# Online 3D Frontier-Based UGV and UAV Exploration Using Direct Point Cloud Visibility

Jason Williams,[*] Shu Jiang,[†] Matthew O'Brien,[†] Glenn Wagner,[‡] Emili Hernandez,[*]
Mark Cox,[*] Alex Pitt,[*] Ron Arkin,[†] Nicolas Hudson[*]

***Abstract*— While robots have long been proposed as a tool to reduce human personnel's exposure to danger in subterranean environments, these environments also present significant challenges to the development of these robots. Fundamental to this challenge is the problem of autonomous exploration. Frontier-based methods have been a powerful and successful approach to exploration, but complex 3D environments remain a challenge when online employment is required. This paper presents a new approach that addresses the complexity of operating in 3D by directly modelling the boundary between observed free and unobserved space (the frontier), rather than utilising dense 3D volumetric representations. By avoiding a representation involving a single map, it also achieves scalability to problems where Simultaneous Localisation and Matching (SLAM) loop closures are essential. The approach enabled a team of seven ground and air robots to autonomously explore the DARPA Subterranean Challenge Urban Circuit, jointly traversing over 8 km in a complex and communication denied environment.**

## I. Introduction

Underground environments, such as mines, are notoriously hazardous to the safety of personnel, particularly during emergencies or disasters. In the last century, there were more than 500 mine disasters in the United States alone [1]. Subterranean environments are difficult to traverse due to terrain which may be rough, dynamic, unstable and prone to landslides [2], flooded, full of methane gas or heavy smoke, and have extreme temperatures [3]. Further, communications is inherently challenging, making reliance on teleoperation problematic, motivating the use of autonomous operation.

The DARPA Subterranean (SubT) Challenge [4], is stimulating the development of integrated systems for operations within underground environments such as mines, tunnels, natural caves and the urban underground. This paper discusses the solution developed by *Team CSIRO* (consisting of CSIRO, Emesent and Georgia Tech) for autonomously exploring unknown environments using a team of robots. The system was deployed in the SubT Urban Circuit (UC) Competition to explore the abandoned Satsop Nuclear Power Plant, a large multi-level facility with lead-lined concrete walls that blocked wireless communications. A heterogeneous team of seven robots autonomously explored the facility, traversing corridors, rubble, doorways, stairways and vertical shafts. The team consisted of three small Superdroid-LT2 based Unmanned Ground Vehicles (UGVs) and two BIA 5 OzBot Titan UGVs, each carrying and launching an Emesent Hovermap Unmanned Aerial Vehicle (UAV). All platforms are equipped with rotating Velodyne Puck Light Detection and Ranging (LiDAR) sensors, as well as cameras (four cameras on UGVs, and a single gimbal-mounted camera on the UAVs). Figure 1 shows two of the platforms and an example of operation.

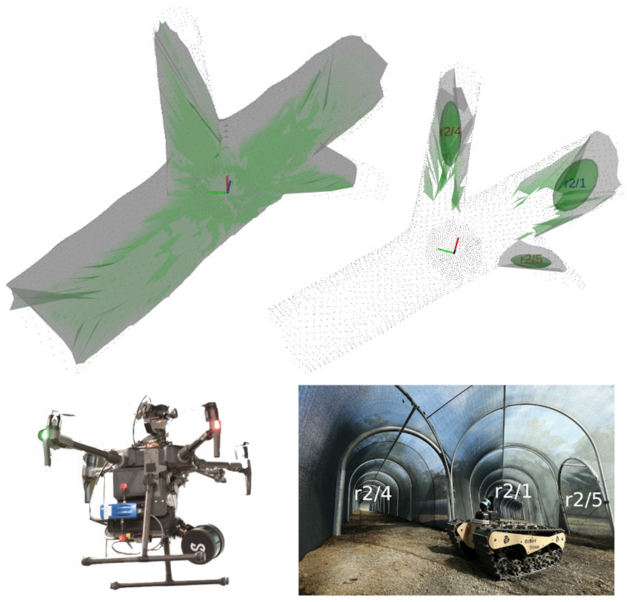

Fig. 1. Example operation of method. Top-left shows mesh representing visible space from viewpoint at axes shown; top-right shows frontiers yet to be explored. Lower-left shows Emesent Hovermap UAV platform used in testing, while lower-right shows BIA 5 OzBot Titan-based platform capturing data, and CSIRO test environment corresponding to results in top figures.

The key differentiator is that the method directly models the boundary between observed free and unobserved space (the *frontier* [5]), as opposed to relying on dense volumetric representations. This is achieved by utilising the efficient direct point cloud visibility algorithm of [6]. The algorithm is demonstrated through results obtained at the SubT UC, where it operated in real time on both UGVs and UAVs providing a core piece of the autonomy capability that permitted a single operator to control seven agents.

## II. Related Work

This section presents a brief overview of the current state of practice of frontier-based exploration of unknown environ-

[*]CSIRO Data61 Robotics and Autonomous Systems Group, `firstname.lastname@csiro.au`

[†]Georgia Tech Mobile Robot Laboratory, {`mjobrien, sjiang, arkin`}`@gatech.edu`

[‡]Emesent, `glenn@emesent.io`

This research was, in part, funded by the U.S. Government under the DARPA Subterranean Challenge. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Department of Defense or the U.S. Government.

ments, describes their limitations, and discusses how this work addresses some of those limitations. First introduced in [5], frontier methods explore a region by repeatedly navigating to the boundary between open space and unexplored space. The concept has been widely used by representing the region as a 2D occupancy grid (e.g., [7], [8]) due to the simplicity and effectiveness of the approach. However, to address more complex environments in a scalable manner, methods beyond 2D occupancy grids are needed.

Representation of the entire space via a single occupancy grid is problematic for large scale problems, where agent localisation is continually optimised via SLAM. Integration of frontier search with SLAM was discussed recently in [9], which utilises the 2D submap occupancy grids generated by Cartographer [10]. The relative position of these occupancy grids is optimised, e.g., due to loop closure. Frontiers are detected locally in each submap, and combined into global frontiers applying the relative pose estimate of each submap. A frontier point in a local submap is only a frontier in the global map if it is unexplored (or a frontier) in all submaps.

The DARPA SubT challenge involves environments that are fundamentally 3D (e.g., mine shafts, stairs, ramps, multi-floor urban environments), and vehicles (e.g., UAVs) that are capable of 3D motion; thus the goal of this work is to develop a method that can directly identify 3D frontiers in LiDAR point cloud data. Previous work on 3D frontiers includes [11], [12]. Both utilise an octree representation of global 3D space, updated using raytracing, walking along the ray for each LiDAR return and updating each traversed voxel. The first seeks to model both frontiers and voids (unobserved volume), where the volume of the void serves to inform the utility of a frontier. The next view is selected to maximise the amount of void space observed. The second presents methods for multiagent coordination. Despite the efficient tree-based methods, neither achieves real-time operation, and integration with a SLAM system performing loop closures is not addressed.

