Resilient Multi-Sensor Exploration of Multifarious Environments with a Team of Aerial Robots

Graeme Best[1*], Rohit Garg[2*], John Keller[2*], Geoffrey A. Hollinger[1], and Sebastian Scherer[2]

[1]CoRIS Institute, Oregon State University, Corvallis OR, USA.
[2]Robotics Institute, Carnegie Mellon University, Pittsburgh PA, USA.
[*]These authors contributed equally to this work.
Corresponding author: Graeme Best graeme.best@uts.edu.au

***Abstract*—We present a coordinated autonomy pipeline for multi-sensor exploration of confined environments. We simultaneously address four broad challenges that are typically overlooked in prior work: (a) make effective use of both range and vision sensing modalities, (b) perform this exploration across a wide range of environments, (c) be resilient to adverse events, and (d) execute this onboard a team of physical robots. Our solution centers around a behavior tree architecture, which adaptively switches between various behaviors involving coordinated exploration and responding to adverse events. Our exploration strategy exploits the benefits of both visual and range sensors with a new frontier-based exploration algorithm. The autonomy pipeline is evaluated with an extensive set of field experiments, with teams of up to 3 robots that fly up to 3 m/s and distances exceeding one kilometer. We provide a summary of various field experiments and detail resilient behaviors that arose: maneuvering narrow doorways, adapting to unexpected environment changes, and emergency landing. We provide an extended discussion of lessons learned, release software as open source, and present a video in the supplementary material.**

I. Introduction

There is a growing need for the autonomous exploration of underground or indoor spaces with realistic constraints on mobility, environmental degradation, and communication limitations. Achieving this will enable applications of robots in situations previously infeasible: for example, finding assets in collapsed mines, preparing for hazardous scenarios in urban infrastructure, and making scientific observations of geometrically-complex natural caves. Prior work has primarily considered partial solutions to parts of the problem of exploration, planning, or ensuring robust behavior. Here, we are interested in complete autonomy solutions, without human intervention from takeoff to landing, for enabling robust flight of teams of aerial robots in a large range of environments—natural caves, limestone and coal mines, building complexes, boiler plants, and the DARPA Subterranean Challenge (SubT) course [9]—without requiring adjustments to the software.

To realize this capability, we believe an effective solution is required to *simultaneously* address four broad challenges:

(a) Make effective use of multiple sensing modalities in a multi-sensor, multi-robot exploration team, especially by making close-range observations of the environment with vision sensors that have limited field of view and range.

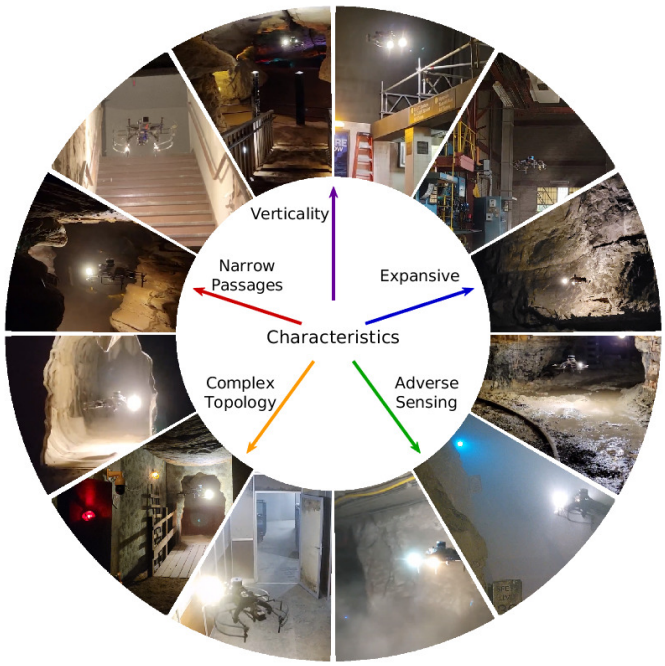


Fig. 1. Our aerial robots exploring a wide range of environments: caves, mines, and abandoned buildings. Each environment has unique characteristics in many different forms, which pose new challenges for navigation, exploration, and perception. In each case, the robots employ our proposed coordinated autonomy pipeline for multi-sensor exploration, and adapt to the various challenges and adverse events, without having any prior knowledge of the environment being explored.

(b) Perform this exploration effectively across a wide range of environments with varying features—e.g., narrow doorways, stairs, expansive rooms, dust, dynamic obstacles, and strongly-attenuated communication—without adjusting the software between environments.

(c) Operate autonomously throughout the mission, from takeoff to landing, while being resilient to environment-specific challenges and system faults.

(d) Execute all of the above on physical robots moving at high speeds, while respecting the limits of onboard computation, sensing, and communication.

This is the problem we solve in this paper.

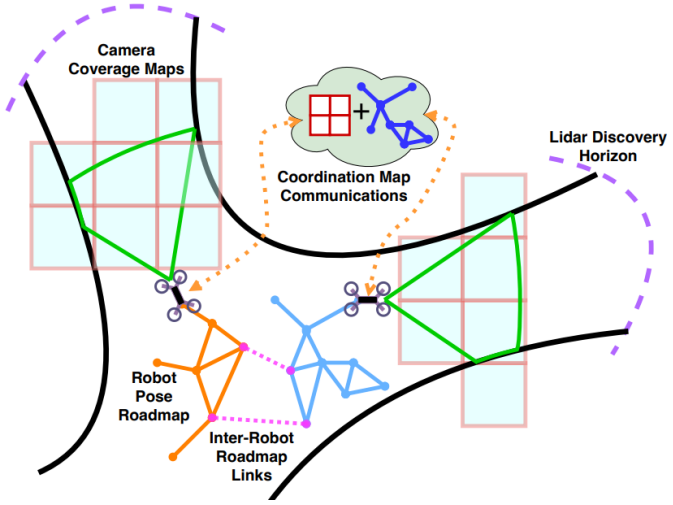


Fig. 2. An illustration of the multi-sensor, multi-robot exploration concept and our solution. The robots visually observe the surfaces of an unknown environment with coordinated autonomy that is resilient to the various challenges present in the cave, mine, and urban environments of Fig. 1.

Our proposed exploration autonomy pipeline directly addresses each of the above challenges in the following ways:

(a) A new exploration strategy that selects viewpoints that benefit both vision and range sensing while making realistic assumptions regarding the sensing and motion. Furthermore, exploration is performed in a coordinated manner, with robots sharing several maps and switching between coordination strategies to adapt to the situation. Figure 2 illustrates this concept.

(b) A robust hierarchy of path planning algorithms and associated map processing techniques that enables adaptive exploration behavior and speed to best suit the environment geometry. Figure 1 illustrates our robots executing missions in a wide range of environments.

(c) Resiliency through the use of a behavior tree plan execution architecture [10] that selects between multiple alternate behaviors when an adverse event is detected, such as recovering from being stuck in dust, emergency landing in a safe zone, and switching to alternate coordination strategies when the primary strategy is failing.

(d) We make careful design decisions regarding how each component of the pipeline uses computational resources. Also, the robots effectively explore even if communication is limited. We tested design choices extensively on physical robots during algorithm development.

We present a selection of results from our extensive field testing and illustrate examples of resilient behaviors in adverse conditions. We demonstrate our autonomy on robots exploring at up to 3 m/s and traveling up to 1100 m in 15 minute flights, with teams of up to 3 robots. These experiments were performed in the environments of Fig. 1, including the SubT Challenge course [9]. We describe the behavior of the algorithms in various circumstances, and comment on lessons learned. The discussion is supported by illustrations in the appendix, and a video in the supplementary material[1].

In summary, the contributions of this paper are:

- A comprehensive approach to resilient multi-sensor, multi-robot exploration and sensor coverage, which makes many specific innovations at the component level, to enable exploration with physical teams of robots in a wide range of unknown environments. The robots leverage LiDAR for discovering the geometry of the environment and cameras for yielding detailed observations. Our approach is demonstrated to be resilient to adverse events, including unexpected battery loss, dynamic obstacles, being stuck in dust, and unreachable areas.
- Evaluation in a large variety of environments with up to 3 robots, and detailed discussion, diagrams, and videos from these extensive field experiments that demonstrate the behavior of our adaptive and resilient autonomy.
- We release our software implementation as open source to the community[2].

Relative to our prior publication regarding the broader SubT system [31], this paper contributes design improvements to all components, several new resilient behaviors, extended technical detail, and recent field experiments.

II. Related Work

While the vast majority of studies on aerial robot exploration have focused solely on optimizing the primary exploration behavior, there has been a few recent studies that propose approaches to full-mission autonomy with finite state machines [14, 16, 17, 25] and behavior trees [23, 26]. We are particularly interested in behavior trees [10], which are common in computer games and have recently gained popularity in robotics [27, 29, 32], due to the ease of design and introspection. We push the boundary of how behavior trees are used for aerial robotics, by instantiating them for missions involving many complex behaviors that aid resiliency.

Exploration algorithms are loosely characterized as information theoretic [6, 11, 19, 34], which are seen as mathematically sound, frontier-based [7, 17, 20, 28, 37, 40], which are computationally efficient, or graph search-based [8, 12, 13, 38], which are scalable. Our pipeline is primarily frontier-based, but with a new frontier definition and selection heuristic, and also uses graph search for specific behaviors.

Multi-sensor exploration combines classic exploration, as the world needs to be discovered, and sensor coverage, since we closely observe the discovered world. Existing approaches to multi-sensor problems involve optimizing for coverage objectives [8], creating separate viewpoints for exploration and coverage [4, 36], or combining exploration and coverage into a single objective [21, 22, 33]. We opt for an approach that creates separate viewpoints for exploration and coverage, similar to [36] in the context of underwater exploration, as we found that viewpoints that favor coverage are most beneficial, but there are specific scenarios where exploration viewpoints

[1] Video: https://youtu.be/c223fYNOmf4
[2] Software: https://theairlab.org/research/2022/05/02/subt_code/

are necessary. Similarly to [36], we select viewpoints that observe frontiers rather than planning directly to frontiers.

For multi-robot exploration, coordination strategies are either explicit [5, 7, 11], where robots jointly optimize plans, or implicit [2, 3], where robots only share knowledge of the world. We employ an implicit strategy as it is less burdensome on communication and our robots are usually spread out.

Aerial exploration strategies are typically combined with local path planners, which enable faster and safer navigation. Common approaches include trajectory optimizers [40], graph search [17], and motion primitives [13]. We use a two-layer approach that combines graph search with motion primitives.

Common map representations include raw point clouds [8] or voxel structures like octrees [35, 36]. We employ OpenVDB [24], which is an alternative hierarchical voxel structure widely used in graphics engines. OpenVDB is yet to receive much attention in robotics, but is befitting due to the efficient queries and iterators. We also employ efficient distance transform maps [30] for fast local planning.

SubT Challenge: While our proposed system is intended to be generally applicable, it was primarily developed for the context of the SubT Challenge [9] by Team Explorer. In the final competition, all teams, including ours, exclusively or heavily used ground robots. However, our aerial robots quickly explored regions that were difficult or impossible to reach by ground robots, which collectively contributed to us winning the "Most Sectors Explored" award. To our knowledge, our aerial robots outperformed the aerial robots deployed by other teams in terms of volume explored, flight speed, doorways navigated, points scored (∼1/3 of our total score), and overall reliability[3]. Further details of our performance are discussed in Sec. VIII.

The autonomy approaches described by the published work of SubT teams share similar planning components, such as roadmap generation [1, 12, 15, 31], frontier-based exploration [1, 12, 15, 17, 31], and map sharing [1, 15, 31]. Notable absences from this published work include explicit multi-sensor exploration and coverage objectives, behavior switching architectures for full-mission autonomy and resiliency, a diverse set of mapping products designed for specialized purposes, and aerial robot navigation through narrow doorways; we propose new solutions to each of these problems within our presented system.

