A Multi-Robot Exploration Planner for Space Applications

Vivek Shankar Varadharajan and Giovanni Beltrame, *Senior Member, IEEE*

***Abstract*—We propose a distributed multi-robot exploration planning method designed for complex, unconstrained environments featuring steep elevation changes. The method employs a two-tiered approach: a local exploration planner that constructs a grid graph to maximize exploration gain and a global planner that maintains a sparse navigational graph to track visited locations and frontier information. The global graphs are periodically synchronized among robots within communication range to maintain an updated representation of the environment. Our approach integrates localization loop closure estimates to correct global graph drift. In simulation and field tests, the proposed method achieves 50% lower computational runtime compared to state-of-the-art methods while demonstrating superior exploration coverage. We evaluate its performance in two simulated subterranean environments and in field experiments at a Mars-analog terrain.**

***Index Terms*—Multi-robot systems, path planning for multiple mobile robots or agents, field robots, space robotics and automation, networked robots.**

I. Introduction

EXPLORING unknown environments with robots offer significant advantages, including minimizing human risk and enabling the parallelization of exploration when deploying a team of robots. Robotic exploration has been attempted on extraterrestrial planets, providing critical insights such as the potential to discover traces of life and identify suitable locations for future human missions [1], [2]. In addition to planetary exploration, robotic exploration has numerous applications on Earth, including search and rescue operations and humanitarian missions [3].

However, exploration in unconstrained environments presents several challenges that need to be addressed concurrently: 1) accurate localization within the environment, 2) efficient path planning that supports the mission objectives, and 3) building a map to generate a reliable environmental model. These three problems—localization, mapping, and planning—are highly interconnected. Localization accuracy depends on establishing correct correspondences between the current observations and the map, mapping accuracy relies on precise localization, and planning effectiveness hinges on the quality of the environmental belief.

Received 21 September 2024; accepted 6 January 2025. Date of publication 20 January 2025; date of current version 29 January 2025. This article was recommended for publication by Associate Editor J. G. Rogers III and Editor G. Loianno upon evaluation of the reviewers' comments. This work was supported in part by NSERC under Grant 2019-05165 and in part by CSA under Grant 19FAPOLA32. *(Corresponding author: Vivek Shankar Varadharajan.)*

The authors are with the MIST Lab, Polytechnique Montreal, Montreal, QC H3T 0A3, Canada (e-mail: vivek-shankar.varadharajan@polymtl.ca).

This letter has supplementary downloadable material available at https://doi.org/10.1109/LRA.2025.3532158, provided by the authors.

Digital Object Identifier 10.1109/LRA.2025.3532158

In this work, we propose a unified approach that integrates localization, mapping, and planning modules, allowing them to communicate and update their states jointly. This unified framework aims to enhance the overall accuracy of exploration by improving each module's performance through this continuous interaction.

The main benefit of our approach is that it achieves higher accuracy, exploration efficiency, and reduced algorithmic runtime compared to state-of-the-art methods [4], [5]. An ideal strategy for exploring unstructured environments involves sampling from the global environmental belief to identify viewpoints that enhance the overall map. However, previous works [4], [5] have shown that global sampling can be computationally expensive. As a solution, we divide exploration planning into two subspaces: 1) local exploration and 2) global exploration. Local exploration focuses on the immediate vicinity of the robot, using a dense set of samples around the robot to identify a high-gain trajectory for exploration. In contrast, global exploration considers the entire environment to determine the next promising local subspace to explore once the local gains diminish.

In our proposed method, the robot constructs a cuboidal grid graph at a predetermined resolution within its current local subspace. This grid graph is then used to identify the trajectory that yields the highest environmental gain. When local exploration yields diminishing returns, the robot switches to a coarser global map to reposition itself toward a higher-gain frontier. Periodically, the robot exchanges gossip messages with other robots, sharing its current global graph and merging environmental information. This shared global map enables all robots to coordinate their exploration strategies for subsequent iterations. Additionally, global graph nodes are linked with pose graph nodes to help correct map drifts over time.