The approach described in this paper provides a fast technique that efficiently generates 3D frontiers and plans observer positions while avoiding walking dense voxel grids for each LiDAR return. It integrates with a conventional graph SLAM pipeline enabling both online operation in complex environments, and scalability to large areas where loop closures are essential. Efficiency is gained by directly representing the frontier as the *boundary* between observed free and unobserved space, as a mesh, rather than maintaining a dense 3D grid of the probability of occupancy in each voxel. This is motivated by the observation that the LiDAR sensor payload has very high SNR, and observes the region around the agent with high confidence, so that the probability grid is effectively binary. Longer range areas need to be visited to map with adequate confidence, and to detect objects using the camera, which is the ultimate goal in the SubT challenge.

## III. Frontier Generation

This section describes the method for generating 3D frontiers. The memoryless processing that is applied independently to each frame to obtain candidate frontiers is described in Section III-A. The sequential processing which fuses these candidate frontiers with the data from previous time steps is described in Section III-B. Finally, methods for accounting for the reachability of frontiers, and multi-agent problems are described in Sections III-C and III-D.

### A. Single viewpoint frontier generation

Our approach uses the direct visibility algorithm of [6], which has previously been applied to robotics problems including vehicle navigation [13] and point cloud colourisation [14]. The method efficiently approximates point cloud visibility via a convex hull in a transformed space, yielding a watertight mesh of the visible space from the current viewpoint.

*1) Pre-processing of point cloud:* To reduce computation and provide the most complete surface possible, a cached, voxelised point cloud is used with resolution $d_{\text{vox}} = 0.3\text{m}$, limited in range to $d_{\max} = 9\text{m}$. The voxelisation maintains a count $n_i$, mean $\boldsymbol{\mu}_i$ and covariance $\boldsymbol{\Sigma}_i$ of points within each voxel $i$, updating with each new point recursively. We will see that surfaces close to the agent can result in errors in visibility determination. This can be mitigated by using the voxel data to subsample points within a range of $\eta_{\text{subsample}} = 1.5\text{m}$ of the agent. Specifically, we replace the single point represented by the mean $\boldsymbol{\mu}_i$ by nine points $\boldsymbol{\mu}_{i,k,l}$ where $k, l \in \{-1, 0, 1\}$, and:

$$\boldsymbol{\mu}_{i,k,l} = \boldsymbol{\mu}_i + \frac{k}{2}\sqrt{\lambda_{i,2}}\boldsymbol{\xi}_{i,2} + \frac{l}{2}\sqrt{\lambda_{i,3}}\boldsymbol{\xi}_{i,3} \tag{1}$$

where $\lambda_{i,j}$ is the $j$-th largest eigenvalue of $\boldsymbol{\Sigma}_i$, and $\boldsymbol{\xi}_{i,j}$ is the corresponding eigenvector. Note that $\boldsymbol{\xi}_{i,1}$ provides an estimate of the normal vector of the surface. Since the sign of the eigenvector is free, we choose it such that $\boldsymbol{\xi}_{i,1}^T(\boldsymbol{p} - \boldsymbol{a}) \leq 0$, where $\boldsymbol{p}$ is the latest point added to the voxel, and $\boldsymbol{a}$ is the agent position at the time of that point. A voxel is excluded if the normal vector points away from the current sensor position, as shown below (2).

We supplement the observed points with those sampled on the surface of a sphere centred on the agent position. Points on this sphere which are determined to be visible represent frontiers, since they are maximum-range points which are unobscured by closer observed points. We refer to $\mathcal{S}(\boldsymbol{a}, d_{\max})$ as the set of points sampled on the sphere centred on the agent position $\boldsymbol{a}$ with radius $d_{\max}$. In practice, these are limited to elevation angles that are able to be viewed by the sensor.

The complete point cloud is denoted as the set:

$$\mathcal{P} = \bigcup_{i \in \mathcal{C}_{d_{\max}} \setminus \mathcal{C}_{\eta_{\text{subsample}}}} \{\boldsymbol{\mu}_i\} \cup \bigcup_{(i,k,l) \in \mathcal{C}_{\eta_{\text{subsample}}} \times \{-1,0,1\}^2} \{\boldsymbol{\mu}_{i,k,l}\} \cup \mathcal{S}(\boldsymbol{a}, d_{\max}) \tag{2}$$

where $\mathcal{C}_r = \left\{ i \,\middle|\, ||\boldsymbol{\mu}_i - \boldsymbol{a}|| \leq r,\ \boldsymbol{\xi}_{i,1}^T(\boldsymbol{\mu}_i - \boldsymbol{a}) \leq 0 \right\}$, the latter condition of which excludes voxels for which the surface faces away from the agent.

*2) Generation of visible mesh:* We distinguish between visible and non-visible points based on the direct point cloud visibility method of [6]. In particular, using the exponential inversion kernel, a point $\boldsymbol{p}$ is transformed to:

$$\tilde{\boldsymbol{p}} = t(\boldsymbol{p} - \boldsymbol{a}) = \frac{\boldsymbol{p} - \boldsymbol{a}}{||\boldsymbol{p} - \boldsymbol{a}||}||\boldsymbol{p} - \boldsymbol{a}||^{\gamma} \tag{3}$$

where $\boldsymbol{a}$ denotes the observer position for the viewpoint. This yields a transformed point cloud:

$$\tilde{\mathcal{P}} = t(\mathcal{P}) = \{t(\boldsymbol{p})|\boldsymbol{p} \in \mathcal{P}\} \tag{4}$$

Using a small negative value for $\gamma$, the transformation reflects about the unit sphere and compresses range making distant points closer to the origin, and close points further away, as illustrated in Figure 2(a)-(b) (depicted with the value that we adopt in our experiments, $\gamma = -0.018$). Subsequently, the convex hull of points in the transformed space is calculated using qhull [15], which has complexity $O(n \log n)$ in the number of query points. We denote this operation:

$$(\tilde{\mathcal{V}}, \tilde{\mathcal{T}}) = \text{qhull}(\tilde{\mathcal{P}}) \tag{5}$$

where $\tilde{\mathcal{V}} \subseteq \tilde{\mathcal{P}}$ is the set of visible points, and $\tilde{\mathcal{T}} \subseteq \tilde{\mathcal{V}}^{\sim 3}$ describes the triangulated mesh, where $\tilde{\mathcal{V}}^{\sim 3}$ denotes the set of three-element subsets of points in $\tilde{\mathcal{V}}$. Subsequently, we can obtain $\mathcal{V}$ and $\mathcal{T}$ by passing $\tilde{\mathcal{V}}$ and $\tilde{\mathcal{T}}$ through the inverse of the transformation (3); practically, this calculation can be avoided by tracking the indexing in reference to $\mathcal{P}$. The set of frontier points is initialised as the visible sphere points:

$$\mathcal{F} = \mathcal{V} \cap \mathcal{S}(\boldsymbol{a}, d_{\max}) \tag{6}$$

We use the term *viewpoint* to refer to the tuple $(\tau, \boldsymbol{a}, \mathcal{V}, \mathcal{T})$, which incorporates the time $\tau$, agent position and mesh of visible points. An example of a viewpoint is shown in the top of Figure 1. We use the notation $\mathcal{H}$ to refer to the set of viewpoints collected by an agent at different times.