III. Problem Formulation

We consider the problem of planning for a team of aerial robots where the goal is to visually observe the surfaces of an unknown environment. Each robot is equipped with range sensors that discover the geometry of the environment, and close-range vision sensors for observing surfaces. We are interested in developing an autonomy pipeline for the full mission—from take-off to landing—that accomplishes the mission objectives and is resilient to any challenges that may arise. The autonomy must be resilient across a wide range of environments, which may feature challenges such as narrow passages, dust, limited communication, unreachable regions, dynamic obstacles, and unexpected battery drain. We formalize this problem as follows.

[3]SubT final event footage: https://www.subtchallenge.com/SubTv.html

A. Environment

The environment is described as a discrete set $\mathcal{E}$ of points $e \in \mathbb{R}^3$. The set $\mathcal{E}$ is split into two disjoint subsets: $\mathcal{E}^{free} \subset \mathcal{E}$ describes all free-space locations, and $\mathcal{E}^{occ} \subset \mathcal{E}$ describes all occupied locations. We additionally define the set of surface points $\mathcal{E}^{surf}$, where $\mathcal{E}^{surf} \subset \mathcal{E}^{occ}$, as all occupied points that are on the boundary of $\mathcal{E}^{occ}$ and $\mathcal{E}^{free}$. These sets are initially unknown to the robots, and their geometric features are unpredictable, with the possible presence of narrow passages, open rooms, vertical passages, stairs, branching, and loops.

B. Robot State and Action Space

A team of R aerial robots, $\mathcal{R} = \{r^1, ..., r^R\}$, are to explore this environment. At time t, the state of robot r^i is defined as a pose $x^i(t) \in SE(3)$. The set of collision-free poses is denoted $\mathcal{X}$, where $\mathcal{X} \subset SE(3)$. Robot r^i takes off at a given time t_0^i with a known pose $x^i(t_0^i)$. The union of the $\mathbb{R}^3$ translation components of each $x \in \mathcal{X}$ forms a subset of $\mathcal{E}^{free}$. For now, we assume $\mathcal{X}$ is constant over time, but we later discuss the possibility of time-varying obstacles. This set is also initially unknown to the robots.

Each robot can move freely within $\mathcal{X}$ with a finite velocity and acceleration. The yaw of each pose is also specifiable by the high-level planners, subject to a finite angular velocity, but the roll and pitch are not controllable and remain near horizontal. Each robot has an onboard controller to facilitate these movements; the details of the controller are out of scope of this paper and the proposed planner does not rely on having a precise model of the robot dynamics.

Each robot has an uncertain battery capacity that allows it to fly for time T^i. Before time $t_0^i + T^i$, robot i is to return to near the take-off pose $x^i(t_0^i)$ and land safely on the ground. Given that T^i is difficult to predict, or a robot may fail to find a path home, each robot must also have a contingency plan to land safely in case $x^i(t_0^i)$ is not reached prior to $t_0^i + T^i$.

C. Sensor Models

Each robot is equipped with one or more range sensors that provide information about the environment geometry. Each observation is a set of range estimates to the nearest surface points $e \in \mathcal{E}^{surf}$ in a fixed set of directions relative to the robot pose $x^i(t)$. These observations may contain noise in the form of false 'hits' off dust or other air particles, or false 'misses' through windows and thin obstacles.

Each robot is also equipped with one or more vision sensors that perform a primary perception task, such as object detection or providing imagery to an operator. It is assumed that these cameras are effective at their task of observing the nearest surfaces up to a fixed range from the robot and within the camera field of view. At time t, the observation yields a set of observed surface points $z^i(t)$, where $z^i(t) \subset \mathcal{E}^{surf}$.

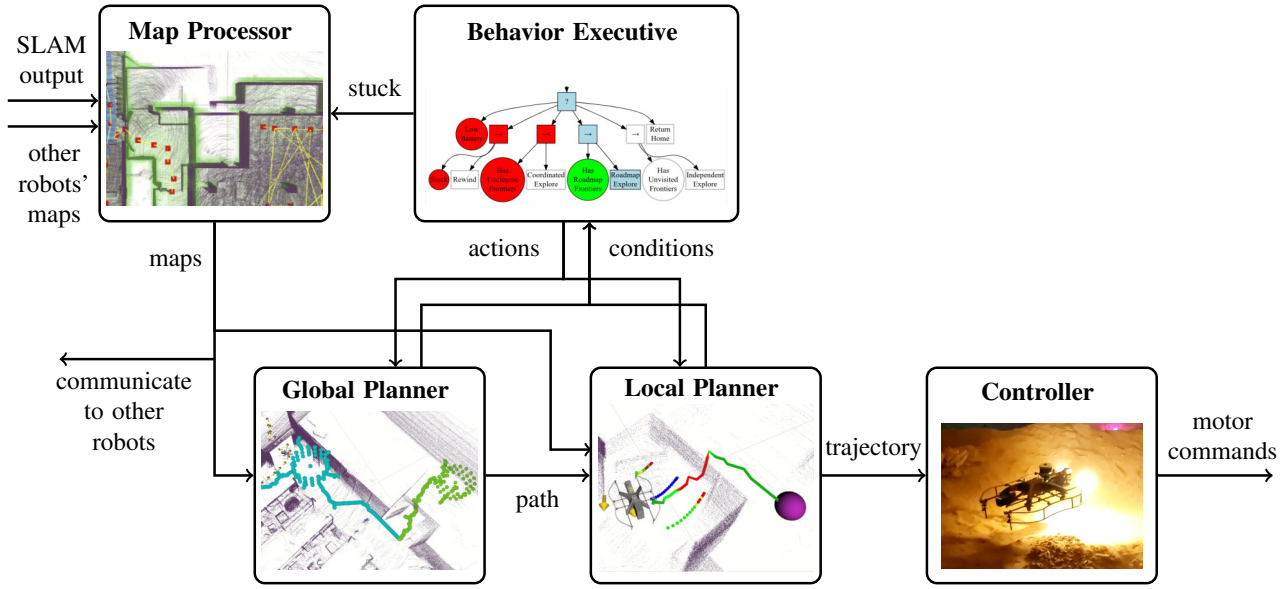


Fig. 3. Overview of the proposed aerial autonomy pipeline, as introduced in Sec. IV

Note that the robot does not necessarily have full knowledge of $z^i(t)$ as $\mathcal{E}^{surf}$ is unknown to the robots, but may be estimated indirectly via information obtained by the range sensors; this is discussed further in Sec. V.

We assume the current pose $x^i(t)$ is accurately estimated by each robot, such as through a SLAM solution, but the proposed pipeline should be resilient to small errors. In our experiments, we utilize the *Super Odometry* SLAM solution [39].

D. Multi-Robot Communication

The robots have onboard communication hardware that enables communicating information between robots. However, the connectivity and bandwidth of the network is unpredictable and limited, and this should be considered when designing what messages are sent between robots.

We assume that robots know the coordinate transforms between their take-off locations; this allows communicated maps to be shared in a common global reference frame.

E. Resilience to Unforeseen Challenges

The robots should be prepared to appropriately respond to all unexpected challenges that may arise during the mission. These challenges include the issues alluded to above—such as dust or fog, changes in environment topology, and unpredictable battery life—as well as any unforeseen challenges that may prevent the continuation of typical mission behavior. In any case, the robots are to respond in a reasonable manner similar to how a remotely-operating human pilot may respond in a similar situation, such as finding alternative paths, backtracking from an area, or landing safely at an alternate location.

F. Problem Statement

Denote the trajectory $x^i(t), \forall t \in [t_0^i, t_0^i + T^i]$ of robot i as X^i, and the collection of trajectories for all robots as X. Define the union of the sets of all points visually observed by all robots throughout the trajectories X as

$$\mathcal{E}^{obs}(X) = \bigcup_i \bigcup_t z^i(t). \tag{1}$$

Given these definitions, the problem to be solved is stated as follows. We aim to find the optimal collection of trajectories, denoted X^*, that respects the assumptions and constraints introduced in the above subsections and maximises the number of points observed by the visual sensors of at least one robot, denoted $\mathcal{E}^{obs}(X)$; i.e., find

$$X^* = \underset{X}{\operatorname{argmax}} |\mathcal{E}^{obs}(X)|. \tag{2}$$

Since communication is intermittent, and the team operates in a decentralized manner, each robot has to solve for its own trajectory based on its own knowledge of the world. Also, since the surface points, $\mathcal{E}^{surf}$, are not known in advance, this problem cannot be solved directly. Instead, strategies must be developed that plan to both discover $\mathcal{E}^{surf}$ and observe these discovered surfaces with vision sensors.

IV. Autonomy Overview

We propose an autonomy solution to this problem that emphasizes resiliency from take-off to landing, across a wide range of environments. Figure 3 presents a high-level overview of our solution and Fig. 4 illustrates a snapshot of these components working together during an example mission.

Central to the decision making is the *behavior executive*: It employs a behavior tree logic structure to reason over various signals that describe the state of the system to decide which behavior to execute right now, such as a particular exploration method, backtrack out of trouble, or a landing sequence. The *map processor* processes mapping data provided by SLAM to generate various mapping products for the different planners

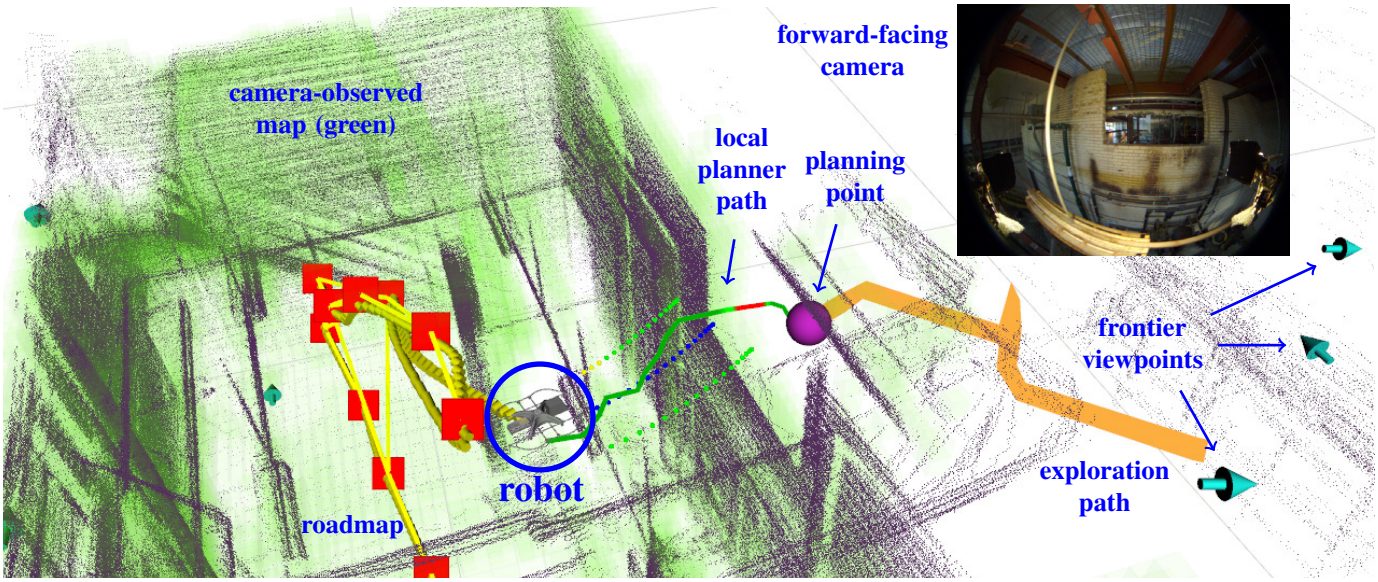


Fig. 4. A robot exploring a boiler plant in Pittsburgh, PA. The robot has finished observing the room on the left (green points), so it plans a path between beams hanging from the ceiling and a small hole in the wall (visible in camera field of view) to reach the frontier viewpoints in the unexplored room on the right. All computation is performed online and onboard the robot. The point cloud is generated with an onboard LiDAR sensor and SLAM solution.

