Place	Systems Track	Virtual Track
1.	\$2M	\$750K
2.	\$1M	\$500K
3.	\$500K	\$250K

the data they acquire during the run. These conditions should mimic the conditions of a real S&R robotic mission. The operator can report the type and position of an artifact. If the type was correct and the reported position was not further than 5 m from the true position, the team was awarded one point. The team with the highest score wins the prize according to [Table 1](#). For a more detailed description of the challenge, see ([Orekhov and Chung, 2022](#)).

To encourage the development of high-level components without worrying about the resilience of the hardware in harsh subterranean conditions and also to enable teams without many resources and/or physical robots to compete, a virtual version (Virtual Track) of the competition was run in parallel to the physical Systems Track. The solutions of the Virtual Track were uploaded as Docker images (one image per robot) to the Gazebo-based Cloudsim simulation environment, where the entire run was simulated. Every team could use the Cloudsim simulator to test their approaches in practice worlds prior to the actual competition.

The competition was further subdivided into individual circuits, which were events in the specific subterranean environments of a tunnel, cave, and urban space. Examples of each environment are shown in [Figure 3](#). The surroundings were chosen to correlate with typical real S&R sites to assure the applicability of the systems developed during the competition. Every type of environment differs in size, geometric dimensions, traversability conditions, and requirements on perception modalities. The specifics of tunnel-like environments are summarized in ([Tardioli et al., 2019](#)) with 10 years of experience in S&R ground robots research. The role of mobile robots in rescue missions after mine disasters is discussed in ([Murphy et al., 2009](#)). The Final Event combined all of the previous environments for the ultimate challenge.

We participated in the competition first as a nonsponsored team. In the Tunnel Circuit, we won 1st place among the nonsponsored teams and 3rd place in total, which earned us \$200 000. The aerial robots explored 340 m of tunnels and found 3 artifacts out of the 10 artifacts discovered by all our robots ([Petrlík et al., 2020](#)). This success was repeated in the Urban Circuit with the same place achieved but this with time larger prize money \$500 000. The UAVs proved their suitability for quick scouting of the environment thanks to their advantage in mobility when the first deployed UAV managed to travel 93 m inside the building in just 200 s while it took about half an hour to reach the same area with the semi autonomously operated ground robots as reported in ([Kratky et al., 2021a](#)). One of the deployed UAVs also served as a retranslating station for other robots



Figure 3. Three types of subterranean environments found in the competition, each challenging for the robot team in a different way. From left to right: tunnel, urban, and cave. The top row shows examples of environments from the Systems Track of the Final Event, while the virtual worlds are pictured in the bottom row.

after navigating to and landing at a strategic position. Thanks to consistent performance in both circuits, DARPA awarded our team the funding for the Final Event, which allowed us to acquire more capable hardware. In the Virtual Track, the UAVs were used as the primary platform for finding artifacts thanks to their high travel speed and the ability to fly over terrain untraversable by UGVs. The ground robots supported longer flights of the UAVs by extending the communication network with breadcrumbs. In total, 215 artifacts were found by the UAVs in the competition worlds (8 artifacts less than the winner). The performance of UAVs in the confined environment of the Systems Track was worse than in the Virtual Track. Nevertheless, while the UGVs detected 5 out of 7 scored artifacts, the aerial robots managed to add 2 unique artifacts not seen by other robots but 1 of them had the wrong class and image and thus could not score. The last point was scored manually by the operator by matching a detection from a UGV that had inconsistent map with the same position in a correct map. The approach presented in this paper is the result of UAV research, development, and testing over the whole 3-year-long period.

2. Related work

The state of the art in rescue robotics is coherently summarized in the survey (Delmerico et al., 2019), which concerns both hardware and software. On the hardware side, different robot morphologies, locomotion types, and platform designs are categorized. Regarding software, the survey concerns perception and control algorithms. The authors interviewed experts on disaster response and humanitarian aid to understand the situation and needs of rescuers.

Here, we provide an overview of the solutions for perception in adverse conditions of the underground environments, methods of localization and mapping for precise and reliable navigation, and techniques for safe traversal of narrow corridors. A summary of systems deployed in previous circuits of DARPA SubT follows. Finally, relevant datasets are referenced in order to prompt further research effort in the S&R area.

2.1. Degraded sensing

Perception in subterranean environments faces constant degradation of the sensor outputs due to the harsh conditions of such places. The underground climate is often filled with impervious dust (particularly in mines), where any movement agitates the settled layer of fine dirt and mineral

particles. On the other hand, caves are typically humid ecosystems, where dense mud replaces the dust layer found in mines. However, the elevated humidity forms droplets of fog, which corrupt the measurements of most visible or Near Infrared (NIR) light-based sensor modalities, and also causes frequent reflections on wet surfaces. Radars can reliably penetrate smoke, dust, and fog, and after postprocessing using, e.g., Generative Adversarial Networks (GANs) (Goodfellow et al., 2014), a 2D occupancy grid for navigation (Lu et al., 2020) can be constructed. Another reliable sensing modality for when images from color (RGB) cameras are polluted by dust or fog is thermal imaging, which, in (Khattak et al., 2019), is used for the localization of robots in areas with airborne obscurants. Our approach goes beyond these works by employing intensity-based filtering of the Light Detection and Ranging (LiDAR) data, and thus no additional sensors are necessary even in dense clouds of dust.

2.2. Localization and mapping

Recent developments in S&R robotics sparked the research of more precise local pose estimation algorithms (also referred to as odometry), as well as long-term globally consistent trajectory and multirobot map fusion of all agents of the robotic team. The state-of-the-art methods were published in (Cadena et al., 2016), where the challenges and future direction of the Simultaneous Localization and Mapping (SLAM) development are also identified. The demands on low control error and robustness to degraded sensor data in the narrow subterranean environments present in the DARPA SubT pushed all contesting teams to either adapt and improve an existing method to be usable in the extreme conditions, or to develop a new SLAM tailored to this specific domain. SLAM methods used by the teams in the Final Event are summarized in (Ebadi et al., 2022) along with expert opinions about the present maturity and future outlook of the field.

Team CoSTAR developed a LiDAR odometry solution (Palieri et al., 2020) based on Generalized Iterative Closest Point (GICP) matching of LiDAR scans with initialization from Inertial Measurement Unit (IMU) and wheel odometry, including the possibility of extension to other odometry sources, such as Visual-Inertial Odometry (VIO). The method is shown to outperform state-of-the-art localization methods on the datasets from Tunnel and Urban circuits. An ablation study presents the influence of individual components on the total Absolute Position Error (APE). The second improved version (Reinke et al., 2022), which was released as open-source is less computationally demanding, less memory intensive, and more robust to sensor failures. All presented experiments are conducted with ground robots. The localization of aerial vehicles is handled by a resilient HeRo state estimation system (Santamaria-Navarro et al., 2019). The state estimation stack considers heterogeneity and redundancy in both sensing and state estimation algorithms in order to ensure safe operation, even under the failure of some modules. Failures are detected by performing confidence tests on both data and algorithm health. If a check does not pass successfully, the resiliency logic switches to the algorithm with the best confidence, similar to our previous solution published in (Baca et al., 2021). The local odometry of (Palieri et al., 2020; Santamaria-Navarro et al., 2019) is accompanied by loop closure detection and pose graph optimization locally on each robot, as well as globally on the base station. This optimizes the trajectories of all robots for a multirobot centralized SLAM solution (Ebadi et al., 2020). After improving the performance of the multirobot loop closure generation and pose estimation, especially in large-scale underground environments, the solution was open-sourced (Chang et al., 2022) and released together with a multirobot dataset from the subterranean environment. A technique for loop closure prioritization (Denniston et al., 2022) improves the Absolute Trajectory Error (ATE) of the multirobot SLAM by prioritizing loop closures based on observability, graph information, and Received Signal Strength Indicator (RSSI) criteria. A decentralized SLAM solution for UAVs (Lajoie et al., 2020) performs distributed outlier-resilient pose graph optimization when another agent is within communication range. This method can be used with either a stereo camera or a LiDAR, and is evaluated on a dataset from the Tunnel Circuit.

The long, featureless corridors that are often present in man-made tunnels lead to unobservability of the motion along the degenerate direction, which leads to significant drift. Promising approaches,

such as (Shan et al., 2020; Xu et al., 2022), constrain the solution of the optimization problem using the preintegrated IMU measurements. This helps to reduce the localization drift under unfavorable environmental geometry. Nevertheless, the vibrations induced by spinning propellers degrade the inertial measurements, and can thus negatively affect the localization precision. Approaches, such as those seen in (Ebadi et al., 2021), detect the geometrical degeneracy using the ratio of the most observable and the least observable directions. This ratio is then used to determine loop closure candidates to reduce the drift along the degenerate direction. Similarly, (Zhang et al., 2016) handles environment degeneracy in state estimation by not updating the solution in detected degenerate directions. Another possibility is to combine the 3D LiDAR method with a direct visual odometry method [e.g., (Alismail et al., 2016)], which tracks image patches by minimizing the photometric error. This approach, which is shown in (Shin et al., 2020), has the advantage over feature-based methods like that of (Zhang and Singh, 2015) in that it provides low drift, even when salient image and geometric features are lacking. The disadvantage is that localization performance is worsened when whirling dust is present in the camera image, as reported in (Petrlik et al., 2020).

Team CERBERUS developed a complementary multimodal sensor fusion (Khattak et al., 2020). The odometry estimated by visual/thermal inertial odometry is used as a prior for LiDAR scan-to-scan and scan-to-map matching. The VIO/TIO priors constrain the scan matching optimization problem, thus reducing drift in a degenerate environment significantly, which is demonstrated in an experiment conducted in a self-similar environment.

Another multimodal approach is the Super Odometry (Zhao et al., 2021) of team Explorer, which was deployed on aerial robots in the tunnel and urban circuits of DARPA SubT. The core of the method is the IMU odometry with biases constrained by VIO and LiDAR-Inertial Odometry (LIO), which are initialized with preintegrated inertial measurements of the constrained IMU. The relative pose factors of VIO and LIO are weighted based on the visual and geometrical degradation, respectively.

Team MARBLE first relied on visual SLAM (Kramer et al., 2021), but after Subterranean Integration Exercise (STIX), they transitioned to the LiDAR-based Cartographer (Hess et al., 2016) due to unstable tracking of motion under poor illumination, reflections, dust, and other visual degradation.

Wildcat SLAM (Hudson et al., 2022) of the CSIRO Data61 team is a multiagent decentralized solution, where each agent computes a global map using the currently available data shared among the robots. The odometry of each agent is based on the work of (Bosse et al., 2012).

Our approach is similar to the other teams' as we also use primarily LiDAR for localization and mapping. An improvement over the state of the art is the compensation of the delay (Pritzl et al., 2022a) caused by the LiDAR scan processing and the delay of the localization itself.

2.3. Mobility

Deploying aerial robots has one great advantage over ground robots due to their full terrain traversability. A UAV can fly over terrain that would compromise the safety of an UGV, e.g., steep decline, mud, water, etc. This allows to neglect the traversability problem necessarily tackled in solutions to UGV navigation (Fan et al., 2021), as the only movement constraint of aerial platforms flying through an enclosed environment is the minimum size of a passage that the robot can safely pass through. The dimensions of such passages depend largely on the size of the UAV, but also on the precision of the pose estimation, the control error of onboard regulators, the map quality, and the reactive behavior in close vicinity of obstacles. Some platforms also tolerate contact with obstacles in the sense that the contact does not endanger the continuation of the mission (Huang et al., 2019). Other types of platforms adapt their morphology and/or locomotion modality to their current surroundings and obstacles (Fabris et al., 2021). In voxel-based map representations, the size of a narrow passage is represented too conservatively, i.e., the size of the narrow passage in the voxel map is the lower bound of the true size. However, in practice, the narrow passage can be up to twice the map resolution larger than its voxel representation, which prevents traversing passages

that are well within the physical limits of the UAV. To better approximate the true shape of the narrow passage, (O’Meadhra et al., 2018) propose continuous representation based on Gaussian Mixture Models (GMM) (Reynolds, 2009), which is converted to a voxel map of arbitrary resolution when queried. However, the information about the exact structure of the environment is lost due to the approximation by Gaussian distribution. We took another approach of locally increasing the resolution of the occupancy voxel map when the size of the environment requires it, which preserves all details.

To fully exploit the capabilities of UAV’s full terrain traversability, the path planning and trajectory generation algorithms have to work with a full 3D representation of the environment and fulfil the real-time requirements. Although several excellent works on planning in constrained environments were recently published (Zhou et al., 2021a; Tordesillas et al., 2022), they focus primarily on generating fast trajectories while the presented application requires maximizing the reliability of the system in the presence of uncertainties imposed by a harsh dynamic environment. In addition, deploying the planning algorithm as part of the complex system running on board UAV with limited computational resources motivates the use of computationally undemanding algorithms. Similarly to team Explorer (Scherer et al., 2022), we make use of a multistage approach consisting of extensively validated computationally undemanding algorithms well-integrated into presented system (Baca et al., 2021; Kratky et al., 2021a).

2.4. DARPA SubT approaches

This paper primarily focuses on the approach developed for and experimentally verified in the Final Event of DARPA SubT. As mentioned, these results are built upon the experience in using the approaches developed for the tunnel and urban circuits. The practical verification of the developed solutions in challenging environments justifies the robustness of these algorithms. Valuable insights on the future of S&R robotics can be drawn from lessons learned by the teams.

Team CoSTAR relied on their uncertainty-aware framework, NeBula, in the tunnel and urban circuits (Agha et al., 2021). The framework supports multimodal perception and localization including radar, sonar, and thermal cameras. Aerial robots were part of their heterogeneous team in STIX and the tunnel circuit, mainly for exploring areas inaccessible to ground robots and data muling with distributed data sharing (Ginting et al., 2021; Saboia et al., 2022). A reactive autonomy approach COMPRA (Lindqvist et al., 2021) was also proposed for UAV underground S&R missions. Their solution gained 2nd and 1st place in the tunnel and urban circuits respectively.

Team Explorer developed a system (Scherer et al., 2022) that achieved 1st place in the tunnel circuit and 2nd place in the urban circuit. Their collision-tolerant platform “DS” with flight time of 13 min was carried on top of a UGV and could be launched by the operator when needed. The authors identified the challenge of combined exploration and coverage problem when their UAVs with limited camera Field Of View (FOV) missed some artifacts along their flight path. The frontier-based exploration pipeline used a custom OpenVDB mapping structure (Museth, 2013) for sampling frontier-clearing viewpoints. Paths to found viewpoints were planned using bidirectional RRT-Connect.

Team CERBERUS deployed legged ANYMAL robots and aerial DJI Matrice M100 robots in the tunnel circuit. Their graph-based system for the autonomous exploration of subterranean environments called GBPlanner was deployed in multiple locations. The exploration of Edgar mine during STIX and the National Institute for Occupational Safety & Health (NIOSH) mine during the tunnel circuit are documented in (Dang et al., 2020b). Specifically, the exploration method for aerial robots (Dang et al., 2019a) consists of a local fast-response layer for planning short collision-free paths and a global layer that steers the exploration towards unvisited parts of the map. This method is part of the solution for underground search by aerial robots found in (Dang et al., 2020a). A mapping and navigation approach (Papachristos et al., 2019a) for autonomous aerial robots based on the next-best-view planner (Papachristos et al., 2017; Bircher et al., 2016) was also proposed, but was later outperformed by the GBPlanner (Dang et al., 2020b). The uncertainty in localization

and mapping is taken into account during the planning in (Papachristos et al., 2019b) in such a way that among all trajectories arriving to the reference waypoint, the one that minimizes the expected localization and mapping uncertainty is selected. To unify the exploration framework across both legged and aerial platforms, (Kulkarni et al., 2021) have revised (Dang et al., 2020b) and added a cooperation framework that identifies global frontiers in a global graph built from the sub-maps of individual robots. The unified strategy for subterranean exploration using legged and aerial robots in tunnel and urban circuits is presented in (Tranzatto et al., 2022b). Team CERBERUS won in the Systems Track of the Final Event and (Tranzatto et al., 2022a) describes their approach that led to this success.

Team MARBLE presents their system deployed to STIX, the tunnel circuit, and the urban circuit in (Ohradzansky et al., 2021). The aerial robots relied on direct vision-based local reactive control and map-based global path planning. Global path planning is common with ground and aerial robots. Viewpoints are selected based on the frontier voxels covered by the camera FOV and the approximate travel time. In the tunnel circuit, the local reactive control generates velocity commands by steering the UAV towards a look-ahead point from the global path, while being repulsed by nearby obstacles. With this planner, traversing narrow passages was problematic due to noise in the depth image. Thus a new planner was developed for the urban circuit based on voxel-based probabilistic tracking of obstacles (Ahmad et al., 2021). In the Systems Track of the Final Event, team MARBLE gained 3rd place.

A heterogeneous team of robots including UAVs was also deployed by team CSIRO Data61 (Hudson et al., 2022), both in the tunnel and urban circuits. The aerial part of the team consisted of a DJI M210 equipped with the commercially available payload of Emesent Hovermap, and a custom gimbaled camera. To explore the environment of the urban circuit, the autonomy utilized an approach based on the direct point cloud visibility (Williams et al., 2020). Team CSIRO Data61 achieved 2nd place in the Systems Track of the Final Event.

Although team NCTU did not participate in the Final Event, their solution (Chen-Lung et al., 2022) to the tunnel and urban circuit showcased originality in the form of autonomous visually localized blimps (Huang et al., 2019). Their navigation was based on policies learned by deep reinforcement learning with simulation-to-world transfer.

Our CTU-CRAS-NORLAB team first participated in the STIX event with a hexarotor platform localized by optic flow (Walter et al., 2018) of the downward-facing camera. The reactive navigation used LiDAR scans to stay in the middle of the tunnel and move forward in a preferred direction at an intersection. The predictive controller (Baca et al., 2016) was forgiving to imprecise localization caused by strenuous optic flow estimation in the whirling dust of the tunnels. The heterogeneous team that secured 3rd place in the tunnel circuit (Roucek et al., 2019) consisted of wheeled, tracked, and aerial robots with different sensor payloads. Instead of unreliable optic flow, the localization of the UAV system (Petrlik et al., 2020) was revamped to rely on 2D LiDAR, HectorSLAM (Kohlbrecher et al., 2011), and state estimation (Petrlik et al., 2021). The hardware platform was also downscaled to a 450 mm diameter quadrotor. The vertical element of the urban circuit called for upgrading the LiDAR to a 3D one, which consequently required a redesign of the whole navigation pipeline (Kratky et al., 2021a) to allow for six Degrees of Freedom (DOF) mobility through the 3D environment. Physically, the platform was based on the same frame as what was used in the tunnel circuit, however prop guards were added to reduce the chance of destructive collision while flying through doors. The CTU-CRAS-NORLAB approach to the urban circuit, which we completed in 3rd place, is described in (Roucek et al., 2020). Although the cave circuit was canceled, extensive preparations were still performed in the sizable Bull Rock cave in South Moravia (Petracek et al., 2021). The exploration depth of the UAV team was greatly extended by a multirobot coordinated homing strategy that focused on extending the communication range of the base station by landing the returning UAVs on the edge of the signal. Based on the lessons learned during these competition and testing deployments (during the 3 years of development UAVs of the CTU-CRAS-NORLAB team achieved > 400 flights and traveled > 50 km in demanding real world environments) the new approaches presented in this paper were designed.

2.5. Datasets

Due to the challenging nature of the subterranean environments, such as narrow passages, degenerate geometry, and perception degradation, datasets that were collected by the competing teams are valuable to the community as the algorithms can be evaluated on demanding data degraded by the previously mentioned issues. In contrast to the verification often conducted under artificially ideal lab conditions, these datasets present a fair way to compare algorithms in realistic conditions. A SLAM dataset (Rogers et al., 2020a) collected during the tunnel circuit and STIX consists of LiDAR scans, images from a stereo camera and thermal camera, IMU measurements, and RSSI, together with a professionally surveyed ground truth map and measured artifact positions. The dataset from the urban circuit (Rogers et al., 2020b) was recorded using the same sensors with the exception of an added carbon dioxide (CO₂) sensor and the lack of a thermal camera. Data from sensors used for autonomous navigation including color-depth (RGBD) camera, event camera, thermal camera, 2D and 3D LiDARs, IMU, and Ultra-Wide Band (UWB) positioning systems were collected (Koval et al., 2022) by a mobile robotic platform moving through a subterranean environment. Another dataset (Kasper et al., 2019) for comparison of VIO methods contains outdoor, indoor, tunnel, and mine sequences, with ground truth poses obtained by laser tracking the sensors rig. Aerial datasets consisting of unsynchronized LiDAR scans and IMU measurements from UAVs flying in the cave, tunnel, and mine environments are included in this paper,¹ with ground truth poses estimated using a professionally surveyed ground truth map. We also publish the labeled visual detection datasets² consisting of images from both UAV and UGV cameras that were used for training of the artifact detection Convolutional Neural Network (CNN). Images from the Tunnel and Urban circuits, Bull Rock Cave, and industrial buildings are included.

3. Contributions

An approach for cooperative exploration of demanding subterranean environments by a team of fully autonomous UAVs in S&R tasks is presented in this paper. Deployment of this approach in the DARPA SubT virtual competition was awarded by 2nd place. The simulation model of the UAV platform designed by our team was used by seven out of nine teams. The crucial contributions of the developed system can be summarized in the following list:

- **A complex approach that can serve as a guide for building a system for Global Navigation Satellite System (GNSS)-denied operations.** The proposed approach was extensively verified in numerous simulated worlds and real physical environments ranging from vast caves, industrial buildings, tunnels, and mines to large outdoor openings. Most importantly, the UAVs were deployed into the intentionally harsh conditions of the DARPA SubT to push them to their limits. The experience gained from hundreds of flights in such conditions are condensed into the lessons learned presented in this paper, which we deem valuable for the field robotics community.
- **Novel mapping structures** are proposed for safety-aware reactive planning over large distances, for compact volumetric inter-robot information sharing, for storing coverage of surfaces by onboard sensors, and for finding a suitable landing spot.
- **Maximization of the probability of detecting a nearby artifact** by searching not only the unexplored space, but also visually covering known surfaces while respecting the limited field of view of the onboard sensors. The detection is coupled with probabilistic estimation of artifact positions based on multitarget tracking and detection-to-hypothesis association, which improves the precision of artifact localization while the robot is moving around the artifact.
- **A novel safety-aware approach to planning** that considers the risk of planned trajectories in addition to the path length in the optimized cost function. In contrast to the state-of-the-art

¹ github.com/ctu-mrs/slam_datasets

² github.com/ctu-mrs/vision_datasets

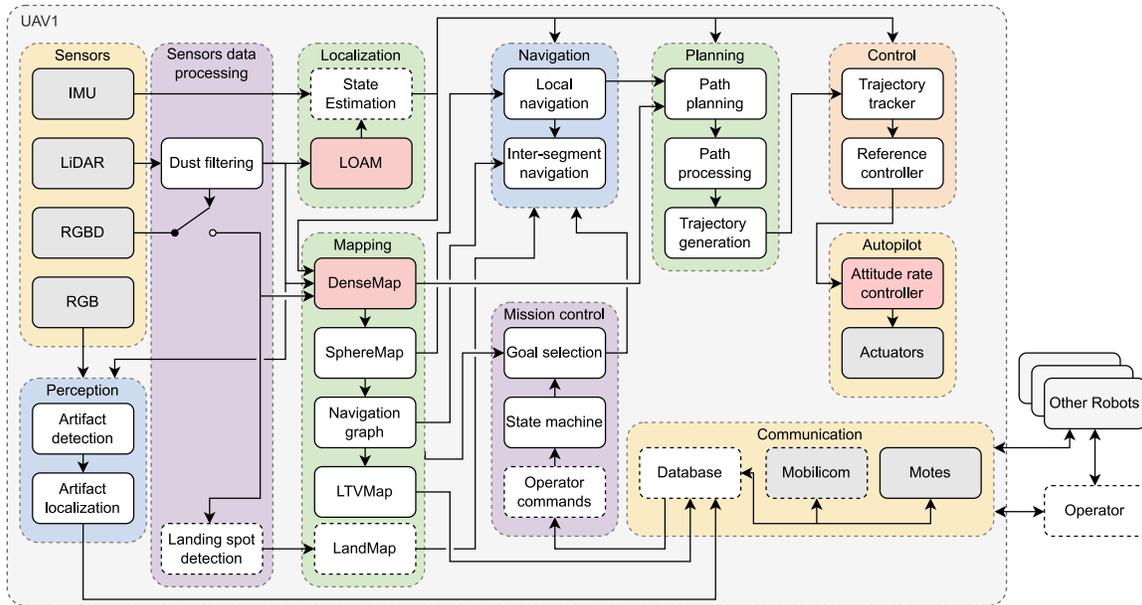


Figure 4. The diagram shows individual modules of the UAV system architecture (as deployed on the hardware platform) grouped into logical categories. Hardware modules are filled with gray, and red distinguishes open source modules not developed by us. The modules marked by dashed lines were used only in Systems Track but not in Virtual Track.

methods, longer paths are selected if the estimated risk of collision is lower than the risk of a shorter path.

- **Full autonomy of the UAV** allows for scalability of the size of the deployed fleet without placing additional workload on the operator. Nevertheless, the operator can override the autonomy with one of the available special commands to change the default search behavior when the UAV is in communication range.
- **The multirobot autonomous search** benefits from a higher number of deployed UAVs that share their topological representations of the environment to cooperatively cover a larger area by biasing the search towards parts unvisited by other agents.

4. System architecture overview

The whole autonomous system of a single UAV consists of software modules, each with different inputs, outputs, and purpose. These modules and their interconnections are depicted in [Figure 4](#) with the individual modules grouped into more general logical categories. The first category includes the physical *Sensors* ([Section 5](#)) of the UAV—the IMU, LiDAR, RGB, and RGBD cameras. The description of the important parameters of the used sensors is available in [Section 12](#). Measurements from IMU and LiDAR enter the *Localization* group ([Section 6](#)), where a full-state estimate of the UAV is obtained. LiDAR is also used in combination with the RGBD camera for building maps in the *Mapping* module group ([Section 7](#)). The *Perception* ([Section 10](#)) category focuses on detection and localization of artifacts using all the available sensor data.

Autonomous search through the environment is governed by the *Mission control* category ([Section 11](#)), which selects goals ([Section 8](#)) based on the current state of the state machine, models of the environment from the *Mapping* group, and possibly also commands from the operator. A coarse path consisting of waypoints to the selected goals is found by the *Navigation* ([Section 9.1](#)) and further refined and time-parametrized in the *Planning* modules ([Section 9](#)) in order to produce a safe and dynamically feasible trajectory. The *Control* blocks ([Baca et al., 2021](#)) track the

trajectory and generate attitude rate references for the low-level *Autopilot* that controls the actuators (Section 12).

The operator receives crucial mission status data, topological maps, and, most importantly, detected artifacts through the *Communication* layer (Roucek et al., 2020). This also allows the operator to influence or override the autonomous behavior of the UAV. All transmitted data are received by other UAVs (or other robots, in the case of a heterogeneous team) in the communication range, which serves two purposes: one, the receiving agent can propagate the message further down the network, and, two, the topological maps allow penalizing goals already visited by other robots to better allocate resources over a large area.

5. Spatial perception

The equipment on board UAV platforms within our research group is modular and replaceable to support a wide spectrum of research areas (Hert et al., 2022). In the proposed system for agile subterranean navigation, however, the aerial platform is fixed to ease fine-tuning of the on-board-running algorithms. From the point of perception, it relies heavily on 3D LiDAR from Ouster (SLAM, dense mapping, and artifact localization), and utilizes vertically oriented RGBD cameras for filling space out of FOV of the primary LiDAR sensor, and uses two RGB Basler cameras for artifact detection, supported by powerful LEDs illuminating the scene. The flow of sensory data within the entire system are shown directly in Figure 4.

5.1. Sensors calibration

The intrinsics of LiDAR sensor and RGBD cameras are factory-calibrated whilst monocular RGB cameras are calibrated with standard OpenCV calibration tools, assuming the pinhole camera model. The extrinsics of the sensors (cameras-to-LiDAR and LiDAR to the flight control unit) are given by the CAD model of the robot. To mitigate the effects of inaccuracies in 3D printing, modeling, and assembly, all the camera-to-LiDAR extrinsics are fine-calibrated using a checkerboard camera calibration pattern with known dimensions. The fine-calibration pipeline detects the pattern in both modalities (LiDAR data and RGB image), finds mutual correspondences, and estimates the extrinsics by defining the problem as perspective-n-point optimization minimizing the reprojection error of the mutual correspondences with Levenberg-Marquardt method.

5.2. Filtering observation noise

The aerodynamic influence of a multirotor UAV on the environment is not negligible, particularly in confined settings. The fast-rotating propellers generate airflow lifting up light particles of dust and whirling them up in clouds. In environments where the clouds are not blown away but are rather rebounded back to the UAV, the effect on sensory performance might be crippling. To minimize deterioration in perception and its dependent systems (e.g., mapping, localization), the incident noise is filtered out from local LiDAR data.

The idea of robust filtering of dust is based on the method presented in (Kratky et al., 2021a) in which LiDAR data are sorted by the intensity field (measured intensity of the reflected light for a given point) and 10 % of the lowest-intensity data in a local radius from the sensor are removed. In contrast to the baseline method, simpler thresholding is adopted such that a subset $\mathcal{P}_F \subseteq \mathcal{P}$ of LiDAR data \mathcal{P} is preserved. The absence of data sorting lowers the computational load and reduces delay in data processing. The set is given as $\mathcal{P}_F = \mathcal{P}_D \cup \mathcal{P}_I$, where

$$\mathcal{P}_D = \{\mathbf{p} \mid \|\mathbf{p}\| \geq \kappa, \mathbf{p} \in \mathcal{P}\}, \quad (1)$$

$$\mathcal{P}_I = \{\mathbf{p} \mid \mathcal{I}(\mathbf{p}) > \Upsilon, \mathbf{p} \in \mathcal{P} \setminus \mathcal{P}_D\}. \quad (2)$$

$\mathcal{I}(\mathbf{p})$ (W m^{-2}) is the intensity of the reflected light from a point \mathbf{p} , κ (m) is a local radius of a filtering sphere with LiDAR data origin at its center, and Υ (W m^{-2}) is the minimal intensity of preserved

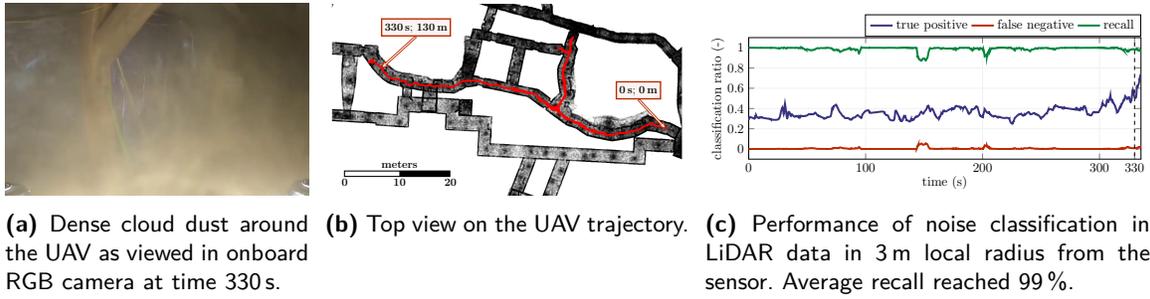


Figure 5. LiDAR-data noise filtration running onboard a UAV during a 154 m flight in the mine part (the dustiest part) of the DARPA SubT finals environment. The true positive classification in (c) denotes the ratio of correctly classified noise whereas the false negative represents the ratio of noise preserved after the filtration process (i.e., the unfiltered noise) to the size of the point cloud. The data for the classification analysis (c) were obtained by spatially comparing the sensor measurements with the map of the environment provided by the organizers.

data points. With n data points within a radius κ , the computational complexity is reduced to $\mathcal{O}(n)$ from baseline $\mathcal{O}(n \log(n))$. Although to achieve optimal performance the method requires calibration to given environmental conditions, a set of reasonable parameters ($\kappa = 5$ m and $\Upsilon = 30$ W m⁻² throughout many of our real-world deployments in the harshest dust conditions) suffices in the majority of applications. The performance of the dust filtering is analyzed in Figure 5 on an example UAV flight in the mine part (the dustiest zone) of the DARPA SubT finals environment.