*3) Discontinuity frontiers:* In addition to points on the maximum range sphere, frontiers can also be identified by searching for long edges ($> \eta_{\text{disc}} = 0.8$m) in the resulting mesh of visible points, $\mathcal{T}$. These edges generally represent depth discontinuities which could be openings, as illustrated in yellow in Figure 2(c) and (g).

Consider a triangle $\mathcal{R} \in \mathcal{T}$ with a single long edge $\mathcal{E} \subset \mathcal{R}$. The vertices are usually[1] observed LiDAR points, representing occupied space. Since the frontier is the boundary between observed free and unobserved space, we need to introduce a new point along the long edge representing the frontier. This is used to test visibility, and determine when the frontier has been observed. Therefore, we split the long edge at its halfway point $\boldsymbol{x} = \text{mean}(\mathcal{E}) \triangleq \frac{1}{|\mathcal{E}|}\sum_{p \in \mathcal{E}} \boldsymbol{p}$, replacing the single triangle by two:

$$\begin{gathered} \mathcal{V}^+ = \mathcal{V}^- \cup \{\boldsymbol{x}\}; \qquad \mathcal{F}^+ = \mathcal{F}^- \cup \{\boldsymbol{x}\} \\ \mathcal{T}^+ = \mathcal{T}^- \backslash \{\mathcal{R}\} \cup \bigcup_{p \in \mathcal{E}} \{\{\boldsymbol{y}, \boldsymbol{p}, \boldsymbol{x}\}\} \end{gathered} \tag{7}$$

[1]In limited cases they will be maximum-range sphere points.

```
Data: $(\mathcal{V}, \mathcal{T}, \mathcal{F})$ from discontinuity splitting
Result: Segmented frontiers $\mathcal{G} = \{(\mathcal{V}_f, \mathcal{T}_f, \mathcal{F}_f)\}_f$
$\mathcal{U} := \{\mathcal{R} \in \mathcal{T} | \mathcal{R} \cap \mathcal{F} \neq \emptyset\}$ // Set of frontier triangles
$\mathcal{G} := \emptyset$ // Set of segmented frontiers
while $\mathcal{U} \neq \emptyset$ do
    Pick $\mathcal{R}_a \in \mathcal{U}$; set $\mathcal{O} := \{\mathcal{R}_a\}$ // Anchor, Open list
    $\mathcal{V}_f = \emptyset,\ \mathcal{T}_f = \emptyset,\ \mathcal{F}_f = \emptyset$ // New frontier
    while $\mathcal{O} \neq \emptyset$ do
        Pick $\mathcal{R} \in \mathcal{O}$; set $\mathcal{O} := \mathcal{O} \backslash \{\mathcal{R}\},\ \mathcal{U} := \mathcal{U} \backslash \{\mathcal{R}\}$
        $\mathcal{V}_f := \mathcal{V}_f \cup \mathcal{R},\ \mathcal{F}_f := \mathcal{F}_f \cup (\mathcal{R} \cap \mathcal{F})$
        $\mathcal{T}_f := \mathcal{T}_f \cup \{\mathcal{R}\}$
        // Add neighbouring triangles to open list
        $\mathcal{O} := \mathcal{O} \cup \big\{\mathcal{X} \in \mathcal{U} \big| |\mathcal{X} \cap \mathcal{R}| \geq 1,$
          $||\text{mean}(\mathcal{X}) - \text{mean}(\mathcal{R}_a)|| \leq \eta_{\text{dia}}\big\}$
    end
    if Area$(\mathcal{T}_f) > \eta_{\text{area}}$ then
        $\mathcal{G} := \mathcal{G} \cup \{(\mathcal{V}_f, \mathcal{T}_f, \mathcal{F}_f)\}$
    end
end
```

**Algorithm 1:** Frontier segmentation algorithm, taking the set of vertices, triangles and frontier points, and returning set of frontiers, resulting from the region growing algorithm. Area$(\mathcal{T}_f)$ denotes the sum of areas of triangles in $\mathcal{T}_f$.

where $^-$ and $^+$ respectively denote pre-update and post-update quantities, and $\{\boldsymbol{y}\} = \mathcal{R} \backslash \mathcal{E}$ is the point opposite the edge being split. Triangles with more than one long edge can be split similarly. We denote the operation splitting all long edges as:

$$(\mathcal{V}^+, \mathcal{T}^+, \mathcal{F}^+) = \text{Disc}(\mathcal{V}^-, \mathcal{T}^-, \mathcal{F}^-) \tag{8}$$

*4) Frontier segmentation:* Connected groups of frontier triangles are extracted via a region growing algorithm operating on the mesh, and those with area $< \eta_{\text{area}} = 0.2\text{m}^2$ are discarded. Region growing proceeds by marking all triangles containing a frontier point as frontier, selecting a frontier triangle, then adding triangles which share an edge with a triangle in the frontier set. The diameter of a frontier is limited to $\eta_{\text{dia}}$ to ensure that navigating to the mean position of vertices is a reasonable approach for observing the frontier. Each frontier is assigned a unique identifier (uid) for use in establishing sequential correspondence and frontier selection. The segmentation method is summarised in Algorithm 1.

*5) Examples:* The operation of the point cloud visibility and frontier generation is illustrated in Figure 2; details are described in the caption. Figure 1 provides a 3D example from data collected in the CSIRO test tunnel, showing the adaptive voxelisation, the determination of visible points and resulting watertight mesh, and the resulting frontiers (the agent entered the tunnel from the left). The frontiers are represented via a triangulated mesh, summarised by a covariance ellipsoid.

## B. Sequential updating

This section describes the sequential processing that combines information between viewpoints. The processing described in this section is repeated after moving a distance

$d_{\text{min}} = 1$m, at a minimum interval of $T_{\text{min}} = 0.5$s. A viewpoint is stored permanently after each $d_{\text{vp}} = 5$m of vehicle motion for subsequent visibility processing. The data required for long-term storage is similar to that retained for a graph SLAM system. Since frontiers and viewpoints are each localised in time, they are stored relative to the SLAM trajectory at the time of observation, so that their positions are adjusted as SLAM corrections are made.

The sequential update process takes candidate frontiers from the memoryless processing described above, and removes triangles that had already been explored by previous viewpoints. Likewise, it examines frontiers carried over from previous processing steps, and removes triangles that have now been observed by the new viewpoint. Triangles are removed if all frontier vertices are visible in a viewpoint, according to the tests described below.

The visibility test compares frontier vertices of new (or old) frontiers to triangles in old (or new) viewpoint(s). The frontier point $\boldsymbol{p} \in \mathcal{F}_f$ is not visible if there exists a triangle $\mathcal{R}$ which is intersected by the line from the agent $\boldsymbol{a}$ to $\boldsymbol{p}$; in practice we consider a point observed if it is at least $\eta_{\text{vis}} = 5$mm in front of a triangle. This can be done using well-known methods, e.g., [16]. Whilst this is a raytracing operation, it is applied to a relatively small number of frontier points, and thus has manageable complexity. This is in contrast to dense mapping frontier methods, where raytracing must be performed for every LiDAR point. The test is skipped for frontiers with no points within the maximum range of a viewpoint.