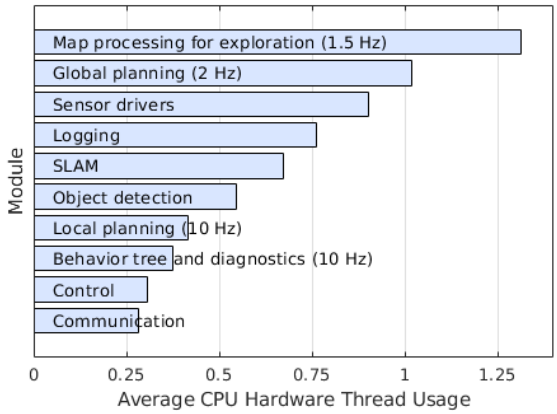


Fig. 5. The number of CPU hardware threads used by the modules of the autonomy pipeline, averaged over a full flight. Execution rate listed for some modules. This was run on a single 8-CPU thread (4-core) computer. Note: This paper focuses on the map processing, global planning, local planning, behavior tree and diagnostics modules.

and other robots. The *global planner* selects goals based on the currently selected behavior, and plans paths to ensure feasibility and achieve these goals. Low fidelity maps are opportunistically communicated between robots. The *local planner* takes intermediate goals from the global planner, plans a path while ensuring a higher degree of safety, adapts the speed, and controls the yaw during tight maneuvers. Trajectory commands are output to an onboard controller.

Our primary design consideration is the trade-off between the prospective strengths of an algorithm and its burden on available time, computation, and communication resources. This is achieved by maintaining a tight loop between development and regular testing, such that we aim for the system to become resilient to issues that arise during experiments. We closely monitored the use of computational resources during development; typical CPU usage is depicted in Fig. 5.

A. Behavior Executive

Our behavior tree, depicted in Fig. 6, commands the high-level decisions. A behavior tree models plan execution with a directed tree [10]. *Condition* nodes are evaluated as true or false based on the system state. An *action* node describes a behavior that is executed when active, and evaluates as success, running, or failure. The tree is evaluated from left to right, with the internal nodes recursively controlling which leaf nodes are active. *Fallback* nodes are true if any of its children are true, while *sequence* nodes are false if any of its children are false. The left-most children of an active *fallback* node are active up to the first evaluated as true/success or running. Similarly, the children of an active *sequence* node are active up to the first false/failure or running.

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

The behavior tree specifies several different exploration behaviors. If the robot is stuck, then it rewinds by backtracking blindly over the previous trajectory. If it has frontier viewpoints not observed by other robots, then it executes coordinated exploration. Otherwise, it travels to viewpoints communicated by other robots. If neither of these are possible, then independent exploration is performed. As a last resort,

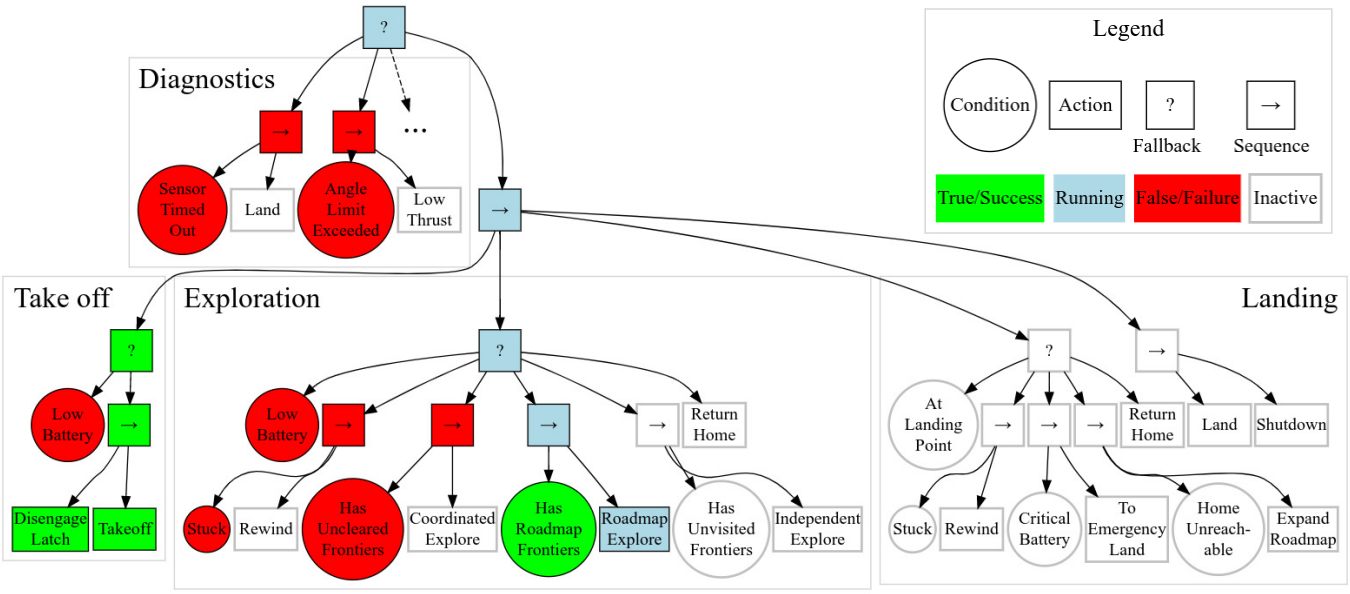


Fig. 6. Our behavior tree is responsible for adaptively switching between autonomous behaviors (actions) as the conditions change. A behavior tree is interpreted from left to right, as introduced in Sec. IV-A. In the pictured state (colors), there are no diagnostics errors, the robot has successfully taken off, the battery is not low, and so it is currently exploring. The 'Roadmap Explore' behavior is selected because all locally-computed frontiers have been cleared by other robots, but other robots have shared a roadmap containing paths to frontiers beyond the sensor range.

the robot returns home. If a higher-priority condition becomes true, then the associated behavior is immediately activated.

Once the battery life is less than the estimated travel time to the take-off location, the robot follows the roadmap back and lands. For improved resiliency, we also have fallback behaviors: backtrack if stuck, emergency land at a safe landing zone, or repair the roadmap if the path is blocked. We further describe these behaviors in the following sections.

V. Map Processing

Our multi-sensor autonomy pipeline uses maps for several purposes: remembering what has been observed, discovering what has not been observed, collision avoidance, coordination, and returning home. As such, we require processing the data from the range and vision sensors to produce mapping products that are specialized for each purpose. *Roadmaps* provide a topological representation of where the robots have been and how the environment is connected. *Collision checking maps* are required by the global and local planners, each with different considerations. *Exploration maps* inform the global planner of the value of future observations.

A. Exploration Maps

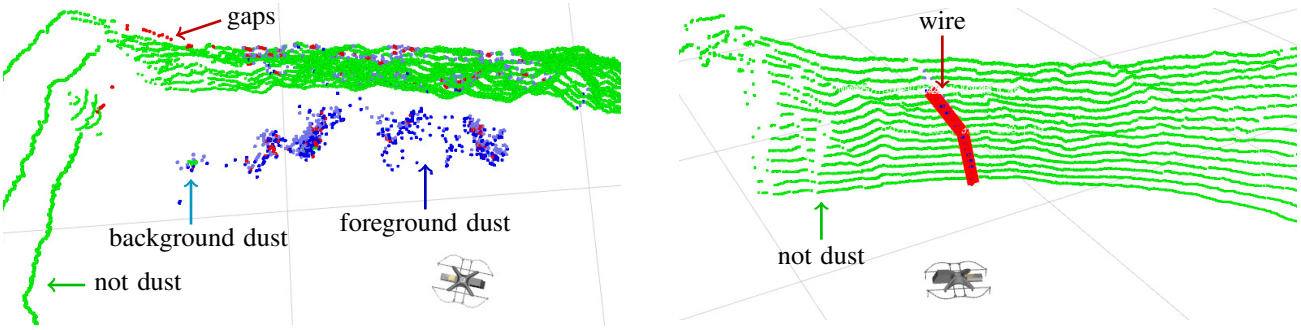
The exploration maps are represented as occupancy grids with the OpenVDB data structure [24]. OpenVDB is a hierarchical grid that resembles a B+ Tree, and provides an average constant-time random access to voxel data. This property enables fast ray-casting operations for probabilistic occupancy updates and collision checking, and therefore OpenVDB is ideal for constructing large-scale maps in real time.

1) Collision Checking: The *global map* used for collision checking is a high resolution (10 cm) probabilistic 3D occupancy grid generated by ray casting with registered range-sensor scans, and thus is a partial estimate of $\mathcal{E}$. This map covers the entire observed environment to facilitate long-horizon planning; this is in contrast to the local planner map discussed below. We write these maps to shared memory to enable low-latency sharing to the global planning modules. Since the global planner often repeats collision-check queries that each require iterating over a region of the global map, we cache the collision checking results in another OpenVDB map, which we found to greatly reduce planning time.

2) Frontier Maps: There are three maps relevant to generating and scoring viewpoints: *vision-observed map*, *vision-frontier map*, and *range-frontier map*.

The *vision-observed map* is a subset of the occupied cells in the *global map* that are estimated to have been observed by the vision sensors, and thus is an estimate of $\mathcal{E}^{obs}$. An example of this map is shown with the green shading in Fig. 4. This map is produced by ray casting from $x^i(t)$ to all points in the *global map* that are within the camera range and field of view. This ray casting is also repeated for a small set of prior poses to account for cases where the camera makes observations of a surface prior to the range sensors. The *vision-observed map* is stored at a lower resolution (0.2 m).

The *vision-frontier map* is a subset of the occupied cells in the *global map* that are estimated to have not yet been observed by the vision sensors, and thus is an estimate of $\mathcal{E}^{occ} \backslash \mathcal{E}^{obs}$. The map leverages the geometry discovered by the range sensors to direct the planning towards surfaces unobserved by the vision sensors. This map is computed by maintaining a low-resolution



(a) Dust filter: Only the "not dust" points are kept.

(b) Wire detector: The "not dust" and "wire" points are kept.

Fig. 7. Example of detecting dust and wires in point clouds. Points detected as dust are removed from the cloud while wires are considered as obstacles by the local planner. Without this dust filtering, the robot would regularly become stuck in environments containing dust.

copy of the *global map* and removing cells when they are added to the *vision-observed map*.

The *range-frontier map* represents the boundary of the range-sensor observations and follows the classical frontier definition [37]. This map contains all free cells in the *global map* that neighbor a significant number of unknown cells.

The advantage of using the *vision-frontier map* is that it directly corresponds to the objective of expanding $\mathcal{E}^{obs}$. However, using this alone would mean that discovering $\mathcal{E}^{surf}$ only happens opportunistically, which can be problematic. This motivates our proposed combination of employing frontiers for both exploration and coverage.

3) Coordination Maps: Our coordination strategy relies on sharing this mapping information between robots. Thus, we define the *communicated map*, which is a map that merges *vision-observed map* and *vision-frontier map*, such that each cell is marked as *vision-observed* or *vision-frontier*. This map has a resolution of 2.0 m to reduce communication, and we also found this to aid resiliency when maps were misaligned between robots. The *communicated map* is merged with maps from other robots to form the *coordination map*, which is used to aid the global planner decision making.

B. Roadmaps

In order to facilitate traveling to areas that robots have been, we generate roadmaps from each robot's odometry. The roadmaps are graphs with vertices placed at 1 m intervals along the traversed path, X^i. Edges always connect consecutive vertices along X^i. Additionally, to find shortcuts, edges are added between vertices that are up to 5 m apart and are line-of-sight collision free. Each robot has two roadmaps: *single-robot roadmap* and *coordination roadmap*.

The *single-robot roadmap* is used for returning home to land. An example is illustrated later in Fig. 17. Each edge records a traversal time estimate, which is used for triggering the 'Low Battery' condition. If the roadmap is blocked, it may be repaired and extended (see Sec. VI).