We evaluate our approach using a team of three ground robots in both simulation and real-world environments. In the simulation, we employ two open-source environments from the gbplanner2 dataset [4]: Darpa Cave One and Pittsburgh Mine. These environments are used to assess the performance of our planner. For physical experiments, we conducted tests in the Canadian Space Agency's Mars Yard, a Mars analog environment (see Fig. 1). In these tests, three ground robots explored a 120 m$\times$60 m area for 15 minutes, with each experiment repeated three times to validate our solution. The robots utilized a full autonomy stack for localization, planning, control, and inter-robot communication. Finally, we open-sourced our planner implementation to encourage further research, available at https://github.com/MISTLab/MGGPlanner.

II. Related Works

The problem of autonomous robot exploration as the primary objective has been widely studied [4], [5], [6], [7], [8], [9],

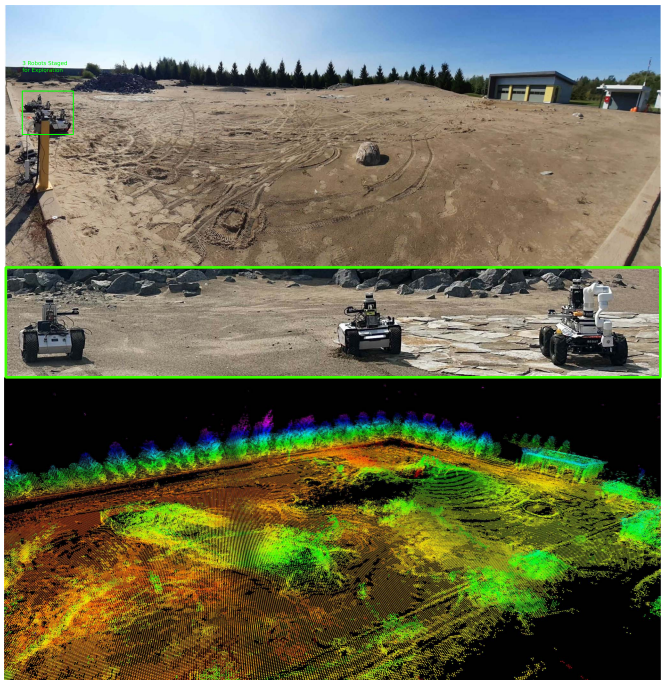

Fig. 1. Mars analog environment used during the deployment. The robots used a full stack autonomy module: Lidar Inertial Odometry (LIO) for localization, the proposed MGG planner for exploration, and a local planner for obstacle avoidance. Communication between the robots and the base station for visualization was done using a dual-band mesh network.

[10], [11]. Some works specifically address local navigational planning [12], [13] to improve traversability in challenging terrains, enabling better interaction with exploration planners. Recent literature in exploration planning can be categorized into frontier-based exploration, topological exploration, and random sampling-based methods.

Frontier-based exploration [8], [14], [15], [16], [17], [18] uses the boundaries of the environment within the robot's map to identify frontiers, the unexplored regions adjacent to explored areas and directs the robot toward them iteratively. These methods often apply local optimization to determine the next best frontier. Some approaches incorporate probabilistic strategies [15], such as expanding a random tree, while others [16] focus on minimizing the number of viewpoints needed to cover the search space efficiently.

Topological exploration approaches [11], [19], [20], [21], [22], [23], [24] create a graph representation of the environment, where nodes represent free space and edges encode the traversability between these nodes. Kim et al. [11] propose a method that identifies topological structures to distribute tasks among multiple robots exploring the environment. Other methods like [21] utilize graph-SLAM to build a topological map and apply depth-first search to find frontiers. However, a common limitation of topological approaches is that they often lack the granularity needed to represent the environment in sufficient detail for effective exploration.

Random sampling-based methods generate random samples in free space and evaluate them to determine the most effective motion trajectory for maximizing exploration objectives. Some methods adopt Next-Best-View (NBV) strategies [25], [26], which sample viewpoints to maximize environmental coverage, though the computational complexity of these approaches grow as the map size increases. Motion primitive-based planning [9] incorporates the traversability of the environment into the planning process, improving the robot's exploration path. Cao et al. [5] introduced a hierarchical framework for concurrent local and global planning, which they later extended to multi-robot exploration [7]. This framework samples a dense set of viewpoints for local space exploration, while the global planner tracks previously visited areas using a coarser representation. Nebula [6] employs a hierarchical planning structure based on a Partially Observable Markov Decision Process (POMDP), using high-density local and low-density global Information Road Maps (IRMs). These local and global IRMs in Nebula are conceptually analogous to the local and global graphs used in this work, and the IRM synchronization among robots uses a ground station while we use a distributed approach in this work.