The visibility test removes triangles from the mesh for which all frontier points were observed in the new viewpoint:

$$\mathcal{T}_f^+ := \mathcal{T}_f^- \setminus \{\mathcal{R} \in \mathcal{T}_f^- | \text{Vis}(\boldsymbol{p}) = 1 \; \forall \; \boldsymbol{p} \in \mathcal{R} \cap \mathcal{F}_f^-\}$$
$$\mathcal{V}_f^+ := \bigcup_{\mathcal{R}\in\mathcal{T}_f^+} \mathcal{R}; \qquad \mathcal{F}_f^+ := \bigcup_{\mathcal{R}\in\mathcal{T}_f^+} \mathcal{R}\cap\mathcal{F}_f^- \tag{9}$$

We refer to the operation in (9) as $(\mathcal{V}_f^+, \mathcal{T}_f^+, \mathcal{F}_f^+) := \text{NotVis}((\mathcal{V}_f, \mathcal{T}_f, \mathcal{F}_f), (\mathcal{V}, \mathcal{T}, \mathcal{F}))$. In order to perform the visibility test, the old frontier or viewpoint is transformed into the current coordinate frame of the agent using our SLAM method, based on [17]. It is possible to defer tests for which there is insufficient knowledge of the frame (i.e., which need loop closures to improve precision), but this was not found to be necessary in our application.

New frontiers and those which have been observed in the viewpoint are then clustered, collecting groups of nearby triangles into new frontiers. Due to the real-time requirement, this is performed using a non-iterative $k$d-tree approach based on [18]. Subsequently, the correspondence of old and new frontiers is determined based on Gaussian approximations of the points in each. This is performed calculating an optimal 2D assignment using the Jonker-Volegant algorithm [19], where the cost of assigning old frontier $f$ to new frontier $g$ is the KL divergence $c(f, g) = D(\mathcal{N}(f)||\mathcal{N}(g))$, where $\mathcal{N}(f)$ and $\mathcal{N}(g)$ respectively refer to the Gaussian approximation of the old and new frontiers $f$ and $g$. The assignment is used as an aid in frontier selection, allowing easy continuation of the same task between updates.

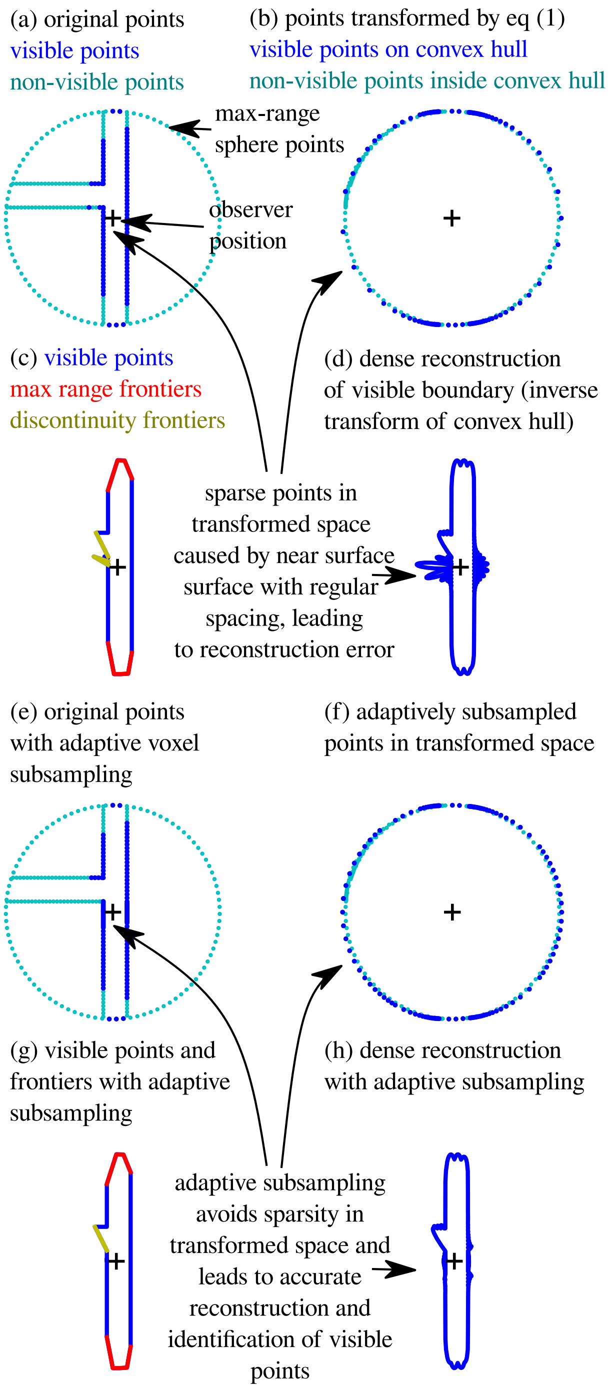
(a) original points
visible points
non-visible points
(b) points transformed by eq (1)
visible points on convex hull
non-visible points inside convex hull
max-range sphere points
observer position
(c) visible points
max range frontiers
discontinuity frontiers
(d) dense reconstruction of visible boundary (inverse transform of convex hull)
sparse points in transformed space caused by near surface surface with regular spacing, leading to reconstruction error
(e) original points with adaptive voxel subsampling
(f) adaptively subsampled points in transformed space
(g) visible points and frontiers with adaptive subsampling
(h) dense reconstruction with adaptive subsampling
adaptive subsampling avoids sparsity in transformed space and leads to accurate reconstruction and identification of visible points