The *coordination roadmap* combines all communicated single-robot roadmaps. Inter-robot edges are added; however, since these are not necessarily covered by the *global map*, we connect edges between all pairs of vertices that are less than 5 m apart, and give them a very high cost. We also associate frontier clusters from the *vision-frontier map* with nearby reachable vertices.

C. Local Planner Map Representation

Since the local planner plans at a higher frequency (10 Hz), over shorter distances, and with greater safety requirements, we opt for a different representation for local planner collision checking. We employ a Euclidean distance transform (EDT) map as it enables efficient querying of distances to closest obstacles [30]. Two $150\times150\times150$ voxel grids containing occupancy probability and distance values are instantiated with 10 cm and 30 cm resolutions. The ray casting and distance updates are executed on the GPU.

1) Dust Filtering: Aerial robots often blow dust into the air, which causes false LiDAR 'hits'. If this dust is not filtered, false obstacles appear in the map, which restricts movements. Dust typically appears as a small group of points, as illustrated in Fig. 7(a). We employ an efficient solution that finds discontinuities in LiDAR scans to label points as dust.

We also employ a wire detector to avoid unintentionally filtering out thin hazardous obstacles. As seen in Fig. 7(b), these wire obstacles are often linear. Thus, a line is fit to the closer points of all discontinuities, and if the fit error is small, then the points are labelled as wires. This is an efficient approximation of intensive methods, e.g., RANSAC, and thus is suitable for executing within the planning loop.

VI. Global Planning and Coordination

The global planning and coordination module seeks to generate paths that result in sufficiently exploring the environment with the range sensors and maximally covering the surfaces with the fields of view of the vision sensors. This is achieved in several steps: candidate viewpoints are generated, these viewpoints are scored, then paths are planned to viewpoints. Figure 4 illustrates these planning components in an example mission: when the local planner locks the *planning point*, the global planner plans from this position. There are several variants of exploration behaviors, which are selected by the behavior tree based on information available from other robots. The global planner also plans for a resilient landing behavior.

A. Viewpoint Generation

1) Vision Viewpoints: A set of *vision viewpoints* are generated by processing the *vision-frontier map*. These viewpoints are poses where it is predicted the robot will observe surfaces that have yet to have been observed. The *vision-frontier map* is first segmented into fixed-size clusters. Next, candidate viewpoints are generated on a cylinder around the cluster centroids with orientations set in the direction of each centroid. A viewpoint is kept if the viewpoint is in free space and has line-of-sight visibility to the centroid. Since this process is CPU-intensive, particularly in expansive environments, we incrementally update the viewpoints only where changes occur.

2) Range Viewpoints: While the *vision viewpoints* are generated to encourage making vision-observations of known surfaces, they do not value the discovery of these surfaces with the range sensors. Thus, we also generate *range viewpoints* that encourage discovering $\mathcal{E}^{surf}$. These viewpoints are generated by segmenting the *range-frontier map* into connected components and placing a collision-free viewpoint in each cluster.

B. Viewpoint Selection

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

- Subtract the Euclidean distance from the planning point to the viewpoint. This favors close viewpoints.
- Subtract an estimate for the change in momentum required between the planning point and the viewpoint. This favors faster flight in straight lines.
- Add a reward for viewpoints approximately 10 m away if there are other viewpoints near the line connecting the planning point and the viewpoint. In open spaces, this favors far viewpoints to increase flight speed.
- Add a large reward if the viewpoint has not been observed by other robots, as defined in the *coordination map*.
- Add a reward for selecting a *vision viewpoint* (not *range viewpoint*), to favor the primary coverage objective.
- Subtract a penalty for viewpoints that the path planner has recently failed for, to avoid repeated failures.

The parameters were tuned through experiments, though we found the behavior was not overly sensitive to changes.

C. Path Planning

A path, like the orange path in Fig. 4, is planned to the viewpoints in order of their scores. We employ RRT-Connect [18] for path planning in 3D over the *global map*. For each viewpoint, we trial a large then a small collision radius. The viewpoints are trialled in order until a feasible path is found. If a viewpoint is unreachable, the viewpoint is added to an *unreachable map*, which is a periodically-cleared OpenVDB map. Points in the *unreachable map* are penalized in the selection heuristic to reduce the computational burden in environments with transparent obstacles. The yaw along the path is a free variable for the controller up until near the viewpoint where it is specified as the viewpoint heading.

D. Coordinated Roadmap Exploration

If all of the locally-generated viewpoints have already been observed by other robots, then the behavior switches to 'Roadmap Explore'. This behavior instead plans to frontiers generated and shared by other robots. However, since they are likely to be beyond this robot's *global map*, we also need to rely on the *coordination roadmap* to find paths to these frontiers. A path is planned to the frontier that has not been observed by any robot and has the shortest path distance. As the robot approaches the selected frontier, new locally-computed viewpoints should be generated, and the behavior tree switches back to 'Coordinated Explore'.

E. Return Home Resiliency

The 'Return Home' behavior is triggered if the exploration is failing or battery life is less than the estimated return-home time. The return home path is found over the *single-robot roadmap*. If the battery is critically low, then the path is redirected to a safe landing location. If the return home path is blocked, the local planner will fail to find a path, and this is communicated to the global planner. If this occurs three times, then all edges near the current position are removed from the roadmap. If no path home is found then the roadmap edge connection radius is increased, which should improve the connectivity at the cost of a higher risk of introducing incorrect edges. If there is still no feasible path, then the robot will return to the closest reachable vertex to the landing location. An example of this behavior is presented later in Fig. 17.

VII. Local Planning

The local planner plans collision free paths through $\mathcal{X}$ to goals given by the global planner, by using the EDT map described in Sec. V-C. The algorithm is designed to increase speed in open space and move carefully through clutter. It achieves this by adjusting the target velocity as a function of the distance to nearest obstacles, and adjusting the collision radius based on the current velocity. Through narrow passages, the planner also aligns the yaw since our robots are longer than they are wide. The algorithm has two steps: perform an A* graph search, then match the path to a library of feasible trajectories. We describe these steps as follows.

A* plans a path through a graph generated from the EDT maps. The graph is 26-connected to neighboring voxels, and has vertices at collision-free voxels of the fine map at close distances and the coarse map when beyond the fine map's boundary. The edge cost is defined as the Euclidean distance plus a penalty p that is introduced to favor safer paths. Specifically, we define p as:

$$p = \begin{cases} A(d_{max} - d_{obst}), & \text{if } d_{obst} < D_{close}, \\ B(d_{max} - d_{obst}), & \text{if } D_{close} \leq d_{obst} \leq D_{far}, \\ 0, & \text{otherwise}, \end{cases} \quad (3)$$

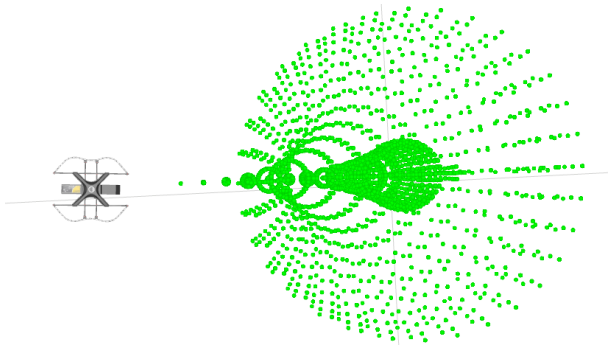


Fig. 8. A trajectory library of dynamically-feasible trajectories as the robot flies to the right. These trajectories are matched to the local planner path.

where d_{obst} is the distance to the nearest obstacle, d_{max} is the maximum voxel–obstacle distance in the EDT, and D_{close}, D_{far}, A and B are constants with $A > B$. With these definitions, A* with a Euclidean heuristic finds the path that minimizes the sum of edge costs. Narrow passages are determined as small d_{obst} values, which triggers a reduction in maximum velocity and collision radius, and sets the heading to align forwards or backwards with the direction of travel.

To ensure dynamically-feasible trajectories are passed to the controller, the path output by A* is matched to a library of feasible trajectories. The library of trajectories adapts based on the current state of the robot; an example is illustrated in Fig. 8. Specifically, the library of trajectories are generated for a discrete set of snap values, $\mathcal{S}$. First, a set of desired velocities $\mathcal{V}_d$ is formed by sampling horizontally in 11.25° increments and pitching at 0°, 12°, 24°, and 80°. Two additional velocities are added in the direction of points on the A* path. Using $\mathcal{V}_d$, the snap values are computed as:

$$\mathcal{S} = \{s : s = -k_v(v_p - v_d) - k_a a_p - k_j j_p, \forall v_d \in \mathcal{V}_d\}, \quad (4)$$

where v_p, a_p, and j_p are the current velocity, acceleration, and jerk, and k_v, k_a, and k_j are constant gains that specify aggressiveness. At the end of each trajectory, the desired velocity is set to 0 so the robot has enough time to stop if necessary. The trajectories are formed by integrating the snap values in $\mathcal{S}$. The output trajectory is selected that is collision free and minimizes the sum of distances to the A* path.

VIII. Field Experiments

Our proposed autonomy pipeline for multi-sensor exploration has been extensively validated on physical robots in a wide range of environments, as illustrated earlier in Fig. 1. These experiments directly motivated and informed the ongoing development of the autonomy. Here, we present a selection of results that aim to showcase a variety of experimental scenarios and provide interesting examples of resilient behaviors. The experiments are summarized in Table I with illustrations provided later in the appendix.

A. Hardware Setup

Our experiments were performed with a team of custom-built quadrotor aerial robots. All sensing and computation is performed onboard the robots. Each robot weighs approximately 5.2 kg, is 68 cm wide and 81 cm long. The battery life varied significantly, but was typically 10 to 15 minutes of flight time. The motors were controlled by a PX4-based flight controller. Each robot was equipped with one Velodyne VLP-16 LiDAR, and upward- and downward-pointing Intel Realsense L515 RGB-D sensors, which provide the range observations. Each robot is also equipped with one forward-facing camera with a 200° fish-eye lens, which provides the visual observations, $z^i(t)$, that are the basis of our objective function in (2). The robots had an Intel NUC computer with an Intel Core i7-8550U CPU that has 8 threads (4 cores). Communication between robots was through a wireless mesh network, but was often intermittent and unreliable. We also performed experiments with a different robot model, with the main differences being increased size and weight, and an additional backward-facing fish-eye camera.

B. Results and Discussion

As described in Table I, each environment presented unique features and challenges. No software changes were made to account for these differences, but rather the autonomy had to adapt to the observed situation.

Adaptive Speeds: The most noticeable difference between the environments that influenced the autonomy was that some had narrow passages only slightly wider then the width of the robot, others had wide open tunnels, while some were mixed. These differences can be clearly seen in Fig. 1. How the autonomy adapted to these differences can be observed by comparing the distances and speeds. The most open environments were the *limestone mine* and *large cavern*, where the robots travelled approximately 1000 m and with speeds up to 3 m/s. The *boiler room* and *office warehouse* were the tightest, resulting in slower speeds to guarantee safety and stability.

Multi-sensor Objectives: The benefits of the multi-sensor exploration formulation, compared to the classic exploration problem, are most evident in the *large cavern* (Fig. 11). The entire environment is observed by the LiDAR very quickly due to the openness of the space, and so the mission would finish very quickly if performing pure exploration. However, many more passes are required to sufficiently observe the environment with the close-range vision sensor, and this is made possible with the use of the vision-frontiers.

Range-frontiers: Conversely, the range-frontiers are still vital in specific scenarios, which is especially evident in the *boiler room*. The range-frontiers were necessary for finding a path through the small hole in the wall illustrated earlier in Fig. 4, as the LiDAR does not sufficiently observe through to discover the geometry and generate vision-frontiers. Hence, generating both frontier types is necessary to complete the full mission shown in Fig. 12.