GBplanner2 [4], a graph-based planner, implements a bifurcated approach to exploration with local and global planning. Locally, it employs a Rapidly exploring Random Graph (RRG) to find the highest exploration gain trajectory, while globally, it maintains a coarse map to guide the robot toward new frontiers.

The key distinction of our work compared to GBplanner2 [4] lies in the structure of the local and global graphs and the optimization process. In our method, the local graph is constructed as a grid rather than using random samples, and the global graph is continuously shared with other robots and optimized using localization estimates. This allows the robots to synchronize their global graph information with neighboring robots. Our approach reduces algorithmic runtime while enhancing exploration efficiency by increasing the surface covered in the explored space.

III. Problem Formulation

Consider a bounded area $\mathbf{Q} \in \mathbb{R}^3$ that needs to be explored by a team of robots using onboard sensors with limited horizontal ($\mathbf{H}_{Fov}$) and vertical ($\mathbf{V}_{Fov}$) fields-of-view. The explorable environment can be classified into three distinct regions: free space ($\mathbf{Q}_{free}$), occupied space ($\mathbf{Q}_{occupied}$), and unknown space ($\mathbf{Q}_{unknown}$). Formally, the environment is discretized into voxels $v \in \mathbb{V}$, where each voxel belongs to one of these three regions, such that:

$$\mathbf{Q} = \mathbf{Q}_{free} \cup \mathbf{Q}_{occupied} \cup \mathbf{Q}_{unknown}$$

The robots' sensors can only perceive surfaces within their line-of-sight, making part of the environment unobservable ($\mathbf{Q}_{unobs}$). Let $\mathbf{q} \in SE(3)$ represent the sensor pose of the robot, with $\mathbf{q} = [p_\mathbf{q}, r_\mathbf{q}]$, where $p_\mathbf{q} \in \mathbf{Q}_{free}$ denotes the position and $r_\mathbf{q} \in SO(3)$ denotes the orientation. Additionally, let $\hat{\mathbf{Q}}_v \subseteq \mathbf{Q}_{free}$ represent the set of sensor poses from which voxel v can be observed and $\hat{\mathbf{Q}}_v^s \subseteq \mathbf{Q}_{free}$ be the set of reachable sensor poses of the robot. The unobservable space is defined as:

$$\mathbf{Q}_{unobs} = \bigcup_{v \in \mathbb{V}} \{v | \hat{\mathbf{Q}}_v^s = \emptyset\}$$

Problem 1: Given the bounded area $\mathbf{Q}$ and the sensor field-of-view parameters $\mathbf{H}_{Fov}$ and $\mathbf{V}_{Fov}$, compute a path represented by a set of viewpoints $\{q_1, \ldots, q_n \mid q_i \in \mathbf{Q}_{free}\}$ that enables the robot to classify the environment into $\mathbf{Q}_{free}$ and $\mathbf{Q}_{occupied}$, while accounting for the robot's motion constraints.

This problem is solved iteratively by each robot in the heterogeneous team, mapping the environment until the entire area is