The above method is utilizable only for sensory data containing information about the intensity of the reflected light. The rest of the sensors (RGBD cameras) are not processed, but their fusion and utilization are controlled by the amount of filtered noise in the primary LiDAR. Having the cardinality of the point sets defined in (2), the estimated amount of noise can be represented as

$$r_d = 1 - \frac{|\mathcal{P}_I|}{|\mathcal{P} \setminus \mathcal{P}_D|}, \quad (3)$$

where $r_d \in \langle 0, 1 \rangle$ is the ratio of the filtered-out observations to all the observations within the local radius κ . The RGBD cameras are then classified as unreliable (and not used in mapping or for detecting landing feasibility, as marked in Figure 4) if

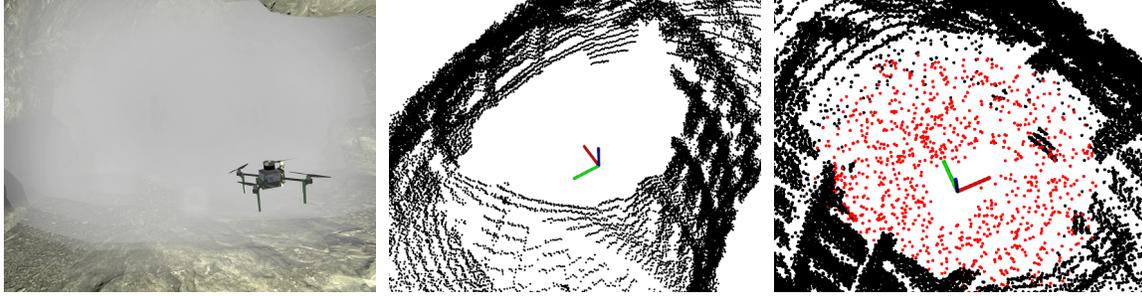
$$r_d > \lambda_d, \quad (4)$$

where $\lambda_d \in \langle 0, 1 \rangle$ is a unitless user-specified threshold. The lower value of λ_d the less amount of noise is integrated into mapping, while greater λ_d lets the connected modules handle the noise themselves. We empirically set the threshold high to $\lambda_d = 0.4$ in our final setup, since our probabilistic mapping pipeline is quite robust to the stochastic noise.

5.2.1. Detecting artificial fog in the virtual environment

The virtual competition contained a fog emitter plugin (see Figure 6) to mimic environmental perception degradation arising from observing smoke, dust, and fog. The plugin spawned a fog cloud when a robot reached the proximity of the emitter. Although our localization pipeline was able to cope with local noise, the inability to filter out the fog particles in a robust way led to a degradation of the local DenseMap, and consequently to blocking local planning which respects strict requirements on collision-free path planning. Thus, in our setup for the virtual challenge, the navigation stack did not try to enter through the fog areas but detected them, maneuvered out of them, and blocked the areas for global planning.

To detect the presence of the UAV within such a fog cloud, a discretized occupancy voxel grid is built from a set of data within a local radius (example data within a radius are shown in Figure 6c). Within this radius is compared the occupancy ratio r (number of occupied voxels to all voxels in the local grid) with maximum occupancy R given by the field of view of the sensor producing the



(a) Visualization of virtual fog in Ignition Gazebo. (b) Example 3D LiDAR data outside fog. (c) Example 3D LiDAR data inside fog (fog colored locally in red).

Figure 6. Simulated fog and its effect on sensory perception in the virtual environment. A fog cloud (a) spawns when a robot reaches its proximity. The cloud then affects the sensory inputs such that a uniform-distribution noise emerges in LiDAR data corresponding to the fog (c).

data. For each LiDAR or depth sensor, the sensor is classified as being in fog if

$$r_f > \lambda_f R, \quad (5)$$

where $\lambda_f \in \langle 0, 1 \rangle$ is a unitless multiplier converting $\lambda_f R$ to a maximal occupancy ratio threshold. The multiplier was set empirically to $\lambda_f = 0.7$ in our final setup.

For depth cameras that are not used for self-localization of the UAV, the in-fog classification solely controls whether the depth data are integrated within the mapping pipeline. However, if a localization-crucial 3D LiDAR is classified to be in fog, a backtracking behavior is triggered within the mission supervisor (see Section 11). The primary purpose of the backtracking is to prevent being stuck in fog and thus the UAV is blindly navigated out of the fog through the recent history of collision-free poses, ignoring occupied cells in the DenseMap (including possible noise from fog measurements). Lastly, detection of fog in a 3D LiDAR blocks the area in global planning.

5.3. Detecting spots safe for landing

For purposes of artifact detection and spatial mapping, the UAV carries a downward-facing RGBD camera, shown in Figure 7a. Apart from mapping the space below the UAV, the depth data of this camera are used in locating areas safe for landing throughout the UAV flight. If the sensor is marked as reliable according to (4), its depth-data frames are continuously fitted with a plane model whose coefficients are used in the binary classification of safe or unsafe landability respecting the plane-fit quality and deviation of its normal vector from the gravitational vector. The process of deciding on safe landability given a single depth-data frame is visualized in Figure 7 and described in Algorithm 1. The classification assumes that the data frame can be transformed into a gravity-aligned world coordinate frame. Inputs to Algorithm 1 are the square size of safe landing spots s (m), the minimal ratio of inliers in RANSAC plane fitting $I_{\min} \in \langle 0, 1 \rangle$, and the minimal z -axis component of unit plane-normal vector $N_{\min}^z \in \langle 0, 1 \rangle$. The square size s specifies the width of an area on which a UAV can land safely. Selection of s constrains the minimal height above the ground in which a safe landing spot detection may occur. Having a sensor with the minimal field of view θ_{\min} observing an even planar surface, the condition on line 12 in Algorithm 1 will be true for distance of the camera from the surface $d > d_{\min}$, where

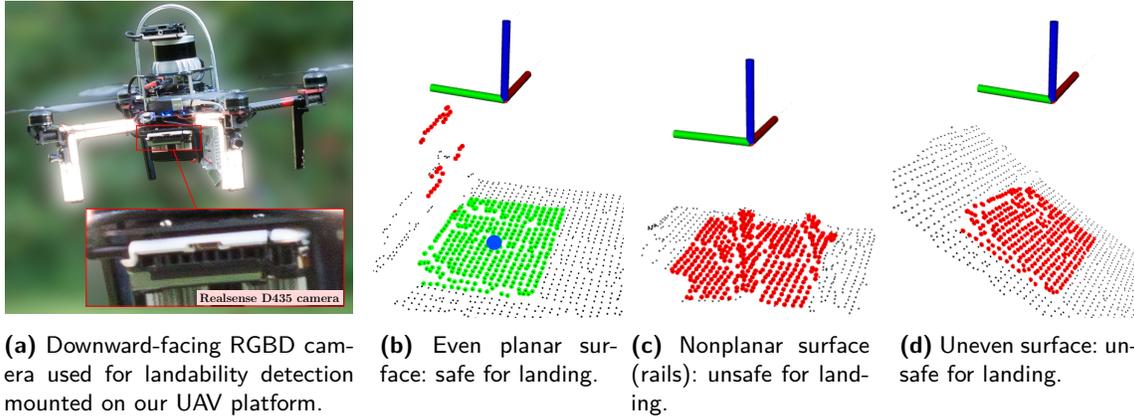
$$d_{\min} = \frac{s}{2 \tan\left(\frac{\theta_{\min}}{2}\right)}. \quad (6)$$

The maximal distance d_{\max} is then given by the maximal range of the sensor. In our setup, we utilized Realsense D435 camera with $\theta_{\min} = 58^\circ$ and set $s = 1.2$ m to be marginally greater than the dimensions of our UAV platform (approximately 0.8 m wide). According to (6), the given parameters

Algorithm 1. Detecting spots safe for UAV landing in downward-facing RGBD camera.

```

1: Input:
2:  $\mathcal{D}$  ▷ Depth-data frame in sensor coordinate frame
3: Output:
4:  $\mathcal{L}$  ▷ Binary classification for landing: {SAFE, UNSAFE}
5:  $\mathbf{p}_W$  ▷ Position of landing area in the world coordinate frame
6: Parameters:
7:  $s$  ▷ Square-size of safe landing spot in meters
8:  $I_{\min}$  ▷ Minimal ratio of inliers in plane fitting
9:  $N_{\min}^z$  ▷ Minimal z-axis component of the normalized plane-normal vector
10: Begin:
11:  $S := \text{cropFrameAtCenter}(\mathcal{D}, s)$  ▷ Crop frame-centered square with size  $s$ 
12: if  $\text{height}(S) < s$  or  $\text{width}(S) < s$  then
13:     return:  $\{\mathcal{L} = \text{UNSAFE}, \mathbf{p}_W = \text{N/A}\}$  ▷ Not safe to land: too close to the ground to decide
14:  $\mathcal{P} := \text{fitPlaneWithRANSAC}(S)$  ▷ Fit data with plane using RANSAC
15: if  $\text{inliers}(\mathcal{P}) / \text{count}(S) < I_{\min}$  then
16:     return:  $\{\mathcal{L} = \text{UNSAFE}, \mathbf{p}_W = \text{N/A}\}$  ▷ Not safe to land: data are not planar
17:  $\mathcal{P}_W := \text{transformToWorldFrame}(\mathcal{P})$  ▷ Transform plane to gravity-aligned frame
18: if  $|\text{normal}(\mathcal{P}_W).z| < N_{\min}^z$  then
19:     return:  $\{\mathcal{L} = \text{UNSAFE}, \mathbf{p}_W = \text{N/A}\}$  ▷ Not safe to land: ground is too steep for landing
20:  $S_W := \text{transformToWorldFrame}(S)$ 
21:  $\mathbf{p}_W := \text{centroid}(S_W)$  ▷ Express landing spot as the centroid of the depth data in the world
22: return:  $\{\mathcal{L} = \text{SAFE}, \mathbf{p}_W\}$ 
    
```



(a) Downward-facing RGBD camera used for landability detection mounted on our UAV platform. (b) Even planar surface: safe for landing. (c) Nonplanar surface (rails): unsafe for landing. (d) Uneven surface: unsafe for landing.

Figure 7. Deciding on landability of a UAV from downward-facing depth data—binary classification to safe (b) and unsafe [(c) and (d)] landing areas. In [(b)–(d)], the UAV is represented by Cartesian axes whereas the depth data are colored in black. The blue sphere in the safe classification (b) denotes the centroid of the plane inliers (colored in green) passed as a feasible landing position to LandMap (see Section 7.5).

yield the minimal distance of the sensor from the ground in detecting the landability to $d_{\min} = 1.08$ m, with $d_{\max} = 10$ m specified by the manufacturer. If the input data frame \mathcal{D} contain noise with nonplanar distribution, the condition on line 15 will classify the data as unsafe. The plane-fit and landability classification parameters $I_{\min} = 0.9$ and $N_{\min}^z = 0.7$ were found empirically for the given sensory setup. Positions classified as safe for landing on line 21 are passed to LandMap described in Section 7.5.

6. Localization

Accurate and reliable localization is critical for most other parts of the system. The ability of the reference controller to track the desired state depends largely on the quality of the available

Table 2. Approximate distribution of the environment cross-section as announced by the organizers before the Final Event.

Cross-section (m ²)	Distribution
<5	65%
5-100	20%
>100	15%

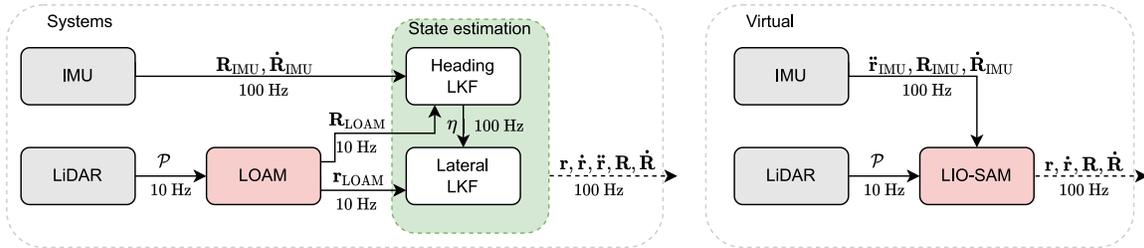


Figure 8. The diagram shows the flow of data among individual localization modules for the Systems Track (left) and Virtual Track (right). The 3D LiDAR supplies A-LOAM or LIO-SAM with the laser scans in the point cloud form \mathcal{P} . Assisted by the orientation \mathbf{R} from the IMU, A-LOAM produces a position estimate $\mathbf{r} = [x, y, z]^T$ that is fed into the *State estimation* block, which outputs the full state estimate. In the case of the virtual pipeline, the IMU data fusion is executed in LIO-SAM, and thus the state estimation module is not needed thanks to the sufficient accuracy of both lateral and heading components.

state estimate. In the narrow environments which are often present in subterranean environments (see Table 2 for cross-section distribution in the Final Event), minimizing the control error is crucial to avoid collisions. Multirobot cooperation assumes the consistency of maps created by individual robots. If the maps of two robots are not consistent due to errors in localization, the multirobot search might be suboptimal. For example, an unvisited goal can be rejected as already reached by a robot with an inconsistent map. Moreover, the localization accuracy influences the position error of a reported artifact. A UAV with localization drift over 5 m can detect and perfectly estimate the position of an artifact. Nevertheless, the report may never score a point since the position of the UAV itself is incorrect.

Our approach relies on a LiDAR sensor for localization as the laser technology proved to be more robust to the harsh conditions of the subterranean environment than the vision-based methods. We have been using LiDAR since the Tunnel circuit (Petrlík et al., 2020) where a lightweight 2D LiDAR aided by a rangefinder for measuring above ground level (AGL) height was sufficient for navigation in tunnels with a rectangular cross-section. The more vertical environment of the urban circuit required redesigning the localization system to use 3D LiDAR for navigating in 3D space (Kratky et al., 2021a).

The localization system deployed in the Final Event and presented in this manuscript builds upon the solution proposed in (Kratky et al., 2021a) and is divided into two modules: the localization algorithm and the state estimation method. Figure 8 shows the data flow in the localization pipeline. We have based the localization on the A-LOAM implementation of the LiDAR Odometry and Mapping (LOAM) algorithm (Zhang and Singh, 2014) for the Systems Track and the LiDAR Inertial Odometry via Smoothing and Mapping (LIO-SAM) (Shan et al., 2020) for the Virtual Track. Our implementation³ has been tested in a real-time UAV control pipeline throughout multiple experimental deployments as part of our preliminary works (Kratky et al., 2021a; Petracek et al., 2021) and in the DARPA SubT competition.

³ github.com/ctu-mrs/aloam

6.1. A-LOAM

The A-LOAM implementation of the LOAM (Zhang and Singh, 2014) algorithm utilizes the laser scans from a multiline LiDAR to obtain its 6-DOF pose. To achieve real-time performance and accurate pose estimation at the same time, the method is divided into two parts.

The first part of the algorithm processes the incoming data at the rate of their arrival and estimates the rigid motion between the consecutive point clouds \mathcal{P}_k and \mathcal{P}_{k+1} obtained at the timestamps t_k and t_{k+1} , respectively. The process starts with finding geometric features in the input point cloud \mathcal{P}_{k+1} . The points are first sorted by the smoothness of their local neighborhood, and then those which are the least and most smooth are selected as edge and planar features, respectively. To achieve a more uniform distribution of features, the point cloud is divided into regions of the same size, and each region can contain only a limited number of edge and planar feature points. A point cannot be chosen as a feature point if there is already a feature point in its local neighborhood. A correspondence is found in \mathcal{P}_k for each edge/planar point from \mathcal{P}_{k+1} . These correspondences are then weighted by their inverse distance, and correspondences with the distance larger than a threshold are discarded as outliers. Finally, the pose transform \mathbf{T}_{k+1}^L between \mathcal{P}_{k+1} and \mathcal{P}_k is found by applying the Levenberg-Marquardt method to align the correspondences.

The second part estimates the pose of the sensor in the map \mathcal{M}_k , which is continuously built from the feature points found by the first part of the algorithm. First, \mathcal{P}_{k+1} is projected into the map coordinate system to obtain \mathcal{P}_{k+1}^W . Then, feature points are searched similarly to as is done in the first part, with the difference being that 10 times more features are found. Their correspondences are found in \mathcal{M}_k , which is divided into cubes with 10 m edges. The correspondences are searched for only in the cubes intersected by the \mathcal{P}_{k+1}^W to keep the run-time bounded. The transform \mathbf{T}_{k+1}^W between \mathcal{P}_{k+1}^W and \mathcal{M}_k is obtained with the same steps as in the first part. Due to the 10-times greater amount of correspondences and search through a potentially larger map, this is a much slower process than the first part.

Thanks to the combination of both parts, the algorithm outputs the pose estimate of the rate of the LiDAR, with drift bounded by slower corrections that snap the pose to the map.

6.2. State estimation

For precise and collision-free navigation through a cluttered narrow environment, which typically appears in subterranean S&R scenarios, the control stack requires a smooth and accurate state estimate at a high rate (100 Hz). The *State estimation* module provides such an estimate through the fusion of data from Advanced implementation of LOAM (A-LOAM) and IMU. It also does this by applying filtering, rejection, and prediction techniques. We provide only a brief description of the estimation process as it is not viewed as the primary contribution and has already been presented in (Baca et al., 2021).

The state vector of the UAV is defined as $\mathbf{x} = [\mathbf{r}, \dot{\mathbf{r}}, \ddot{\mathbf{r}}, \mathbf{R}, \dot{\mathbf{R}}]^T$. The position $\mathbf{r} = [x, y, z]^T$, its first two derivatives of $\dot{\mathbf{r}}$ and $\ddot{\mathbf{r}}$, the orientation in the world frame \mathbf{R} , and the angular velocities $\dot{\mathbf{R}}$ include all the dynamics required by other onboard algorithms. Even though the position \mathbf{r} is provided by the A-LOAM algorithm, the rate of the position updates is too low for the control loop. Furthermore, the velocity and acceleration vector is not known, and must thus be estimated. A Linear Kalman Filter (LKF) of a point mass model with position, velocity, and acceleration states is employed to estimate the unknown variables at the desired rate.

While the IMU of the onboard autopilot provides the orientation \mathbf{R} , the heading⁴ η is prone to drift due to the bias of the gyroscopes in Micro-Electromechanical Systems (MEMS) IMUs. We correct this drift in a standalone heading filter, which fuses $\dot{\mathbf{R}}$ gyro measurements with A-LOAM

⁴Heading is the angle between the heading vector and the first world axis. The heading vector is the direction of the forward-facing body-fixed axis projected onto the plane formed by the horizontal axes of the world frame, as formally defined in (Baca et al., 2021).

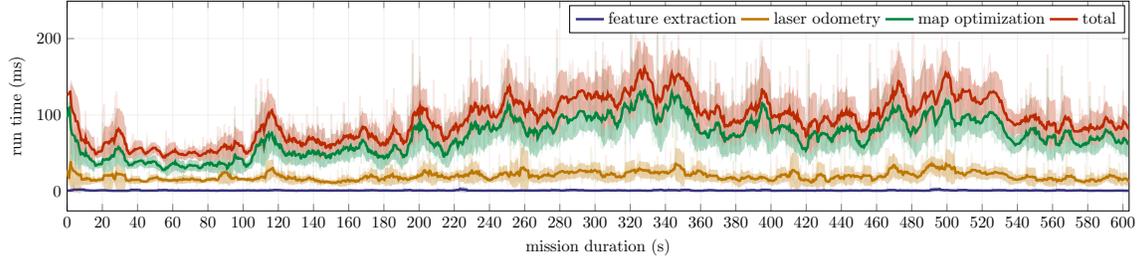


Figure 9. The computation time of the most demanding parts of the A-LOAM algorithm is plotted with respect to the time in the mission that was conducted in simulation. The total time is the sum of all three parts. The darkest colors depict moving mean, the medium dark bands represent the moving standard deviation, and raw data are shown by the lightest colors. The moving statistics are calculated over 1 s long time window. On average, the feature extraction takes 1 ms, the laser odometry 19 ms, the map optimization 91 ms, and, in total, the pose estimate is obtained in 111 ms.

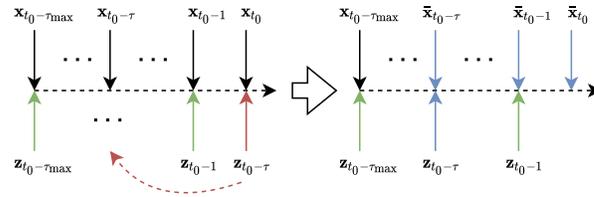


Figure 10. The left time sequence shows the situation in the filter after the arrival of delayed correction $\mathbf{z}_{t_0-\tau}$ at time t_0 . The green arrows represent corrections applied at the correct time. The delayed $\mathbf{z}_{t_0-\tau}$ would be fused at t_0 in a traditional filter, resulting in a suboptimal state estimate. However, thanks to the buffering of state and correction history, it is fused into the correct state at time $t_0 - \tau$. The states after $t_0 - \tau$ had to be recalculated to reflect the correction $\mathbf{z}_{t_0-\tau}$, which is shown by the blue color in the right time sequence.

η corrections. Corrections from the magnetometer are not considered, due to the often-occurring ferromagnetic materials and compounds in subterranean environments.

The processing of a large quantity of points from each scan and matching them into the map takes 111 ms on average (see Figure 9 for run time analysis) for the onboard Central Processing Unit (CPU). The empirical evaluation shows that the controller of the UAV becomes increasingly less stable when the state estimate is delayed for more than 300 ms. To reduce the negative effect of the delay on the control performance, we employ the time-varying delay compensation technique (Pritzl et al., 2022a). We define the delay as $\tau = t_{\mathbf{T}_{k+1}} - t_{\mathcal{P}_{k+1}}$, i.e., the time it took LOAM to compute the pose transform after receiving the point cloud from LiDAR. The core of the method is a buffer \mathbf{Q}_x containing the past states $\mathbf{x}_{(t_0-\tau_{\max}, t_0)}$, and buffer \mathbf{Q}_z having the past corrections $\mathbf{z}_{(t_0-\tau_{\max}, t_0)}$ of the filter. The length of the buffer is not fixed, but data older than the expected maximum delay τ_{\max} are discarded to keep the buffer size bounded. When a new delayed measurement $\mathbf{z}_{t_0-\tau}$ arrives at time t_0 , it is applied as a correction to the state $\mathbf{x}_{t_0-\tau}$ in \mathbf{Q}_x . The corrected state $\bar{\mathbf{x}}_{t_0-\tau}$ replaces $\mathbf{x}_{t_0-\tau}$. All subsequent states $\mathbf{x}_{(t_0-\tau, t_0)}$ are discarded from \mathbf{Q}_x , and replaced by the states $\bar{\mathbf{x}}_{(t_0-\tau, t_0)}$ propagated from $\bar{\mathbf{x}}_{t_0-\tau}$, using regular prediction steps of the filter with all corrections from \mathbf{Q}_z . Figure 10 visualizes the sequence of performed actions. Thus we acquire a time-delay compensated state estimate which, when used in the feedback loop of the UAV controller, allows for stable flight with a delay of up to 1 s. The effect that increasing the delay has on the control error is plotted in Figure 11.

6.3. LIO-SAM

LIO-SAM (Shan et al., 2020), used in the Virtual Track approach, utilizes IMU integration on top of dual factor-graph optimization. The first factor-graph optimization is similar to the A-LOAM

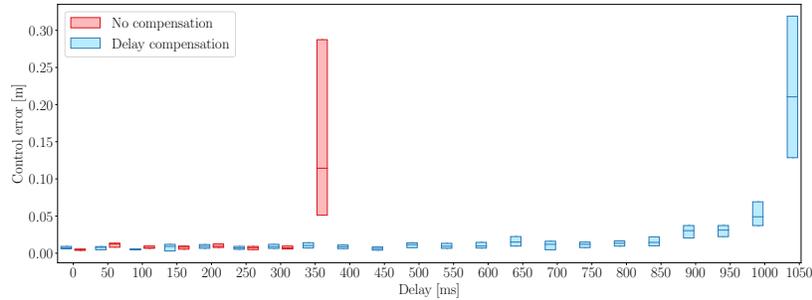


Figure 11. The box plot shows the median with lower and upper quartiles of the control error with respect to the delay of the position estimate used in the feedback loop. The data were obtained in simulation by artificially increasing the delay of ground truth position in 50 ms increments. Without compensation, the system becomes unstable after exceeding 300 ms delay, which results in oscillation-induced control error at 350 ms. The control error for the longer delay is not shown, because the high amplitude of oscillations led to a collision of the UAV. The highest delay with compensation is 1000 ms when the system has over a 5 cm control error, but is still stable. The UAV stability is lost at 1050 ms delay.

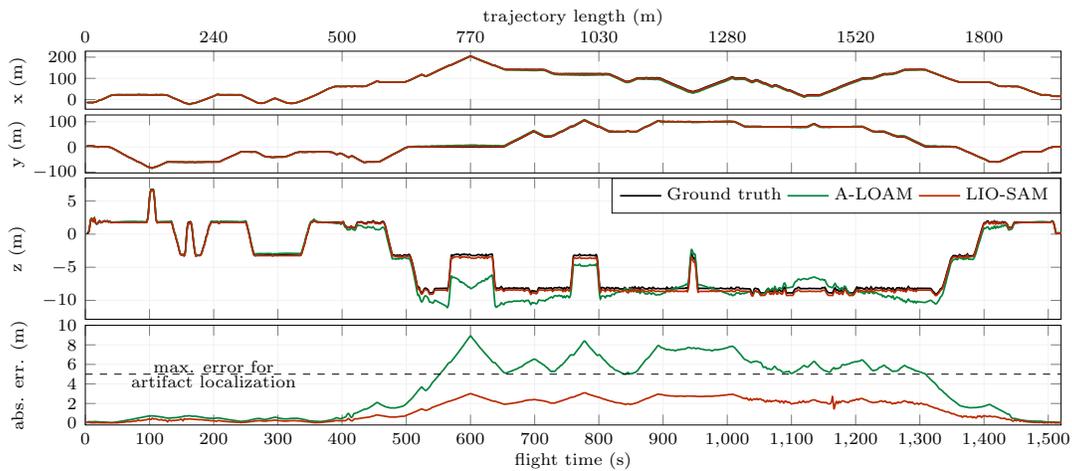


Figure 12. The performance of A-LOAM and LIO-SAM during a single flight within *Finals Prize Round World 01* (see Figure 46) of the DARPA SubT virtual environment. A-LOAM does not fuse the inertial measurements which assist LIO-SAM during LiDAR-scan matching in areas of the environment where such matching suffers from geometric degeneration, in the context of solving optimization problems. The selected environment contains a variety of narrow vertical passages where the performance of narrow-FOV LiDAR perception is limited, leading to drift in the ego-motion estimation that is clearly visible in the A-LOAM method. The LIO-SAM method was shown to achieve sufficient accuracy and low drift during long-term and arbitrary 3D navigation within a simulated environment.

mapping pipeline as it first extracts geometrical features out of raw LiDAR data and registers them to a feature map, with the motion prior given by the second optimization pipeline. The second factor-graph optimization fuses the mapping output with IMU measurements and outputs fast odometry used in the state estimation pipeline. The first graph is maintained consistently throughout the run, whereas the second graph optimization is reset periodically to maintain real-time properties.

In a simulated environment, LIO-SAM yields greater accuracy than A-LOAM for its fusion of inertial measurements with precisely modeled and known characteristics. A comparison of both the methods within the simulated environment is summarized in Figure 12. In the real world, the measurements of an IMU rigidly mounted on board a UAV contain a wide spectrum of large stochastic noise. During empirical testing, the integration method in LIO-SAM was shown to not be

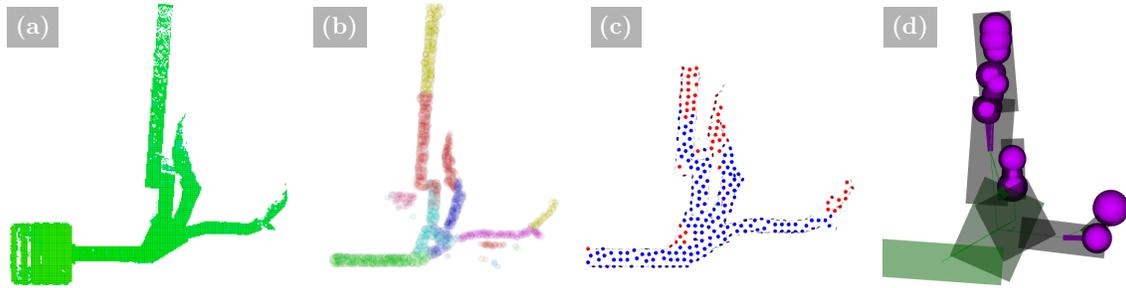


Figure 13. Top view of the used mapping structures from the intersection of the virtual Final Event map. DenseMap (a) is used for short-distance planning, SphereMap (b) for safety-aware long-distance planning, FacetMap (c) for storing surface coverage, and LTVMap (d) for compact topological information sharing among robots.

robust towards the unfiltered noise while frequency-band and pass filters induced significant time delays, destabilizing the pipeline completely. For the inability to accurately model the noise, real-world laser-inertial fusion is done manually by smoothing over a short history of past measurements (see [Section 6.2](#)).

7. Mapping

In this section, we present our approach to mapping the explored environments. As each task has specific requirements on the map properties, we designed multiple spatial representations, each of which is structured for a particular task. In particular, DenseMap ([Figure 13a](#)) is utilized for short-distance path planning; FacetMap ([Figure 13b](#)) for surface coverage tracking; SphereMap ([Figure 13c](#)) for fast and safe long-distance path planning; lightweight topological-volumetric map (LTVMap) ([Figure 13d](#)) for compressed, topological, and mission-specific information sharing between robots in low bandwidth areas; and LandMap ([Figure 15](#)) for representing feasible spots for safe UAV landing. These maps and the methods for building them are presented in this section.

7.1. DenseMap

Local information of the UAV is combined within a dense map to serve as the basis for the entire navigation stack, as described in ([Kratky et al., 2021a](#)). The map integrates information in a dense, probabilistic manner using an efficient octree structure implemented within the OctoMap ([Hornung et al., 2013](#)) library. During the map update, the data of each input modality producing spatial measurements are used to update the map with respect to the pose estimate correlating to the timestamp of the respective measurement. The data to be integrated are first cleared of any observation noise (see [Section 5](#)). The ray of each remaining spatial measurement is integrated within a discretized representation of the environment using the Bayes rule and ternary classification to the unknown, free, and occupied voxels. The output of dense mapping is convertible to other navigation representations and serves as the fundamental structure for local planning and dynamic obstacle detection.

To retain maximum information under constraints on real-time performance, the voxelization resolution is selected such that a scan insertion is processed at 5 Hz, at worst. The resolution can be locally increased if path planning demands a decrease in discretization errors. This is a useful feature for improving safety and repeatability in navigating highly narrow passages. To maintain the map structure, the local resolution is controlled by a factor n such that the local resolution equals $r/2^n$ with r being the default resolution of the dense map. In our sensory and computation setup, the default resolution is empirically set to 20 cm, reduced by a factor of $n = 2$ to 5 cm for navigating narrow passages, if required. The integrated data consist of LiDAR measurements and depth estimates of two RGBD cameras. These sensors are mounted on-board UAVs so that

the spatial observations cover roughly all directions around the robot, enabling almost arbitrary UAV-motion planning in collision-free 3D space.