Coordination Behaviors: In the missions with multiple robots, different coordination behaviors arose. In the *limestone mine* (see Fig. 9), there were enough branches in the environment for the robots to primarily rely on the shared observed maps to spread out into different directions; the robots mostly

TABLE I
A summary of the selection of presented field experiments. The Fig. 1 references are specified in parentheses, with '1' at the top of the diagram then counting clockwise. The cell counts in the last column refer to the number of occupied voxels (0.2 m cubes) observed by the vision sensors.

Environment	Location	Figures	Features & Challenges	Robots	Distances Traveled (m)	Average Speeds (m/s)	Maximum Speeds (m/s)	Average Number View-points	Vision-observed Voxels ('000s)
Natural cave	Laurel Caverns, PA	1 (10, 12), 10	open & narrow, steep slopes, unstructured, unreachable areas	1	439	0.7	2.0	46	283
Large cavern	Louisville Mega Cavern, KY	1 (3), 11	very open, high ceilings	1	874	1.3	3.0	334	224
Boiler room	Abandoned complex, Pittsburgh PA	1 (2), 4, 12	multi-level, vertical passages, thin obstacles	1	365	0.6	2.2	141	239
Office warehouse	Louisville Mega Cavern, KY	1 (7), 13	doorways, clutter, dynamic obstacles	2	189, 273	0.5, 0.5	1.9, 1.8	124, 2267	162, 208
Subway station	Louisville Mega Cavern, KY	1 (1, 5, 9), 14, 15	long distances, fog, unreachable areas	3	224, 473, 385	0.9, 0.8, 0.8	1.9, 2.1, 2.2	80, 2461, 2651	236, 248, 235
Limestone mine	Brady's Bend, PA	1 (6), 9	long distances, wide tunnels, dust	3	1106, 660, 1073	1.5, 1.2, 1.7	2.6, 2.9, 2.7	881, 652, 1617	552, 396, 412
Auditorium corridors	Abandoned complex, Pittsburgh PA	16	doorways, open & narrow, unreachable areas	2	555, 233	0.9, 0.5	2.7, 2.0	425, 374	292, 229
Abandoned hospital	Abandoned complex, Pittsburgh PA	17	doorways, dynamic obstacles, structured	1	491	0.7	1.7	435	365

executed the 'Coordinated Explore' behavior tree action. In the *auditorium corridors* (see Fig. 16), the robots had no option but to follow each other down the corridor until the branching point; the second robot executed the 'Roadmap Explore' action and leveraged the roadmap shared by the first robot to find the unexplored rooms. A similar coordination behavior was observed in the *subway station*, where the robots followed similar paths until they reached the open space of Fig. 14(b).

Constrained Environments: Comparing the behavior in the two most open environments, the *large cavern* was mostly one large room while the *limestone mine* was unconstrained in size. While similar speeds and distances were achieved in these environments, the number of cells in the vision-observed voxel map was significantly lower in the *large cavern* since the later part of the flight was spent revisiting small areas missed in the earlier passes of the environment (see Fig. 11).

Viewpoint Density: The number of global planning viewpoints varied greatly between environments. The number of viewpoints increases when the LiDAR can observe much farther than the cameras, resulting in more vision-frontiers; this was particularly evident in the *limestone mine* and the *large cavern*. In contrast, in the *natural caves* and *boiler room*, the LiDAR field of view was significantly occluded by obstacles. The algorithm is designed to ensure sufficient viewpoints for full coverage, while also being mindful of CPU usage.

Unreachable Regions: Another challenge for viewpoint generation was that the *office warehouse*, *subway station*, and *auditorium corridors* had large unreachable regions that were visible to the LiDAR, e.g., through glass and netting. During initial testing, the unreachable viewpoints would introduce latency due to repeated path planning failures. This motivated the *unreachable map* and incremental viewpoint updates.

Resilient Behaviors: We encountered numerous examples of resilient behaviors; here, we describe three examples:

- Figure 13 illustrates how the local planner adapts to the current situation: the robot begins at high speed in the wide corridor, then decreases speed when near obstacles for safety, then rotates to fit through a doorway that is barely wider than the width of the robot.
- Figure 17 demonstrates the resilient return home behavior: when the battery is low the robot plans a path home, attempts to follow that path, detects this path is blocked by a door, so repairs and extends the roadmap to find an alternative path home, then safely lands.
- Figure 10 shows a case where the return home time is underestimated due to an extended steep climb at slower speed. The battery falls critically low, so the robot backtracks and lands at the safe landing zone at the bottom of the stairs.

SubT Challenge Final Event: Our autonomy pipeline was employed on our aerial robots during the DARPA Subterranean Challenge final event. During the prize round, labeled as *subway station* and illustrated in Fig. 14, we deployed three aerial robots that focused on the 'urban' section while our ground robots explored the other areas. The first aerial robot launched from a ground robot near the stairs that lead to the subway station. The other two aerial robots were deployed

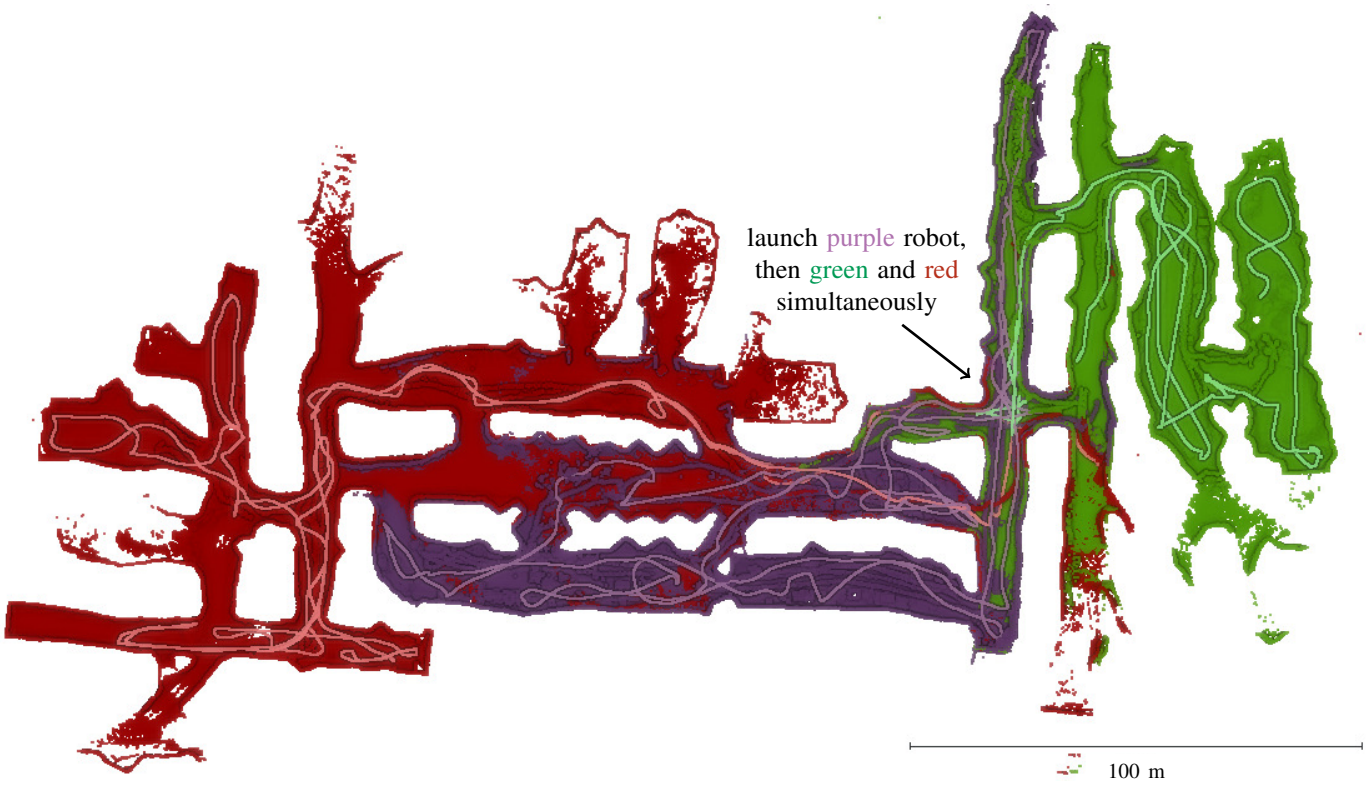


Fig. 9. An example coordination mission in a *limestone mine*. Three robots were launched at the same location, with the purple robot launching earlier. The next two robots used coordination maps from the purple robot to discover new areas. In this open environment, the achieved speeds of up to 3 m/s and distances exceeding 1000 m in 10–15 minute flights. Other missions with different behaviors are illustrated in the appendix.

from the start gate, guided by the human operator towards the urban section, then explored autonomously. Unlike the other presented experiments, this mission involved human input and UGV map sharing; the details are out of the scope of this paper. The robots flew between 200 m and 400 m each, with speeds sometimes exceeding 2 m/s, then landed safely. The aerial robots detected, localized and communicated all of the 8 objects in that area, which was the competition objective. Three object detections are shown in Fig. 15.

IX. Lessons Learned and Future Work

Our extensive experiments and ongoing development revealed several key challenges and trade-offs:

- **Avoid Overfitting:** When learning from an experiment, it is essential to respond in a way that both fixes any issues that arose and does not cause new issues in other experimental scenarios. Overfitting to particular environments and scenarios may be avoided by testing in a wide range of environments, and having the scenarios be unknown to both the robots and the developers.
- **Solution Complexity:** Algorithm design involves a trade-off between selecting complex algorithms that provide benefits such as theoretical guarantees, and less complicated algorithms that may be resource-efficient and easier to get working. Our system is evidence that an appropriate answer to this trade-off varies between components; e.g., frontier-based exploration is an established method that we generalized for multi-sensor exploration, but OpenVDB is a complicated data structure that provides practically-meaningful query-time guarantees.
- **Multi-Objective Exploration:** While classic robot exploration poses the problem of mapping for the sake of mapping, we are increasingly seeing robots being useful for other tasks where mapping is a secondary objective. If the primary objective is neglected in favor of exploration, then this may be detrimental to the mission success. However, exploration objectives cannot be ignored either as high quality maps may be essential for achieving the primary objective.
- **Resiliency through Redundancy:** One of our aims was to develop a system that never stopped moving, no matter what went wrong, even if it meant behaving suboptimally. We found the best way to achieve this was to provide options to the autonomy that worked in different scenarios. We found the behavior tree architecture to be a convenient tool for facilitating this resiliency.

Regarding future work, we are interested in incorporating other behaviors, such as rendezvous and data muling behaviors for sharing information. We are also interested in studying other ways to balance exploration and coverage; perhaps employing machine learning to generate policies from data would be fruitful. Our implicit coordination strategy was

appropriate for our scenarios, but scenarios requiring tightly-coupled coordination would benefit from planning in the joint space. We aim to continue expanding the autonomy for systems with more robots, faster flying, and complex mission objectives.

Acknowledgments

We thank the rest of SubT Team Explorer for the development of other parts of the system beyond the scope of this paper, including hardware development and repairs, safety piloting, software infrastructure, SLAM, communication, and user interface, which are discussed further in [31].

Approved for public release; distribution is unlimited. This work was in part sponsored by DARPA under agreement #HR00111820044. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsor.

G. Best has recently moved to the University of Technology Sydney, Ultimo NSW, Australia.

References

[1] Ali Agha et al. NeBula: Quest for robotic autonomy in challenging environments; team CoSTAR at the DARPA subterranean challenge. *arXiv preprint arXiv:2103.11470*, 2021.

[2] Monica Anderson and Nikolaos Papanikolopoulos. Implicit cooperation strategies for multi-robot search of unknown areas. *Journal of Intelligent and Robotic Systems*, 53(4):381–397, 2008.

[3] Antoine Bautin, Olivier Simonin, and François Charpillet. Towards a communication free coordination for multi-robot exploration. In *Proc. National Conference on Control Architectures of Robots*, 2011.

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

[5] Graeme Best, Oliver Cliff, Timothy Patten, Ramgopal R. Mettu, and Robert Fitch. Dec-MCTS: Decentralized planning for multi-robot active perception. *International Journal of Robotics Research*, 38(2-3):316–337, 2019.