Fig. 2. 2D example of point cloud visibility algorithm for a simple hallway scenario. (a) shows input point cloud, consisting of observed points, supplemented by points on sphere, and the observer located at the black $+$. (b) shows transformed points, colouring visible points (those lying on the convex hull) and non-visible points (corresponding points in (a) coloured similarly). Note the sparse distribution of points at azimuths where the wall is close to the observer. (c) shows the visible points, connected by straight edges, with color denoting frontiers identified via points on the sphere, and those identified via long edges. Note the anomalous frontier caused by a point marked as visible on the other side of the near wall. (d) shows the inverse transform of the convex hull of (b), revealing the implied visible region, again showing the consequence of the sparse sampling of near surfaces in the transformed space in (b). (e)-(h) shows the same results where the sampling density is increased by $3\times$ in the region within 1.5m of the observer, eliminating the anomalous frontier, demonstrating how adaptive subsampling mitigates the reconstruction error in (c).

```
Data: Time $\tau$, agent position $\boldsymbol{a}$, point cloud $\mathcal{P}$ from
      (2), previous frontiers $\mathcal{G}$ and viewpoints $\mathcal{H}$
Result: New set of frontiers $\mathcal{G}$ and viewpoints $\mathcal{H}$
Calculate $\tilde{\mathcal{P}}$ from (4)
$(\tilde{\mathcal{V}}, \tilde{\mathcal{T}}) :=$ qhull$(\tilde{\mathcal{P}})$ (5)
Calculate $(\mathcal{V}, \mathcal{T})$ by correspondence of $\mathcal{V}$ and $\tilde{\mathcal{V}}$
Initialise $\mathcal{F}$ by (6)
$(\mathcal{V}, \mathcal{T}, \mathcal{F}) :=$ Disc$(\mathcal{V}, \mathcal{T}, \mathcal{F})$ (8)
Calculate new frontiers $\mathcal{G}'$ from $(\mathcal{V}, \mathcal{T}, \mathcal{F})$ using
  Algorithm 1
for each new frontier $(\mathcal{V}_f, \mathcal{T}_f, \mathcal{F}_f) \in \mathcal{G}'$ and old
  viewpoint $(\tau', \boldsymbol{a}', \mathcal{V}', \mathcal{T}') \in \mathcal{H}$ within range do
    $(\mathcal{V}_f, \mathcal{T}_f, \mathcal{F}_f) :=$
      NonVis$((\mathcal{V}_f, \mathcal{T}_f, \mathcal{F}_f), (\tau', \boldsymbol{a}', \mathcal{V}', \mathcal{T}'))$ (9)
end
$\bar{\mathcal{G}} := \emptyset$ // Set of updated frontiers
for each old frontier $(\mathcal{V}_f, \mathcal{T}_f, \mathcal{F}_f) \in \mathcal{G}$ in range of
  new viewpoint do
    $(\mathcal{V}'_f, \mathcal{T}'_f, \mathcal{F}'_f) :=$ NotVis$((\mathcal{V}_f, \mathcal{T}_f, \mathcal{F}_f), (\tau, \boldsymbol{a}, \mathcal{V}, \mathcal{T}))$
      (9)
    if $(\mathcal{V}'_f, \mathcal{T}'_f, \mathcal{F}'_f) \neq (\mathcal{V}_f, \mathcal{T}_f, \mathcal{F}_f)$ then
        $\bar{\mathcal{G}} := \bar{\mathcal{G}} \cup \{(\mathcal{V}'_f, \mathcal{T}'_f, \mathcal{F}'_f)\}$ // Frontier updated
        $\mathcal{G} := \mathcal{G} \backslash \{(\mathcal{V}'_f, \mathcal{T}'_f, \mathcal{F}'_f)\}$
    end
end
$\mathcal{G}' :=$ Cluster$(\mathcal{G}' \cup \bar{\mathcal{G}})$ // New frontier clusters
$\mathcal{G}' :=$ Assign$(\mathcal{G}', \bar{\mathcal{G}})$ // Old/new correspondence
$\mathcal{G}' := \{(\mathcal{V}_f, \mathcal{T}_f, \mathcal{F}_f) \in \mathcal{G}' | \text{Area}(\mathcal{T}_f) > \eta_{\text{area}}\}$
$\mathcal{G} := \mathcal{G} \cup \mathcal{G}'$ // New set of frontiers
$\mathcal{H} := \mathcal{H} \cup \{(\tau, \boldsymbol{a}, \mathcal{V}, \mathcal{T})\}$ // New set of viewpoints
```

**Algorithm 2:** Processing performed to generate and update frontiers at each iteration.

The complete process for generating and updating frontiers is given in Algorithm 2.

### C. Observer planning

While the algorithm of Sections III-A and III-B yields frontiers that are robust to variations in ground slope and are suitable for ground and aerial vehicles, additional checks are required to determine from where it will be best to observe the frontier. The first step of this process is to determine a collection of potential observer locations; as with [13], this is performed using the mesh generated in (5). Mesh vertices are initially classified as traversable if the angle between the normal and the $z$-axis is less than $\eta_{\text{norm}} = 30°$, i.e.,

$$\mathcal{A}' = \left\{ \boldsymbol{p} \in \mathcal{V} \,\middle|\, |\boldsymbol{e}_z^T \boldsymbol{\xi}_{i,1}| \geq \cos \eta_{\text{norm}} \right\} \tag{10}$$

where $\boldsymbol{e}_z = [0, 0, 1]^T$, and $\boldsymbol{\xi}_{i,1}$ denotes the eigenvector corresponding to the voxel from which $\boldsymbol{p}$ originated. Subsequently, traversable regions are found as those for which there are no non-traversable points within a neighbourhood of size $\eta_{\text{trav}} = 0.5$m:

$$\mathcal{A}'' = \left\{ \boldsymbol{p} \in \mathcal{A}' \,\middle|\, \boldsymbol{p}' \in \mathcal{A}' \; \forall \, \boldsymbol{p}' \in \mathcal{V} \text{ s.t. } ||\boldsymbol{p} - \boldsymbol{p}'|| \leq \eta_{\text{trav}} \right\} \tag{11}$$

Points in $\mathcal{A}''$ are subsequently subsampled, retaining a single point within a $0.8$m neighbourhood, but retaining the connectivity between neighbourhoods according to the constituent points. Only points that are reachable from the agent according to this simplified graph are retained. We refer to this final set of traversable points as $\mathcal{A}$.

We consider the points in $\mathcal{A}$ as potential locations from which we may observe the frontier. The reward for observing a frontier triangle $\mathcal{R} \in \mathcal{T}_f$ from a location $\boldsymbol{a} \in \mathcal{A}$ is a heuristic based on an approximation of the volume that would be observed from the frontier triangle to the maximum distance $d_{\text{max}}$ if viewed from agent position $\boldsymbol{a}$, assuming that the view was unimpeded. The approximation is based on the volume of a spherical sector. The area of the sector with radius $r$ is $A = \Omega r^2$, where $\Omega$ is the solid angle of the sector, while the volume is $\Omega \frac{r^3}{3}$. Therefore, if the triangle has an area $A = \text{Area}(\mathcal{R})$ at range $r = ||\,\text{mean}(\mathcal{R}) - \boldsymbol{a}||$ from the agent, the new volume observed could be approximated as $\frac{A}{3r^2}(d_{\text{max}}^3 - r^3)$. Further accounting for the angle $\theta$ between the triangle's normal vector and the vector from the triangle midpoint to the agent, we arrive at an estimate of

$$\hat{r}(\boldsymbol{a}, \mathcal{R}) = \frac{A \cos^+ \theta}{3r^2}(d_{\text{max}}^3 - r^3) = \frac{A \cos^+ \theta}{3} \left( \frac{d_{\text{max}}^3}{r^2} - r \right) \tag{12}$$

where $\cos^+ \theta \triangleq \max\{\cos \theta, 0\}$. Empirically, it was found that the resulting frontier selection performs better if the dynamic range of the reward is reduced. One way of achieving that is to replace the term $\frac{d_{\text{max}}^3}{r^2}$ with $\frac{d_{\text{max}}^2}{r}$; this may be viewed as accounting for a limiting of growth in the $z$ axis due to the floor and ceiling.