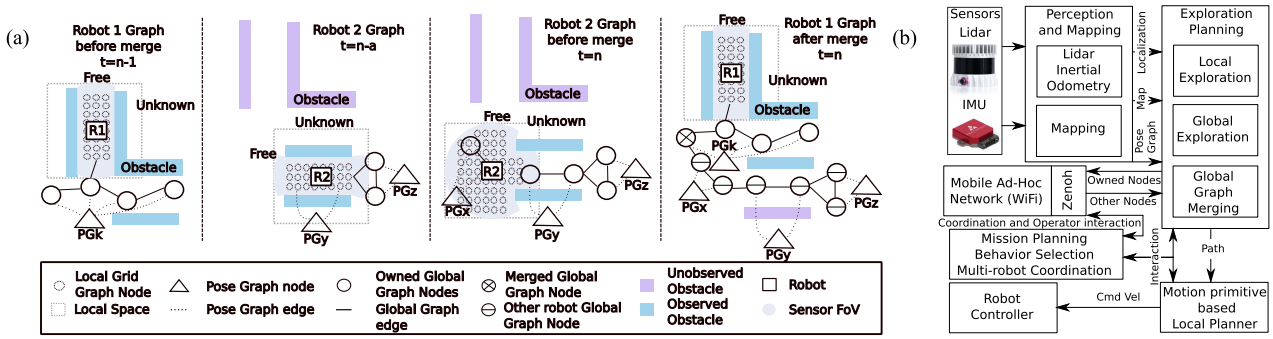

Fig. 2. (a) Graphical illustration of the local grid graph, global graph, traversable space, and their association with the pose graph nodes: the leftmost image shows the graphs of robot R1 one time step before the global graph merge. The second image from the left depicts the graphs of a neighboring robot R2 a time steps before the global graph merge. The third image illustrates R2's graphs at time n, just prior to merging, while the rightmost image displays the merged graphs on R1 at time n. (b) System integration block diagram of the planner and autonomy components: the robot uses LiDAR and IMU to generate a belief pose estimate and constructs an environmental map that integrates merged map updates via a pose graph. The exploration planner generates exploration trajectories through three components: local planning, global planning, and global graph merging. The robot periodically broadcasts its owned global graph nodes to the network, enabling neighboring robots to merge these updates into their own graphs. Similarly, broadcasts received from neighboring robots are merged into the robot's current global graph. The generated exploration plan is executed by a motion-primitive-based local planner running in a closed loop to navigate the robot. Mission behaviors are triggered and monitored by a mission planning component, which also facilitates operator interaction and monitoring when communication is available.

explored, such that:

$$\mathbf{Q}_{unknown} / \mathbf{Q}_{unobs} = \emptyset$$

Problem 2: Given a robot x with its local environmental information $\mathbf{Q}^x$, update all neighboring robots within the communication range $y \in N_{rcom}$ to ensure a unified environmental representation across the team.

This problem is addressed by iterative building and broadcasting the explored regions using a global graph representation of the environment. To achieve effective synchronization, the robots require a common reference frame for environmental updates. We apply a Collaborative Simultaneous Localization and Mapping (CSLAM) approach [27], which aligns the poses of all robots into a unified reference frame, enabling the synchronization of environmental information.

IV. Methodology

The proposed approach for multi-robot exploration in unconstrained environments leverages each robot's onboard sensing and communication capabilities to autonomously explore and map the environment. Previous literature [4], [5], [28] has demonstrated the computational benefits of a bifurcated approach, where detailed planning is conducted in the local subspace, while a global subspace keeps track of unexplored areas. Building on this foundation, we propose a Multi-robot Grid Graph (MGG) planner, which enhances this method by using grid-based local planning and distributed environmental synchronization across multiple robots.

The primary advantage of our approach is that it allows robots to operate in a distributed manner, synchronizing environmental information and performing local planning on a grid graph. This structure improves the solution quality while reducing algorithmic runtime. The local grid planner efficiently identifies exploration paths by taking into account the robot's motion capabilities and sensor configuration. Meanwhile, the global planner maintains a connected graph of previously visited locations, tracking environmental frontiers to assist in revisiting key areas or repositioning the robot when local planning reaches a dead end.

Our method utilizes the global graph to synchronize environmental information among robots. Synchronizing the global graph uses minimal bandwidth, in contrast to synchronizing full point cloud maps, while still preserving a shared representation of both known (Q_{known}) and unknown ($Q_{unknown}$) regions. Furthermore, each global graph node is associated with a pose graph node from the robot's SLAM system. This enables the robot to correct odometry drift through loop closures, incorporating both inter-robot and intra-robot SLAM data. Fig. 2(a) provides a graphical overview of the planning process, and (B) shows the system components used with the exploration planner.

A. Local Exploration Planning

The local exploration process begins by constructing a grid graph in the robot's local cuboidal subspace, considering all feasible discretized configurations. Leaf vertices in the grid graph are identified, and Dijkstra's shortest path algorithm is applied to compute paths to these vertices. The volumetric gain of each leaf vertex is then calculated to identify exploration frontiers, and the best vertex to visit is selected based on this gain. Before execution, the computed path is refined for safety to ensure the robot can traverse it without encountering obstacles.