[6] Frederic Bourgault, Alexei A Makarenko, Stefan B Williams, Ben Grocholsky, and Hugh F Durrant-Whyte. Information based adaptive robotic exploration. In *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, volume 1, pages 540–545, 2002.

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

[8] Chao Cao, Hongbiao Zhu, Howie Choset, and Ji Zhang. TARE: A hierarchical framework for efficiently exploring complex 3D environments. In *Proc. Robotics: Science and Systems (RSS)*, 2021.

[9] Timothy Chung. DARPA Subterranean Challenge, 2021. URL https://www.subtchallenge.com/. Accessed on 01.28.2022.

[10] Michele Colledanchise and Petter Ögren. *Behavior Trees in Robotics and Al: An Introduction*. CRC Press, 2018.

[11] Micah Corah and Nathan Michael. Distributed matroid-constrained submodular maximization for multi-robot exploration: Theory and practice. *Autonomous Robots*, 43(2):485–501, 2019.

[12] Tung Dang, Marco Tranzatto, Shehryar Khattak, Frank Mascarich, Kostas Alexis, and Marco Hutter. Graph-based subterranean exploration path planning using aerial and legged robots. *Journal of Field Robotics*, 37(8): 1363–1388, 2020.

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

[14] David Gómez-Anaya, Rodrigo Munguía, Edmundo Guerra, and Antoni Grau. Full autonomous navigation for an aerial robot using behavior-based control motion and SLAM. In *Proc. IEEE Emerging Technology and Factory Automation (ETFA)*, 2014.

[15] Nicolas Hudson et al. Heterogeneous ground and air platforms, homogeneous sensing: Team CSIRO Data61's approach to the DARPA subterranean challenge. *arXiv preprint arXiv:2104.09053*, 2021.

[16] Michail Kalaitzakis, Brennan Cain, Nikolaos Vitzilaios, Ioannis Rekleitis, and Jason Moulton. A marsupial robotic system for surveying and inspection of freshwater ecosystems. *Journal of Field Robotics*, 38(1):121–138, 2021.

[17] Vít Krátký, Pavel Petráček, Tomáš Báča, and Martin Saska. An autonomous unmanned aerial vehicle system for fast exploration of large complex indoor environments. *Journal of Field Robotics*, 38(8):1036–1058, 2021.

[18] James J Kuffner and Steven M LaValle. RRT-Connect: An efficient approach to single-query path planning. In *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA)*, volume 2, pages 995–1001, 2000.

[19] Mikko Lauri and Risto Ritala. Planning for robotic exploration based on forward simulation. *Robotics and Autonomous Systems*, 83:15–31, 2016.

[20] Chris Yu Hsuan Lee, Graeme Best, and Geoffrey A Hollinger. Optimal sequential stochastic deployment of multiple passenger robots. In *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 8934–8940, 2021.

[21] Ki Myung Brian Lee, Felix H Kong, Ricardo Cannizzaro, Jennifer L Palmer, David Johnson, Chanyeol Yoo, and Robert Fitch. An upper confidence bound for simultaneous exploration and exploitation in heterogeneous multi-robot systems. In *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 8685–8691, 2021.

[22] Liyang Liu, Simon Fryc, Lan Wu, Thanh Long Vu, Gavin Paul, and Teresa Vidal-Calleja. Active and inter-

active mapping with dynamic Gaussian process implicit surfaces for mobile manipulators. *IEEE Robotics and Automation Letters*, 6(2):3679–3686, 2021.

[23] Martin Molina, Abraham Carrera, Alberto Camporredondo, Hriday Bavle, Alejandro Rodriguez-Ramos, and Pascual Campoy. Building the executive system of autonomous aerial robots using the Aerostack open-source framework. *International Journal of Advanced Robotic Systems*, 17(3):1–20, 2020.

[24] Ken Museth. VDB: High-resolution sparse volumes with dynamic topology. *ACM Trans. on Graphics*, 32(3):1–22, 2013.

[25] Alejandro Rodriguez-Ramos et al. Autonomous aerial robot for high-speed search and intercept applications. *arXiv preprint arXiv:2112.05465*, 2021.

[26] Evgenii Safronov, Michael Vilzmann, Dzmitry Tsetserukou, and Konstantin Kondak. Asynchronous behavior trees with memory aimed at aerial vehicles with redundancy in flight controller. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 3113–3118, 2019.

[27] Evgenii Safronov, Michele Colledanchise, and Lorenzo Natale. Task planning with belief behavior trees. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 6870–6877, 2020.

[28] Manish Saroya, Graeme Best, and Geoffrey A. Hollinger. Online exploration of tunnel networks leveraging topological CNN-based world predictions. In *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, pages 6038–6045, 2020.

[29] Emily Scheide, Graeme Best, and Geoffrey A. Hollinger. Behavior tree learning for robotic task planning through Monte Carlo DAG search over a formal grammar. In *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 4837–4843, 2021.

[30] Sebastian Scherer, Dave Ferguson, and Sanjiv Singh. Efficient C-space and cost function updates in 3D for unmanned aerial vehicles. In *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 2049–2054, 2009.

[31] Sebastian Scherer et al. Resilient and modular subterranean exploration with a team of roving and flying robots. *Field Robotics*, 2:1–58, 2022.

[32] Christopher Iliffe Sprague, Özer Özkahraman, Andrea Munafo, Rachel Marlow, Alexander Phillips, and Petter Ögren. Improving the modularity of AUV control systems using behaviour trees. In *Proc. IEEE/OES Autonomous Underwater Vehicle Workshop (AUV)*, 2018.

[33] Fouad Sukkar, Graeme Best, Chanyeol Yoo, and Robert Fitch. Multi-robot region-of-interest reconstruction with Dec-MCTS. In *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 9101–9107, 2019.

[34] Wennie Tabib, Kshitij Goel, John Yao, Curtis Boirum, and Nathan Michael. Autonomous cave surveying with an aerial robot. *IEEE Trans. on Robotics*, 38(2):1016–1032, 2022.

[35] Yulun Tian, Katherine Liu, Kyel Ok, Loc Tran, Danette Allen, Nicholas Roy, and Jonathan P. How. Search and rescue under the forest canopy using multiple UAVs. *International Journal of Robotics Research*, 39(10-11):1201–1221, 2020.

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

[37] Brian Yamauchi. A frontier-based approach for autonomous exploration. In *Proc. IEEE International Symposium on Computational Intelligence in Robotics and Automation*, pages 146–151, 1997.

[38] Fan Yang, Dung-Han Lee, John Keller, and Sebastian Scherer. Graph-based topological exploration planning in large-scale 3D environments. In *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 12730–12736, 2021.

[39] Shibo Zhao, Hengrui Zhang, Peng Wang, Lucas Nogueira, and Sebastian Scherer. Super Odometry: IMU-centric LiDAR-visual-inertial estimator for challenging environments. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 8729–8736, 2021.

[40] Boyu Zhou, Yichen Zhang, Xinyi Chen, and Shaojie Shen. FUEL: Fast UAV exploration using incremental frontier structure and hierarchical planning. *IEEE Robotics and Automation Letters*, 6(2):779–786, 2021.

X. Appendix: Figures for Field Experiments

Below, we provide several figures that illustrate a selection of our field experiments and resilient behaviors. These experiments are described further in the captions and earlier in Table I.

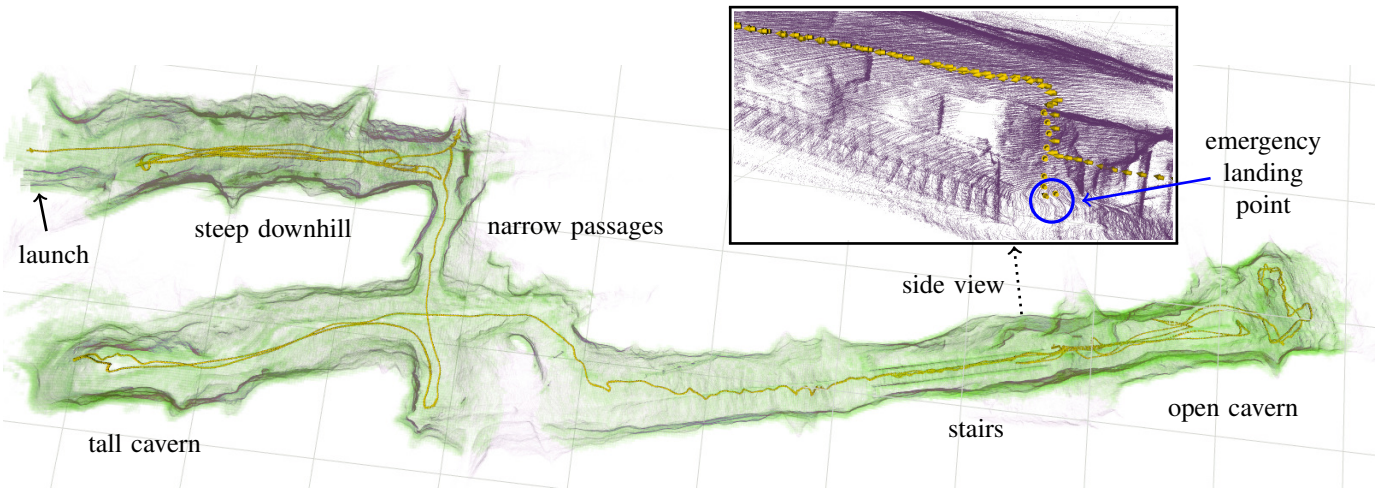


Fig. 10. A single-robot mission in a *natural cave*. The green shaded is the vision-observed map. This environment features a steep from left to right. Inset: The robot underestimated the return home time while returning up the steep stairs. It therefore returned to a suitable emergency landing area at the bottom of the stairs. Background: 10 m grid.