7.2. SphereMap

To enable the UAV to quickly evaluate the travel time and risk caused by flying near obstacles while also pursuing any given goal, we developed a multilayer graph structure that uses volumetric segmentation and path caching, called SphereMap (Musil et al., 2022). All three layers of the SphereMap are updated near the UAV in every update iteration, which runs at approximately 2 Hz.

Path planning in the SphereMap depends on only one parameter c_R , which we call *risk avoidance*. It is used to trade path safety for path length. For long-distance planning, we disregard UAV dynamics and only take into account the path length and obstacle clearance along the path. We define the path cost between points \mathbf{p}_1 and \mathbf{p}_2 as

$$D(\mathbf{p}_1, \mathbf{p}_2) = L + c_R R, \quad (7)$$

where L is the path Euclidean length summed over all edges of the path in the sphere graph, and $R \in [0, L]$ is a risk value computed by examining the radii of the spheres along the path. For example, a path with all spheres with radii at the minimal allowed distance from obstacles would have $R = L$, and a path through open space with large sphere radii would have $R = 0$.

The lowest layer of the SphereMap is a graph of intersecting spheres, shown in Figure 13b. It is constructed by filling the free space of an obstacle k -d tree built from the DenseMap with spheres at randomly sampled points. The graph is continuously built out of intersecting spheres, and then by pruning the spheres that become unsafe or redundant. The radii of the spheres carry obstacle clearance information, which is used for path risk evaluation.

The second layer of the SphereMap is a graph of roughly convex segments of the sphere-graph. It is updated after every update of the sphere graph by creating and merging segments until every sphere in the graph belongs to a segment.

The third and last layer of the SphereMap is a navigation graph. For every two adjacent segments, we store one sphere-sphere connection, which we call a *portal* between the segments, as in (Blochliger et al., 2018). These portals form the vertices of the navigation graph. At the end of every SphereMap update iteration, we compute which paths are optimal according to the path cost from (7) between all pairs of portals of a given segment. The paths are computed only inside that given segment. If the segments are kept small (tens of meters in length), the recomputation is reasonably fast. The optimal portal-portal paths form the edges of the navigation graph. The UAV uses the navigation graph to quickly find long-distance paths between any two points in the known space by planning over the edges of the navigation graph, and then by only planning over the sphere graph in the first and last segments of the path.

7.3. FacetMap

The occupancy octree and SphereMap maps are sufficient for volumetric exploration. However, the goal of the DARPA SubT challenge was to locate artifacts, most of which could be detected only from cameras. Because the FOV of our UAVs' cameras did not cover the entire FOV of the LiDAR and depth cameras, not all occupied voxels in the occupancy map could be considered as "covered by cameras." For this reason, we developed another map, called FacetMap, illustrated in Figure 14. This map is a simple surfel map, with the facets stored in an octree structure, each having an orientation, a coverage value, and a fixed size. The FacetMap is built by computing the normals of the occupancy map at sampled occupied points, and creating facets with a set resolution if there are no existing facets with a similar normal nearby. The facets are updated (i.e., added or deleted) periodically at approximately 2 Hz in a cube of pre-defined size around the UAV.

Each facet holds a coverage value that is, for simplicity, defined as binary. A facet is marked as covered if the facet center falls into the FOV of any camera, and the ray from the camera to

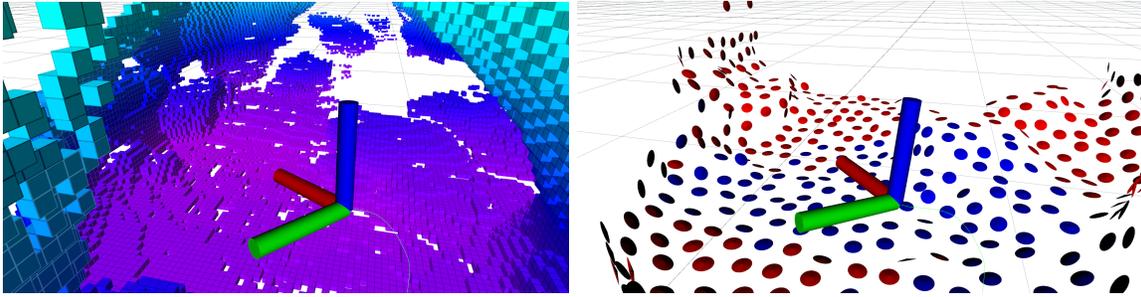


Figure 14. Illustration of the FacetMap in simulation as described in Section 7.3. The map is built from the DenseMap (left) by finding normals of sampled points. The orientation of the visualization discs (right) is determined by the facet’s normal, and the color by whether the facet was covered by the UAV’s front-facing cameras or not.

the facet center is at an angle lower than a defined threshold from the facet’s normal, so as to not mark surfaces as covered if they are viewed at a very skewed angle. The angle threshold was set empirically to 78° in the competition. Angles larger than the threshold reduced the probability of successfully detecting artifacts. The covered facets stay in the map even if the underlying occupancy map shifts (e.g., when an obstacle moves). As described in Section 8.2.3, one strategy used in our system uses this map to cover as much of the surface as possible while flying between volumetric exploration viewpoints. The strategy in Section 8.2.2 uses this map to completely cover surfaces of a dead-end corridor before backtracking to search a different area. Coverage of entire regions of the SphereMap can also be easily computed and then stored in the LTVMap, as described in Section 7.4.

7.4. LTVMap

Distributing all of the maps described in this chapter among the UAVs would be highly demanding for the communication network. As such, we have developed the lightweight topological-volumetric map (LTVMap), which combines the necessary mission-related information from the other maps and can be quickly extracted from the SphereMap and sent at any time.

This map consists of an undirected graph, where each vertex is created from a free-space segment in the original SphereMap and the edges are added for all of its adjacent segments. Each vertex holds an approximation of the segment’s shape. In our implementation, we use four DOF bounding boxes (with variable size and rotation along the vertical axis) for shape approximation, though any other shape could be used.

For cooperative exploration purposes, the frontier viewpoints (described in Section 8.1) found by a given UAV are also sent in the LTVMap, with each viewpoint being assigned an information value and segment from which the viewpoint is reachable. For surface coverage purposes, every segment in the LTVMap also holds a single numerical value representing the percentage of relevant surfaces covered in that segment. This value is computed by projecting points from the facets of the FacetMap and counting the points that fall into every segment. Further description and analysis of LTVMaps can be found in (Musil et al., 2022). These LTVMaps are shared among robots, and are used for cooperative search planning onboard UAVs, as described in Section 8.3.

7.5. LandMap

As described in Section 5.3, a downward-facing RGBD camera detects areas safe for landing. These areas are continuously collected within an unconnected set and stored in a sparse point-cloud manner with minimum mutual distance 5.0 m, low enough for avoiding unnecessary long paths to the nearest landing spot while keeping the LandMap memory-light even for large environment. An example of

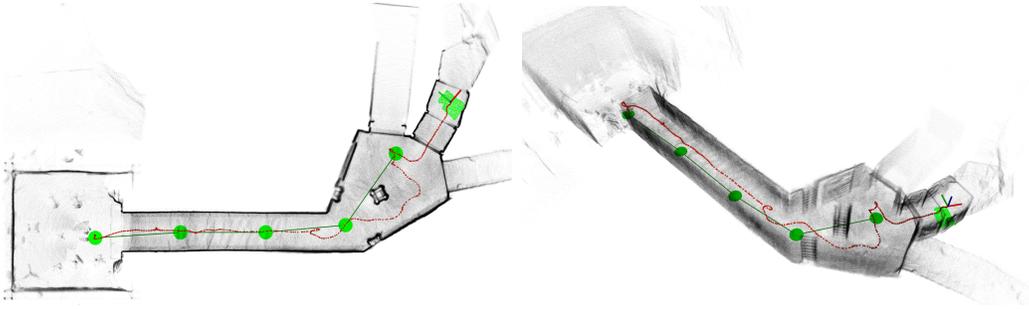


Figure 15. Example of the LandMap with resolution of 5 m built in the beginning of the DARPA SubT systems Final Event after 70 s of a UAV flight. The UAV is represented by the Cartesian axes with its trajectory colored in red. The LandMap incorporates the spots classified as safe for UAV landing (green circles) which are used during the UAV homing phase of the mission to ensure safety during the landing procedure.

the LandMap is shown in Figure 15. During the homing phase of the mission, the UAV navigates to an area connected to the ground station via the communication network (see Section 11.4). After reaching this area, the UAV navigates towards a safe landing spot as indicated by the LandMap, which is closest to its current pose (see mission state machine in Figure 27). While flying towards the LandMap-selected spot, the UAV lands sooner if the ground below the UAV is classified as safe-for-landing in the current RGBD data. The landing spots previously identified as safe are, once more, verified before landing in order to ensure safety in dynamic environments. If the spot is no longer safe for landing, it is invalidated and the UAV is navigated to the next closest landing spot.

8. Autonomous search

Since communication between robots in subterranean environments can never be ensured, the UAVs in our system operate completely autonomously and only use information from other robots to update their goal decision (e.g., blocking frontiers leading to areas explored by other robots). The system can also be controlled at a very high level by the human operator, which is described in Section 11.2. This section describes the high-level search autonomy of our system.

8.1. Informative viewpoint computation and caching

For exploration purposes, the UAVs in our system do not consider the information gain along trajectories, but rather sequences of discrete viewpoints, so that we can have a unified goal representation for both local and global search planning. These viewpoints are divided into places at which a UAV could obtain some volumetric information, called *frontier viewpoints*, and the points at which a UAV could cover some not-yet-covered surfaces with its cameras, called *surface coverage viewpoints*. Each viewpoint ξ , comprising of position \mathbf{p}_ξ and heading φ_ξ , is therefore assigned some information value $I(\xi)$. In our approach, the information gain of frontier viewpoints ξ_F and surface viewpoints ξ_S is computed as

$$I(\xi_F) = c_F \frac{n_{\text{unk}}}{n_{\text{rays}}}, \quad I(\xi_S) = c_S n_{\text{unc}}, \quad (8)$$

where $n_{\text{unk}}/n_{\text{rays}}$ is the ratio of rays cast in the UAV's depth cameras' and LIDAR's FOVs that hit an unknown cell of the occupancy map before hitting an occupied one or going out of range. Similarly, n_{unc} is equal to the number of uncovered facets of the FacetMap, hit by rays that are cast in the UAV's RGB cameras' FOVs. The constants c_F and c_S are empirically tuned to alter the UAV's next viewpoint selection and hence, its behavior. Additionally, a positive or negative bias c_{FS} can be added to the information value of either function to make the UAV prefer one type of viewpoints more.

The UAV does not sample and evaluate viewpoints on-demand after reaching some viewpoint, rather it continually samples viewpoints in its vicinity at a given rate and stores them into a map of cached viewpoints. Only viewpoints that have $I(\xi)$ above some threshold, are safe, not too close to another informative viewpoint, and not blocked by mission control are stored. The viewpoints are also pruned from the map if they become uninformative or if a better viewpoint is nearby. Lastly, viewpoints that were found in a previous update and are now outside the local update box, are kept as global goals and are pruned more aggressively than the local goals. This approach continually produces a map of informative viewpoints that is denser near the UAV and sparse in the rest of the environment.

8.2. Single-UAV autonomous search planning

In our approach, the UAV can be in three states of autonomous search—*locally searching*, *traveling to goal* or *returning*, and the goal planning and evaluation is divided into local and global planning, as in (Dang et al., 2019b). In all of these states, reachability determination and path planning to any given goal is performed using the rapid long-range path finding provided by the SphereMap, described in Section 7.2. The transitions between the three states are fairly simple—if there are informative and reachable viewpoints near the UAV, the UAV is in the locally searching state and tries to always keep a sequence of two viewpoints. These are given to the trajectory planning pipeline so that the UAV doesn’t stop at each viewpoint and compute the next best one. This is done by performing a local replanning of the sequence whenever the UAV is getting close to a viewpoint.

When there are no reachable viewpoints near the UAV or when new information is received from the operator or other robots, a global replanning is triggered.

The global replanning, inspired by (Dang et al., 2020b), computes paths to all stored informative viewpoints (not only in the local search box) and evaluates them. The best viewpoint is then set as a goal to the long-distance navigation pipeline described in Section 9.1. Finally, the *returning* state is triggered when the global planning does not find any reachable goals, or if the operator demands it, or if $t_{\text{home}} < c_H t_{\text{battery}}$, where t_{home} is the estimated time of flight needed to return to the base station, t_{battery} is the estimated remaining flight time, and c_H is an empirically tuned constant. The value of t_{home} is computed from the UAV’s average flight speed, and a path found through the SphereMap to the base station. If there is no path to the base station, the UAV will instead try to return along a tree of visited positions, which is built specifically for this purpose, so that for example if a path is only temporarily blocked, the UAV will fly to the roadblock, and if it is removed, will continue flying to the base station. The UAV can also recover from this state, if it is returning due to having found no reachable goals, and suddenly some goals become reachable again. When the UAV gets close to the goal, it switches back to the *locally searching* state.

The reward functions used to evaluate goals govern the behavior of the UAV while searching the environment, and as such, they define the search strategy of the UAV. For simplicity, we made the local planner and global planner use the same reward function in a given strategy, with only one difference, that the local planner can add a penalty to local goals, based on the UAV’s current momentum and heading, to allow for smoother local search, which is a highly simplified version of the local viewpoint tour planning done by (Zhou et al., 2021b). These strategies and their corresponding reward functions were utilized in the challenge:

8.2.1. Greedy search strategy (GS)

The chosen reward function for selecting the next best viewpoint ξ from the current UAV viewpoint ξ_{UAV} (the UAV’s current position and heading) can be written as

$$R_{\text{GS}}(\xi_{\text{UAV}}, \xi) = I(\xi) - D(\xi_{\text{UAV}}, \xi), \quad (9)$$

where $I(\xi)$ is the information value of the viewpoint (described in Section 8.1) and D is the best path cost computed in the SphereMap (described in Section 7.2). This type of reward was selected for its simplicity, which allows easy debugging and tuning of UAV behavior. It is also easier to

extend this reward function to the multi-UAV cooperation reward functions in [Section 8.3](#). This reward function for controlling the next best goal selection thus depends on the constants c_F , c_S , c_{FS} described in [Section 8.1](#) and the risk-awareness constant c_R used in path planning, which can be used to tune the search based on the desired behavior. The constants c_F , c_S , c_{FS} control whether and how much the UAV prefers frontier viewpoints or surface viewpoints, while c_R is set according to the desired risk avoidance. This reward function is very simple and can take the UAV in various directions, leaving behind uncovered surfaces in faraway places. The next strategy aims to solve this.

8.2.2. *Dead end inspection strategy (DEI)*

A more thorough reward function can be written as

$$R_{\text{DEI}}(\xi_{\text{UAV}}, \xi) = I(\xi) - D(\xi_{\text{UAV}}, \xi) + (D(\mathbf{p}_{\text{HOME}}, \xi) - D(\mathbf{p}_{\text{HOME}}, \xi_{\text{UAV}})). \quad (10)$$

This strategy adds the difference in path costs to the base station position \mathbf{p}_{HOME} from the evaluated viewpoint and from the UAV. This greatly increases the value of viewpoints that are deeper in the environment, relative to the UAV. Using this reward function, the UAV will most likely first explore frontiers until reaching a dead-end, and then thoroughly cover surfaces from the dead end back to the base, analogous to a depth-first search.

8.2.3. *Viewpoint path enhancement strategy (VPE)*

The third strategy used on the UAVs is not a change of the reward function, but rather a simple way to increase surface coverage when the UAV is flying through long stretches of explored but not perfectly covered space, either in the DEI or GS strategy. If VPE is enabled and the UAV is flying to a distant goal, then we periodically take the short-distance trajectory from the local path planner (described in [Section 9](#)), sample it into multiple viewpoints, and try to perturb these viewpoints to increase surface coverage, while not increasing the flight time too much. Thus we fully utilize the agility of quadcopter UAVs, as they can easily turn from side to side while flying in a given direction.

8.2.4. *Comparison of the strategies*

During pre-competition testing, the three strategies mentioned above proved to be nearly identical in the total amount of covered volume and surfaces. However, there are serious differences in the overall behavior and what it means for cooperation. The GS strategy on average covers the most volume and surfaces but leaves behind many patches of surfaces or frontiers in very far-away places, due to its greedy nature. The VPE strategy just slightly alters the GS strategy to cover more surfaces in total at the cost of less explored volume but also leaves unfinished goals behind. This has generally been very useful in areas with long corridors that have a high amount of short branches leading off from the main corridor, such as in tunnels or cramped urban areas because the VPE strategy will force the UAV to peek into the corridors, but not to rigorously explore them as with the DEI strategy. The DEI strategy usually covers less space and surfaces in total, but what is most important—it does not leave uncovered and unexplored parts of the environment behind, meaning that for cooperative missions, no other UAV needs to go to that space again, as that space has been completely covered. This is essential in longer missions to ensure complete coverage of the environment.

8.3. Probabilistic cooperative search planning

Our approach to multi-UAV search planning was to make the UAVs completely autonomous and decentralized by default, while also being able to share important information and use it for their own planning. Each UAV always keeps the latest version of the LTVMap (described in [Section 7.4](#)) received from a given UAV. When a new LTVMap is received, every newest received map currently being stored onboard the UAV is updated by every other newest received map, as well as by the LTVMap constructed from the UAV’s own SphereMap.

The updating is done so that the frontier viewpoints, sent along with each LTVMap, which fall into explored space in other LTVMaps, are blocked. This is difficult to do in a deterministic manner due to map drift and other inaccuracies. Therefore, we approached this problem similarly as in (Burgard et al., 2005) by gradually decreasing the reward of frontier viewpoints whenever the viewpoint falls into the explored space of any segment’s bounding box in a received LTVMap. Because the LTVMap bounding boxes are a very rough approximation of the segments’ true shape, this reward decreasing is weaker at the edges of the bounding boxes and strongest when the viewpoint lies deep inside the bounding box. Each frontier viewpoint in any LTVMap is assigned a likelihood $l(\xi \in V_{\text{exp}})$ to represent how likely it is that the viewpoint has already been visited by any other UAV. The $l(\xi \in V_{\text{exp}})$ of any viewpoint is the maximum of a function describing the likelihood that the point lies in a given segment’s bounding box, computed over all segments of all the other received LTVMaps. This likelihood function can be selected arbitrarily; for our approach, we selected a function, which is equal to 0 outside of the segment’s bounding box, and grows linearly to 1 the closer it is to the center of the bounding box. The updates of these $l(\xi \in V_{\text{exp}})$ values for a three UAV mission can be seen in Figure 17.

For a frontier viewpoint ξ_L in the UAV’s local map, which has $l(\xi_L \in V_{\text{exp}}) > 0$, the reward function changes into

$$R(\xi_{\text{UAV}}, \xi_L, \mathbb{M}) = l(\xi_L \in V_{\text{exp}})R_R(\xi_{\text{UAV}}, \xi_L, \mathbb{M}) + (1 - l(\xi_L \in V_{\text{exp}}))R_L(\xi_{\text{UAV}}, \xi_L), \quad (11)$$

where R_L is the reward function defined by the employed single-UAV search strategy described in Section 8.2. This does not take into account any information from other UAVs. R_R is a reward function which takes into account other frontiers in received LTVMaps that could be reachable through ξ_L , as illustrated in Figure 16. If $l(\xi \in V_{\text{exp}}) = 0$, it means that the viewpoint does not fall into the space of any received LTVMap and the UAV only decides based on its own maps. If $l(\xi \in V_{\text{exp}}) = 1$, the viewpoint surely lies in explored space of another UAV, hence it does not

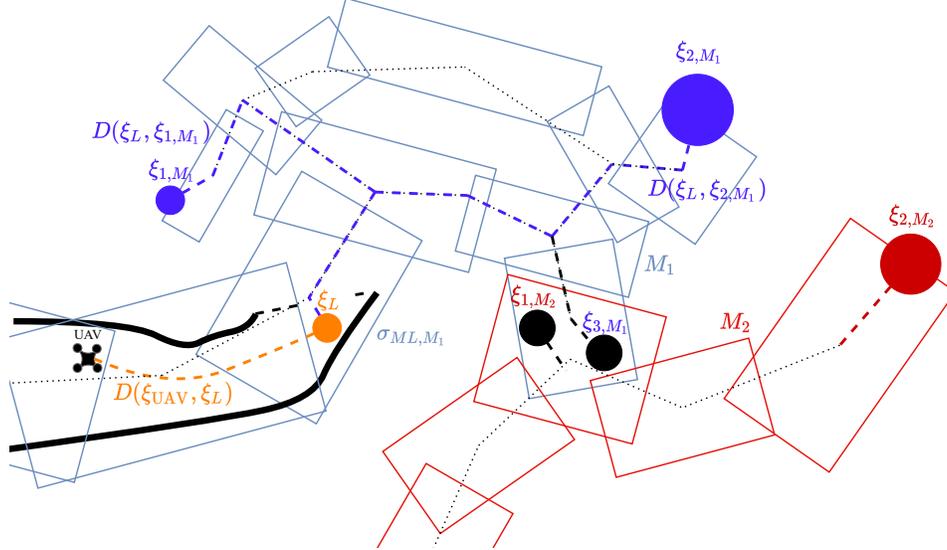


Figure 16. Diagram illustrating the computation of the cooperative exploration reward function, as described in (12). The image shows a UAV evaluating a frontier viewpoint ξ_L (orange) in its local occupancy map (black lines). The UAV has received two LTVMaps M_1, M_2 from two other UAVs. As the local map frontier ξ_L falls into one of the free space segments σ_{ML, M_1} of M_1 , it is assigned as belonging to that segment and acts as an edge in planning paths between the local map and the received map M_1 . Therefore, the frontier viewpoints $\xi_{1, M_1}, \xi_{2, M_1}$ should be reachable through ξ_L . A path to them is estimated across the centroids of the segments of M_1 . The viewpoints $\xi_{3, M_1}, \xi_{1, M_2}$ (black) are marked as having $l(\xi \in V_{\text{exp}}) = 1$, since they fall deep into the explored space of the other received map, and are therefore not considered.

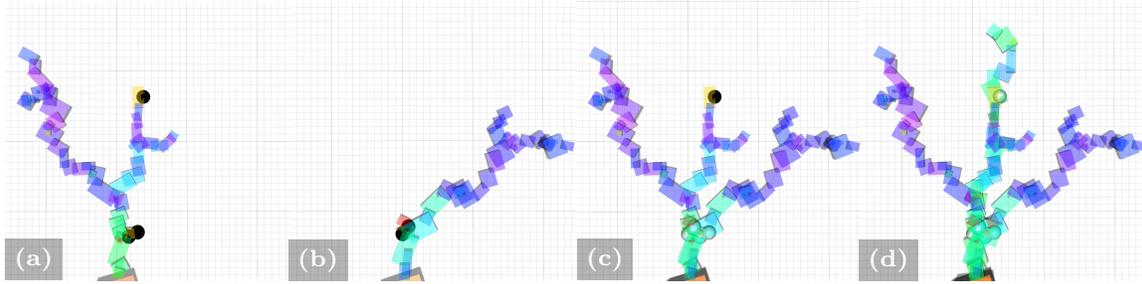


Figure 17. Illustration of LTVMap sharing and utilization during a cave exploration mission in simulation with three UAVs running the DEI strategy (described in Section 8.2.2). The heatmap color of the LTVMap segments shows surface coverage of the individual segments, with purple signifying complete coverage. The colors of the exploration viewpoints signify their $l(\xi \in V_{\text{exp}})$ value, with white having a value equal to 1 and black being 0. Image (a) shows the LTVMap sent by UAV1 after returning to communication range with the base station. This map is given to UAV2, which then launches and chooses to explore the nearest unexplored frontier in the map of UAV1. Image (b) shows the LTVMap sent by UAV2 when it is returning. Image (c) then shows how the maps are co-updated onboard UAV3, which launches after receiving the LTVMap from UAV2. The only nonexplored viewpoint remaining is in the top part of the image. Image (d) shows the maps received by the base station from all three UAVs at the end of the mission with no unexplored viewpoints remaining.

bring any volumetric information to the team, so the UAV considers whether exploring it would eventually lead it to globally unexplored viewpoints. Figure 17 illustrates how sharing the LTVMap helps UAVs to not explore already explored parts of the environment. The function R_R that achieves this behavior was designed as

$$R_R(\xi_{\text{UAV}}, \xi_L, \mathbb{M}) = \max_{M \in \mathbb{M}} \max_{\xi_R \in M} I(\xi_R) - D(\xi_{\text{UAV}}, \xi_L) - \frac{D_R(\xi_L, \xi_R, \sigma_{ML,M})}{1 - l(\xi_R \in V_{\text{exp}})}, \quad (12)$$

where \mathbb{M} is the set of all received LTVMaps, and $\sigma_{ML,M}$ is the most likely segment that ξ_L belongs to in a map M . The function D_R is a special path cost function computed as a sum of Euclidean distances of segment centers in a given map, spanning from ξ_L , through the center of $\sigma_{ML,M}$, and towards a given frontier viewpoint ξ_R . The value of D_R is also scaled by a user-defined parameter. This is done so as to increase the cost of viewpoints in received maps as there is more uncertainty about the path to these viewpoints. The division by $1 - l(\xi_R \in V_{\text{exp}})$ serves to gradually decrease the reward of exploring the viewpoint up to $-\infty$ when the viewpoint was surely explored by another UAV. Computation of this reward function is illustrated in Figure 16.

The percentage of covered surfaces inside segments received in the LTVMap is used for blocking the surface coverage viewpoints in segments, where the percentage is above a user-defined threshold. The segments with low surface coverage could be used as additional goals in a similar manner as shared frontiers in Figure 16. However, for simplicity, this was not implemented.

8.4. Autonomy robustness enhancements

One important problem is that in the case of dark and nonreflective surfaces (common in the DARPA SubT Finals course) the LiDAR beam does not return with enough energy. Such surfaces will not be marked as occupied and essentially become permanent frontiers, which means that some informative viewpoints, as defined in Section 8.2, are noninformative. To solve this, the UAV builds a map of visited positions. With time spent near a visited position, we linearly decrease the value of nearby viewpoints. After some time, the sampling is blocked near those positions completely.

Another problem arising is due to highly dynamic obstacles in the occupancy map, such as other robots, fog, or very narrow corridors where the discretization of occupancy can oscillate. As such, the reachability of a given viewpoint can oscillate. This was solved by putting a timeout on trying to reach a given viewpoint and was triggered if the UAV did not get closer to the goal within a defined

time. After this timeout, an area around the viewpoint is blocked until the end of the mission, or until a manual reset by the operator. This approach may cause the UAV to block some goals that are only temporarily blocked by another robot in narrow passages, but it was deemed preferable rather than having the UAV permanently oscillate in such passages.

The autonomy system can be easily controlled by operator commands (described in Section 11.2) which can block viewpoints in a set cylinder in space, force the UAV to explore towards some goal, or simply move to a given position and stay there. In this way, problematic situations not covered by our solution, such as organizing multiple robots in a tight corridor, can be resolved by the operator.

9. Path planning, trajectory generation and tracking

Planning collision-free paths and generating dynamically feasible trajectories is another vital component of the presented UAV system operating in a constrained environment. The sequence of waypoints (a waypoint in this context is either only a point in space, when we do not care about the heading, or a point in space and heading, for example when using the VPE strategy in Section 8.2.3, that the local planner should move the UAV through) that efficiently guides the UAV through the environment is produced by the long-distance navigation module, described in Section 9.1. Given the navigation waypoints, a computationally undemanding multistage approach is applied to obtain a trajectory lying at a safe distance from obstacles, while also respecting dynamic constraints (limits on velocity, acceleration, and jerk) and minimizing the time of trajectory following. In particular, the solution can be divided into three modules: path planning to obtain the local reference path, path processing to increase the safety margin of the path, and the trajectory generation to obtain a time-parametrized trajectory respecting the dynamic constraints of the UAV. The diagram illustrating connections and data transfer between particular modules in path planning and trajectory generation pipeline is shown in Figure 18. The long-distance path found in SphereMap, the local path found in DenseMap, the postprocessed path, and the dynamic trajectory are depicted in Figure 19.

9.1. Safety-aware long-distance navigation

When a goal, or a sequence of goals, is set to the navigation stack, the long-distance navigation module computes a path through the SphereMap, optimal according to (7). The module then keeps this path and utilizes the trajectory planning and tracking modules to follow it. This is done simply by a “carrot and stick” approach, where the trajectory planning module is given a near waypoint

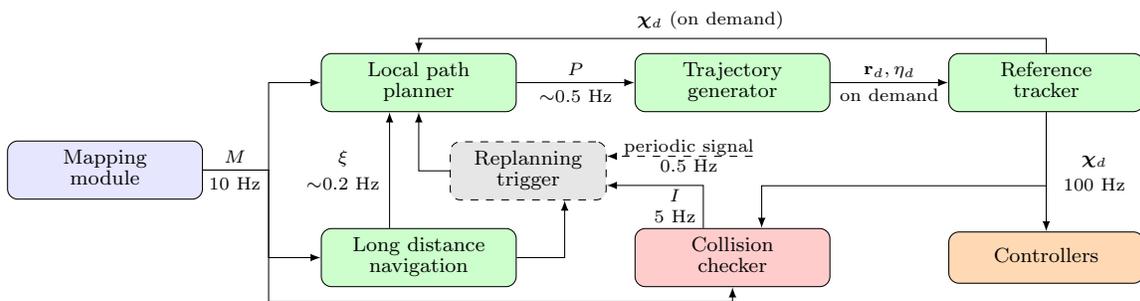


Figure 18. A diagram of the path planning and trajectory generation pipeline. Given 3D map M provided by *Mapping module*, the *Local path planner* produces path P connecting a start position for planning derived from full state reference χ_d provided by *Reference tracker*, with the viewpoints ξ supplied by *Long distance navigation* module. *Trajectory generator* produces feasible trajectory along the collision-free path P and supplies the position and heading reference (\mathbf{r}_d, η_d) to a *Reference tracker*. *Reference tracker* creates a smooth and feasible reference for the reference feedback controllers. The *Local path planner* is triggered by a new set of goals, periodic signal or by an interrupt I generated by *Collision checker* responsible for detection of collisions with respect to most recent map of the environment.

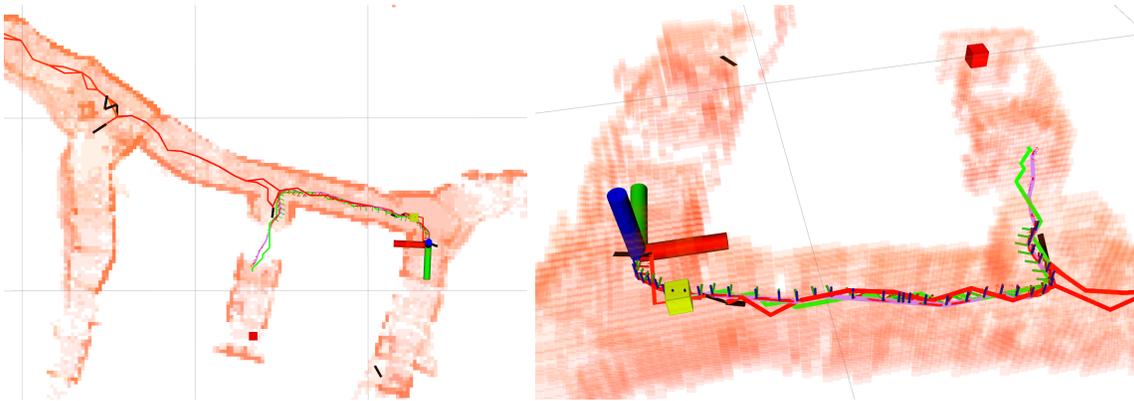


Figure 19. A two-view visualization of the path produced by all stages of the planning pipeline. The cached long-distance paths (—) between portals (—) are found in the SphereMap. A geometric path (—) is found in the DenseMap to the next waypoint given by the SphereMap. This path is then postprocessed (—) to be further away from obstacles, and a time-parametrized trajectory respecting the dynamics of the UAV is sampled (small axes). The small axes represent samples from the trajectory with constant time step, so axes further away from each other mean that the velocity of that part is higher. The current UAV pose is shown as large axes.