The observer position and reward for a new or newly updated frontier is then calculated as the position which observes the largest total new volume:

$$\boldsymbol{o}_f = \arg\max_{\boldsymbol{a} \in \mathcal{A}} \sum_{\mathcal{R} \in \mathcal{T}_f} \hat{r}(\boldsymbol{a}, \mathcal{R}); \quad \hat{r}_f = \max_{\boldsymbol{a} \in \mathcal{A}} \sum_{\mathcal{R} \in \mathcal{T}_f} \hat{r}(\boldsymbol{a}, \mathcal{R}) \tag{13}$$

Observer positions for frontiers that are not updated in a given iteration are left unchanged.

### D. Multi-agent frontiers

While it would be preferable to replicate the viewpoints of each agent at every other agent, the communications involved in doing so would be prohibitive. The data already transmitted to solve the multiagent SLAM problem (i.e., submaps of 3D surfels, as described in [17]) was found to be insufficient to reconstruct a reliable estimate of the boundary between known free and unknown space. For this reason, the approach taken was to share summaries of frontiers (i.e., the mean and covariance of the points in a frontier, the reward estimate and the best observer location) between agents, and penalise the rewards according to agents' trajectories. Thus, if a remote agent's trajectory passes through a frontier from the local agent, its reward is discounted, and likewise, if the local agent's trajectory (or that of another remote agent) passes through a frontier from a remote agent, its reward is discounted. Since trajectories are shared for the purpose of multiagent SLAM, the communications overhead is very small. The concept is similar to the implicit communication of the grazing behaviour in [20]. The reward discount is achieved by storing a point every five seconds along each agent's trajectory; we refer to the set of points for all agents up to and including the current time as $\mathcal{L}$. The discounted reward is then:

$$\tilde{r}_f = \hat{r}_f \prod_{p \in \mathcal{L} \,\middle|\, ||\boldsymbol{p}-\boldsymbol{o}_f|| \leq \eta_d} d_{\rho,\eta_d}(||\boldsymbol{p} - \boldsymbol{o}_f||) \tag{14}$$

where $d_{\rho,\eta}(r) = \rho + (1-\rho)\sin^2\left(\frac{\pi r}{2\eta}\right)$. The discount reduces the reward by a factor of $\rho = 0.2$ if the goal is in a position previously visited by the agent, and increases according to a sine wave to have no discount if $||\boldsymbol{p} - \boldsymbol{o}_f|| \geq \eta_d = 6$m. Similar discounts are applied to goal locations which have been attempted and found to be unreachable.

## IV. Frontier-Based Exploration

To explore the environment, each robot repeatedly selects, and navigates to, frontiers in a distributed manner. The selection mechanism differs between the UGV and UAV platforms, due to different planners, and a rapid integration effort onto the UAV before the event.

### A. UGV frontier selection

The most common methods for frontier selection choose the nearest frontier to an agent, or the largest frontier. In the complex environments presented by the SubT Urban Challenge, both of these were found to be ineffective. The solution adopted was a utility function, which selects the frontier which provides the highest reward per unit distance:

$$f^* = \arg\max_f u(f, \boldsymbol{a}); \quad u(f, \boldsymbol{a}) = \tilde{r}_f / d(\boldsymbol{o}_f, \boldsymbol{a}) \tag{15}$$

This performed adequately for both well-structured areas and more open regions. The distance in (15) is calculated using the topometric map, described in Section IV-C.

Agents operate independently, and coordinate when communications are available through two mechanisms. Firstly, other agents' latest goals were included in the set $\mathcal{L}$ when calculating the frontier discounts in (14). Secondly, agents are prevented from selecting a frontier that is currently selected by another agent.

In challenging real-world environments, it is common for navigation to a frontier to fail. Progress towards the frontier is monitored, and if the closest distance reached to the currently selected frontier does not improve by at least 1m in any 15s period, an additional penalty $\rho_{\text{fail}}$ is applied to the frontier, which is cleared when the frontier is updated.

The frontier selection is summarised in Algorithm 3.

```
Data: Agent position $\boldsymbol{a}$; frontier observer positions
      $\boldsymbol{o}_f$; $\mathcal{L}$ containing agents' historical trajectories,
      failed goals, and current goals; frontiers
      selected by other agents $\mathcal{M}$
Result: Newly selected frontier $f^*$
for each frontier $f \notin \mathcal{M}$ do
    Calculate $\boldsymbol{o}_f$ and $\hat{r}_f$ using (13), $\tilde{r}_f$ using (14)
    if $f$ has failed and not been updated since then
        $\tilde{r}_f := \tilde{r}_f - \rho_{\text{fail}}$
    end
end
Calculate $f^*$ using (15), excluding $f \in \mathcal{M}$
```

**Algorithm 3:** Algorithm for multiagent frontier selection.

### B. UAV frontier selection

Frontier selection for the UAV did not use the set of sparse candidate observer positions and instead navigates to the mean of the selected frontier. Due to flight time, selection was biased to avoid backtracking, and the drones were only launched near large novel spaces. Frontier selection is independent of other agents, and uses the reward:

$$\hat{r}_f = p_f \frac{\min\{2\sqrt{5\lambda_{f,2}}, \sqrt{A_f}, \chi_{\text{size}}\}^2}{\max\{\chi_{\text{blacklist}}, d_w(\boldsymbol{o}_f, \boldsymbol{a})\}} \tag{16}$$

The term $2\sqrt{5\lambda_{f,2}}$ (where $\lambda_{f,2}$ is the second-largest eigenvalue of the covariance matrix of frontier vertices) and the square root of the frontier area both represent estimates of the minimum dimension of the frontier, taken as information regarding whether it is likely that the UAV will be able to proceed through the region. The limit on maximum size $\chi_{\text{size}} = 5$m limits the pull associated with large, distant frontiers. The weighted distance $d_w(\boldsymbol{o}_f, \boldsymbol{a})$ from the agent position $\boldsymbol{a}$ to the frontier centre $\boldsymbol{o}_f$ weights the $z$-component by a factor of two in order to discourage movement to frontiers far above or below, while the limit on minimum size $\chi_{\text{blacklist}} = 3$m prevents fixation on very close frontiers.

The multiplier $p_f = s_f w^{\hat{\boldsymbol{v}}_f^T \hat{\boldsymbol{v}}'}$ provides momentum, where $s_f = 4$ if the frontier is the same as that selected at the previous iteration, and one otherwise. The weight $w = 2$ is raised to the power of the dot product of the unit vector from the agent to the frontier $\hat{\boldsymbol{v}} = (\boldsymbol{o}_f - \boldsymbol{a})/||\boldsymbol{o}_f - \boldsymbol{a}||$ and $\hat{\boldsymbol{v}}'$, the same quantity when the frontier was first selected, in order to encourage continuous motion in the same direction.

The same criterion is utilised to blacklist frontiers that do not make sufficient progress within 15s. The minimum size of frontiers considered for selection is $\chi_{\text{size}} = 1.7\text{m}^2$. A frontier is considered complete if the agent reaches within $\chi_{\text{blacklist}}$ of the frontier centre.

### C. Topometric map

When frontiers are in the local vicinity of the agent, local navigation methods are employed to find a feasible path. However, when the frontier being investigated by an agent is completed (e.g., reaching a dead end), to achieve scalability an alternate method must be employed to navigate to the subsequently selected frontier, which may be distant.