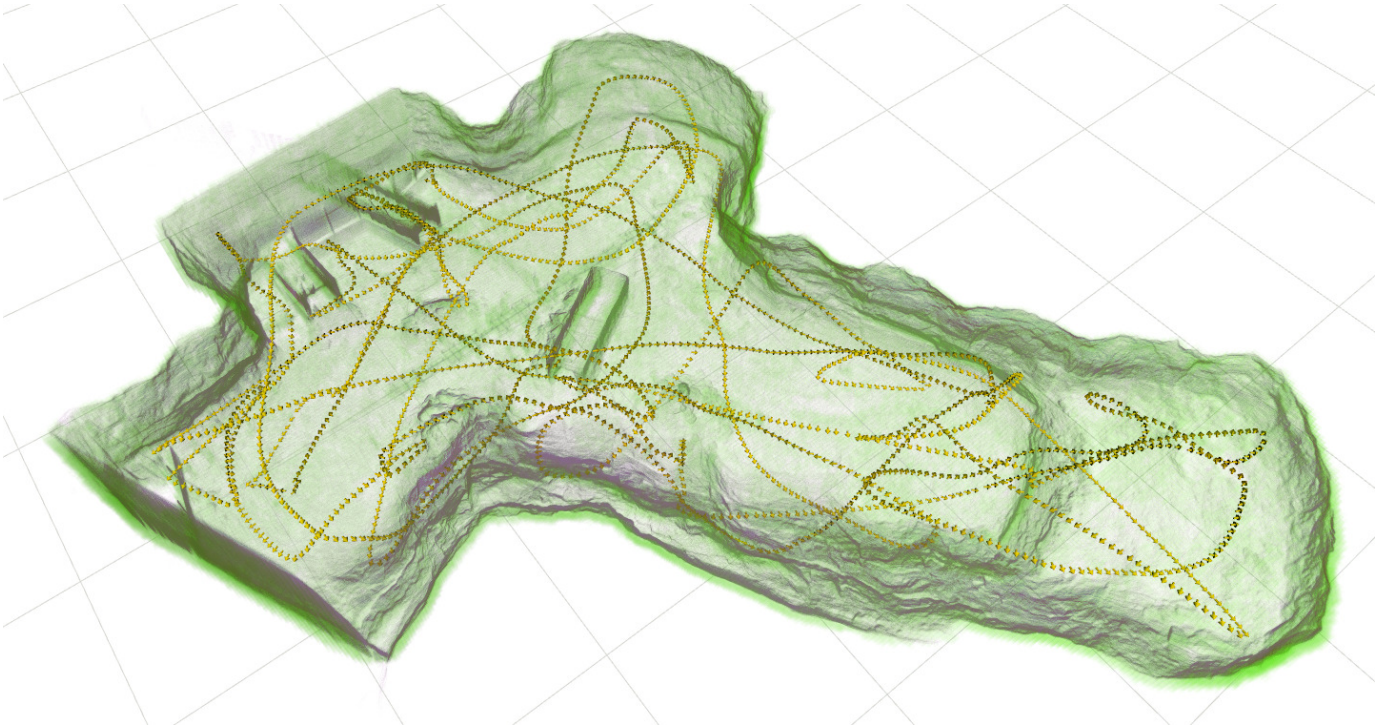
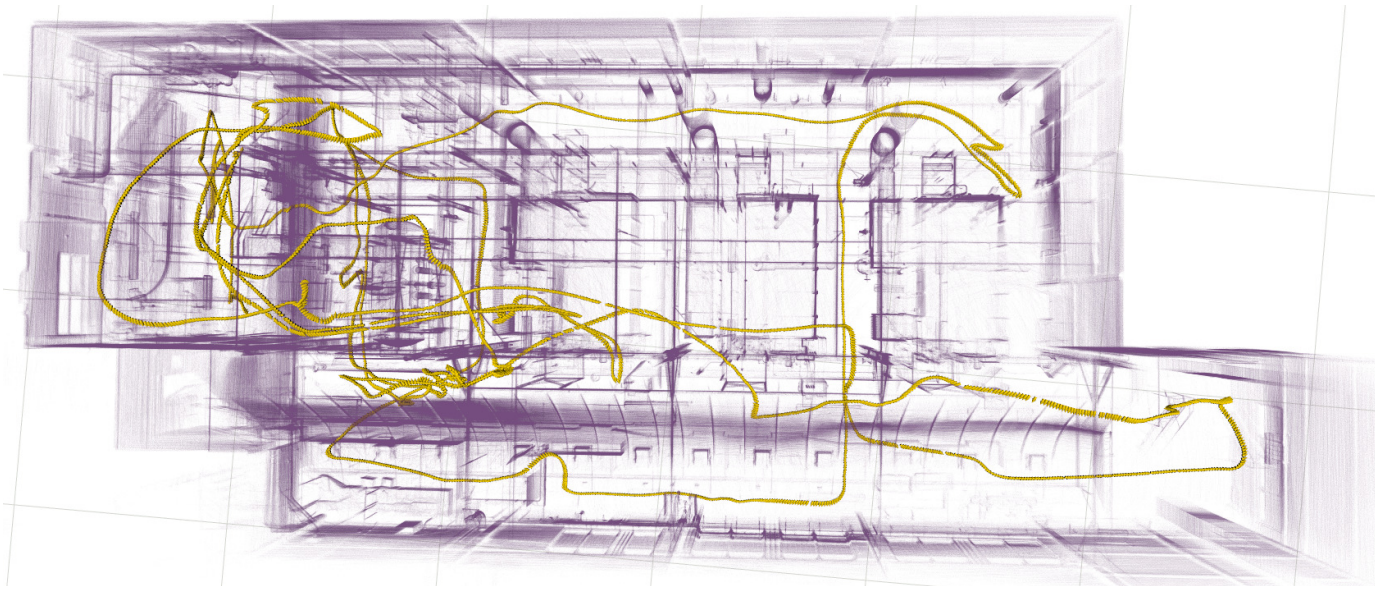
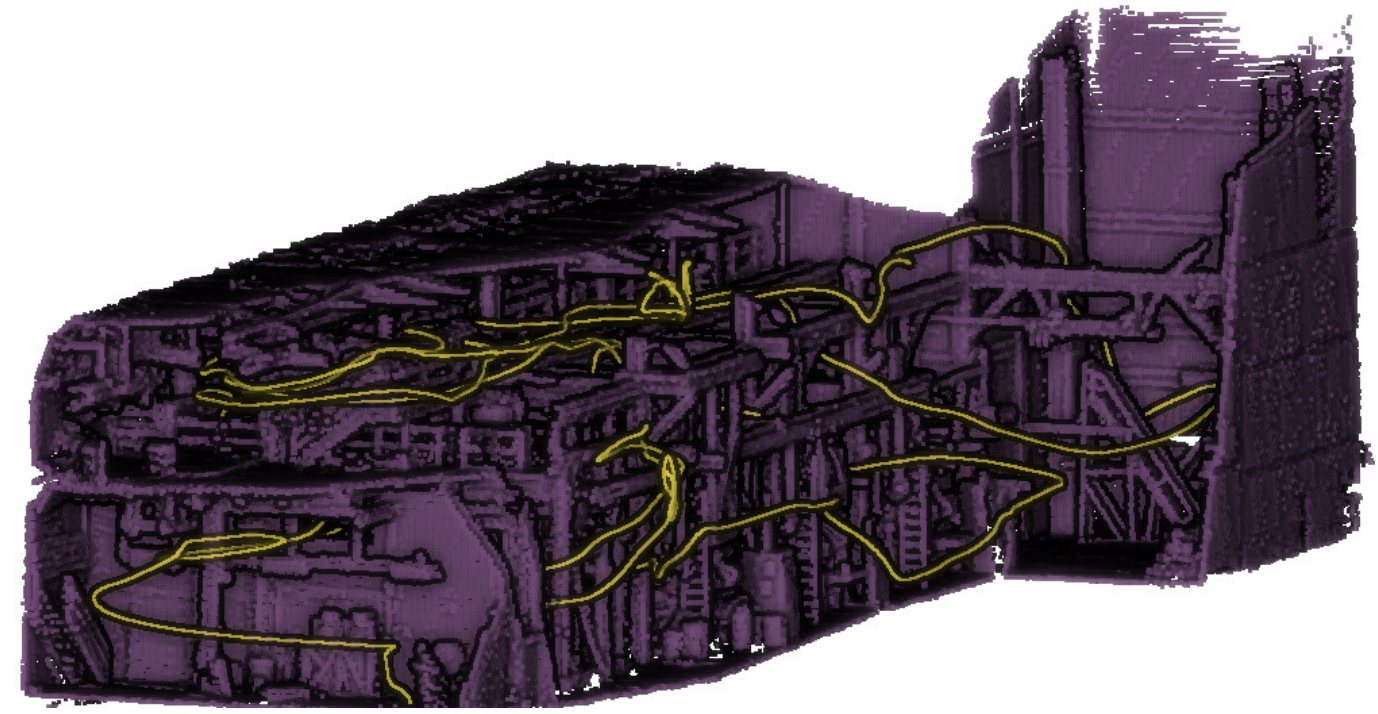


Fig. 11. A single robot explores a *large cavern*. This was a wide open area that enabled long-range LiDAR observations with a few high-velocity straight trajectories, but many more passes of the environment were required to sufficiently observe with the vision sensor. The ceiling was also very high, meaning that each location in the horizontal plane had to be observed from multiple altitudes.

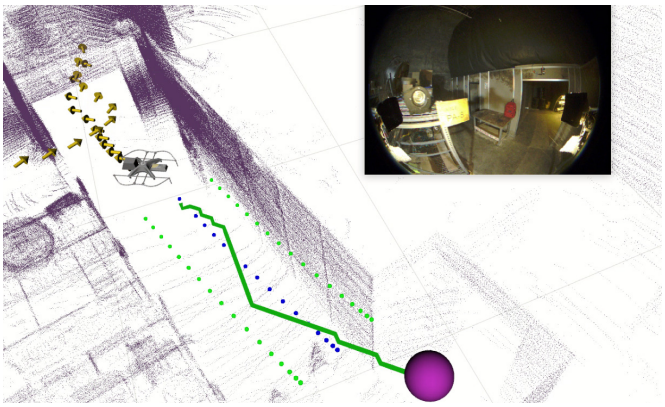


(a) Top-down view of the 3D structure.

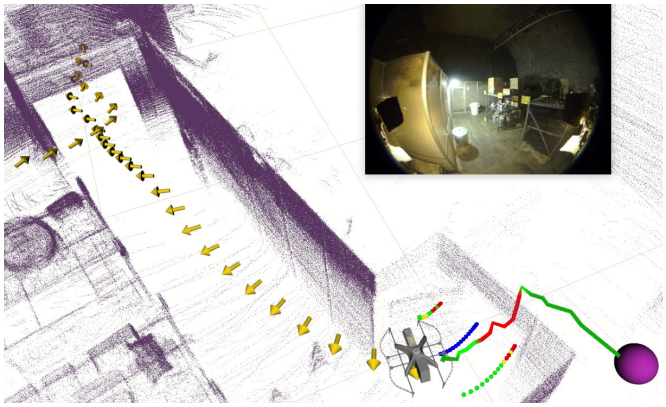


(b) Side-on view with walls cut out.

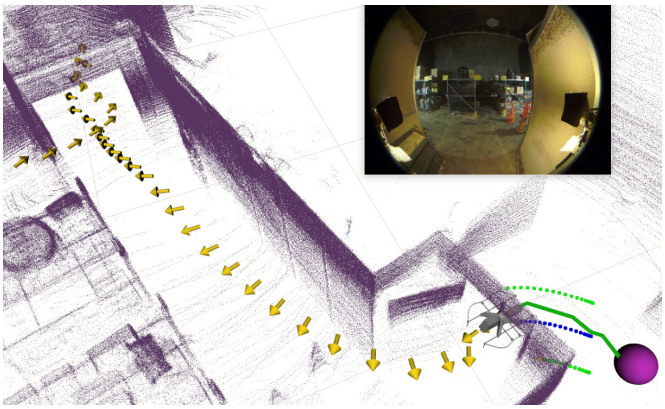
Fig. 12. A single-robot mission in a *boiler room*, which is a complex multi-level structure with many narrow and vertical openings between beams. The take-off location is in the left of both images. Same mission as earlier in Fig. 4.



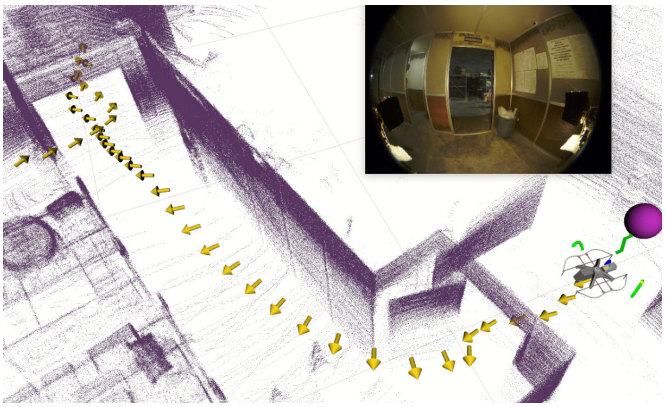
(a) Time: 0s. Robot increases speed in the wider corridor.



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

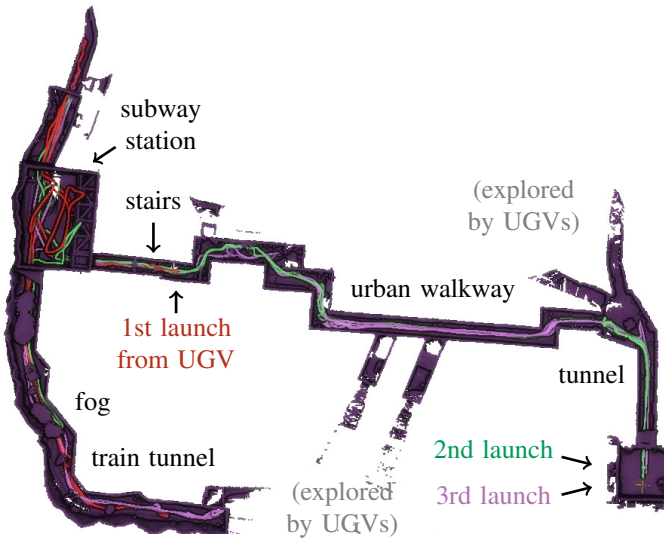


(c) Time: 13s. Robot rotates to fly backwards through doorway.