(approx. 20m away from the UAV at maximum, to keep planning time short) on the path. This temporary goal waypoint is then slid across the path towards the goal.

If the trajectory planning and tracking modules cannot advance along the SphereMap path for a specified amount of time, which can be caused by a dynamic obstacle such as a rockfall, fog, or another robot, the SphereMap path following is stopped and an unreachability flag is raised. The UAV then chooses a different goal or tries to find a new path to the same goal based on the current state of mission control.

When the search planning requires the UAV to fly through multiple nearby goal viewpoints, such as when covering the surfaces in a room with cameras or when visiting multiple viewpoints while traveling and using the VPE strategy described in Section 8.2.3, the local path planning module is instead given a sequence of waypoints (containing both the goal viewpoints for surface coverage, which require heading alignment, and waypoints that do not require heading alignment and only serve to guide the local path planning). Thus the output of this module is always a sequence of one or more waypoints, which may or may not require heading alignment, and through which the local path planning module should find a path in a short time, which we can control by changing the look-ahead distance.

9.2. Local path planning

The grid-based path planner coupled with iterative path processing was adopted from (Kratky et al., 2021a) to obtain the primary reference path. The proposed approach presents a path planning and processing algorithm, which is based on the traditional A* algorithm applied on a voxel grid with several modifications to decrease the computational demands. The first modification lies in avoiding the computationally demanding preprocessing of the map representation (e.g., obstacle dilation by Euclidean distance field), which often requires more time than the actual planning on the grid. This holds true especially for shorter direct paths that leave a significant portion of the previously processed environment unexploited. For this reason, the presented approach builds a k -d tree representation of the environment which is then used to conclude the feasibility of particular cells, based on their distance to the nearest obstacle. As a result, the computational demands are partially moved from the preprocessing phase to the actual planning phase. This approach is particularly efficient in the case of paths that do not require exploiting a significant part of the environment. The second important modification is applying node pruning, similar to the jump point

search algorithm (Harabor and Grastien, 2011). This modification helps to decrease the number of unnecessarily expanded nodes. As such, it lowers the computational time required for obtaining the solution. A detailed analysis of the influence of particular modifications on the performance of the planning algorithm is provided in (Kratky et al., 2021a).

To allow the generated paths to lead through narrow passages, the limits on safety distance are set to the dimension of the narrowest opening that is supposed to be safely traversable by the UAV. However, setting this distance to a value that ensures safety in the event of the maximum possible deviation from the path caused by any external or internal source would lead to the preclusion of entering narrow passages of the environment. On the contrary, setting this distance to a minimum value without considering safety margins would increase the probability of collision along the whole path. To balance the traversability and safety of the generated path, the minimum required UAV-obstacle distance applied in the planning process is set to the lowest traversability limit, and iterative path postprocessing is applied to increase the UAV-obstacle distance in wider parts of the environment. The employed postprocessing algorithm proposed in (Kratky et al., 2021a) iteratively shifts the path towards the free part of the environment, while continually maintaining the path’s connectivity. As such, this anytime algorithm increases the average UAV-obstacle distance throughout the flight, which significantly improves the reliability of the navigation with respect to imprecisions in the reference trajectory tracking.

The generated path is periodically replanned at a rate of 0.5 Hz to exploit the newly explored areas of the environment and handle dynamic obstacles. The continuous path following is achieved by using the predicted reference generated by the MPC tracker (Baca et al., 2018) to identify the starting position for the planner at time T_s in the future. Apart from the periodic replanning, the planning is also triggered by the detection of a potential collision on the prediction horizon of the trajectory reference produced by the MPC tracker. The potential collisions are checked at a rate of 5 Hz by comparing the distance of particular transition points of the predicted trajectory to the nearest obstacle in the most recent map of the environment. Depending on the time left to the time instant of a potential collision, the UAV is either requested to perform a stopping maneuver or to trigger replanning with the most up-to-date map.

9.3. Trajectory generation

The path generated by the path planning pipeline is a series of waypoints, each consisting of a 3D position and heading. A trajectory (a series of dense time-parameterized waypoints) is generated for each new path, so that the motion of the UAV satisfies translational dynamics and dynamic constraints up to the 4th derivative of position. The dynamics of the trajectory can be changed according to the current safety distance limit. However, in the Final Event, this feature was disabled, as the UAV was already constrained to 1 m s^{-1} and further slowdown would unnecessarily prolong the time spent in a narrow passage, where the risk of collision is higher. The trajectory generation system is based on the polynomial trajectory generation approach (Richter et al., 2016; Burri et al., 2015), but it was significantly extended to perform in a constrained, real-world environment (Baca et al., 2021). This approach was modified to minimize the total flight time while still satisfying the dynamic constraints. Furthermore, an iterative sub-sectioning algorithm was added to force the resulting trajectory into a feasible corridor along the original path. Moreover, a fallback solver was added to cope with invalid QP solver results caused by numerical instabilities or in case of the solver timeout. The QP solver sometimes fails to produce a feasible trajectory, e.g., by violating the dynamic constraints, or by violating the corridor constraints. In such cases, we find an alternative solution by linearly sampling each original path segment. The time duration for each segment is estimated heuristically as an upper bound using the same method as in the initialization of the polynomial trajectory generation (Baca et al., 2021). Although the trajectory produced by this method violates the dynamics in each waypoint, the underlying MPC Trajectory tracking mechanism provides smooth control reference even at these points. Most importantly, despite the fallback solution not being optimal, it is tractable and is guaranteed to finish within a fraction of the

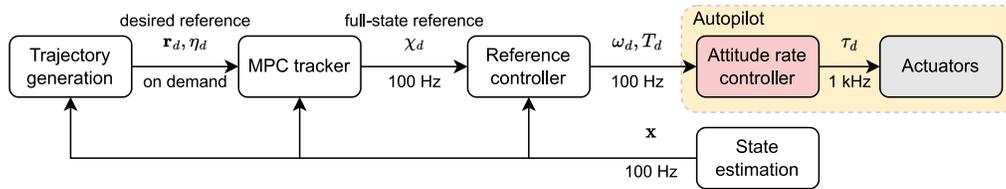


Figure 20. A diagram of the system architecture (Baca et al., 2021): *Trajectory generation* supplies the position and heading reference (\mathbf{r}_d, η_d) to the *MPC tracker*. The *MPC tracker* creates a smooth and feasible reference for the *Reference controller*. The *Reference controller* produces the desired angular velocities and thrust (ω_d, T_d) for the embedded *Attitude rate controller*, which sets the desired speed of the motors τ_d .

time of the polynomial optimization. Finally, a dynamic initialization mechanism and a time-outing system were added to cope with the nonzero trajectory generation and path planning computation times. Even though the path planning and the trajectory generation can last for several hundreds of milliseconds, the resulting trajectory always smoothly connects to the currently tracked trajectory. Therefore, no undesired motion of the UAV is produced. The updated trajectory generation approach was released and is maintained as part of the MRS UAV System (Baca et al., 2021).

9.4. Trajectory tracking and feedback control

The low-level guidance of the UAV is provided by a universal UAV control system, as developed by the authors of (Baca et al., 2021). The onboard control system supports modular execution of UAV reference generators, feedback controllers, and state estimators. During the SubT Finals, the system exclusively utilized the geometric tracking control on $SE(3)$ (Lee et al., 2010) to follow the desired states generated by the MPC Tracker (Baca et al., 2018). First, the MPC Tracker is supplied with a time-parametrized reference trajectory, from which a smooth and feasible reference state consisting of position, velocity, acceleration, jerk, heading, and heading rate is generated by controlling a virtual model of the UAV. Second, the feedback controller minimizes the control error around the generated reference state and produces an attitude rate reference for the low-level attitude rate controller embedded in the Flight Control Unit (FCU). Figure 20 depicts the pipeline diagram of the control system with data flow among individual modules.

10. Artifact detection, localization, and reporting

Objects of interest (artifacts) in the explored area are detected visually using a CNN that processes images from several onboard RGB cameras covering the frontal, top, and bottom sectors of the UAV. The CNN detector is trained on our manually labeled dataset and outputs predicted bounding boxes and corresponding classes of the artifacts in the input images. To estimate the 3D positions of the detections, we have leveraged the onboard 3D LiDAR sensor and the mapping algorithm described in Section 7. These positions are processed by an artifact localization filter based on our previous work (Vrba et al., 2019), which fuses the information over time to filter out sporadic false positives and improve the localization precision. The artifact detection, localization, and filtering pipeline is illustrated in Figure 21.

Our approach consolidated into a similar principle of early recall and late precision proposed by (Lei et al., 2022). The CNN generates a high amount of detections to not miss any artifact at the cost of a high false positive rate. The false positives are later filtered out by the localization filter and the resulting hypotheses are further pruned by the human operator to improve the precision of the pipeline as a whole.

10.1. Artifact detection

The artifact detection is executed in parallel on image streams from all cameras at the same time, which would require a dedicated Graphical Processing Unit (GPU) onboard the UAV. Therefore,

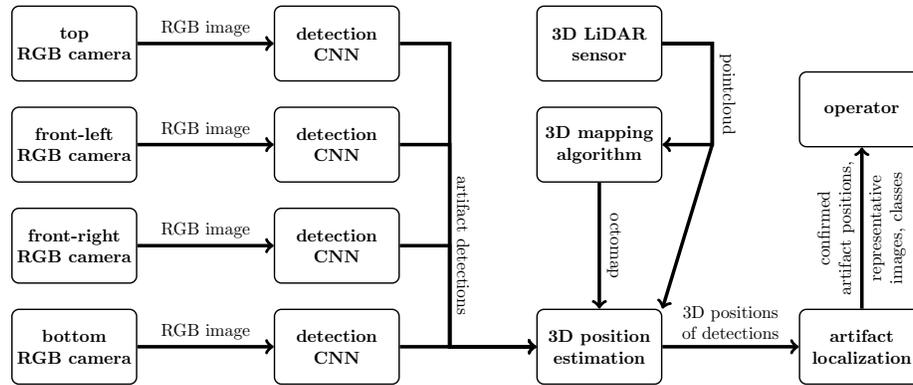


Figure 21. Schematic of the artifact detection and localization pipeline.



Figure 22. Training images containing artifacts captured by the onboard cameras in cave (a), tunnel (b), and urban (c) environments.

we have chosen the lightweight MobileNetV2 CNN (Sandler et al., 2018), in order to achieve a high detection rate and keep the load on the onboard computer as low as possible.

The CNN is running on the Intel UHD GPU that is integrated within the onboard CPU of the UAV. The integrated Intel GPU interfaces with our pipeline using the OpenVino⁵ framework. The OpenVino framework together with the Intel GPU achieves more than 5 Hz detection rate on 4 cameras in parallel but due to fixed resource allocation, we are locking the camera rates to 5 Hz. This artificial throttling of the detection rate avoids issues when the integrated GPU locks the memory resources for the CPU, which might lead to lag in the control pipeline.

The MobileNetV2 base model is modified for training using the OpenVino open-source tools. The evaluation of the model is based on the mean average precision metric (mAP) and recall. The mAP metric is a standard metric for object detection models since it provides information about how accurate the prediction is. Recall provides an understanding what is the ratio between true positive predictions and the total number of positive samples in the dataset.

The main challenge for the model is to adapt to different domains—mine, urban, and cave environments have different lighting and backgrounds (see Figure 22), which affect the detection performance. Moreover, the angle from which the images were taken is different as part of the images in the dataset were taken by ground vehicles and the rest by UAVs.

As the whole dataset was initially not available, we had to train the model incrementally whenever we gathered data from a new type of environment or camera angle to ensure we represented all cases uniformly in the training data. The incremental training was more time-efficient than retraining on the whole dataset each time new data was added. Training from scratch or checkpoints took us 2–3 days using our GPU capabilities, while incremental training produced good results in only 4–6 hours

⁵ docs.openvino.ai/latest/index.html

of training. Having the possibility to relatively quickly update the model was critical when we were doing practical experiments or hot-fixing some false-positive detections during competition days or our experimental campaign.

For training the model on a growing dataset, we used a variety of learning schedulers from the MMDetection toolbox (Chen et al., 2019). The Cosine scheduler designed by (Loshchilov and Hutter, 2016) is used for warm-restarts of the training pipeline to overcome the loss of learned features. The main challenge of transfer learning is to overcome the loss of learned distribution on the previous dataset when training the model on the new dataset (in this case the new dataset is a combination of the previous dataset and newly collected data).

In our experience, different learning rate schedulers should be used depending on the size of newly added data:

- *Cosine scheduler* (Loshchilov and Hutter, 2016) is used during clean model training on the initial dataset.
- *Cyclic scheduler* (Smith, 2015) is used when the size of new data is more than 15 % of the size of the initial dataset.
- *Step decay scheduler* is used when less than 15 % of the initial dataset size is added.

The model was trained using NVIDIA GeForce RTX 3090 video card with 24 GB of RAM, with 64 images per batch. The training size initially contained around 13 000 images and incrementally increased to 37 820 as new backgrounds and false negative samples were gradually added. Out of the 37 820 images 31 000 were labeled artifacts and 6820 were background images without any artifact to reduce the false positive rate. The train and validation split was 70 % to 30 % per training size. We open-sourced our training pipeline to facilitate replicating the achieved results by the community: github.com/ctu-mrs/darpa_subt_cnn_training. This method resulted in a score of 49.1 % mAP on the whole dataset. Such a value is acceptable on the onboard computation unit with limited resources, due to which a trade-off between accuracy and detection was necessary.

The dataset was collected using the off-the-shelf objects that were specified by the organizers, see Figure 1. The data have been recorded from the onboard cameras on the UAVs and UGVs, in particular:

- Intel RealSense D435,
- Basler Dart daA1600,
- Bluefox MLC200w.

The Basler cameras do not have an IR filter installed to maximize the amount of captured light. Altogether the dataset has 37820 images, sometimes with multiple objects in one frame. An example of images from the dataset is shown in Figure 22.

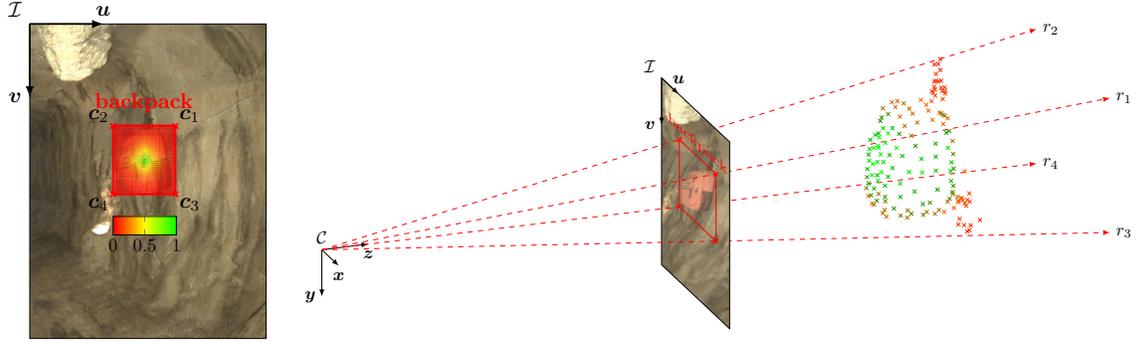
We publish the labeled detection datasets that were used for training of the neural network at github.com/ctu-mrs/vision_datasets. In addition, we also publish the tools to convert it into PASCAL VOC or COCO formats for immediate usage on most of the open-source models.

10.2. Estimation of 3D position of detections

Positions of the detected objects are estimated using data from the onboard LiDAR sensor and the mapping algorithm. Each detection is represented by four corner points $\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3, \mathbf{c}_4$ of its bounding rectangle in the image plane of the corresponding camera, as estimated by the detector (see Figure 23a). These points are expressed as undistorted pixel coordinates in the image frame \mathcal{I} .

The mathematical projection model of the camera $f_{\text{proj}} : \mathbb{R}^3 \rightarrow \mathbb{R}^2$ is assumed to be known. In our case, we have used the standard pinhole camera model formulated as

$$k \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_u & 0 & u_0 \\ 0 & f_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \quad (13)$$



(a) Example of an artifact detection with an overlay visualization of the sample weighting function f_w . (b) Model of the camera and the point cloud-based sampling method. Rays r_1, r_2, r_3, r_4 are projections of c_1, c_2, c_3, c_4 , respectively. Only the points within the area defined by these rays are selected. The selected points are colored based on their weight. Nonselected points are not drawn for clarity.

Figure 23. Illustration of the point sampling for 3D position estimation of detected artifacts with an example detection of a backpack.

where f_u, f_v, u_0, v_0 are parameters of the model (focal length and image center), $[x, y, z]^T$ is a 3D point in the camera coordinate frame \mathcal{C} , and u, v are distortion-free pixel coordinates in the image frame \mathcal{I} , corresponding to the 3D point (see Figure 23b for illustration). To model the distortion of the real-world camera, we have used a standard radial-tangential polynomial distortion model. It is worth noting that the output of f_{proj}^{-1} is a 3D ray and not a single point, which is represented in the model by the free scaling factor $k \in \mathbb{R}$.

The input LiDAR scan is represented as a set of 3D points $\mathcal{S} = \{\mathbf{p}_i\}$ expressed in the camera coordinate frame \mathcal{C} . The occupancy map is represented using the DenseMap data structure that is described in Section 7, and which provides a raycasting function $f_{\text{raycast}} : \mathcal{R} \rightarrow \mathbb{R}^3$ where \mathcal{R} is the set of all 3D rays. The function f_{raycast} returns the point, corresponding to the first intersection of the specified ray with an obstacle in the environment (or nothing if there is no such intersection).

The position of each detected object is estimated from a number of points that are sampled using two methods: a primary one that utilizes the latest available point cloud from the LiDAR and a secondary backup method using the latest DenseMap estimated by the mapping algorithm. The primary method is more accurate and less computationally intensive, but for artifacts lying outside of the FOV of the LiDAR scan, it may not provide enough samples for accurate 3D position estimation, which is when the secondary method is employed. For each sampled point $\mathbf{s}_i \in \mathcal{S}$, its weight w_i is calculated. The position estimate \mathbf{d} and its corresponding uncertainty covariance matrix \mathbf{Q}_d are obtained as a weighted mean of the sampled points:

$$\mathbf{d} = \sum_{i=1}^{|\mathcal{S}|} \mathbf{s}_i w_i, \quad \mathbf{Q}_d = \frac{1}{1 - \sum_{i=1}^{|\mathcal{S}|} w_i^2} \sum_{i=1}^{|\mathcal{S}|} w_i (\mathbf{s}_i - \mathbf{d}) (\mathbf{s}_i - \mathbf{d})^T, \quad (14)$$

where \mathcal{S} is the set of sampled points and the weights w_i are normalized so that $\sum_{i=1}^{\mathcal{S}} w_i = 1$.

The weight of a point \mathbf{s} is obtained based on the distance of its reprojection to the image coordinates $\mathbf{s}' = [s_u, s_v]^T = f_{\text{proj}}(\mathbf{s})$ from the center of the detection's bounding box $\mathbf{c}_0 = [c_u, c_v]^T$ using the function

$$f_w(\mathbf{s}', \mathbf{c}_0) = \left(1 - \frac{2|s_u - c_u|}{w_{\text{bb}}}\right)^2 \left(1 - \frac{2|s_v - c_v|}{h_{\text{bb}}}\right)^2, \quad (15)$$

where $w_{\text{bb}}, h_{\text{bb}}$ are the width and height of the bounding box, respectively. The weighting function serves to suppress points further from the center of the bounding box. This is based on our empirical observation that the center provides the most reliable estimate of the detected object's position,

Algorithm 2. Algorithm for the estimation of a detection’s position and covariance.

```

1: Input:
2:    $\mathcal{D} = \{\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3, \mathbf{c}_4\}$ ,  $\mathbf{c}_i \in \mathbb{R}^2$                                 ▷ undistorted coordinates of the detection's bounding box
3:    $f_{\text{proj}} : \mathbb{R}^2 \rightarrow \mathcal{R}$                                         ▷ the projection model of the camera
4:    $\mathcal{P} = \{\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_{|\mathcal{P}|}\}$ ,  $\mathbf{p}_i \in \mathbb{R}^3$                     ▷ the latest point cloud from the LiDAR
5:    $f_{\text{raycast}} : \mathcal{R} \rightarrow \mathbb{R}^3$                                 ▷ the raycasting function of the occupancy map
6:    $n_{\text{desired}} \in \mathbb{N}$                                             ▷ the desired number of sampled points
7: Output:
8:    $\mathbf{d} \in \mathbb{R}^3$                                                     ▷ estimated position of the detection
9:    $\mathbf{Q}_{\mathbf{d}} \in \mathbb{R}^{3 \times 3}$                                         ▷ covariance matrix of the position estimate
10: Begin:
11: ▷ First, the desired number of points is sampled using the primary and secondary methods.
12:  $r_1 := f_{\text{proj}}^{-1}(\mathbf{c}_1)$ ,  $r_2 := f_{\text{proj}}^{-1}(\mathbf{c}_2)$ ,  $r_3 := f_{\text{proj}}^{-1}(\mathbf{c}_3)$ ,  $r_4 := f_{\text{proj}}^{-1}(\mathbf{c}_4)$  ▷ project the corners of the bounding box to 3D rays
13:  $\mathcal{S}_1 := \{\mathbf{p}_i \in \mathcal{P} \mid \mathbf{p}_i \text{ within the area defined by edges } r_1, r_2, r_3, r_4\}$  ▷ try to sample  $n_{\text{desired}}$  points using the primary method
14:  $n_{\text{remaining}} := \max(n_{\text{desired}} - |\mathcal{S}_1|, 0)$                                 ▷ calculate the remaining number of points to be sampled
15:  $\mathcal{S}_2 := \text{sampleRectangle}(\{\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3, \mathbf{c}_4\}, n_{\text{remaining}}, f_{\text{proj}}, f_{\text{raycast}})$  ▷ sample any remaining points from the occupancy map
16:  $\mathcal{S} := \mathcal{S}_1 \cup \mathcal{S}_2$                                             ▷ complement  $\mathcal{S}_1$  with the remaining points from  $\mathcal{S}_2$ 
17: ▷ Then, the weight of each sampled point is calculated using the weighting function  $f_w$ .
18:  $\mathbf{c}_0 := \text{mean}(\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3, \mathbf{c}_4)$                                 ▷ calculate the center of the bounding box
19: for each  $\mathbf{s}_i \in \mathcal{S}$  do
20:    $\mathbf{s}'_i := f_{\text{proj}}(\mathbf{s}_i)$                                         ▷ project the point back to the image frame  $\mathcal{I}$ 
21:    $w_i := f_w(\mathbf{s}'_i, \mathbf{c}_0)$                                     ▷ calculate its weight
22: ▷ Finally, the position and its uncertainty are calculated as a weighted mean and covariance and returned.
23:  $\mathbf{d} := \sum_{i=1}^{|\mathcal{S}|} \mathbf{s}_i w_i$ 
24:  $\mathbf{Q}_{\mathbf{d}} = \frac{1}{1 - \sum_{i=1}^{|\mathcal{S}|} w_i^2} \sum_{i=1}^{|\mathcal{S}|} w_i (\mathbf{s}_i - \mathbf{d})(\mathbf{s}_i - \mathbf{d})^T$ 
25: return  $\mathbf{d}$ ,  $\mathbf{Q}_{\mathbf{d}}$ 

```

while the bounding box’s corners typically correspond to the background and not the object, as illustrated in Figure 23a. The whole 3D position estimation algorithm is presented in Algorithm 2. The `sampleRectangle` routine used in Algorithm 2 is described in Algorithm 3.

The estimated positions and the corresponding covariance matrices serve as an input to the *artifact localization filter* described in the next section (refer to Figure 21). To avoid bias and numerical singularities in the filter, some special cases of the covariance calculation have to be handled. Namely, these are the following.

1. *All extracted points lie on a plane.* This happens, e.g. when all the cast rays of the secondary position estimation method intersect the same voxel of the DenseMap. The covariance matrix is then singular, which causes numerical problems with integrating the measurement.
2. *All extracted points are too close to each other.* This typically happens when the detected object is too far or too small. The covariance matrix’s eigenvalues are then too small, biasing the fused position estimate of the artifact.

To avoid these problems, the estimated covariance matrix is rescaled, so that all eigenvalues conform to a specified minimal threshold before being processed by the artifact localization filter.

10.3. Artifact localization filter

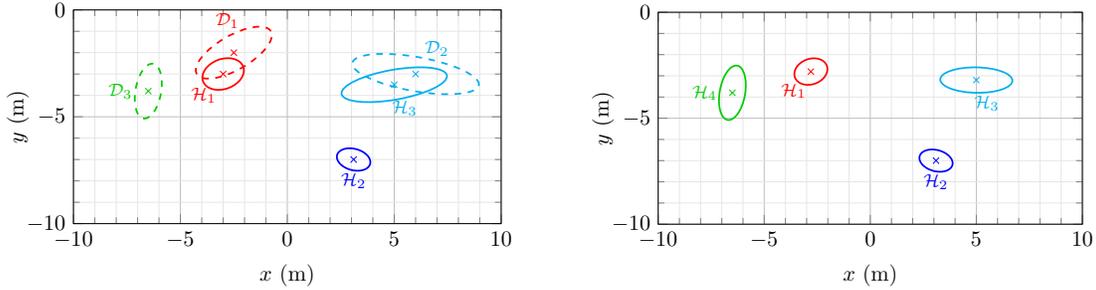
Artifact detections are filtered using an approach based on our previous work, where a multi-target tracking algorithm was employed for detection, localization, and tracking of micro aerial vehicles (Vrba et al., 2019). The filtering serves to improve the precision of the artifacts’ estimated positions and to reject false positives. Only artifacts that are consistently detected multiple times with sufficient confidence are confirmed, and only the confirmed artifacts are then reported to the operator to save the limited communication bandwidth. A single step of the algorithm is illustrated in Figure 24.

Algorithm 3. The `sampleRectangle` routine for sampling a number of 3D points from the occupancy map.

```

1: ▷ This routine samples points within a rectangle in the image plane  $\mathcal{I}$  by raycasting pixels on inscribed ellipses with
   an increasing radius.
2: Routine sampleRectangle:
3:   Input:
4:      $\{\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3, \mathbf{c}_4\}, \mathbf{c}_i \in \mathbb{R}^2$                                 ▷ corners of the rectangle to be sampled in the image frame  $\mathcal{I}$ 
5:      $n_{\text{remaining}} \in \mathbb{N}$                                           ▷ the desired number of samples
6:      $f_{\text{proj}} : \mathbb{R}^2 \rightarrow \mathcal{R}$                             ▷ the projection model of the camera
7:      $f_{\text{raycast}} : \mathcal{R} \rightarrow \mathbb{R}^3$                         ▷ the raycasting function of the occupancy map
8:   Output:
9:      $\mathcal{S} = \{\mathbf{s}_i\}$                                             ▷ a set of sampled points in the image frame  $\mathcal{I}$  such that  $|\mathcal{S}| \leq n_{\text{remaining}}$ 
10:  Parameters:
11:     $n_r \in \mathbb{N}, n_\alpha \in \mathbb{N}$  ▷ number of radial sampling steps and number of circumferential steps per unit circumference
12:  Begin:
13:     $w := c_{1,u} - c_{3,u}, h := c_{1,v} - c_{3,v}$                     ▷ calculate the width and height of the rectangle
14:     $r_{\text{step}} := 1/n_r$ 
15:    for  $r \in \{0, r_{\text{step}}, 2r_{\text{step}}, \dots, 1\}$  do
16:       $\alpha_{\text{step}} := r/n_\alpha$ 
17:       $\Delta\alpha := u, u \sim \mathcal{U}(-\pi, \pi)$                     ▷ generate a random angular offset to avoid biasing some directions
18:      for  $\alpha \in \{0, \alpha_{\text{step}}, 2\alpha_{\text{step}}, \dots, 2\pi\}$  do
19:         $\mathbf{s}' := [wr \cos(\alpha + \Delta\alpha)/2, hr \sin(\alpha + \Delta\alpha)/2]^\top$  ▷ calculate a sample point on an ellipse
20:         $r := f_{\text{proj}}(\mathbf{s}')$                                 ▷ project the point to a 3D ray
21:         $\mathcal{S} := \mathcal{S} \cup f_{\text{raycast}}(r)$                     ▷ find an intersection of the ray with an obstacle and add it to  $\mathcal{S}$ 
22:        if  $|\mathcal{S}| = n_{\text{remaining}}$  then
23:          return  $\mathcal{S}$ 
24:    return  $\mathcal{S}$ 

```



(a) Situation before the update step. The detections \mathcal{D}_1 and \mathcal{D}_2 are associated to the hypotheses \mathcal{H}_1 and \mathcal{H}_3 , respectively. The detection \mathcal{D}_3 is not associated to any hypothesis. The hypothesis \mathcal{H}_2 has no detection associated.

(b) Situation after the update step. The detections \mathcal{D}_1 and \mathcal{D}_2 updated the hypotheses \mathcal{H}_1 and \mathcal{H}_3 , respectively. The detection \mathcal{D}_3 initialized a new hypothesis \mathcal{H}_4 and the hypothesis \mathcal{H}_2 remained unchanged.

Figure 24. Illustration of one step of the artifact localization filter (a top-down view). Hypotheses \mathcal{H}_i are shown as covariance ellipsoids with the mean $\hat{\mathbf{x}}_i$, marked by an “x” symbol. Detections \mathcal{D}_i are represented in the same way using dashed lines. Associations between hypotheses and detections are highlighted using color.

The filter keeps a set of hypotheses about objects in the environment. Each hypothesis \mathcal{H} is represented by an estimate of the object’s position $\hat{\mathbf{x}}$, its corresponding covariance matrix \mathbf{P} , and a probability distribution of the object’s class $p_{\mathcal{H}} : \mathcal{C} \rightarrow [0, 1]$, where \mathcal{C} is the set of considered classes. For every hypothesis \mathcal{H} , up to one detection $\mathcal{D}_{\mathcal{H}}$ is associated according to the rule

$$\mathcal{D}_{\mathcal{H}} = \begin{cases} \operatorname{argmax}_{\mathcal{D}} l(\mathcal{D} | \mathcal{H}), & \text{if } \max_{\mathcal{D}} l(\mathcal{D} | \mathcal{H}) > l_{\text{thr}}, \\ \emptyset, & \text{else,} \end{cases} \quad (16)$$

where $l(\mathcal{D} | \mathcal{H})$ is the likelihood of observing \mathcal{D} given that it corresponds to \mathcal{H} , and l_{thr} is a likelihood threshold. The associated detections are used to update the corresponding hypotheses. The detections that are not associated initialize new hypotheses.