To keep track of frontiers at a global scale, we have developed a multi-agent topometric map (i.e., a topological graph with metric information) for UGV navigation purposes. The map consists of a graph, where nodes represent points along agent trajectories (spaced by around 3m), and edges represent the connections along the trajectory, supplemented by edges generated when the logic in Section III-C detects a path between two nodes. The map is shared between agents, allowing one agent to navigate to another agent's frontier. Frontier nodes are incorporated into the graph by connecting them to the node on the agent's trajectory closest in time to the time when the frontier was last updated. An example of the topometric map generated over multiple agents over a mission is shown below in Figure 3(b), in white. To navigate to a distant frontier, we first navigate to the associated node using the path generated by A* on the topometric graph. The same A* planner is leveraged to compute the distance from the agent to each frontier in the map, as used in (15).

## V. Experimental Results

Frontier exploration was the primary method used by the CSIRO team to explore the abandoned Satsop Nuclear Power Plant, as part of the DARPA SubT UC. A heterogeneous robot team of seven UGVs and UAVs was deployed (Figure 3(a)), commanded by a single operator during four one-hour trials. The operator had no prior knowledge of the courses, which spanned multiple levels, included stairwells, vertical shafts and had lead-lined concrete walls blocking direct communications. Each robot was equipped with a LiDAR based perception package, and ran the same SLAM and frontier generation algorithms. The goal of the challenge was to detect as many artefacts as possible, e.g., backpacks, "survivors", and cell phones. The inset in Figure 3(a) provides an example of an artefact detected during an explore mission, annotated and relayed to the operator for confirmation.

The most successful strategy used at the UC depended on the *explore-sync* behaviour: An agent explores into the facility, rapidly losing communication contact, while building a map. After three minutes it drives back to communication range and uplinks data (maps, remaining frontiers, and object detections) to the operator. Given this new map, subsequent robots were commanded to drop relay nodes (expanding communication range), or were sent on their own explore-sync missions. Communication was performed using Rajant ES1 (UGVs and relay nodes), and DX2 (UAV) radios. Agents communicate with each other and with the operator station in a peer-to-peer, disruption-tolerant fashion as links are available. The distributed processing described in Section III-D tolerates high latency due to dropped communications; this is mitigated by the fact that agents operating in the same vicinity generally have good communications.

The UAV was explicitly launched from the BIA 5 Titan UGVs near vertical shafts, or areas hard to observe from the ground. Robots without directed commands would automatically start explore-sync behaviours. When the system worked well, the operator was required only to confirm object detections, but due to the complexity of the course, the operator also had to provide commands to descend stairwells, or command robots to a global pose to help direct exploration.

The explore-sync behaviour accounted for the majority of detected objects and distance travelled: One UGV covered 782m after descending a stairwell and detecting 3 of 5 potential objects in that area; Another UGV traversed 1.3km on an upper level, observing 7 of 11 objects. Not all observed objects were scored due to perception and communications. Explore-sync also produced several difficulties: Agents often did not return to the distant region being explored after completing their sync task, as frontier selection was myopic. In hindsight, a simple heuristic for addressing this would have been to return to the region being explored after sync. Rooms with small doors were often not explored as observer positions were not found due to the insufficient voxel resolution.

Snapshots of exploration are shown in Figure 4 (for the UGV) and Figure 5 (for the UAV). In both cases, each frontier is shown in magenta (depicted via the mean of the frontier's vertices). Figure 3(b) shows the final map merged from all agents, with frontiers and the topometric map overlaid.

The Satsop facility contained vastly different building scales and structure types compared to environments considered during development and prior testing. Highly complex roof structures were found to provide many frontiers; the UGV observer planning and UAV frontier selection largely avoided undesirable effects due to these. Improvements are expected by tuning the logic that selects which viewpoints to retain, and performing finer splitting of the long edges in discontinuity frontiers. Occasional instances were observed where frontiers reappear in previously explored space; again, this can be improved by tuning the logic for retaining viewpoints.

One challenge for the UGVs was an instance where the 3D frontier was correctly identified, but due to barriers the path to the frontier was not autonomously planned. This highlights the need for a traversability frontier to plan paths to the 3D frontiers. The relative performance of a method based purely on a traversability frontier, to the proposed and alternate 3D frontier methods, and hybrid traversability/3D methods is an interesting question for further study.

## VI. Conclusion

This paper presents an approach to 3D frontier generation which enables online application in complex environments by directly modelling the boundary of explored and unexplored space rather than computing and storing dense 3D spatial maps. The method works alongside a graph SLAM pipeline to scale to large environments being explored by multiple agents. The method was a core part of the autonomy capability in our team's entry to the SubT UC, where it was utilised on seven agents, including both UGVs and UAVs. A major focus of future work will be on incorporating frontier assignment into a more general multi-agent task allocation framework, considering non-myopic plans.

## References

[1] R. R. Murphy, *et al.*, "Mobile robots in mine rescue and recovery," *IEEE Robotics Automation Magazine*, vol. 16, no. 2, pp. 91–103, 2009.

[2] D. Silver, *et al.*, "Topological exploration of subterranean environments," *Journal of Field Robotics*, vol. 23, no. 6-7, pp. 395–415, 2006.

[3] S. Thrun, *et al.*, "Autonomous exploration and mapping of abandoned mines," *IEEE Robotics Automation Magazine*, vol. 11, no. 4, pp. 79–91, 2004.

[4] T. Chung. (2019) DARPA subterranean (SubT) challenge. [Online]. Available: https://www.darpa.mil/program/darpa-subterranean-challenge

[5] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *CIRA*, 1997, pp. 146–151.

[6] S. Katz and A. Tal, "On the visibility of point clouds," in *ICCV*, 2015, pp. 1350–1358.

[7] R. G. Simmons, *et al.*, "Coordination for multi-robot exploration and mapping," in *AAAI/IAAI*, 2000, p. 852–858.

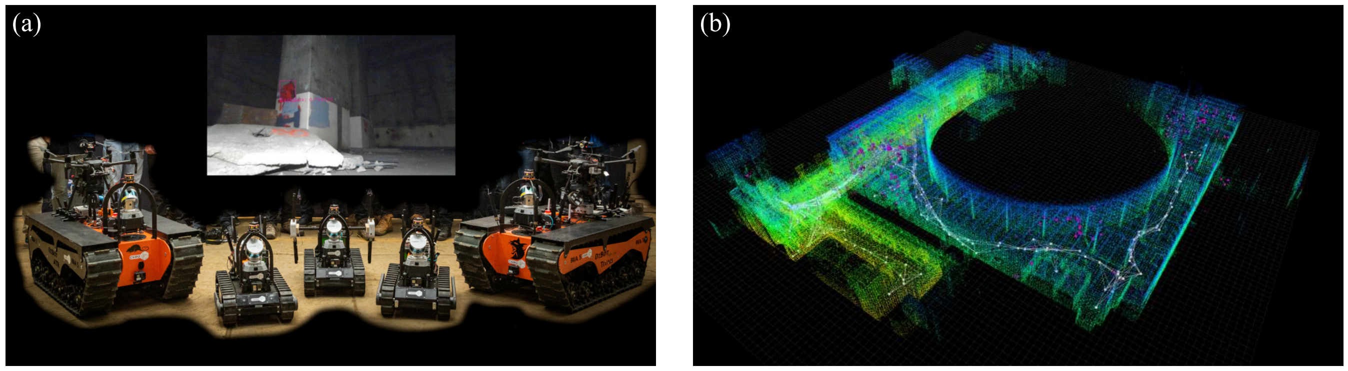

Fig. 3. (a) shows the heterogeneous robot team, with UAVs mounted on the BIA 5 OzBot Titans; example artefact detection (backpack) inset. (b) shows final merged map from all agents (UGV and UAV), with UGV topometric map in white. (Frontier data for some agents is missing in the image). Grid squares represent 1m.