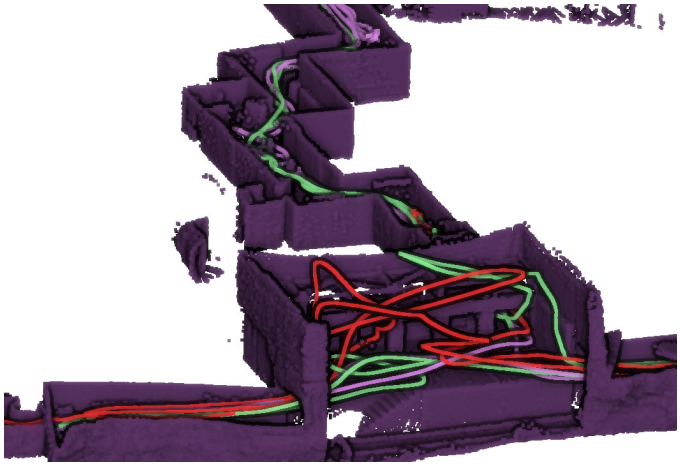


(d) Time: 17s. Doorway maneuver successfully completed.

Fig. 13. A timelapse sequence of a robot exploring an *office warehouse*. The local planner adjusts the speed and yaw of the robot as the robot maneuvers through the doorway. The yellow arrows are a history of poses; inset is the view from the forward facing camera; the purple sphere is the current local planner goal; the solid path is the local planner path (with red indicating a narrow passage, and green otherwise); the dots and coloring near the future trajectory represent clearance checking.



(a) Top-down view with ceilings cut out.



(b) Subway station viewed from the side with walls cut out.

Fig. 14. Three robots (red, green, purple) coordinate during the DARPA Subterranean Challenge prize run. The red robot was launched from the back of a ground robot near the stairs that lead to the subway station. The other two robots explored a similar area since our human operator in the competition deemed this area to be most suitable for our aerial robots. Only the purple robot's point cloud is visible. All robots landed at their launch location.

(a) *Backpack* artifact hidden on the roof of the subway platform.

(b) *Vent* artifact behind pillars of subway platform.

(c) *Survivor* artifact in the walkway.

Fig. 15. Three visually-detected hidden objects during the DARPA SubT Challenge final event. The proposed pipeline aims to generate good viewpoints for visual detection algorithms.

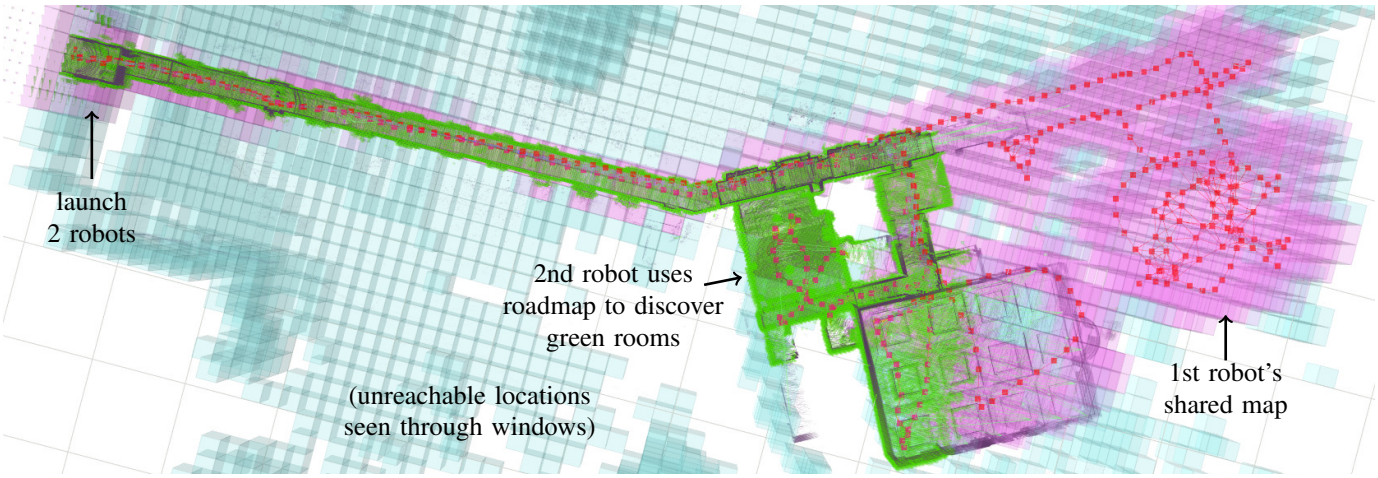
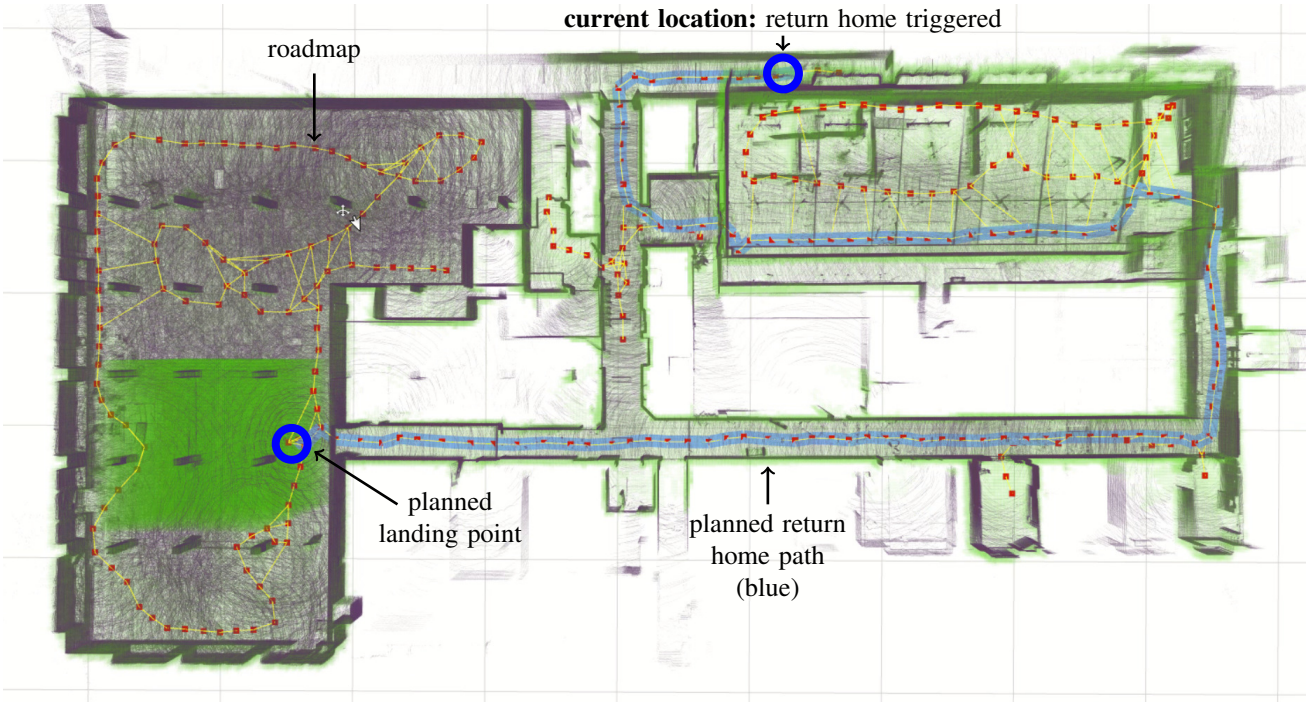
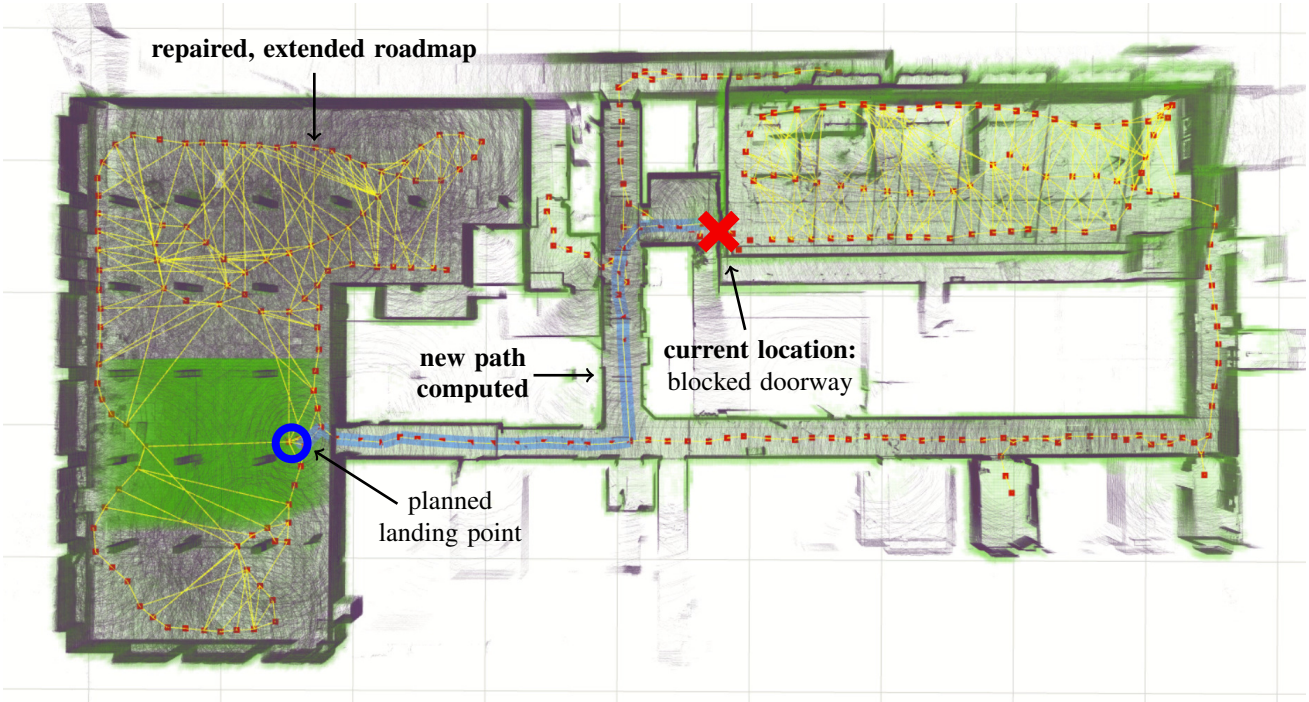


Fig. 16. Two robots sequentially explore the *auditorium corridors*. Figure shows information from the perspective of the second robot. The first robot went down the corridor then explored the auditorium to the right (coarse pink map), while skipping several rooms in the middle. This information was shared with the second robot, which used the shared coordination roadmap to discover the rooms in the middle then switches to the primary exploration behavior. Background: 10 m grid.



(a) The planned return home path from the time that a low battery was detected.



(b) A blockage is detected while returning home, which triggers the repairing and replanning behavior: Edges are removed near the blocked doorway and the maximum edge length is increased throughout the roadmap to find a new path home.

Fig. 17. An example of the return home behavior by a single robot in an *abandoned hospital*. A path home is planned over the roadmap, but it is discovered that this path has been blocked. Typically only paths close to the roadmap are planned to minimize the risk of crashes on a return. The planner repairs the roadmap, discovers there is no path home on the repaired roadmap, so then extends the roadmap to find a new feasible path. Note: The green cube in the room on the left is predefined as 'observed' to avoid that area.