The position estimate $\hat{\mathbf{x}}$ of a hypothesis \mathcal{H} and its covariance \mathbf{P} are updated using the Kalman filter's update equation and an associated detection $\mathcal{D}_{\mathcal{H}}$ at time step t as

$$\mathbf{K}_{[t]} = \mathbf{P}_{[t]} \mathbf{H}^{\top} (\mathbf{H} \mathbf{P}_{[t]} \mathbf{H}^{\top} + \mathbf{Q}_{\mathbf{d}[t]})^{-1}, \quad (17)$$

$$\hat{\mathbf{x}}_{[t+1]} = \hat{\mathbf{x}}_{[t]} + \mathbf{K}_{[t]} (\mathbf{d}_{[t]} - \mathbf{H} \hat{\mathbf{x}}_{[t]}), \quad (18)$$

$$\mathbf{P}_{[t+1]} = (\mathbf{I} - \mathbf{K}_{[t]} \mathbf{H}) \mathbf{P}_{[t]}, \quad (19)$$

where $\mathbf{K}_{[t]}$ is a Kalman gain, \mathbf{I} is an identity matrix, \mathbf{H} is an observation matrix (in our case, equal to \mathbf{I}), $\mathbf{d}_{[t]}$ and $\mathbf{Q}_{\mathbf{d}[t]}$ are the estimated position of $\mathcal{D}_{\mathcal{H}[t]}$ and its corresponding covariance matrix, respectively. The class probability distribution $p_{\mathcal{H}}$ is updated as

$$p_{\mathcal{H}[t+1]}(c) = \frac{n_{\text{dets}[t]} p_{\mathcal{H}[t]}(c) + p_{\mathcal{D}_{\mathcal{H}[t]}}(c)}{n_{\text{dets}[t]} + 1}, \quad (20)$$

where $c \in \mathcal{C}$ is an object's class and $n_{\text{dets}[t]}$ is the number of detections, associated to \mathcal{H} thus far.

Because the artifacts are assumed to be immobile, the Kalman filter's prediction step is not performed, which has the effect that the uncertainty of a hypothesis (represented by \mathbf{P}) can decrease without bounds. This can cause the likelihood $l(\mathcal{D} | \mathcal{H})$ of new measurements corresponding to the same object to be below the association threshold, breaking the association algorithm. To avoid this, the covariance matrix \mathbf{P} is rescaled after each update so that its eigenvalues are larger than a specified minimal value, which enforces a lower bound on the position uncertainty of the hypotheses.

10.3.1. Association likelihood

To calculate the likelihood $l(\mathcal{D}_{[t]} | \mathcal{H}_{[t]})$ of observing a detection $\mathcal{D} \equiv \{\mathbf{d}, \mathbf{Q}_{\mathbf{d}}\}$ given that it corresponds to a hypothesis $\mathcal{H} = \{\hat{\mathbf{x}}, \mathbf{P}\}$ at time step t , we use a measurement model

$$\mathbf{d}_{[t]} = \mathbf{H} \mathbf{x} + \xi_{[t]}, \quad \xi_{[t]} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_{\mathbf{d}[t]}), \quad (21)$$

where \mathbf{H} is the observation matrix, \mathbf{x} is a hidden state (the real position of the artifact), $\xi_{[t]}$ is measurement noise, and $\mathcal{N}(\mathbf{0}, \mathbf{Q}_{\mathbf{d}[t]})$ denotes the Gaussian probability distribution with zero mean and covariance matrix $\mathbf{Q}_{\mathbf{d}[t]}$. Using this model, the probability density function of the expected measurement given \mathbf{x} is

$$p(\mathbf{d}_{[t]} | \mathbf{x}) = f(\mathbf{d}_{[t]} | \mathbf{H} \mathbf{x}, \mathbf{Q}_{\mathbf{d}[t]}), \quad (22)$$

where $f(\cdot | \mu, \Sigma)$ denotes the density function of the Gaussian distribution with mean μ and covariance matrix Σ .

The Kalman filter described by equations (17) to (19) can be interpreted as an estimator of the probability density of the hidden state given previous measurements. This probability density is represented as a random variable with a Gaussian distribution:

$$p(\mathbf{x} | \mathbf{d}_{[1]}, \dots, \mathbf{d}_{[t]}) = f(\mathbf{x} | \hat{\mathbf{x}}_{[t]}, \mathbf{P}_{[t]}). \quad (23)$$

The likelihood $l(\mathbf{d}_{[t]})$ of observing a new measurement $\mathbf{d}_{[t]}$ given previous measurements $\mathbf{d}_{[1]}, \dots, \mathbf{d}_{[t-1]}$ is the value of a probability density function $p(\mathbf{d} | \mathbf{d}_{[1]}, \dots, \mathbf{d}_{[t-1]})$ at $\mathbf{d}_{[t]}$. By combining equations (21) and (23), the likelihood may be expressed as

$$\begin{aligned} l(\mathbf{d}_{[t]}) &= p(\mathbf{d}_{[t]} | \mathbf{d}_{[1]}, \dots, \mathbf{d}_{[t-1]}) = \int p(\mathbf{d}_{[t]} | \mathbf{x}) p(\mathbf{x} | \mathbf{d}_{[1]}, \dots, \mathbf{d}_{[t-1]}) d\mathbf{x} \\ &= \int f(\mathbf{d}_{[t]} | \mathbf{H} \mathbf{x}, \mathbf{Q}_{\mathbf{d}[t]}) f(\mathbf{x} | \hat{\mathbf{x}}_{[t-1]}, \mathbf{P}_{[t-1]}) d\mathbf{x} \\ &= f(\mathbf{d}_{[t]} | \mathbf{H} \hat{\mathbf{x}}_{[t-1]}, \mathbf{Q}_{\mathbf{d}[t]} + \mathbf{H} \mathbf{P}_{[t-1]} \mathbf{H}^{\top}), \end{aligned} \quad (24)$$

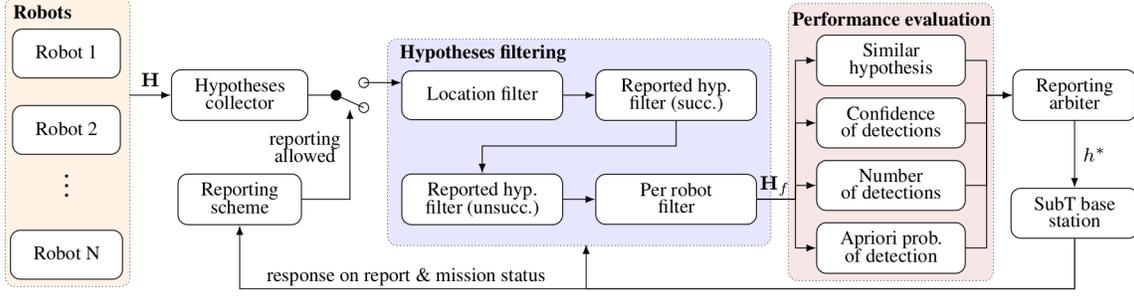


Figure 25. Illustration of the automatic reporting process from the Virtual Track.

which is the value of the probability density function of a Gaussian distribution with mean $\mathbf{H}\hat{\mathbf{x}}_{[t-1]}$ and covariance $\mathbf{Q}_{\mathbf{d}[t]} + \mathbf{H}\mathbf{P}_{[t-1]}\mathbf{H}^\top$ at $\mathbf{d}_{[t]}$. This expression is used to determine the detection-to-hypothesis association at each step according to equation (16), as described in the previous section.

10.4. Arbiter for artifact reporting

In contrast to the system part of the competition, the Virtual Track requires substituting the human operator with an autonomous arbiter for artifact reporting. The main functionality of the autonomous base station resides in collecting the hypotheses from the robots and reporting the location of artifacts. The number of reports in each run is limited and usually lower than the number of hypotheses collected from all robots. Therefore, a subset of hypotheses needs to be chosen so that the expected score is maximized. The implemented reporting strategy is based on filtering the collected hypotheses by considering their location and artifact type, followed by evaluating the performance index of particular hypotheses. The entire workflow is illustrated in Figure 25.

The autonomous base station collects the hypotheses from individual robots throughout the entire run. The predefined reporting scheme specifies the maximum allowed number of reports at particular time instants of the mission. Most of the reports are saved to the last minutes of the mission when the base station holds most of the information collected from the robots. However, some reports are allowed sooner during the mission to tackle the problem of unreliable communication and prevent a failure to report all hypotheses before the time limit exceeds. When the reporting scheme allows for submitting a report, the collected hypotheses are processed to obtain the best available hypothesis h^* in a set of all collected hypotheses \mathbf{H} . First, the hypotheses are filtered using information about previous reports, their validity, location, and per robot limits on the number of reports and minimum success rate. The final set of filtered hypotheses is obtained as

$$\mathbf{H}_f = \mathbf{H} \setminus \{\mathbf{H}_{\text{area}} \cup \mathbf{H}_{\text{succ}} \cup \mathbf{H}_{\text{unsucc}} \cup \mathbf{H}_r\}, \quad (25)$$

where \mathbf{H}_{area} stands for the hypotheses located outside of the competition course, \mathbf{H}_{succ} stands for hypotheses in the vicinity of the successful reports of the same artifact class, $\mathbf{H}_{\text{unsucc}}$ contain hypotheses in the vicinity of the unsuccessful reports of the same artifact class, and \mathbf{H}_r represents the hypotheses of robots that have exceeded their own limit on reports and concurrently have a low success rate of their submitted hypotheses. The performance index for a hypothesis h_i is computed as

$$P(h_i) = \alpha p_r + \beta p_c + \gamma p_n + \delta p_a, \quad (26)$$

where the values p_r, p_c, p_n, p_a represent the percentile of particular performance indices of hypothesis h_i among all hypotheses in \mathbf{H}_f , and $\alpha, \beta, \gamma, \delta$ are the weight coefficients. The particular performance indices are related to the number of robots with a similar hypothesis (p_r), the overall confidence of the detections assigned to the hypothesis (p_c), the number of detections assigned to the hypothesis (p_n), and the apriori probability of detection of a particular object (p_a). The next hypothesis to be

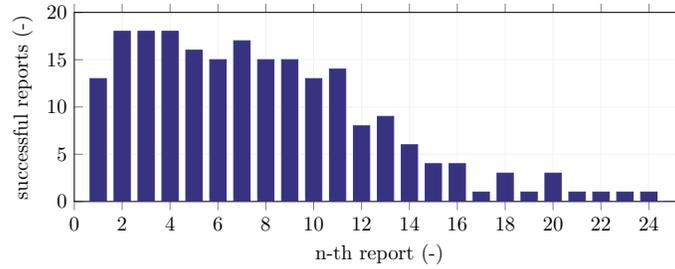


Figure 26. The distribution of successful reports over particular reporting attempts during all runs of the SubT Virtual Track Prize Round. The lower success rate of the first attempt in comparison to later attempts is caused by the early time of the first report, which was allowed 100 s after the start of the mission. By this time, only a single UAV had already entered the course, and thus the number of available hypotheses to choose from was low.

reported h^* is chosen based on the following equation:

$$h^* = \arg \max_{h_i \in \mathbf{H}_f} P(h_i). \quad (27)$$

The distribution of successful reports over particular reporting attempts during all runs of the SubT Virtual Track Prize Round is shown in Figure 26. In the Systems Track, the autonomous arbiter was not used as the decision-making of the human operator regarding which hypotheses to report was superior to the autonomous arbiter, which operated based on a fixed set of rules.

11. Mission control

The proposed system is designed for fully autonomous operation, so that the rescue team can benefit from the autonomous reconnaissance of the UAV without the need for any additional personnel operating the UAV. The DARPA SubT competition reflects this requirement on autonomy by allowing only robots without human operators to enter the course. In theory, the robots could be teleoperated (Moniruzzaman et al., 2022). However, this is not scalable with the number of robots. Moreover, for teleoperation, a reliable communication link between the robot and the operator is required, but is often not available, especially deeper in the subterranean environment where impenetrable walls diminish signal propagation. Thus the correct execution of an autonomous mission relies on a state machine that governs the high-level actions of the UAV.

11.1. State machine

The state machine applied in the SubT System Finals consists of 12 fundamental states. In the first state, the status of components that are vital to the mission is checked to ensure that the mission will be accomplished. Both the software components (*localization, mapping, planning, artifact detection, artifact localization, database*) and hardware components (*LiDAR, RGB cameras, depth cameras, mobilicom unit*) are checked prior to the mission. This component health check is crucial as, while still in the staging area, any potential component failures can be addressed, but it is not possible when the UAV is already flying.

When all components are running correctly, the UAV enables the output of the reference controller, transits to *WAITING FOR TAKEOFF* state, and waits for approval from the safety operator to start the mission. The approval required to guarantee the safety of the personnel moving in the vicinity of the UAV is given by arming the UAV and transferring the control of the UAV fully to the onboard computer by toggling the Radio Controller (RC) switch. After the approval to start, the UAV waits for a specified safety timeout in the *READY FOR TAKEOFF* state while signaling the imminent takeoff by flashing LEDs. In this state, the approval can be taken back by the safety operator. After the timeout elapsed, the *PERFORMING TAKEOFF* state is entered, during which the UAV ascends until reaching the desired takeoff height.

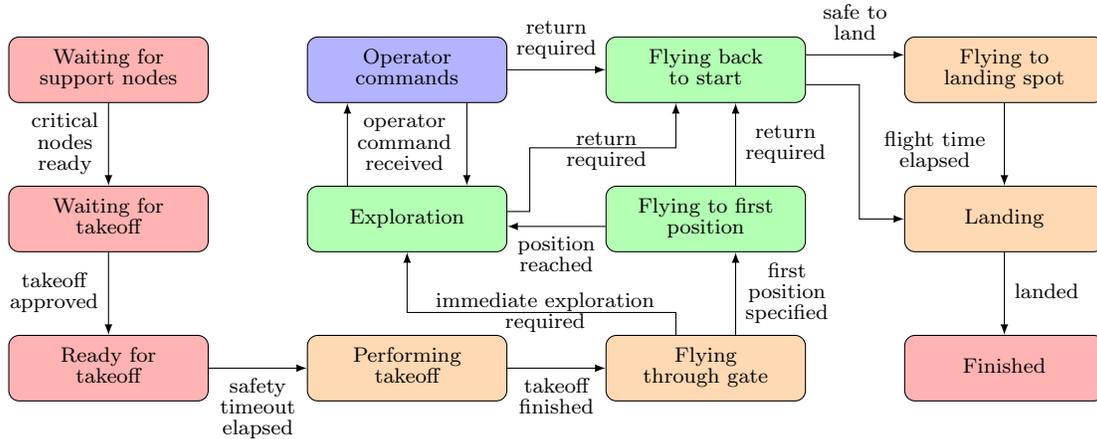


Figure 27. Simplified version of the state machine governing the autonomous mission in SubT Systems Track.

In the next state (*FLYING THROUGH GATE*), the UAV is navigated to a position inside the area to be explored. Once this position is reached, the space behind the UAV is virtually closed to prevent flight back towards the rescue personnel. If the rescuers have some prior knowledge about the environment, e.g., they see a door to which they want to send the UAV, they can optionally specify this first position to steer the UAV in that direction. After reaching this first position or if the flight to the first position is not requested, the UAV enters the *EXPLORATION* state. In this state, the UAV fulfills the primary mission goals until the upper bound of the estimated time to return is equal to the remaining flight time. Then the UAV initiates returning to the takeoff position in the state *FLYING BACK TO START*.

The return position is the takeoff position by default, but the operator can request any other position (e.g., to serve as a communication retranslation node) to which the UAV tries to return. After the position is reached, the UAV flies to the nearest safe landing spot as described in Section 5.3, and the *LANDING* state is entered. The landing is also triggered when the flight time is elapsed during the *FLYING BACK TO START* or *FLYING TO LANDING SPOT* states. When the UAV lands, it enters the *FINISHED* state, in which it turns off the motors, Light-Emitting Diodes (LEDs), LiDAR, and other components except the communication modules to conserve battery power for retranslating communications.

The required communication between the UAV and its operator during the start of the mission is limited to signals provided by the RC and visual signals provided by flashing LEDs. This enables very fast deployment of the UAV that automatically starts all necessary software components once the onboard computer is powered on and provides the information about being prepared to start by a single long flash of LEDs. After that, the operator can approve the mission by the remote controller without the need for any additional communication or commanding of the UAV. Following this automated procedure, the UAVs are prepared to start one minute after the battery is plugged in.

A *FAULT* state (not shown in the simplified diagram in Figure 27) can be entered from all states in which the UAV is in the air (all states except the ones colored red in Figure 27). The *FAULT* state is entered only when it is detected that the mission cannot continue safely. In such a case, a controlled emergency landing is initiated if a position estimate is available. When a position estimate cannot be provided the emergency landing escalates into the failsafe landing, during which the UAV gradually lowers its thrust, while maintaining zero tilt. After contact with the ground is detected, the motors are turned off and the UAV is disarmed. The *FAULT* state is final, i.e., the mission cannot continue due to the failures, which triggered the transition into this state. The conditions for entering the *FAULT* state are the following.

- Data from a sensor critical for localization are not available for 1 s. This situation can happen in case of a hardware failure, detached cable, power supply failure, sensor driver bug, etc.

- The control error exceeds 2 m, which can occur with a diverging state estimate, overloaded CPU, or insufficient thrust.
- The state estimate is not available for 0.1 s, which can be caused by a bug in the state estimation module or an overloaded CPU.
- The innovation of the state estimation exceeds 2 m. Innovation is the difference between current state and a correction coming from a localization algorithm. Large innovation indicates a discrete step in the localization algorithm.
- A maximum thrust threshold of 80 % is exceeded for 1 s. This condition is triggered when a discharged battery cannot provide enough current to perform the desired motion. A faulty or older battery with many discharge cycles might struggle to provide sufficient current sooner than is the expected flight time.
- A tilt over 75° is detected, which can happen if a discrete step appears in the state estimate or when the UAV collides with an obstacle.

The state machine applied in the Virtual Track of the SubT Challenge differs only in a few states given by the specifics of the simulation environment. First, it does not contain the operator commands states that are not available in a virtual environment. Second, it contains two additional states, *BACKTRACKING* and *AVOIDING COLLISIONS*. The *BACKTRACKING* state is entered when the UAV is stuck in a fog and tries to escape from it by backtracking to the most recent collision-free poses, ignoring the occupied cells in the current map (see [Section 5.2.1](#) for details). In the *AVOIDING COLLISIONS* state, the UAV is avoiding collision with the UAVs of higher priority by stopping the lateral motion and decreasing its altitude. We have decided against using collision avoidance in the Systems Track due to the low probability of collision, and high probability of deadlocks in narrow corridors.

11.2. Operator commands

While the UAV is capable of carrying out the mission on its own in the fully autonomous mode, the operator can intervene by issuing an operator command to influence the behavior of the UAV. All operator commands can be activated only in the *EXPLORATION* state and in the operator command states, in which the UAV performs its primary goal. Allowing operator commands in other states would interfere with the takeoff, returning, and landing processes. The commands are transmitted from the operator’s base station to the UAV through the available communication modalities described in [Section 11.4](#). The following commands are available for the operator:

- **Explore to position.** The operator can bias the automatic goal selection process by issuing the *Explore to position* command. After the command is received by the UAV, the currently used reward function for evaluating viewpoints is extended by a term that penalizes the Euclidean distance of the viewpoint from the desired position \mathbf{p}_D . The term added to the reward function for a viewpoint ξ is simply

$$\Delta R(\xi_{\text{UAV}}, \xi, \mathbf{p}_D) = -c_{oc} |\mathbf{p}_\xi - \mathbf{p}_D|. \quad (28)$$

Such modification of the reward function causes the viewpoints closer to the desired positions to be preferred over farther viewpoints. The assertiveness of reaching the desired position can be controlled by the coefficient c_{oc} . If this is set too high, it might force the viewpoints with a minimal distance from obstacles and low information value to be selected.

- **Plan to position.** The *Plan to position* command bypasses the viewpoint selection process and requests the planner to find a path directly to the specified position. When the requested position is not reachable, i.e., it is in an occupied or unknown space, the planner will find the path to the closest point using the Euclidean distance heuristic function. Thus this command should be used primarily for reaching an already visited position, e.g., to land there and retranslate communication from robots that are already further in the environment, or to approach a stuck robot to retrieve its data.

- **Set return position.** Under normal operation, the UAV returns to the staging area when its battery is depleted. The operator can change the return position by issuing the *Set return position* command. This can save valuable flight time of the UAV when a communication chain is already established.
- **Stop.** The operator can also halt the movement of the UAV by issuing the *Stop* command. This command is useful when the operator wants to inspect an interesting area in more detail, prevent the UAV from going into a noninformative or dangerous area, or temporarily retranslate communications. Moreover, this command is a prerequisite for calling the *Land* command.
- **Land.** It is possible to land the UAV prematurely before the end of the mission by issuing the *Land* command. The expected use case involves landing the UAV at a position advantageous for extending the communication network. Before calling the *Land* command, the *Stop* command must be called to prevent an accidental landing at an incorrect location, due to the arbitrary delay of the command sent through an unreliable network. The system does not guarantee landing at the exact specified position, as a safe landing spot is found in the vicinity of the requested position.
- **Return home.** The *Return home* command switches the UAV to the *returning* state, as defined in [Section 8.2](#). In this state, the UAV uses the navigation module to get as close as possible to the specified return position.
- **Resume autonomy.** The last operator command cancels the behavior that was forced by previous operator commands (except *Land* and *Set return position*). This causes the UAV to resume autonomous exploration, start its return, or land (depending on the flight time left).

11.3. Operator interface

Only a single human (operator) could view the mission-specific data sent by the robots to the base station. His main task was to analyze the artifact hypotheses and report ones that seemed correct to the DARPA server to score points. He could also influence the behavior of the robots by issuing high-level operator commands ([Section 11.2](#)).

To facilitate his responsibility, each of the two tasks has a dedicated interface. Commands are issued from the RViz-based interface with each command mapped to a unique keyboard shortcut. The operator also often used live camera streams from the robots to get contextual information about the environment. This information was essential for deciding where each robot should be sent (e.g., quadrupeds to urban sections) and also for quick assessment of why a robot could be stuck.

The second interface for artifact hypotheses management is also based on RViz with a custom rqt plugin for viewing the details of each hypothesis including the image, number of detections, class probabilities, and position. These properties help the operator decide whether to send the hypothesis to the DARPA scoring server or decline it. Manual refinement of hypotheses poses is also possible by dragging them on the map.

The GUI was displayed on a semi-mobile workstation with 3 integrated displays and one external monitor standing on top of the workstation. The arrangement of the 4-displays is shown in [Figure 28](#). From our experience, the more the operator sees without keyboard and mouse interaction, the better for his performance.

Apart from the human operator who could view all mission data, the rules also allowed the other staging area personnel to view status data. We have thus set up a diagnostics console on a computer outside the staging area. This console showed useful diagnostics information that could be relayed via voice to the human operator.

11.4. Communication

The developed system assumes an unreliable bidirectional low-bandwidth communication network with intermittent dropouts. It should be mentioned that two meshing-capable wireless technologies are used on the hardware level—2.3 GHz Mobilicom and 868 or 915 MHz motes, with details of

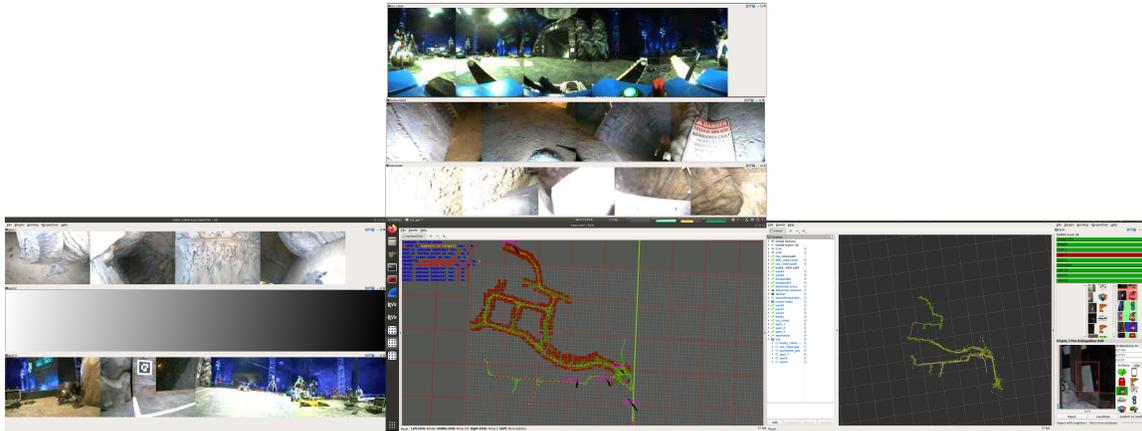


Figure 28. The operator interface display arrangement. Top screen shows live views of cameras from 3 UGVs. In bottom row, left to right, are screens with another 3 live UGV streams, the control GUI and artifact reporting GUI. The bottom left screen could also show a terminal window that was used for direct execution of scripts on the robots (as a fallback for a failure case that was not handled by the control GUI). [Figure 2](#) illustrates the physical look of the setup of the workstation with 4 displays.

both available in (Roucek et al., 2020). The notes are also dropped by UGVs as deployable range-extending battery-powered modules (Bayer and Faigl, 2020) similarly to (Ginting et al., 2021; Saboia et al., 2022) to build a communication mesh network. Our custom-made notes have lower bandwidth (100 B s^{-1}) than (Ginting et al., 2021; Saboia et al., 2022), which is compensated by sending only necessary compressed data. Moreover, bandwidth-intensive data are sent through a 1 MB s^{-1} Mobilicom network. This multimodal communication approach is robust to the failure of either Mobilicom or notes as both are able to transfer mission-critical data.

This paper focuses on high-level usage of the communication network, which is used as a black box, and as such the low-level layers of the communication protocol are not discussed.

The developed system benefits from available connections to other agents and the base station in multiple ways. First, when a robot detects an artifact, the detection with its estimated position is shared over the network instead of returning physically to the base station, thus saving time valuable for the success of the mission. Second, the individual agents can share the information about the already explored volume in the form of a topological-volumetric map (LTVMap) introduced in [Section 7.4](#). The knowledge of other agents’ topological-volumetric maps penalized regions already explored by other robots, which encourages splitting of the robot team and covering a larger volume over the same time period as shown in [Figure 29](#). Third, each robot shares its position with the base station, so that the operator has an overview of where all robots are located. The operator can then influence the future behavior of any robot in the communication range by sending an operator command ([Section 11.2](#)). Last, positions of the communication nodes (breadcrumbs or landed UAVs), which form the communication network shown in [Figure 30](#), are sent to be used for returning to the communication range when the remaining flight time is low.

11.5. Calibrating global reference frame

The entire navigation system of heterogeneous robots within the CTU-CRAS-NORLAB team is decentralized under the assumption of a shared coordinate frame—the world coordinate frame O_W . To obtain the transformation of a robot’s local origin within the world frame, the staging area of the competition environment provides a set of visual tags and a set of reflective markers, both with precisely known poses within the world (see the markers mounted on the entrance to the environment in [Figure 31](#)). The reflective markers are used within our 6-DOF calibration procedure in which a Leica TS16 total station is employed to measure 3D points with sub-millimeter accuracy. The origin

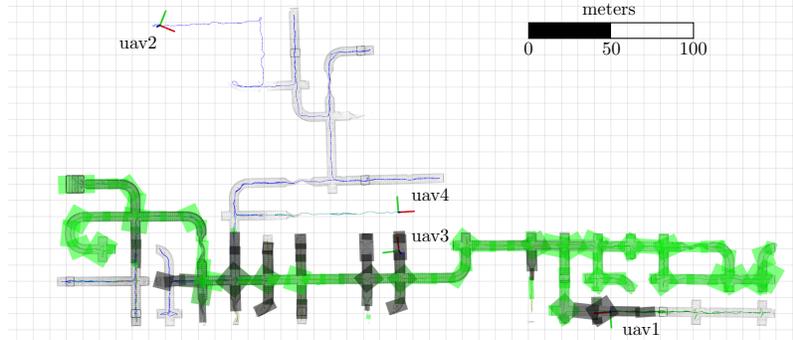


Figure 29. Example of the dispersed exploration of a tunnel system during the first run in world 1 of the virtual track. Only LTVMap from UAV1 is shown for clarity, other UAVs received this map and maps from the other UAVs. Instead of exploring again the same places as UAV1, both UAV2 and UAV4 explore previously unvisited corridors. Dark parts of LTVMap in this figure are not yet fully explored, so UAV3 flies to inspect these areas to not miss any potentially hidden artifacts.

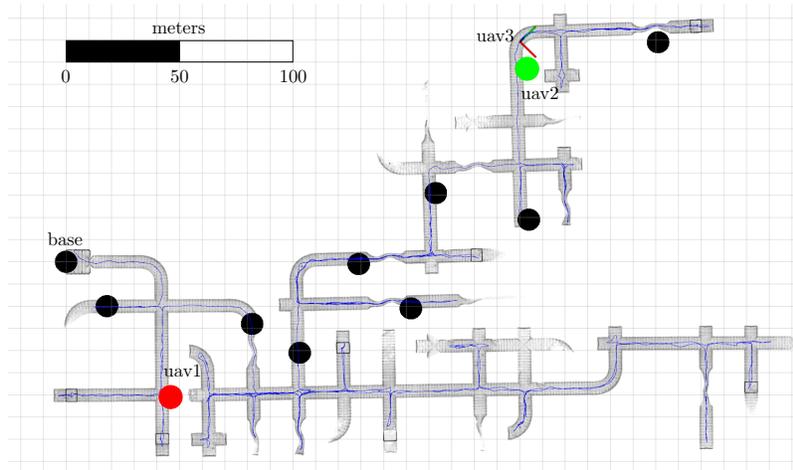


Figure 30. A communication network consisting of a base station and 8 breadcrumbs (black) deployed by the UGVs and 2 UAVs from the 3rd run in world 1 of the virtual track. UAV3 with its trajectory shown in blue could explore further thanks to the deployed communication nodes. Without the communication network, the UAV would have to return to the staging area, thus traveling additional 500 m from its final landing position.

\mathbf{T}_{TS}^W of the total station in the world is derived from measuring known in-world marker poses and used in deriving \mathbf{T}_B^W of a robot B .

To calibrate the pose of a single robot B after \mathbf{T}_{TS}^W is known, 4 known points on the robot's frame need to be measured, used in estimating \mathbf{T}_B^W , and sent to the information database (see Section 11.4) or directly to the robot. As the number of robots in the CTU-CRAS-NORLAB team deployments reached up to 9 robots per run (see Figure 31), the overhead for robots-to-world calibration decelerated the rate of robot deployments as well as limited the possibilities for quick in-situ decision-making. To speed up the calibration pipeline for UAVs with limited flight distance (and hence with greater room for calibration errors), just a single UAV A needs to be calibrated with the total station wherein the initial pose of the remaining UAVs B is estimated from on-board LiDAR data. The known transformation \mathbf{T}_A^W and pre-takeoff LiDAR data \mathbf{D}_A of a robot A are shared throughout the robots and used to estimate \mathbf{T}_B^W . The transformation \mathbf{T}_B^A is estimated by registering source LiDAR data \mathbf{D}_B onto target data \mathbf{D}_A using Iterative Closest Point (ICP) with extremely tight constraints in matching the rotation component of the transformation. The tight rotation

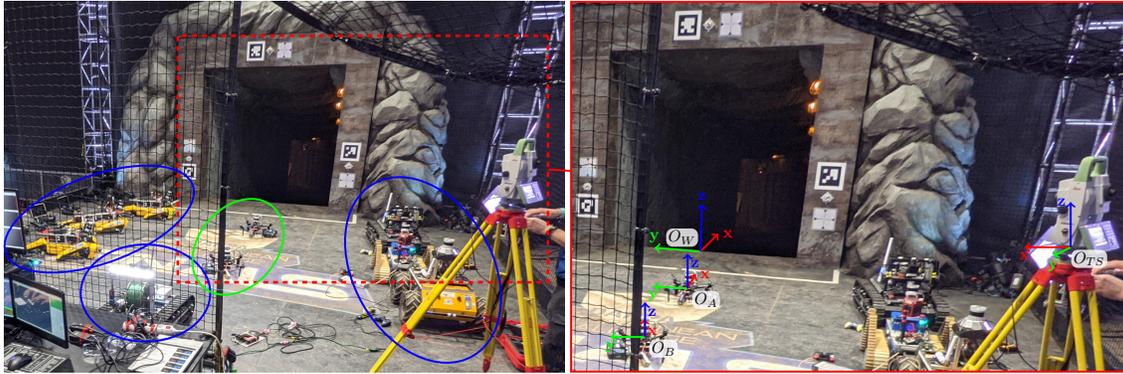


Figure 31. Example robot distribution (7 UGV robots in blue, 2 UAV robots in green) of team CTU-CRAS-NORLAB within the staging area of Systems Track environment of DARPA SubT Challenge, 2021. The Right figure highlights the reference frames of interest—the world origin O_W together with the origin of the Leica total station O_{TS} used for calibrating local robot origins O_A and O_B within the world.