A key distinction in local exploration planning of the MGG planner compared to gbplanner2 lies in the structured construction of the local grid graph (Algorithm 1), which spans the robot's immediate environment comprehensively. In contrast, gbplanner2 [4] employs random sampling to build its graph. The MGG planner systematically loops through the local subspace, generating a grid that covers all feasible robot configurations, thereby enhancing both the accuracy of exploration and the efficiency of computation. Algorithm 1 outlines the sampling and local graph building process: vertices in free space are first identified within the local planning space (line 6) and projected onto the robot's ground plane belief (line 7) using the *ProjectPoint* function, which predicts the ground plane based on neighboring voxels in the map and projects samples. The nearest neighbor vertex is determined (line 8) using the *NearestVertex* function and evaluated for inclusion in the graph (lines 8-9). Accepted vertices are connected via edges (lines 10-11), and

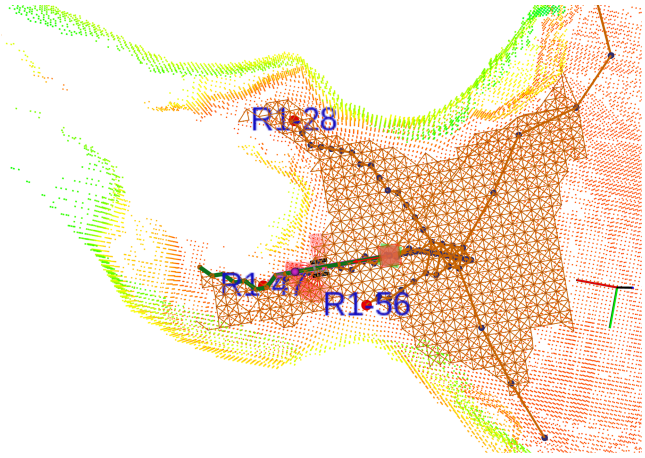

Fig. 3. Illustration of the various graphs built by the robot during one of the exploration test runs: The dark orange represents the local graph, while the light orange with dark blue circles denotes the global graph. The green path indicates the computed local path, and the magenta path represents the refined path. Global graph frontier nodes are marked in red, with numbering in blue prefixed by a robot identifier. Red trajectories show the valid motion primitives generated by the local planner. The voxel map is colored by height for added context.

Algorithm 1: Sampling and Local Grid Graph Building, Input:(min:[x,y,z,res], max:[x,y,z,res])

1: $\mathbf{V} \leftarrow [x_0, y_0, z_0, \theta]$; $\mathbf{E} \leftarrow \emptyset$
2: **for** $i \leftarrow x_{min}$ to $x_{max}; step : x_{res}$ **do**
3: **for** $j \leftarrow y_{min}$ to $y_{max}; step : y_{res}$ **do**
4: **for** $k \leftarrow z_{min}$ to $z_{max}; step : z_{res}$ **do**
5: $v_{cur} \leftarrow [x, y, z]$ Current sample vertex
6: **if** $v_{cur} \in \mathbf{Q}_{free}$ **then** Determines if sample is free
7: $v_{cur} \leftarrow$ ProjectPoint(v_{cur}) Project sample to ground plane
8: $v_{near} \leftarrow$ NearestVertex(v_{cur}) Find nearest vertex
9: **if** inclination $<$ max inclination and $z_{diff} < z_{max}$ **then**
10: $V \leftarrow \mathbf{V} \cup \{v_{cur}\}$
11: $E \leftarrow \mathbf{E} \cup \{e_{cur}\}$
12: **for** [**do** Creates a dense edge set]$v \in nearestvertex$
13: **if** Admissible edge **then**
14: $E \leftarrow \mathbf{E} \cup \{e_{near}\}$
15: **end if**
16: **end for**
17: **end if**
18: **end if**
19: **end for**
20: **end for**
21: **end for**
22: return $\mathbf{V}, \mathbf{E}$

Algorithm 2: Neighbour Graph Merging, Input:($\mathbf{G}_{nei}$, $\mathbf{G}$)