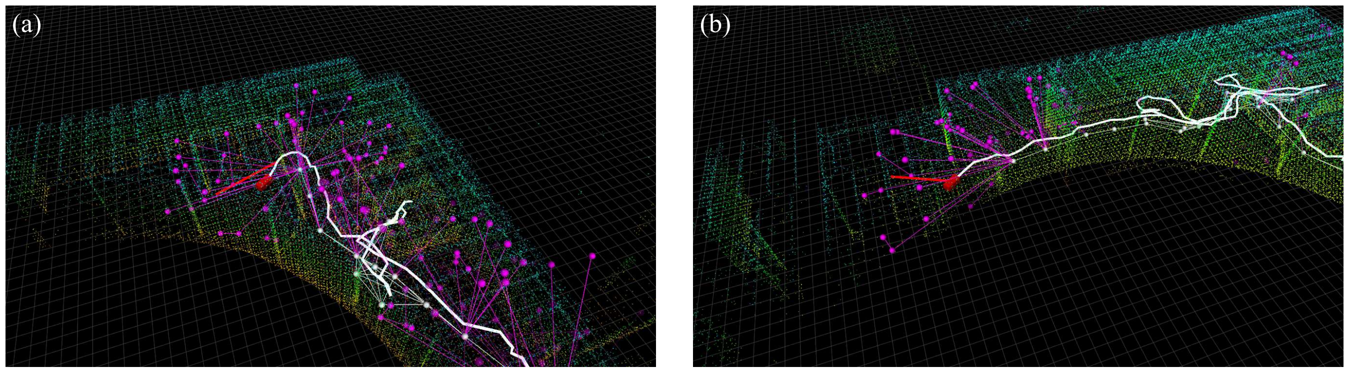

Fig. 4. Snapshots of exploration mission in UC using Titan UGV platform based on BIA 5 OzBot Titan. Frontier positions shown in magenta, agent position and path to selected frontier shown in red, and trajectory to date and topometric graph shown in white. In (a) agent is exploring towards left, before following a promising frontier to the right, and returning to left in (b).

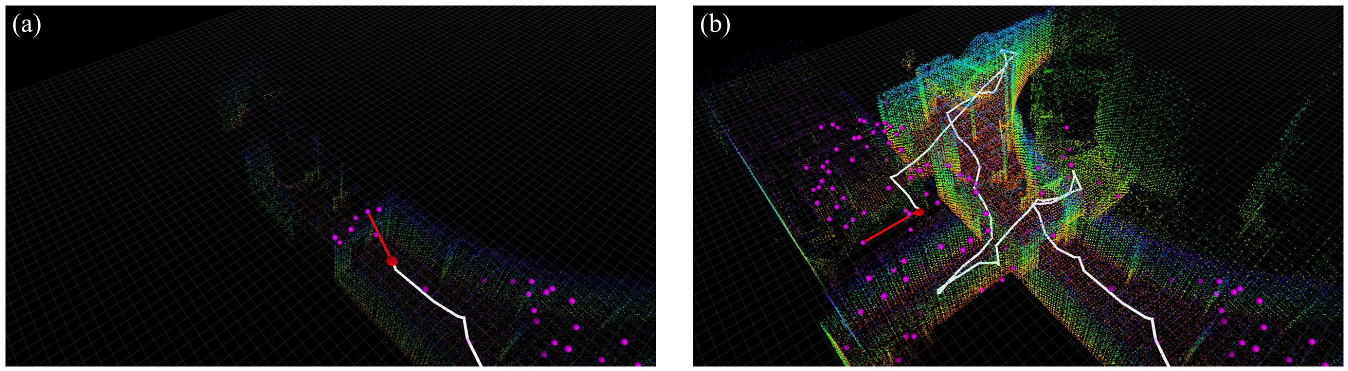

Fig. 5. Snapshots of exploration in UC using Emesent Hovermap UAV. Frontier positions shown in magenta, agent position and path to currently selected frontier shown in red, trajectory to date shown in white.

[8] M. Keidar and G. A. Kaminka, "Efficient frontier detection for robot exploration," *The International Journal of Robotics Research*, vol. 33, no. 2, pp. 215–236, 2014.

[9] J. Oršulić, *et al.*, "Efficient dense frontier detection for 2D graph SLAM based on occupancy grid submaps," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3569–3576, 2019.

[10] W. Hess, *et al.*, "Real-time loop closure in 2D LIDAR SLAM," in *ICRA*, 2016, pp. 1271–1278.

[11] C. Dornhege and A. Kleiner, "A frontier-void-based approach for autonomous exploration in 3D," in *SSRR*, 2011, pp. 351–356.

[12] A. Mannucci, *et al.*, "Autonomous 3D exploration of large areas: A cooperative frontier-based approach," in *Modelling and Simulation for Autonomous Systems*, 2018, pp. 18–39.

[13] F. Ruetz, *et al.*, "OVPC mesh: 3D free-space representation for local ground vehicle navigation," in *ICRA*, 2019, pp. 8648–8654.

[14] P. Vechersky, *et al.*, "Colourising point clouds using independent cameras," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3575–3582, 2018.

[15] C. B. Barber, *et al.*, "The quickhull algorithm for convex hulls," *ACM Trans. Math. Softw.*, vol. 22, no. 4, p. 469–483, Dec. 1996.

[16] P. Shirley and R. Morley, *Realistic Ray Tracing*, 2nd ed. Taylor & Francis, 2008.

[17] M. Bosse, *et al.*, "Zebedee: Design of a spring-mounted 3D range sensor with application to mobile mapping," *IEEE Transactions on Robotics*, vol. 28, no. 5, pp. 1104–1119, 2012.

[18] A. T. Ihler, *et al.*, "Particle filtering under communications constraints," in *SSP Workshop*, July 2005, pp. 89–94.

[19] R. Jonker and A. Volgenant, "A shortest augmenting path algorithm for dense and sparse linear assignment problems," *Computing*, vol. 38, no. 4, p. 325–340, Nov. 1987.

[20] T. Balch and R. C. Arkin, "Communication in reactive multiagent robotic systems," *Auton. Robots*, vol. 1, no. 1, p. 27–52, Feb. 1994.