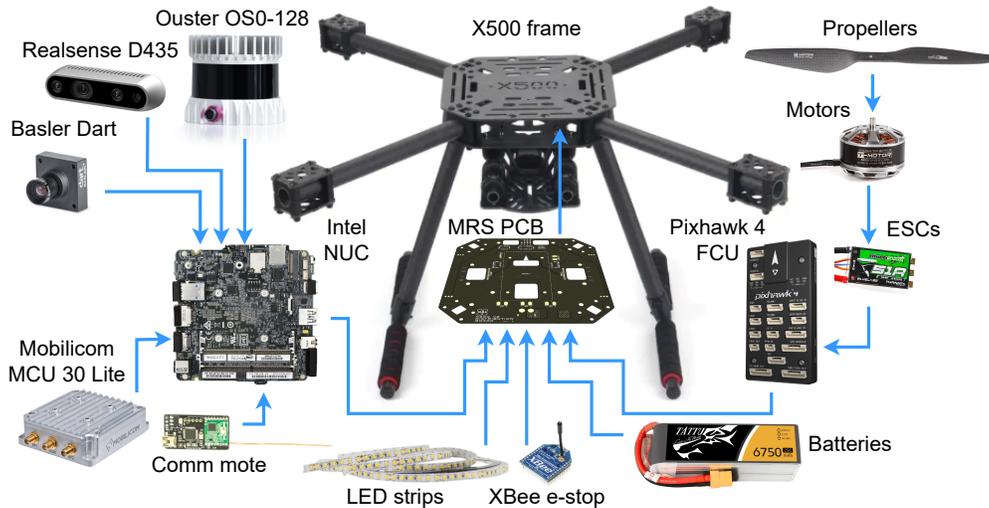


Figure 32. The interconnection of hardware components that were used is the Systems Track.

constraints are important as frame-orientation misalignments are the largest source of absolute error during deep deployments. The pose of robot B in the world is then given by $\mathbf{T}_B^W = \mathbf{T}_B^A \mathbf{T}_A^W$.

12. Hardware platform

The components of our S&R UAV were carefully selected to optimize the flight time and perception capabilities based on years of experience with building aerial robots for research (Ahmad et al., 2021), competitions (Walter et al., 2022), inspection (Silano et al., 2021), documentation (Kratky et al., 2021b) and aerial filming (Kratky et al., 2021). All platforms we have designed for diverse tasks and purposes including DARPA SubT are presented in (Hert et al., 2022).

Our platform is built upon the Holybro X500 quadrotor frame. The 500 mm frame is made entirely of carbon fiber, therefore it is stiff and light. Moreover, the arm length can be changed to accommodate different propellers. A description of all components that are mounted on the UAV frame follows. The connections of the components are depicted in Figure 32. Our team designed and manufactured a custom Printed Circuit Board (PCB) that replaced the top board of the X500 frame. This PCB (see Figure 34) supplies battery power to individual Electronic Speed Controllers (ESCs),

integrates several independent power supplies and provides a communication interface among the main computer, the Pixhawk flight controller, and MRS Modules. MRS Modules are small expansion boards that provide additional functionality and have a standardized electrical and mechanical interface. The UAV can be configured with different MRS Modules depending on the required capabilities. The PCB is connected to the main flight 4S lithium-polymer battery, which provides 14.0 V–16.8 V depending on the state of charge. The battery voltage is used to directly power the ESCs and the Intel NUC main computer. The board then integrates three independent 5 V/3 A buck converters, two to provide redundant power for the Pixhawk flight controller and one to power the MRS Modules. One 24 V/2 A boost converter is used to power the Ouster OS0-128 3D LiDAR scanner. The board has two slots for MRS Modules, one is used to control 12 V LED strips which provide illumination for the onboard RGB cameras. The second module is an interface for an XBee radio module, used as an e-stop receiver. Communication among the main computer, Pixhawk, and MRS Modules is provided by FT4232 Quad USB-UART bridge, which is integrated into the PCB. We selected MN3510 KV700 motors from T-motor and paired them with 13-inch carbon fiber propellers for large payload capacity and propulsion efficiency. The motors are driven by Turnigy Bl-Heli32 51A ESCs, as they are lightweight and easily configurable.

The 3D LiDAR was upgraded to the OS0-128 model, which features 128 scanning lines and wide 90° vertical field of view, which allows for perceiving the surroundings of the UAV in the challenging underground environments. Despite the wide coverage of the LiDAR sensor, there are still blind spots above and below the UAV when mounted horizontally. To cover these spots, we use two Intel Realsense D435 RGBD cameras, facing up and down. This enables the UAV to fly directly upwards, even in cluttered vertical shafts, without risking collision. Both of the RGBD cameras are also used for mapping and artifact detection. Additionally, the bottom facing RGBD camera is used for landing site detection. The platform is equipped with two (left and right) dedicated artifact detection cameras, the Basler Dart daA1600 with 97° horizontal FOV lens, and sufficient lighting provided by LED strips. All algorithms run on the onboard Intel NUC i7-10710U CPU with 6 physical cores and the detection CNN utilizes the integrated Intel UHD GPU.

The high-power Mobilicom MCU-30 Lite wireless communication module provides long-range connection between robots and the base station while keeping low weight of 168 g. In some topologically complex areas, even the high-power Mobilicom cannot assure reliable connection between the units, so it is supported by smaller communication motes, which are also dropped as breadcrumbs by the UGVs to improve the signal range. These motes are compact communication modules based on the RFM69HCW transceiver working at 868 MHz or 915 MHz with 100 mW transmission power and 100 B s⁻¹ data bandwidth. The performance of the motes was analyzed in the Bull Rock cave (Bayer and Faigl, 2020) and first deployed at the Urban Circuit (Roucek et al., 2020). The WiFi unit of the onboard Intel NUC computer was not used for any communication.

Finally, the large payload capacity of the UAV allowed us to extend the flight time by using a larger battery. We used two 4S 6750 mA h Li-Po batteries in parallel. Instead of a larger battery, two smaller batteries were used due to the 100 Wh limit for aircraft transportation. This gave the UAV a flight time of 25 min with a total mass of 3.3 kg.

The X500 platform (Figure 33) is capable of flying in dense indoor environments, even in tight vertical shafts, while being able to localize itself with the required accuracy. It has four different cameras for artifact detection, is able to communicate and form mesh networks with other robots, and possesses a long flight time.

Furthermore, this platform was also replicated in the virtual competition with the same parameters as the physical counterpart. All of the teams except for two used the X500 platforms in the Virtual Track due to its long flight time, substantial sensor suit, and agile dynamics.

13. Technical details of hardware deployment

With a few exceptions, the components of the UAV software stack deployed in the Virtual and Systems tracks are equal, yet the available processing powers are not. The Virtual Track yields a

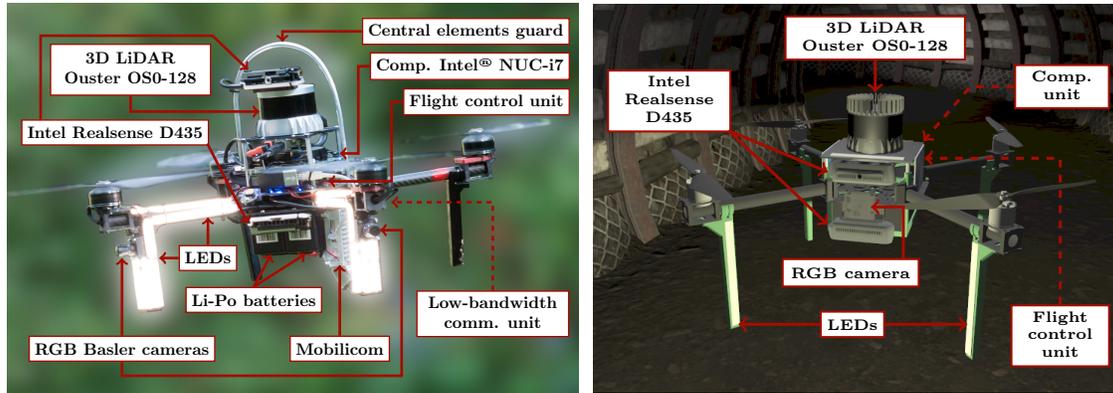


Figure 33. X500 platform used in the Systems Track (left) and Virtual Track model counterpart (right).

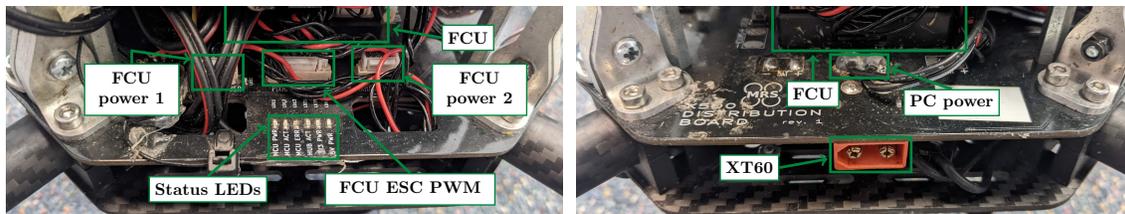


Figure 34. Custom PCB replacing the top board of the X500 frame from the front (left) and back (right). The FCU is powered by a dual redundant 5V power supply, while the Intel NUC computer is powered directly from the 4-cell battery at 14.0V–16.8V. The battery is connected using the XT60 connector and the status of individual modules of the PCB are signaled by 6 status LEDs. The ESCs are also connected to our PCB.

low real-time simulation factor. Together with the computational capacities of each simulated robot, it provides almost unlimited computational resources for running all algorithms with any desired resolution or maximal settings. On the other hand, the simulation-to-world transition requires the algorithms to run on the onboard processing units. This imposes hard requirements on the algorithms' optimization, as well as on minimization of the amount of data transfers and their latency. These requirements force us to

- compromise between accuracy and real-time performance in the system design (i.e., cutting out global optimization in on-board running SLAM),
- ensure real-time properties for systems handling critical factors of the mission (i.e., UAV control),
- optimize the data flow and the priorities of processing order within the software stack, and
- prevent any possible deadlocks from arising from outages of both synchronous, and asynchronous data.

Ensuring real-time settings for all systems of a robotic deployment is implausible, particularly in complex robotic-research projects where the stack design must allow for the system to function as a whole under limited real-world conditions. We summarize the specific aspects of the proposed ROS-based software stack, allowing us to transfer all components to on-board processing capacities. Thus providing full decentralization within a UAV team.

Software based on ROS 1 allows for connecting components under a *nodelet manager* in order to group *nodelet* plugins. In contrast to *node* configuration, the *nodelets* under a *manager* have shared memory and do not require copying data, a tool useful particularly in the case of passing large maps within the navigation stack. Our deployment stack consists of several *managers*, each of which

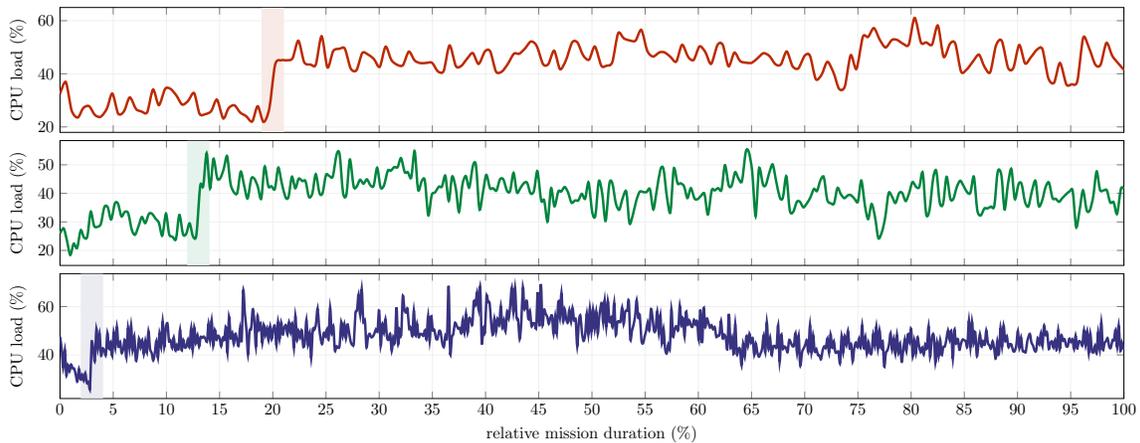


Figure 35. The CPU load of onboard computers of individual UAVs (*red, green, blue*) during the prize round of SubT Systems Track. The highlighted parts of the graph correspond to the start of processing onboard images by the object detection pipeline.

handles a distinctive part of the system. These include UAV control, preprocessing of LiDAR data and SLAM, preprocessing of RGBD data and dense mapping, navigation and path planning, and perception. The data flowing between these *managers* are copied, and thus the rate of sharing is subject to maximal reduction. To decrease the temporal and memory demands of algorithms, the resolution of input data and the output maps is decreased as much as possible within the scope and requirements of the desired application. The rate of saving data for after-mission analyses is also limited as much as possible, with no postreconstructable data being recorded at all.

In contrast to the system designs for UGV platforms, the delays in state estimation and control inputs are a critical subject for reduction. This is because excessive delays lead to destabilization of a multirotor aerial platform (see analysis on delay feasibility in Figure 11) as it is a dynamically unstable system requiring frequent feedback, even for simple hovering. The *nodelet managers* handling such critical parts of the system are prioritized at the CPU level, utilizing the negative *nice* values that prioritize the related processes during CPU scheduling. To decrease asynchronous demands on the CPU, nonprioritized components are penalized with positive *nice*. Furthermore, their scheduling is restricted on a predetermined set of threads in a multithreaded CPU. The primary subject of scheduling restriction is the perception pipeline containing a computationally heavy CNN, where static allocation reduces its asynchronous influence on the rest of the system at the cost of a limited processing rate. The effect of switching on the perception pipeline is visible in Figure 35, showing the CPU load of the three deployed UAVs during the DARPA SubT Systems Track. In other validation tests, the CPU load reached up to 90% in 1500s long missions within vast underground environments. Such an overloaded CPU results in frequent asynchronous delays, culminating to unpredictable and destructive behavior.

To limit the power consumption and hence, increase the maximum flight time, unsolicited hardware and software components can be temporarily powered off. These include switching off on-board lights in meaningless settings, disabling CNN processing when not needed, or powering off the LiDAR in the after-landing phase when the UAV is serving solely as a retranslation unit for communication.

14. System deployment

Throughout the development of the system presented in this paper, the individual components were extensively tested before integration. Deployments of the whole system were less frequent, but allowed testing the interaction of individual modules and verifying the ability to fulfill the primary objective of finding objects of interest in subterranean environments.



Figure 36. The verification of localization and perception in the following scenarios: data degraded by insufficient lighting and whirling dust (a), traversal of vertical narrow passage (b), performance in humid caves (c), multirobot exploration (d), and scalability with the environment size (e).

14.1. Continuous field verification

The S&R UAV system was continuously tested to empirically verify the correctness and reliability of the developed algorithms, strategies, and hardware. The UAVs were deployed into diverse types of environments, including historical and industrial buildings of varied levels of disintegration, in humid unstructured caves, a decommissioned underground military fortress, and vast outdoor rural areas. Some of these environments are shown in [Figure 36](#). Such tests are critical for evaluating the performance under the stochastic influence of real-world conditions, which are typically not modeled in simulations. In particular, each perception mode is more or less degraded by ambient lighting or the lack of it, the fog with microscopic condensed droplets of water, smoke or dust particles, reflections on water or smooth surfaces, etc. The filtration of LiDAR and depth data from [Section 5.2](#) therefore had to be tuned correctly to prevent the integration of false positives into the map, while keeping the actual obstacles. Moreover, the artifact detection system needed to work under a wide range of visibility conditions and chromatic shifts, for which it was necessary to collect artifact datasets from the mentioned environments.

14.2. DARPA SubT Final Event Systems Track

The Final Event, which was the culmination of the DARPA SubT competition, was organized in the Louisville Mega Cavern in Kentucky on September 23, 2021. The course consisted of all three environments from the previous circuits and contained all artifacts from previous events plus *the cube*, which was a new artifact for the Final Event. This section reports on the results achieved by the aerial part of the CTU-CRAS-NORLAB team. A total of 40 artifacts were distributed over 880m long course, which was divided into 28 smaller sectors to track the team’s progress. Every robot starts in the staging area, from which a single corridor leads to an intersection that branches into three ways. Each of the branches leads to one of the three specific environment types (tunnel, urban, and cave).



Figure 37. All robotic platforms used in the Prize Round. (From left to right) X500, Spot, Husky, TRADR, Marmotte.

Table 3. The summary of deployed robots in the Final Event sorted by deployment times. *Operable time* means how long the robot was operable, i.e., its computers were running and it could move. *Motion time* is the time the robot was moving faster than 0.1 m s^{-1} . The row *Artifacts* shows the number of confirmed hypotheses as defined in Table 5.

Robot	Spot 1	Red	Spot 2	Marmotte	Husky	Spot 3	TRADR	Blue	Green
Locomotion	Legged	Aerial	Legged	Tracked	Wheeled	Legged	Tracked	Aerial	Aerial
Deploy time	0:20	2:00	4:00	7:20	12:40	17:32	28:20	36:00	46:30
Operable time	6:00	3:00	7:00	44:00	20:00	11:00	32:00	22:25	6:10
Motion time	3:00	2:32	2:00	6:00	5:00	9:00	4:00	15:22	4:33
Traveled	111 m	69 m	47 m	181 m	131 m	195 m	97 m	304 m	119 m
Artifacts	4	1	2	1	2	0	3	3	3
Sectors explored	4	2	2	2	2	5	4	4	4

Our team deployed a heterogeneous lineup of robots. A total of 3 legged robots (Spots), 2 tracked robots (TRADR, Marmotte), 3 aerial robots (X500), and 1 wheeled robot (Husky) robot were deployed in the Final Event Prize Round (see Figure 37). The Husky robot is a fast wheeled platform (3.6 km h^{-1} max. speed) for exploration of easy terrain. Tracked Marmotte was also fast (4 km h^{-1} max. speed) but could overcome obstacles larger than Husky could. Spots were the universal ground platform thanks to the ability to pass most terrain except slippery surfaces (max speed 5 km h^{-1}). The highest traversability among the ground platforms was offered by the tracked TRADR robot thanks to its controllable flippers (however, maximum speed is approximately 2 km h^{-1}). The primary role of the aerial robots was to explore areas unreachable by ground robots such as vertical shafts or paths blocked by obstacles. The detailed composition of the team is summarized in Table 3 together with deployment times and mission statistics. The payload of all UGVs consisted of Ouster OS0-128, 5–6× Basler Ace 2 or 1× PointGrey Ladybug 2 cameras, Xsens MTI-30 IMU, Mobilicom MCU-30 Lite, Nvidia Jetson Xavier AGX, Intel NUC 10i7FNK, LED illumination, SDC30 gas sensor, and communication notes.

14.2.1. UAV deployment summary

Three UAVs in total (*red*, *green*, and *blue*) were deployed in the 60 min long run. All UAVs used the Greedy strategy (Section 8.2.2) without VPE for the simplicity of its reward function, which made it easier to fine-tune the reward function coefficients for the competition environment and debug the UAVs’ behavior. The UAV performance is summarized in Table 4 and the flight trajectories are plotted in Figure 44. The first UAV (*red*) took off just after the first UGV, arrived to the first intersection, explored 10 m of the tunnel section, returned to the intersection, flew to the cave branch where it collided with the Spot UGV (Figure 42a). The chronologically second deployed UAV was *blue*, which went into the urban branch where it traveled to a vertical alcove with a phone artifact. Then it returned to the start of the urban section, where it hovered until exhausting the battery (Figure 42c), because all viewpoints were blocked in its map corrupted by drift in the featureless urban corridor. The last deployed UAV was *green* that explored the tunnel section, where it was blocked by a dynamically added artificial wall (Figure 41). After flying through a cluttered tunnel corridor, the UAV collided with a metal rod protruding from the wall (Figure 42b).

Table 4. The mission statistics from the prize round of the Final Event. The localization accuracy was not evaluated for UAV blue. Obtaining the ground truth position using scan matching would have been extremely strenuous due to the degenerate geometry of LiDAR scans in the urban tunnel. This degeneracy also caused the onboard localization to drift several meters.

UAV	Red	Green	Blue
Localization accuracy:			
avg max error in translation (m)	0.38 0.63	0.97 2.66	-
avg max error in heading (°)	0.64 4.06	1.48 5.37	-
Safety clearance	0.4 m	0.11 m	0.21 m
Landing cause	Collision with UGV	Collision with a metal rod protruding from the wall	Depleted battery after being trapped in degraded map

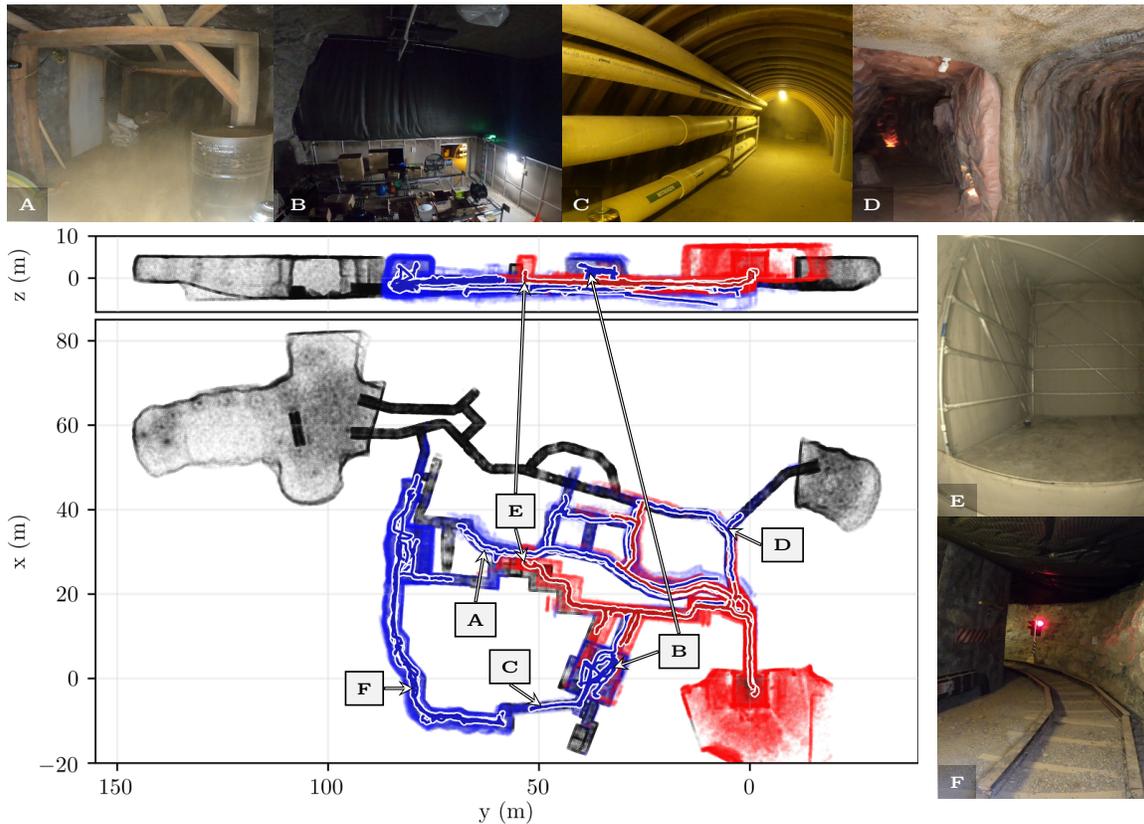


Figure 38. UAV trajectories and on-board-built maps of the environment from all flights during the prize round (colored in red) and the postevent testing (colored in blue) overlaid over the ground truth map (colored in black). The photos from on-board camera highlight the diversity and the narrow confines of the environment.

The maps and the trajectories of all our UAV flights during the prize round and the postevent testing are shown in Figure 38, together with summary of the mapping errors from these flights in Figure 39. The distance of the UAVs from the nearest obstacle during all flights in the prize round are shown in Figure 40.

14.2.2. Artifact detection discussion

The performance of the artifact detection and localization system is summarized in Table 6, and the number of artifacts detected by each UAV in Table 5. A total of seven artifacts appeared in

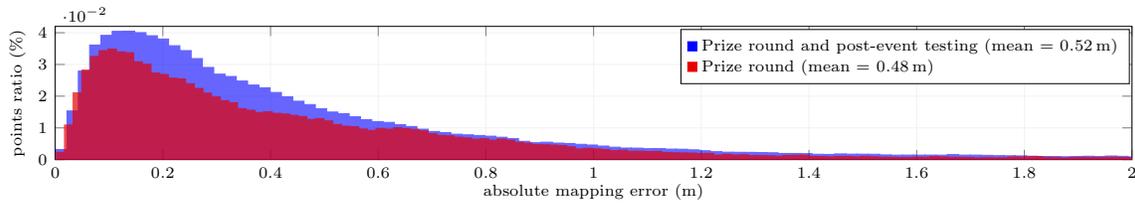


Figure 39. Distribution of mapping errors throughout the prize round and the postevent testing flights (colored in red and in blue in Figure 38) of DARPA SubT. The absolute mapping error denotes the distance between the ground truth map and concatenation of DenseMaps built with resolution of 20 cm on-board during particular UAV flights. The error metric is the Euclidean distance between a point from the on-board maps to its closest point in the ground truth map.

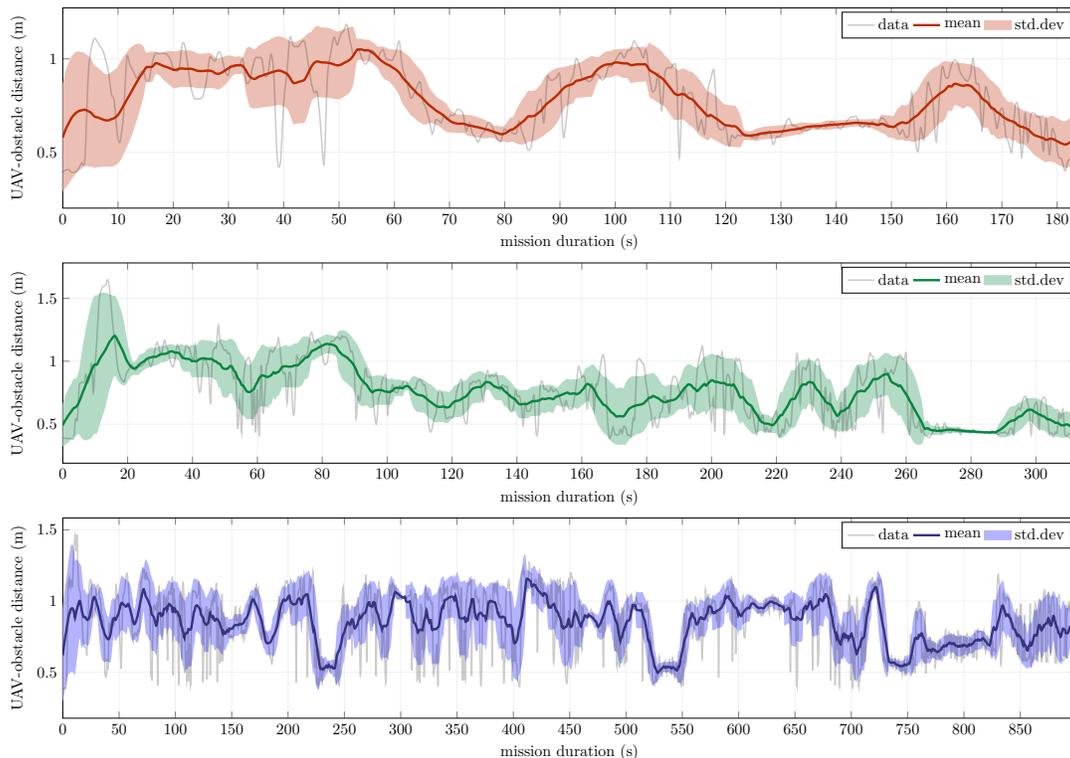


Figure 40. Distance between the center of the UAVs and the measured nearest obstacle during the prize round of the SubT Systems Track. The moving mean and standard deviation are computed over a 10 s long time window.

the camera images, and six artifacts were detected by the CNN. The detections with estimated bounding boxes from all UAVs are shown in Figure 45. The survivor *s2* was seen in three frames of the bottom camera. However, only a small part of the survivor sleeve was visible and the images were further degraded by motion blur, as can be seen in Figure 43. Thus the CNN did not manage to detect the artifact. From the six detections, the cellphone artifact *p1* was detected only on one image frame when the UAV *blue* peeked into the vertical shaft in the urban part. However, as explained in Section 10, a total of four detections are necessary to create a hypothesis and to confirm the position, and thus this single detection was discarded. Another missed point was the survivor *s1*, which was detected and localized within the 5 m limit, but the artifact was labeled as a cube instead of a survivor. The hypothesis was merged with a high number of false positives and, consequently, the correct image was not sent to the operator, who could not determine the correct class to report.

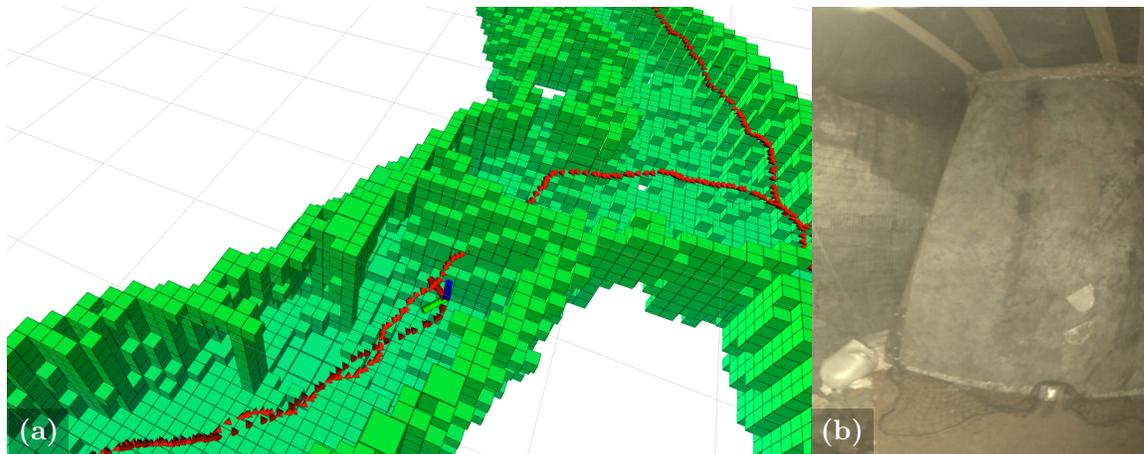


Figure 41. The artificial wall that blocked the way back for UAV *green* in the map (a) and in the camera image (b).

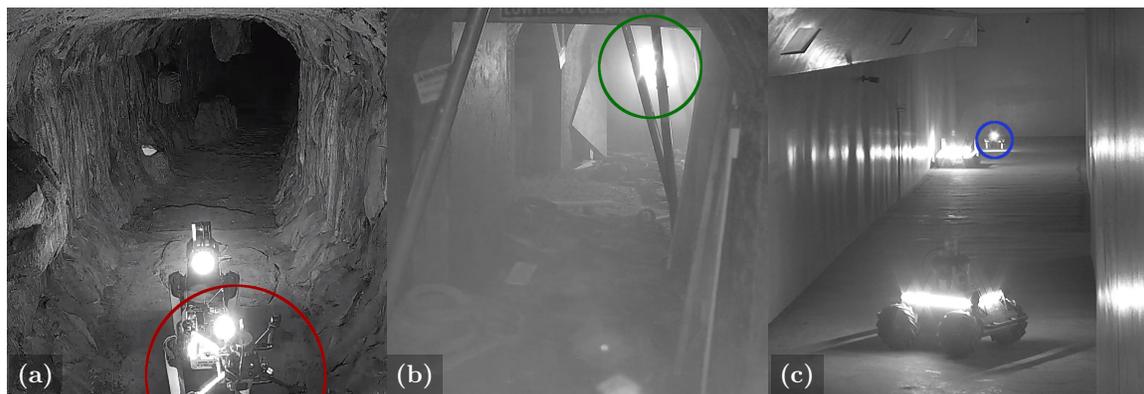


Figure 42. The landing events of all three UAVs. The UAV *red* (a) collided with the Spot UGV, UAV *green* (b) hit a metal rod protruding from the wall, and UAV *blue* (c) landed after its battery was exhausted by hovering while being trapped in a map corrupted by drift in the featureless corridor.



Figure 43. The only three image frames of the survivor *s2* captured by the downward-facing camera. The artifact was not detected as there is only a small part of the survivor's sleeve visible in the image, which is also degraded by motion blur.

Both vent *v1* and drill *d1* were detected, localized, and correctly labeled. The drill *d4* was incorrectly classified as a backpack, nevertheless, the operator reported the correct class based on the detection image. All three UAVs detected the *d4* drill, but UAV *green* provided the highest accuracy, which is reported in Table 6. In total, four artifact hypotheses arrived to the base station with sufficient information for obtaining a point for the report.

Table 5. Statistics of artifact detection for each deployed UAV from the prize round of the Final Event. The *seen* column yields the number of artifacts that appeared in the image of one of the on-board cameras. If the artifact was detected by the CNN, it is listed in the *detected* column and the detection is shown in Figure 45. Artifacts that were *confirmed* had enough consistent detections to establish a hypothesis. *Confirmed unique* artifacts were not detected by another robot, including UGVs.

UAV	Artifacts			
	seen	detected	confirmed	confirmed unique
Red	1	1	1	0
Green	4	3	3	1
Blue	4	4	3	1

Table 6. Unique artifacts detected by lightweight CNN running on-board UAVs in real time. The total error e_{tot} of the artifact position is the sum of the UAV localization drift error e_{loc} and the error of estimating the artifact position e_{est} from the detected bounding box. Artifacts detected by more UAVs are listed only once with values from the most accurate hypothesis among the UAVs. The hypothesis was *Confirmed* when more than four images were associated with it. Some artifacts were correctly detected and localized, but the wrong label was assigned to them. This is documented in the *Correct class* column. Even with a wrong label, the operator could still deduce the correct class by looking at the image sent with the hypothesis. Only one image was sent with each hypothesis, and if it was possible to deduce the correct class, then the image was listed as *Correct image*.

Artifact	Frames detected	Confirmed	Correct class	Correct image	e_{loc} (m)	e_{est} (m)	e_{tot} (m)
v1	27	✓	✓	✓	1.94	4.61	3.08
s1	60	✓	×	×	2.93	4.57	2.89
p1	1	×	×	×	-	-	-
d4	11	✓	×	✓	0.77	1.61	1.30
f1	13	✓	✓	✓	0.85	1.33	1.31
d1	9	✓	✓	✓	1.46	2.30	1.55

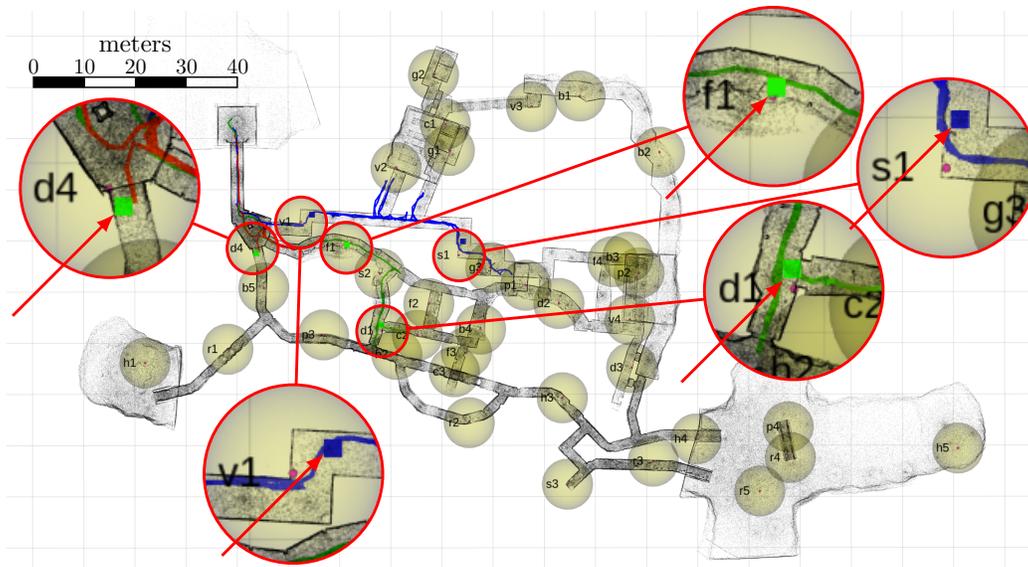


Figure 44. The map of the Final Event course was obtained by the organizers by scanning the course with a laser scanner station. The paths traveled by all three UAVs (*red*, *green*, and *blue*) during the Final Event are depicted by their respective colors. The ground truth positions of artifacts are surrounded by a yellow sphere in order to visualize the 5m limit for the reported artifact to be counted as a point in the competition. The five artifacts that were detected and localized within this 5m limit are shown as squares colored by the detecting UAV and highlighted in the magnified sections with red arrows.



Figure 45. Images of artifacts detected by the UAVs in the Final Event. The color of the rectangle shows which UAV detected the artifact and at what mission time as shown in the bottom right corner.

14.3. DARPA SubT Final Event Virtual Track

In parallel to the Systems Track, the competition was also running in the simulated form of the Virtual Track. The teams had to submit a solution consisting of docker images of a robotic team put together within a limited budget to buy the robots and their sensory packages.

The Systems Track included a single run (with two preliminary rounds) conducted in a single world and was therefore focused on the reliability of the robots, which had to overcome challenging terrain with narrow passages and adverse conditions for perception. On the other hand, the virtual teams were deployed three times in each of the eight worlds, ranging from vast models of artificially created environments to scanned courses from the previous events, including the Final Event course. Moreover, in the Virtual Track, the whole mission must be fully autonomous and no human interventions are possible. The purpose of the virtual event was to evaluate the high-level planning, cooperation, decision-making, and efficient coverage of the large worlds. As the cooperative searching strategy is one of the core contributions of this work, we have presented the results from the virtual course here as most of the worlds allowed for efficient deployment and cooperation of the multirobot teams.

14.3.1. Differences from the Systems Track

The simulation model of the IMU provides much better data compared to the real sensor with the same parameters. Thus is due to the measurements in the simulation not being corrupted by propeller-induced vibrations, wind gusts, or saturation, as well as having the IMU rigidly attached to the UAV body with known extrinsic parameters. The higher quality of the simulated data allows for the use of LiDAR-inertial odometry. In addition to the LiDAR, it also relies on the IMU preintegration in its optimization process, thus providing a smooth and drift-free position estimate, even when there are few geometrically rich features present. Specifically, the LIO-SAM (Shan et al., 2020) algorithm was chosen for its low drift and high precision over the A-LOAM deployed in the Systems Track. Both algorithms are detailed in Section 6.

The computation power available for artifact detection in the Virtual Track was not constrained by the weight of the onboard computation hardware as was the case in the Systems Track. As a result, compromises of the performance/weight ratio had to be made on the detector in the Systems Track, as reported in Section 10.

Reporting of the found artifacts is handled by the operator in the Systems Track, which is not possible in the fully autonomous Virtual Track. A virtual artifact reporter algorithm was developed to gather artifact hypotheses from all robots and decide which hypotheses are the most likely to score a point (described in detail in [Section 10.4](#)).

The control interface of the simulated UAV was also different from the real one. While the FCU of the real UAV accepted attitude rate commands generated by the Special Euclidean group of dimension 3 (SE(3)) controller, the simulated UAV was controlled on a higher level by velocity commands. This did not allow for precise control of the UAV motion, as was the case for the low-level attitude rate control.

The deployment sequence of individual robots in the Systems Track could be decided by the operator based on the requirements of locomotion modality, dynamics, and sensory payload during the progress of the mission. In contrast, the sequence in the Virtual Track was fixed before the start of the run.

LandMap introduced in [Section 7.5](#) was not used in the Virtual Track where the UAV was not destroyed even after a rough landing. As long as the UAV landed in the communication range of the network it could send its hypotheses to the base station and further retranslate messages from/to other robots.

14.3.2. Virtual Track results

In the virtual deployment, our team consisted of five UAVs and two UGVs. The UAVs were the superior platform in the Virtual Track due to their greater movement speed, smaller form-factor, and better mobility to fly over terrain untraversable by the UGVs. We deployed two UGVs to build a communication network consisting of breadcrumbs dropped at the edges of the wireless signal range. This allowed for the UAVs to maximize the time for searching for artifacts as they could return to the nearest breadcrumb instead of to the base station back at the staging area. Both UGVs were deployed at the start of the run. The deployment times and exploration strategies of individual UAVs are listed in [Table 7](#). Our solution achieved 2nd place with a total of 215 scored points. [Table 8](#) summarizes the points scored by the top three teams on each world of the Virtual Track ([Figure 46](#)). The lower number of points on worlds 4, 5, 6, and 8 can be explained by the fact that these worlds were not made of the tiles that were used in the qualification and practice worlds. The details on traveled distance and collected hypotheses by particular UAVs during all runs of the SubT Virtual Finals are provided in [Figure 47](#) and [Figure 48](#), respectively.

Table 7. The times of deployment and assigned strategies from [Section 8.2](#) in the Virtual Track. The second UAV was scheduled to take off after the first UAV returned to communication range so that it can take advantage of the LTVMap of the first UAV. Both DEI and Greedy strategies were used as DEI guarantees covering dead-end corridors at the cost of lower average velocity and lower total surface covered. The first UAV used DEI so that the rest of the team did not need to return to where the first UAV already had been. The next two UAVs maximize the searched volume with the Greedy strategy and the last two UAVs cover any missed surfaces with DEI.

UAV	1	2	3	4	5
Start (s)	60	1560	1680	1800	1920
Strategy	DEI	Greedy	Greedy	DEI	DEI

Table 8. The score achieved by the top three teams on each world of the Virtual Track. The reported values are the sums of three runs on each world.

World	1	2	3	4	5	6	7	8	total
Dynamo	21	52	48	18	15	11	44	14	223
CTU-CRAS-NORLAB	31	39	45	16	18	13	36	17	215
Coordinated Robotics	44	41	27	23	17	14	26	20	212

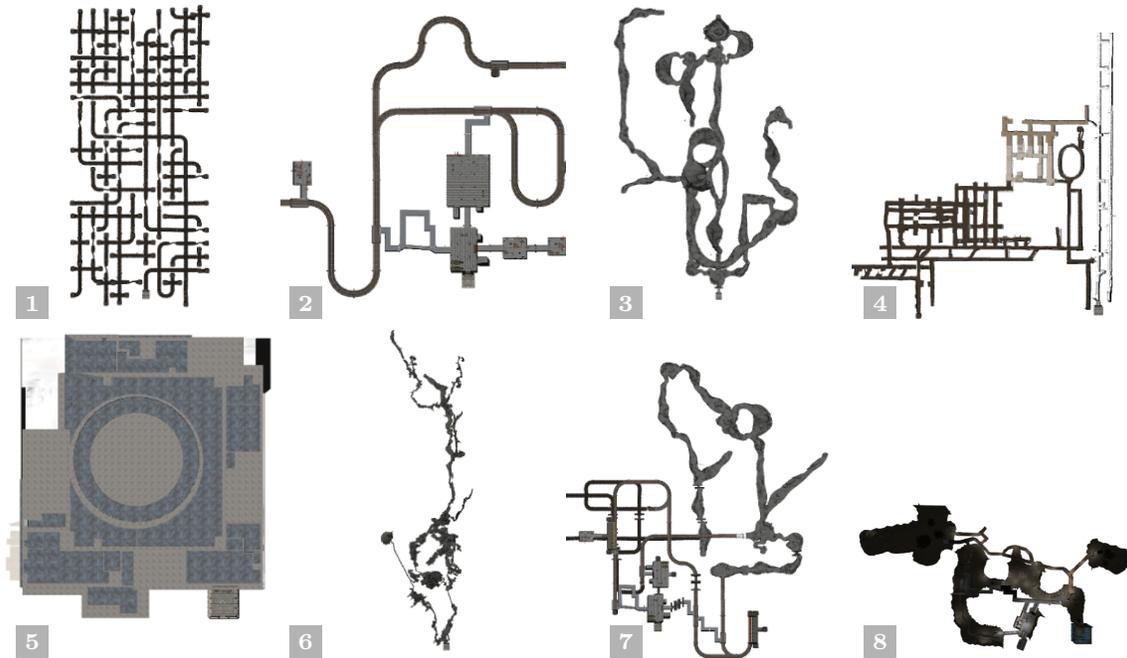


Figure 46. All eight worlds used in the Virtual Track of the DARPA SubT Finals. The worlds 1, 2, 3, and 7 are built from tiles that were used in the preliminary and practice rounds. World 4 is the model of the NIOSH research mine, where the tunnel circuit was held. Similarly, world 5 corresponds to the model of the location of the urban circuit—the unfinished Satsop nuclear power plant. World 6 is a model of a narrow cave system. World 8 is modeled based on the Systems Track Finals.

15. Lessons learned and future work

In this section, we present our view on the state of the S&R UAVs, the lessons learned, which problems are solved, and what areas require more research to achieve reliable performance suitable for deployment as a tool for assisting rescue workers. These findings were collected throughout the preparation for as well as during the DARPA SubT Competition, which aimed to push the state of the art of S&R robotics. Furthermore, this discussion should be of some interest to the community as we highlight aspects that could be explored in future research and development. In general, most of the individual subproblems, such as localization, mapping, detection, and communication, are solved to the point of being capable of performing an autonomous mission in extremely challenging conditions. The developed algorithms are now used in actual field deployment instead of just laboratories and simulations, which introduces disturbances, noise, dust, and other detrimental effects that negatively impact the algorithms' performance and reliability. It is essential to focus on the reliability of the employed methods to make the UAVs a valuable asset to the S&R team.

The role of the aerial robot in a heterogeneous S&R robotic team is a quickly deployable agent that can provide swift situation awareness, environment type, and topology information that allows for informed decision-making about the rest of the mission. Furthermore, areas such as caves, collapsed buildings or high openings can often be reached only by UAVs. On the other hand, ground robots have the advantage of higher payload capacity, which results in improved perception capabilities compared to UAVs.

The localization method based on 3D LiDAR provides precise position estimates, even under severe degradation by dust. However, as proved by the UAV *blue*, the estimate can begin to drift when the solved optimization is ill-conditioned due to low-variance geometry, typically in long corridors with straight walls. The unpredictable nature of subterranean environments requires a localization method that is reliable and drift-free under arbitrary conditions. Solutions based

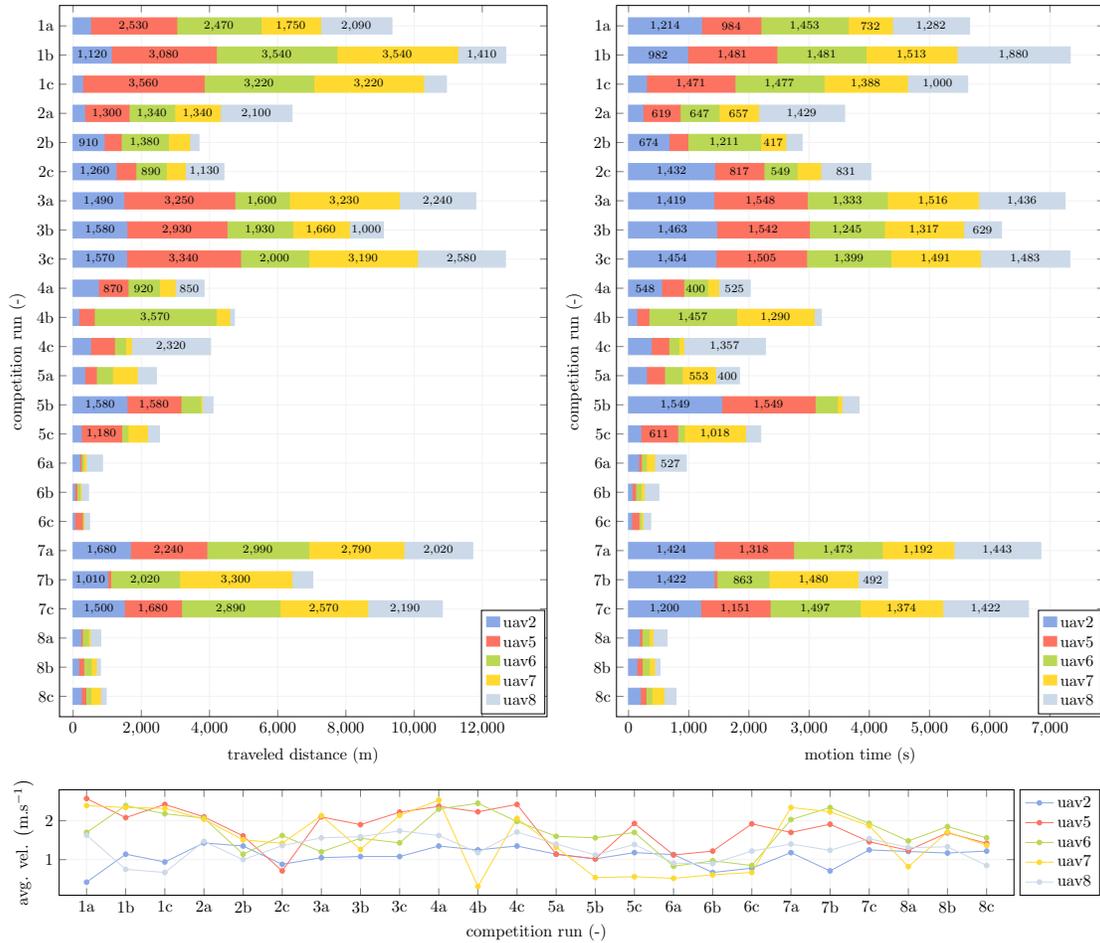


Figure 47. Overall traveled distance, time of active motion, and average velocity of particular UAVs in all runs of the SubT Virtual Finals. The maximum traveled distance throughout all runs was achieved by UAV5 in run 1c (3560m). The maximum active time was achieved by UAV2 in run 5b (1539s). The presented average velocity incorporates the entire flight, including hovering states.

on detecting geometrical degeneracy, and multimodal fusion of LiDAR and visual methods were described in Section 2.2. The results seem promising but due to high unpredictability and challenges of subterranean environments more research in localization algorithms is still required for truly robust pose estimation in arbitrary conditions.

In addition to map drift caused by errors in the localization, the volumetric occupancy grid did not contain the smaller obstacles like ropes, cables, and thin poles, which led to the collision of UAV *green* as seen in Figure 42b. Although some LiDAR rays hit these thin obstacles, the occupied cells generated by these rays were often changed to free when multiple rays that passed through these cells hit the wall behind them. As a result, the navigation pipeline planned a path through these cells that appeared free, but contained a thin metal pole, causing a collision. The ability to traverse narrow passages is also impaired since the passages appear narrower than they really are due to grid discretization. We propose to locally increase the resolution of the grid of DenseMap on demand to approximate the free space more accurately, while keeping the scan integration times bounded. This approach is however only a partial solution as the need for a more granular resolution might not always be reliably detected. Consequently, the need arises for a flexible map that is not bound by fixed cell size, similarly to the SphereMap, possibly based on surfel mapping as seen in (Behley and Stachniss, 2018), or based on GMM (O’Meadhra et al., 2018).

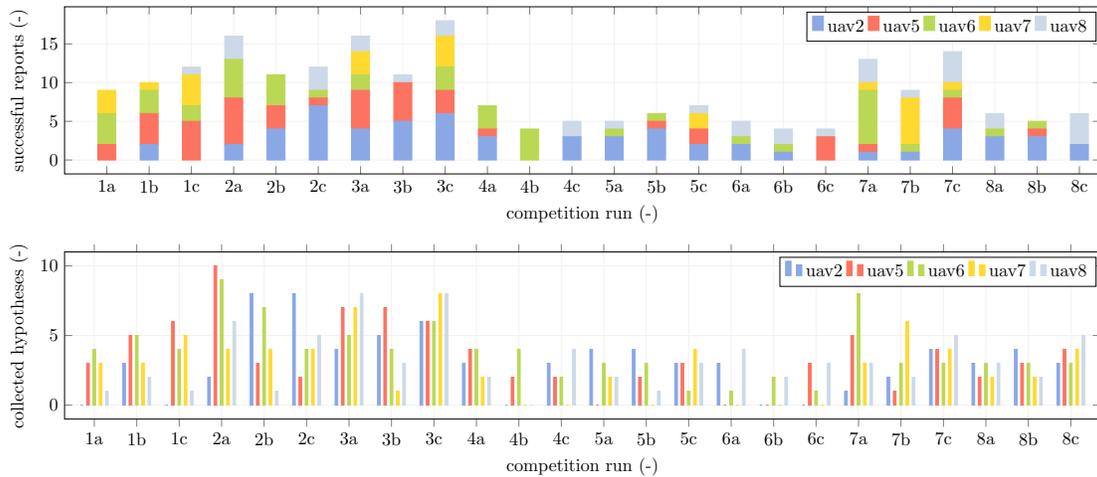


Figure 48. Distribution of successful reports among the UAVs in particular runs of SubT Virtual Finals (top) and the number of valid hypotheses collected throughout particular runs by individual robots (bottom). The number of successful reports of individual robots is mostly influenced by the ordering of robots and their delayed starts in the mission.

Related to the narrow passage traversal was also the decision to not include prop guards in the platform design. With the experience from Urban Circuit (Kratky et al., 2021a) where our platform featured prop guards, we decided against prop guards in the Final Event as they further increase the size and mass of the UAV. The advantage of the prop guards is uncertain as they do not automatically allow the UAV to continue operation after a collision. The control and localization software needs to be designed to handle these dampened collisions and even then it is not guaranteed that the UAV will continue in flight as the collision is caused by a failure of a module (perception failure to detect an obstacle, localization error, etc.), which cannot be solved by prop guards. Nevertheless, platforms with prop guards were deployed successfully by other teams [e.g., (Scherer et al., 2022; Agha et al., 2021)] so a consensus on this design choice has not been reached yet.

We experienced a surprising issue when our UAV equipped with the Ouster OS0-128 LiDAR was passing around a UGV with LeiShen C16 LiDAR. The rays emitted by the LeiShen corrupted some of the Ouster measurements, which manifested as points in random distance within the FOV of the LiDAR. These false positives were not filtered out by the intensity filter from Section 5.2, because the intensities fall into the same range of values as true positives. As a result, the points get integrated into the map, as shown in Figure 49. Nevertheless, the performance of the UAV was not degraded as the navigation pipeline is robust to such sparse noise. This experience highlights the importance of testing the compatibility of robotic platforms deployed in heterogeneous teams.

The flight time of the UAV over 20 min was achieved as the payload was limited only to crucial components. However, the presence of only a single computational unit without CNN acceleration or dedicated GPU led to compromises in the artifact detection CNN. Large-size models such as YOLOv3 (Redmon and Farhadi, 2018) were too slow for achieving satisfactory frame rates on the CPU, so lighter models had to be used. As explained in Section 10, the lightweight MobileNetV2 CNN allowed for lightweight models (7MB) that could fit into the cache of the CPU. Furthermore, the OpenVino framework supports accelerating the CNN on the GPU integrated with the CPU, which helped to achieve sufficient frame rates. Although the lightweight model successfully detected all artifact types, the labeling was not very reliable and many false positives were detected. This impacted the artifact localization process, as the false positives were fused into the artifact hypotheses, which shifted the estimate further from the true position. Also, the images of these false positives were sometimes sent as the representative image of the hypothesis. Thus the operator could not correctly decide the artifact class when the label produced by the CNN was incorrect. When

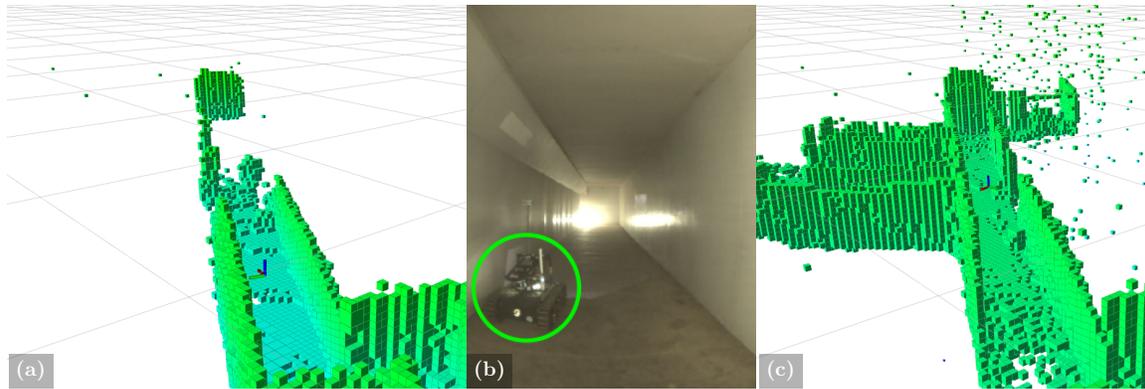


Figure 49. DenseMap before (a) approaching the UGV with LeiShen C16 LiDAR (b) and (c) when it gets corrupted by random points in the FOV of the LiDAR mounted on the UAV after flying in close vicinity (≈ 1 m) to the UGV. Notice, a few false positives were integrated into the map even when the UAV was 8 m away from the UGV (a).

payload capacity prevents the use of more capable hardware, the issue must be compensated by the sensing strategy. In contrast to UGVs the mobility of UAVs allows reaching closer to the artifact to verify the detection. Approaches of perception-driven navigation can improve the performance of the lightweight detector by planning a trajectory to inspect the artifact from a closer distance and other angles after the initial detection.

Although our platform is quite compact (500 mm without propellers), it could not pass through all of the narrow passages, even during the postevent testing. Apart from the discrete map and conservatively set distance from obstacles (see Table 4), the size of the UAV prevented flying through some of the narrower passages of the circuit. As even smaller passages are to be expected during deployment of robots in real S&R scenarios, the UAV platforms should be further miniaturized to allow safe traversal of narrow passages. Deployment of visually localized UAVs could decrease the size significantly but the capabilities of visual navigation pipelines still underperform compared to the LiDAR solutions, which was the preferred approach of most teams. A possible workaround that compensates for the lower flight time of smaller platforms is the marsupial deployment (Lindqvist et al., 2022; De Petris et al., 2022). When such miniaturization is not possible due to, e.g., insufficient payload of smaller platforms, a heterogeneous aerial team consisting of both large and small platforms can be deployed. In such case, the large platform carrying a LiDAR can command and send position corrections to smaller visually localized UAV that can inspect tight narrow passages that are unreachable by the large UAV (Pritzl et al., 2022b).

A mutual collision avoidance module is a necessity for any application where multiple robots share the same workspace. The developed priority-based module uses the already shared information about the robots' positions when communication is available, as it should since the risk of collision arises when robots are in close proximity. This module prevented collisions in the Virtual Track, where despite the vastness of most of the worlds, the collisions happened often in the practice runs before implementing the collision avoidance. We decided against using the collision avoidance module in the Systems Track. This was done as the robots could easily become deadlocked in tight corridors and also due to the collision probability being reasonably low because of the delay between each UAVs launch. Additionally, the operator could override the full autonomy to prevent collision, if necessary. Nevertheless, the UAV *red* collided with a Spot UGV shortly after the start of the run, which could have been prevented if collision avoidance was enabled. A deadlock-free solution based on agent theory approaches can be devised for situations when communication is available, and behavior prediction methods can provide a backup when communication is not possible.

Even though the organizers did a great job at providing a realistic simulation environment for the Virtual Track, many phenomena, unexpected situations, and issues from the real world are

not present in the simulation. Moreover, the rules of the competition are different for the two tracks. For example, the absence of a human operator in the virtual team changes the approach drastically as all decision-making needs to be automated. For details about the differences between the Systems and Virtual tracks see [Section 14.3.1](#). As a team that participated in both Systems and Virtual tracks, we want to list the greatest hurdles encountered in the simulation to real-world transfer:

- **Degraded sensor output.** The output of real-world sensors is corrupted by various negative effects. On the other hand, the imperfections in simulated sensors are typically modeled only by an additive noise, which most algorithms can cope with by smoothing or filtering. The performance of such algorithms severely deteriorates with input data degraded by the harsh conditions of underground environments. In the case of laser-based sensors, rays get reflected by small airborne particles such as dust, smoke, and fog to produce false measurements. Although the fog was modeled in the simulation, the distribution of fog points in the point cloud did not coincide with the distribution in the real world, and thus we implemented different approaches for Virtual and Systems tracks, which are detailed in [Section 5.2](#). Cameras in addition to the airborne particles suffer from insufficient illumination, high-contrast scenes, motion blur, and rolling shutter aliasing. Using neural networks for object detection proved to be robust to these effects when trained on datasets with similar data.
- **Environment scale.** The scale of the environment was much smaller in the Systems Track than in the Virtual Track. Most of the simulation worlds spanned several-kilometers-long corridors with vast caves to challenge the cooperative exploration abilities of the teams. The vastness and openness of the worlds favored fast flights to cover as much space as possible during flight time. In contrast, the Systems Track was narrow from the beginning of the course (see the cross-section distribution in [Table 2](#)) and the UAV was closer than 1 m from obstacles for most of the flight (see the distance to obstacle plot in [Figure 40](#)). To minimize the chance of collision, the velocity of the UAV was constrained to 1 m s^{-1} instead of 5 m s^{-1} in simulation and the control was tuned for low error by commanding the UAV in attitude rates instead of linear velocities.
- **Safety.** In simulation, the robots cannot harm anyone and to perform another run after a collision it is sufficient to restart the simulation. Contrary to that, in the real world, special care must be taken to make the robots, especially fast aerial robots with quickly spinning propellers, safe for the environment, operators, and any other humans in the vicinity. To assure maximum safety during takeoff, status checks are performed automatically but then the safety operator has to approve the takeoff by toggling a switch on the RC as described in [Section 11.1](#). During the flight, health checks of the rate of crucial data streams are performed, control errors are monitored, and innovation of state estimation corrections are analyzed. When any of the monitored values exceeds a critical threshold, an emergency landing is initiated to minimize the damage to the platform. A technique based on data from the downward-facing depth camera was developed to assure safe regular (not emergency) landing on planar low-slope surfaces stored in LandMap ([Section 7.5](#)).
- **Decision-making in artifact reporting.** Due to the limited payload of the UAV, a lightweight CNN was used for artifact detection, which forced us to choose a compromise between precision and recall. Having a human operator to verify the artifact hypotheses allowed us to maximize recall to not miss any artifact at the cost of a higher false positive count. A correctly detected but misclassified artifact could still score a point if the operator correctly deduced the class from the image (see [Table 6](#)). This would not have been possible using the autonomous arbiter and thus the flexibility of a human operator overperforms the autonomous arbiter, which optimizes a fixed criterion when reporting an artifact hypothesis.
- **Computational resources** The simulated run in Virtual Track was running only at a fraction of real time and thus the employed algorithms had more computation time available. In the Systems Track, the developed algorithms had to run on the onboard processing units in real

time. Thus, for the simulation to real-world transfer, the algorithms need to be optimized, critical systems prioritized, and often a compromise between accuracy and computation time has to be found. We discuss all modifications and optimizations in [Section 13](#).

16. Conclusion

This paper has presented the complex UAV system deployed in the final round of the DARPA SubT Challenge after 3 years of development and testing in numerous real world demanding environments (including gold mine, coal mine, abandoned nuclear power plant, caverns, military fortress, natural caves, old factory hall, subway station, etc.). Based on these unique opportunities and experience, we have designed both the hardware UAV platform and the multi-UAV software with a focus on the exploration of such vast, complicated, and varying environments.

In the Systems Track of DARPA SubT Challenge, three UAVs were deployed alongside ground robots into the competition course consisting of a heterogeneous environment of the tunnel, urban, and cave sections, where the aerial team detected and localized four artifacts and traveled 492 m in total. The austere conditions of the circuit, such as narrow passages, dust, featureless corridors, and dynamic obstacles, tested the reliability of the system as a whole, including the hardware design of a compact platform with a considerable flight time of 25 min. Most of the testing was realized in environments where it performed exceptionally well, including a former brewery where the UAV had to explore an abandoned building with partially collapsed floor and ceiling, or during the exploration of Byci Skala (Bull Rock Cave) in the Moravian Karst cavern system. Compared to ground robots the UAVs could search a larger volume of space because they could easily fly over any encountered problematic terrain such as mud, water, and rubble and thus had an advantage in the exploration of unknown terrains with unexpected obstacles. Furthermore, the worlds of the Virtual Track of the competition were also very large; even with our UAV possessing a 25 min flight time and fast dynamics, they were not able to reach the furthest parts of some worlds. Although our system was designed primarily for these large-scale environments, its performance in the challengingly tight corridors of the prize round was also impressive. The difficulty of UAV deployment in such adverse environments motivated numerous achievements beyond the state of the art that are summarized in this paper. Many lessons were learned in the process that could facilitate and support designing complex robotic systems in similar applications in the future.

A larger team of five aerial robots was deployed in the Virtual Track, alongside two UGVs. By employing the proposed cooperative exploration strategies based on topological map sharing, the exploration effort of our team was spread out over a wider area. Together with dynamic flight and reliable artifact detection/localization, this helped to achieve the 2nd place with 215 scored points. Moreover, seven out of the nine participating teams used our X500 UAV, which was modeled according to the specification of the physical platform thanks to its long flight time, a wide array of sensors, modest size, and reasonable price.

Based on the successful deployment in the DARPA SubT, which focused on providing challenging conditions typically encountered during rescue missions in underground environments, we conclude that the presented UAV system is a valuable addition to teams of first responders, as it can provide situational awareness and even find survivors after a catastrophe without risking the lives of rescuers in dangerous environments.

Acknowledgments

We would like to thank the members of the CTU-CRAS-NORLAB team who participated in the design and development of the UAV and UGV hardware platforms, software development, simulations, testing and general support. Namely: Ruslan Agishev, Afzal Ahmad, Teymur Azayev, Jan Bayer, Tommy Bouchard-Lebrun, Petr Čížek, Simon-Pierre Deschênes, Jan Faigl, Olivier Gamache, Alexandre Guénette, Bedřich Himmel, Jakub Janoušek, Tomáš Krajník, Vladimír Kubelka, Denis Ouellet, Tomáš Petříček, François Pomerleau, Miloš Prágr, Tomáš Rouček, Vojtěch Šalanský, Martin

Škarytka, Vojtěch Spurný, Pavel Stoudek, Arsen Tkachev, Maxime Vaidis, Volodymyr Zabulskyi, Karel Zimmermann, and Martin Zoula.

This work was partially funded by the Defense Advanced Research Projects Agency (DARPA), by the CTU grant no. SGS20/174/OHK3/3T/13, by the Czech Science Foundation (GAČR) under research project no. 20-29531S, by TAČR project no. FW03010020, by the OP VVV funded project CZ.02.1.01/0.0/0.0/16 019/0000765 “Research Center for Informatics,” by the European Union’s Horizon 2020 research and innovation programme AERIAL-CORE under grant agreement no. 871479, and by the NAKI II project no. DG18P02OVV069.

ORCID

Matěj Petrлік  <https://orcid.org/0000-0002-5337-9558>
 Pavel Petráček  <https://orcid.org/0000-0002-0887-9430>
 Vít Krátký  <https://orcid.org/0000-0002-1914-742X>
 Tomáš Musil  <https://orcid.org/0000-0002-9421-6544>
 Yurii Stasinchuk  <https://orcid.org/0000-0002-2197-1442>
 Matouš Vrba  <https://orcid.org/0000-0002-4823-8291>
 Tomáš Báča  <https://orcid.org/0000-0001-9649-8277>
 Daniel Heřt  <https://orcid.org/0000-0003-1637-6806>
 Martin Pecka  <https://orcid.org/0000-0002-0815-304X>
 Tomáš Svoboda  <https://orcid.org/0000-0002-7184-1785>
 Martin Saska  <https://orcid.org/0000-0001-7106-3816>

References

- Agha, A., Otsu, K., Morrell, B., Fan, D. D., Thakker, R., Santamaria-Navarro, A., Kim, S., Bouman, A., Lei, X., Edlund, J. A., Ginting, M. F., Ebadi, K., Anderson, M., Pailevanian, T., Terry, E., Wolf, M. T., Tagliabue, A., Vaquero, T. S., Palieri, M., et al. (2021). Nebula: Quest for robotic autonomy in challenging environments; TEAM costar at the DARPA subterranean challenge. *CoRR*, abs/2103.11470.
- Ahmad, A., Walter, V., Petracek, P., Petrлік, M., Baca, T., Zaitlik, D., and Saska, M. (2021). Autonomous aerial swarming in gnss-denied environments with high obstacle density. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 570–576. IEEE.
- Ahmad, S., Sunberg, Z. N., and Humbert, J. S. (2021). End-to-end probabilistic depth perception and 3d obstacle avoidance using pomdp. *Journal of Intelligent & Robotic Systems*, 103(2):1–18.
- Alismail, H., Kaess, M., Browning, B., and Lucey, S. (2016). Direct visual odometry in low light using binary descriptors. *IEEE Robotics and Automation Letters*, 2(2):444–451.
- Alotaibi, E. T., Alqefari, S. S., and Koubaa, A. (2019). Lsar: Multi-uav collaboration for search and rescue missions. *IEEE Access*, 7:55817–55832.
- Baca, T., Hert, D., Loianno, G., Saska, M., and Kumar, V. (2018). Model Predictive Trajectory Tracking and Collision Avoidance for Reliable Outdoor Deployment of Unmanned Aerial Vehicles. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1–8. IEEE.
- Baca, T., Loianno, G., and Saska, M. (2016). Embedded Model Predictive Control of Unmanned Micro Aerial Vehicles. In *2016 IEEE International Conference on Methods and Models in Automation and Robotics (MMAR)*, pages 992–997.
- Baca, T., Petrлік, M., Vrba, M., Spurny, V., Penicka, R., Hert, D., and Saska, M. (2021). The mrs uav system: Pushing the frontiers of reproducible research, real-world deployment, and education with autonomous unmanned aerial vehicles. *Journal of Intelligent & Robotic Systems*, 102(26):1–28.
- Bayer, J. and Faigl, J. (2020). Handheld localization device for indoor environments. In *2020 4th International Conference on Automation, Control and Robots (ICACR)*, pages 60–64.
- Behley, J. and Stachniss, C. (2018). Efficient surfel-based slam using 3d laser range data in urban environments. In *Robotics: Science and Systems*, volume 2018, page 59.
- Bircher, A., Kamel, M., Alexis, K., Oleynikova, H., and Siegwart, R. (2016). Receding horizon” next-best-view” planner for 3d exploration. In *2016 IEEE international conference on robotics and automation (ICRA)*, pages 1462–1468. IEEE.

- Blochlinger, F., Fehr, M., Dymczyk, M., Schneider, T., and Siegwart, R. Y. (2018). Topomap: Topological mapping and navigation based on visual slam maps. *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–9.
- Bosse, M., Zlot, R., and Flick, P. (2012). Zebedee: Design of a spring-mounted 3-d range sensor with application to mobile mapping. *IEEE Transactions on Robotics*, 28(5):1104–1119.
- Burgard, W., Moors, M., Stachniss, C., and Schneider, F. (2005). Coordinated multi-robot exploration. *Robotics, IEEE Transactions on*, 21:376 – 386.
- Burri, M., Oleynikova, H., , Achtelik, M. W., and Siegwart, R. (2015). Real-Time Visual-Inertial Mapping, Re-localization and Planning Onboard MAVs in Unknown Environments. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems*.
- Cadena, C., Carlone, L., Carrillo, H., Latif, Y., Scaramuzza, D., Neira, J., Reid, I., and Leonard, J. J. (2016). Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on robotics*, 32(6):1309–1332.
- Chang, Y., Ebadi, K., Denniston, C. E., Ginting, M. F., Rosinol, A., Reinke, A., Palieri, M., Shi, J., Chatterjee, A., Morrell, B., Agha-mohammadi, A.-a., and Carlone, L. (2022). Lamp 2.0: A robust multi-robot slam system for operation in challenging large-scale underground environments. *IEEE Robotics and Automation Letters*, 7(4):9175–9182.
- Chen, K., Wang, J., Pang, J., Cao, Y., Xiong, Y., Li, X., Sun, S., Feng, W., Liu, Z., Xu, J., Zhang, Z., Cheng, D., Zhu, C., Cheng, T., Zhao, Q., Li, B., Lu, X., Zhu, R., Wu, Y., Dai, J., Wang, J., Shi, J., Ouyang, W., Loy, C. C., and Lin, D. (2019). MMDetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*.
- Chen-Lung, L., Jui-Te, H., Huang, C.-I., Zi-Yan, L., Chao-Chun, H., Yu-Yen, H., Siao-Cing, H., Po-Kai, C., Zu Lin, E., Po-Jui, H., Po-Lin, L., Bo-Hui, W., Lai-Sum, Y., Sheng-Wei, H., MingSian R., B., and Hsueh-Cheng, W. (2022). A heterogeneous unmanned ground vehicle and blimp robot team for search and rescue using data-driven autonomy and communication-aware navigation. *Field Robotics*, 2:557–594.
- Dang, T., Khattak, S., Mascarich, F., and Alexis, K. (2019a). Explore locally, plan globally: A path planning framework for autonomous robotic exploration in subterranean environments. In *2019 19th International Conference on Advanced Robotics (ICAR)*, pages 9–16. IEEE.
- Dang, T., Mascarich, F., Khattak, S., Nguyen, H., Nguyen, H., Hirsh, S., Reinhart, R., Papachristos, C., and Alexis, K. (2020a). Autonomous search for underground mine rescue using aerial robots. In *2020 IEEE Aerospace Conference*, pages 1–8. IEEE.
- Dang, T., Mascarich, F., Khattak, S., Papachristos, C., and Alexis, K. (2019b). Graph-based path planning for autonomous robotic exploration in subterranean environments. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3105–3112.
- Dang, T., Tranzatto, M., Khattak, S., Mascarich, F., Alexis, K., and Hutter, M. (2020b). 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 preprint arXiv:2205.05477*.
- Delmerico, J., Mintchev, S., Giusti, A., Gromov, B., Melo, K., Horvat, T., Cadena, C., Hutter, M., Ijspeert, A., Floreano, D., et al. (2019). The current state and future outlook of rescue robotics. *Journal of Field Robotics*, 36(7):1171–1191.
- Denniston, C. E., Chang, Y., Reinke, A., Ebadi, K., Sukhatme, G. S., Carlone, L., Morrell, B., and Agha-mohammadi, A.-a. (2022). Loop closure prioritization for efficient and scalable multi-robot slam. *IEEE Robotics and Automation Letters*, 7(4):9651–9658.
- Ebadi, K., Bernreiter, L., Biggie, H., Catt, G., Chang, Y., Chatterjee, A., Denniston, C. E., Deschenes, S.-P., Harlow, K., Khattak, S., et al. (2022). Present and future of slam in extreme underground environments. *arXiv preprint arXiv:2208.01787*.
- 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 *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 80–86. IEEE.
- Ebadi, K., Palieri, M., Wood, S., Padgett, C., and Agha-mohammadi, A.-a. (2021). Dare-slam: Degeneracy-aware and resilient loop closing in perceptually degraded environments. *Journal of Intelligent & Robotic Systems*, 102(1):1–25.

- Fabris, A., Kirchgeorg, S., and Mintchev, S. (2021). A soft drone with multi-modal mobility for the exploration of confined spaces. In *2021 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pages 48–54.
- Fan, D. D., Otsu, K., Kubo, Y., Dixit, A., Burdick, J., and Agha-Mohammadi, A.-A. (2021). Step: Stochastic traversability evaluation and planning for safe off-road navigation. *arXiv preprint arXiv:2103.02828*.
- Ginting, M. F., Otsu, K., Edlund, J. A., Gao, J., and Agha-Mohammadi, A.-A. (2021). Chord: Distributed data-sharing via hybrid ros 1 and 2 for multi-robot exploration of large-scale complex environments. *IEEE Robotics and Automation Letters*, 6(3):5064–5071.
- Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. *Advances in neural information processing systems*, 27.
- Harabor, D. and Grastien, A. (2011). Online graph pruning for pathfinding on grid maps. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 25, pages 1114–1119.
- Hert, D., Baca, T., Petracek, P., Kratky, V., Spurny, V., Petrlik, M., Matous, V., Zaitlik, D., Stoudek, P., Walter, V., Stepan, P., Horyna, J., Pritzl, V., Silano, G., Bonilla Licea, D., Stibinger, P., Penicka, R., Nascimento, T., and Saska, M. (2022). MRS Modular UAV Hardware Platforms for Supporting Research in Real-World Outdoor and Indoor Environments. In *2022 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE.
- Hess, W., Kohler, D., Rapp, H., and Andor, D. (2016). Real-time loop closure in 2d lidar slam. In *2016 IEEE international conference on robotics and automation (ICRA)*, pages 1271–1278. IEEE.
- Hornung, A., Wurm, K. M., Bennewitz, M., Stachniss, C., and Burgard, W. (2013). OctoMap: An Efficient Probabilistic 3D Mapping Framework Based on Octrees. *Autonomous Robots*, 34:189–206.
- Huang, Y.-W., Lu, C.-L., Chen, K.-L., Ser, P.-S., Huang, J.-T., Shen, Y.-C., Chen, P.-W., Chang, P.-K., Lee, S.-C., and Wang, H.-C. (2019). Duckiefloat: a collision-tolerant resource-constrained blimp for long-term autonomy in subterranean environments. *arXiv preprint arXiv:1910.14275*.
- Hudson, N., Talbot, F., Cox, M., Williams, J. L., Hines, T., Pitt, A., Wood, B., Frousheger, D., Surdo, K. L., Molnar, T., Steindl, R., Wildie, M., Sa, I., Kottege, N., Stepanas, K., Hernandez, E., Catt, G., Docherty, W., Tidd, B., Tam, B., Murrell, S., Bessell, M., Hanson, L., et al. (2022). Heterogeneous ground and air platforms, homogeneous sensing: Team CSIRO data61’s approach to the DARPA subterranean challenge. *Field Robotics*, 2:595–636.
- Kasper, M., McGuire, S., and Heckman, C. (2019). A benchmark for visual-inertial odometry systems employing onboard illumination. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5256–5263. IEEE.
- Khattak, S., Mascarich, F., Dang, T., Papachristos, C., and Alexis, K. (2019). Robust thermal-inertial localization for aerial robots: A case for direct methods. In *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1061–1068. IEEE.
- Khattak, S., Nguyen, H., Mascarich, F., Dang, T., and Alexis, K. (2020). Complementary multi-modal sensor fusion for resilient robot pose estimation in subterranean environments. In *2020 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1024–1029.
- Kohlbrecher, S., Meyer, J., von Stryk, O., and Klingauf, U. (2011). A flexible and scalable slam system with full 3d motion estimation. In *Proc. IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*. IEEE.
- Koval, A., Karlsson, S., Mansouri, S. S., Kanellakis, C., Tevetzidis, I., Haluska, J., Agha-mohammadi, A.-a., and Nikolakopoulos, G. (2022). Dataset collection from a sub environment. *Robotics and Autonomous Systems*, 155:104168.
- Kramer, A., Kasper, M., and Heckman, C. (2021). Vi-slam for subterranean environments. In *Field and Service Robotics*, pages 159–172. Springer.
- Kratky, V., Alcantara, A., Capitan, J., Stepan, P., Saska, M., and Ollero, A. (2021). Autonomous aerial filming with distributed lighting by a team of unmanned aerial vehicles. *IEEE Robotics and Automation Letters*, 6(4):7580–7587.
- Kratky, V., Petracek, P., Baca, T., and Saska, M. (2021a). An autonomous unmanned aerial vehicle system for fast exploration of large complex indoor environments. *Journal of Field Robotics*, 38(8):1036–1058.
- Kratky, V., Petracek, P., Nascimento, T., Cadilova, M., Skobrtal, M., Stoudek, P., and Saska, M. (2021b). Safe documentation of historical monuments by an autonomous unmanned aerial vehicle. *ISPRS International Journal of Geo-Information*, 10(11):738/1–16.
- Kulkarni, M., Dharmadhikari, M., Tranzatto, M., Zimmermann, S., Reijgwart, V., De Petris, P., Nguyen, H., Khedekar, N., Papachristos, C., Ott, L., et al. (2021). Autonomous teamed exploration of subterranean environments using legged and aerial robots. *arXiv preprint arXiv:2111.06482*.

- Lajoie, P.-Y., Ramtoula, B., Chang, Y., Carlone, L., and Beltrame, G. (2020). Door-slam: Distributed, online, and outlier resilient slam for robotic teams. *IEEE Robotics and Automation Letters*, 5(2):1656–1663.
- Lee, T. et al. (2010). Geometric tracking control of a quadrotor UAV on SE(3). In *2010 IEEE Conference on Decision and Control*, pages 5420–5425. IEEE.
- Lei, X., Kim, T., Marchal, N., Pastor, D., Ridge, B., Scholler, F., Terry, E., Chavez, F., Touma, T., Otsu, K., et al. (2022). Early recall, late precision: Multi-robot semantic object mapping under operational constraints in perceptually degraded environments. *arXiv preprint arXiv:2206.10062*.
- Lindqvist, B., Kanellakis, C., Mansouri, S. S., Agha-mohammadi, A.-a., and Nikolakopoulos, G. (2021). Compra: A compact reactive autonomy framework for subterranean mav based search-and-rescue operations. *arXiv preprint arXiv:2108.13105*.
- Lindqvist, B., Karlsson, S., Koval, A., Tevetzidis, I., Haluska, J., Kanellakis, C., Agha-mohammadi, A.-a., and Nikolakopoulos, G. (2022). Multimodality robotic systems: Integrated combined legged-aerial mobility for subterranean search-and-rescue. *Robotics and Autonomous Systems*, 154:104134.
- Loshchilov, I. and Hutter, F. (2016). Sgdr: Stochastic gradient descent with warm restarts.
- Lu, C. X., Rosa, S., Zhao, P., Wang, B., Chen, C., Stankovic, J. A., Trigoni, N., and Markham, A. (2020). See through smoke: robust indoor mapping with low-cost mmwave radar. In *Proceedings of the 18th International Conference on Mobile Systems, Applications, and Services*, pages 14–27.
- Martinez-Rozas, S., Rey, R., Alejo, D., Acedo, D., Cobano, J. A., Rodriguez-Ramos, A., Campoy, P., Merino, L., and Caballero, F. (2022). An aerial/ground robot team for autonomous firefighting in urban gnss-denied scenarios. *Field Robotics*, 2:241–273.
- Moniruzzaman, M., Rassau, A., Chai, D., and Islam, S. M. S. (2022). Teleoperation methods and enhancement techniques for mobile robots: A comprehensive survey. *Robotics and Autonomous Systems*, 150:103973.
- Murphy, R. R., Kravitz, J., Stover, S. L., and Shoureshi, R. (2009). Mobile robots in mine rescue and recovery. *IEEE Robotics & Automation Magazine*, 16(2):91–103.
- Museth, K. (2013). Vdb: High-resolution sparse volumes with dynamic topology. *ACM transactions on graphics (TOG)*, 32(3):1–22.
- Musil, T., Petrlík, M., and Saska, M. (2022). Spheremap: Dynamic multi-layer graph structure for rapid safety-aware uav planning. *IEEE Robotics and Automation Letters*, 7(4):11007–11014.
- 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*.
- Orekhov, V. and Chung, T. (2022). The darpa subterranean challenge: A synopsis of the circuits stage. *Field Robotics*, 2:735–747.
- O’Meadhra, C., Tabib, W., and Michael, N. (2018). Variable resolution occupancy mapping using gaussian mixture models. *IEEE Robotics and Automation Letters*, 4(2):2015–2022.
- Palieri, M., Morrell, B., Thakur, A., Ebadi, K., Nash, J., Chatterjee, A., Kanellakis, C., Carlone, L., Guaragnella, C., and Agha-mohammadi, A.-a. (2020). 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.
- Papachristos, C., Khattak, S., and Alexis, K. (2017). Uncertainty-aware receding horizon exploration and mapping using aerial robots. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 4568–4575. IEEE.
- Papachristos, C., Khattak, S., Mascarich, F., and Alexis, K. (2019a). Autonomous navigation and mapping in underground mines using aerial robots. In *2019 IEEE Aerospace Conference*, pages 1–8. IEEE.
- Papachristos, C., Mascarich, F., Khattak, S., Dang, T., and Alexis, K. (2019b). Localization uncertainty-aware autonomous exploration and mapping with aerial robots using receding horizon path-planning. *Autonomous Robots*, 43(8):2131–2161.
- Petracek, P., Kratky, V., Petrlík, M., Baca, T., Kratochvil, R., and Saska, M. (2021). Large-Scale Exploration of Cave Environments by Unmanned Aerial Vehicles. *IEEE Robotics and Automation Letters*, 6(4):7596–7603.
- Petrlík, M., Baca, T., Hert, D., Vrba, M., Krajník, T., and Saska, M. (2020). A Robust UAV System for Operations in a Constrained Environment. *IEEE Robotics and Automation Letters*, 5(2):2169–2176.
- Petrlík, M., Krajník, T., and Saska, M. (2021). Lidar-based stabilization, navigation and localization for uavs operating in dark indoor environments. In *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 243–251. IEEE.

- Pritzl, V., Stepan, P., and Saska, M. (2021). Autonomous flying into buildings in a firefighting scenario. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 239–245. IEEE.
- Pritzl, V., Vrba, M., Stepan, P., and Saska, M. (2022a). Cooperative navigation and guidance of a micro-scale aerial vehicle by an accompanying uav using 3d lidar relative localization. In *2022 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 526–535. IEEE.
- Pritzl, V., Vrba, M., Stepan, P., and Saska, M. (2022b). Cooperative navigation and guidance of a micro-scale aerial vehicle by an accompanying uav using 3d lidar relative localization. In *2022 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 526–535. IEEE.
- Queraltá, J. P., Taipalmaa, J., Pullinen, B. C., Sarker, V. K., Gia, T. N., Tenhunen, H., Gabbouj, M., Raitoharju, J., and Westerlund, T. (2020). Collaborative multi-robot search and rescue: Planning, coordination, perception, and active vision. *Ieee Access*, 8:191617–191643.
- Redmon, J. and Farhadi, A. (2018). Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*.
- Reinke, A., Palieri, M., Morrell, B., Chang, Y., Ebadi, K., Carlone, L., and Agha-Mohammadi, A.-A. (2022). Locus 2.0: Robust and computationally efficient lidar odometry for real-time 3d mapping. *IEEE Robotics and Automation Letters*, 7(4):9043–9050.
- Reynolds, D. A. (2009). Gaussian mixture models. *Encyclopedia of biometrics*, 741(659-663).
- Richter, C., Bry, A., and Roy, N. (2016). Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments. In *Robotics Research*, pages 649–666. Springer.
- Rogers, J. G., Gregory, J. M., Fink, J., and Stump, E. (2020a). Test your slam! the subt-tunnel dataset and metric for mapping. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 955–961.
- Rogers, J. G., Schang, A., Nieto-Granda, C., Ware, J., Carter, J., Fink, J., and Stump, E. (2020b). The darpa subt urban circuit mapping dataset and evaluation metric. In *International Symposium on Experimental Robotics*, pages 391–401. Springer.
- Roucek, T., Pecka, M., Cizek, P., Petricek, T., Bayer, J., Salansky, V., Hert, D., Petrlik, M., Baca, T., Spurny, V., et al. (2019). Darpa subterranean challenge: Multi-robotic exploration of underground environments. In *International Conference on Modelling and Simulation for Autonomous Systems*, pages 274–290. Springer.
- Roucek, T., Pecka, M., Cizek, P., Petricek, T., Bayer, J., Salansky, V., Hert, D., Petrlik, M., Baca, T., Spurny, V., Pomerleau, F., Kubelka, V., Faigl, J., Zimmermann, K., et al. (2020). Darpa subterranean challenge: Multi-robotic exploration of underground environments. In Mazal, J., Fagiolini, A., and Vasik, P., editors, *Modelling and Simulation for Autonomous Systems*, pages 274–290, Cham. Springer International Publishing.
- Saboia, M., Clark, L., Thangavelu, V., Edlund, J. A., Otsu, K., Correa, G. J., Varadharajan, V. S., Santamaria-Navarro, A., Touma, T., Bouman, A., et al. (2022). Achord: Communication-aware multi-robot coordination with intermittent connectivity. *arXiv preprint arXiv:2206.02245*.
- Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L.-C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520.
- Santamaria-Navarro, A., Thakker, R., Fan, D. D., Morrell, B., and Agha-mohammadi, A.-a. (2019). Towards resilient autonomous navigation of drones. In *The International Symposium of Robotics Research*, pages 922–937. Springer.
- Scherer, S., Agrawal, V., Best, G., Cao, C., Cujic, K., Darnley, R., DeBortoli, R., Dexheimer, E., Drozd, B., Garg, R., et al. (2022). Resilient and modular subterranean exploration with a team of roving and flying robots. *Field Robotics*, 2:678–734.
- Shakhatreh, H., Sawalmeh, A. H., Al-Fuqaha, A., Dou, Z., Almaita, E., Khalil, I., Othman, N. S., Khreishah, A., and Guizani, M. (2019). Unmanned aerial vehicles (uavs): A survey on civil applications and key research challenges. *Ieee Access*, 7:48572–48634.
- Shan, T., Englot, B., Meyers, D., Wang, W., Ratti, C., and Daniela, R. (2020). Lio-sam: Tightly coupled lidar inertial odometry via smoothing and mapping. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5135–5142. IEEE.
- Shin, Y.-S., Park, Y. S., and Kim, A. (2020). Dvl-slam: sparse depth enhanced direct visual-lidar slam. *Autonomous Robots*, 44(2):115–130.
- Silano, G., Baca, T., Penicka, R., Liuzza, D., and Saska, M. (2021). Power line inspection tasks with multi-aerial robot systems via signal temporal logic specifications. *IEEE Robotics and Automation Letters*, 6(2):4169–4176.
- Smith, L. N. (2015). Cyclical learning rates for training neural networks.

- Spurny, V., Pritzl, V., Walter, V., Petrlík, M., Baca, T., Stepan, P., Zaitlik, D., and Saska, M. (2021). Autonomous firefighting inside buildings by an unmanned aerial vehicle. *IEEE Access*, 9:15872–15890.
- Tardioli, D., Riazuelo, L., Sicignano, D., Rizzo, C., Lera, F., Villarroel, J. L., and Montano, L. (2019). Ground robotics in tunnels: Keys and lessons learned after 10 years of research and experiments. *Journal of Field Robotics*, 36(6):1074–1101.
- Tomic, T., Schmid, K., Lutz, P., Domel, A., Kassecker, M., Mair, E., Grixia, I. L., Ruess, F., Suppa, M., and Burschka, D. (2012). Toward a fully autonomous uav: Research platform for indoor and outdoor urban search and rescue. *IEEE robotics & automation magazine*, 19(3):46–56.
- Tordesillas, J., Lopez, B. T., Everett, M., and How, J. P. (2022). Faster: Fast and safe trajectory planner for navigation in unknown environments. *IEEE Transactions on Robotics*, 38(2):922–938.
- Tranzatto, M., Dharmadhikari, M., Bernreiter, L., Camurri, M., Khattak, S., Mascarich, F., Pfreundschuh, P., Wisth, D., Zimmermann, S., Kulkarni, M., et al. (2022a). Team cerberus wins the darpa subterranean challenge: Technical overview and lessons learned. *arXiv preprint arXiv:2207.04914*.
- Tranzatto, M., Mascarich, F., Bernreiter, L., Godinho, C., Camurri, M., Khattak, S., Dang, T., Reijgwart, V., Loeje, J., Wisth, D., Zimmermann, S., Nguyen, H., Fehr, M., Solanka, L., Buchanan, R., Bjelonic, M., Khedekar, N., Valceschini, M., Jenelten, F., and Alexis, K. (2022b). Cerberus: Autonomous legged and aerial robotic exploration in the tunnel and urban circuits of the darpa subterranean challenge. *Field Robotics*, 2:274–324.
- Vrba, M., Hert, D., and Saska, M. (2019). Onboard marker-less detection and localization of noncooperating drones for their safe interception by an autonomous aerial system. *IEEE Robotics and Automation Letters*, 4(4):3402–3409.
- Walter, V., Novak, T., and Saska, M. (2018). Self-localization of unmanned aerial vehicles based on optical flow in onboard camera images. In *Lecture Notes in Computer Science, vol 10756.*, Cham. Springer International Publishing.
- Walter, V., Spurny, V., Petrlík, M., Baca, T., Zaitlik, D., Demkiv, L., , and Saska, M. (2022). Extinguishing real fires by fully autonomous multirotor uavs in the mbzirc 2020 competition. *Field Robotics*, 2:406–436.
- Williams, J., Jiang, S., O’Brien, M., Wagner, G., Hernandez, E., Cox, M., Pitt, A., Arkin, R., and Hudson, N. (2020). Online 3d frontier-based ugv and uav exploration using direct point cloud visibility. In *2020 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, pages 263–270. IEEE.
- Xu, W., Cai, Y., He, D., Lin, J., and Zhang, F. (2022). Fast-lio2: Fast direct lidar-inertial odometry. *IEEE Transactions on Robotics*.
- Zhang, J., Kaess, M., and Singh, S. (2016). On degeneracy of optimization-based state estimation problems. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 809–816. IEEE.
- Zhang, J. and Singh, S. (2014). Loam: Lidar odometry and mapping in real-time. In *Robotics: Science and Systems*, volume 2.
- Zhang, J. and Singh, S. (2015). Visual-lidar odometry and mapping: Low-drift, robust, and fast. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2174–2181. IEEE.
- Zhao, S., Zhang, H., Wang, P., Nogueira, L., and Scherer, S. (2021). Super odometry: Imu-centric lidar-visual-inertial estimator for challenging environments. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8729–8736.
- Zhou, B., Pan, J., Gao, F., and Shen, S. (2021a). Raptor: Robust and perception-aware trajectory replanning for quadrotor fast flight. *IEEE Transactions on Robotics*, 37(6):1992–2009.
- Zhou, B., Zhang, Y., Chen, X., and Shen, S. (2021b). Fuel: Fast uav exploration using incremental frontier structure and hierarchical planning. *IEEE Robotics and Automation Letters*, 6(2):779–786.

How to cite this article: Petrlík, M., Petráček, P., Krátký, V., Musil, T., Stasinchuk, Y., Vrba, M., Báča, T., Heřt, D., Pecka, M., Svoboda, T., & Saska, M. (2023). UAVs beneath the surface: Cooperative autonomy for subterranean search and rescue in DARPA SubT. *Field Robotics*, 3, 1–68.

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