1: **if** not Merged($\mathbf{G}_{nei}$) **then** {Identify correspondences}
2: **for** $v \in \mathbf{G}_{nei}$ **do**
3: $v_{near} \leftarrow$ nearest vertex(v,$\mathbf{G}$)
4: **for** $x \in v_{near}$ **do**
5: **if** edge obstacle free (v,x) **then**
6: add edge(v,x)
7: **end if**
8: **end for**
9: **end for**
10: **else** {Graph already merged}
11: **for** $v \in \mathbf{G}_{nei}$ **do**
12: **if** not exist v $\in \mathbf{G}$ **then** {add missing vertex}
13: add vertex(v,$\mathbf{G}$)
14: **end if**
15: **end for**
16: **for** $e \in \mathbf{E}_{nei}$ **do**
17: **if** not exists edge **then** {add missing edges}
18: add edge(e, $\mathbf{G}$)
19: **end if**
20: **end for**
21: **end if**
22: $\mathbf{V} \leftarrow [x_0, y_0, z_0, \theta]$; $\mathbf{E} \leftarrow \emptyset$
23: return $\mathbf{V}, \mathbf{E}$

additional admissible edges are added to the graph, forming a densely connected graph (lines 12-15). Fig. 3 illustrates the local graph and other planning graphs generated during a test run.

The computational complexity of generating the grid graph is $\mathcal{O}(N_x \times N_y \times N_z)$, where N_x, N_y, and N_z represent the dimensions of the grid in the x, y, and z directions, respectively. Collision checking is performed using the Voxblox [29], which has a complexity of $\mathcal{O}(1)$. Additionally, nearest neighbor vertex lookups are performed using a kd-tree with a complexity of $\mathcal{O}(\log(N))$.

The runtime of local exploration planning constitutes the most significant portion of the overall planning runtime, as it involves detailed sampling of viewpoints, ground plane validation, and exploration gain calculations. For grid-based sampling, runtime is proportional to the grid resolution and is upper-bounded by the total number of samples. In contrast, random sampling incurs additional overhead due to repeated validation of samples, especially when ground plane constraints are applied. Moreover, grid sampling offers a uniformly distributed and well-structured set of vertices, which enhances the quality of exploration paths. This uniformity reduces the likelihood of revisiting unexplored regions in subsequent iterations, thereby improving exploration efficiency and optimizing the use of available runtime.

B. Cooperative Global Exploration Planning

Global exploration planning allows the robot to maintain a coarse representation of the environment, reposition itself to a more promising frontier, or navigate back to previously explored areas. At the end of each local planning iteration, the robot selectively adds a subsampled set of nodes, including the best trajectory identified during local planning, to the global graph. This global graph serves as a navigation map, enabling the robot to efficiently move to specific locations in the observed environment, such as the home base or a high-priority frontier.

Each node in the global graph also encodes information from the robot's pose graph, which is updated when the localization system detects a loop closure. This ensures that the robot's position and environmental map remain consistent and accurate. When the robot repositions to a different area, it first identifies the frontier with the best information gain and distance in the

global graph. It then computes the shortest path using Dijkstra's algorithm to the frontier, allowing it to more efficiently explore.

To ensure collaboration between robots, global graphs are synchronized periodically by broadcasting graph updates (see Fig. 2(a). The prerequisite for merging the global graph between two robots is that the robots are in the communication range and have visited a common location in the environment. Each robot broadcasts its owned global graph nodes (nodes corresponding to physically visited locations). Robots receiving the global graph of other robots attempt to merge the graphs using Algorithm 2. As shown in the Algorithm 2, the graph merging process begins by identifying correspondences between nodes in the neighbor's graph and the robot's current graph (lines 2-9). If no prior merge has occurred, new correspondences are established (lines 5-7). Once the graphs are successfully merged, the nodes are synchronized to ensure consistency between the two robots in the team (lines 11-20). While this mechanism prevents conflicts, it does limit information propagation across multiple hops in the network. The global graph of a specific robot is only merged when two robots are within direct communication range (80 m).

The computational complexity of the graph merging process is polynomial, with a time complexity of $\mathcal{O}(n^2)$, where n represents the number of nodes in both the neighbor's and the current robot's graphs. As the graph's size increases, the merging process's complexity grows accordingly.

V. Results

Simulations: To assess the performance of the proposed Multi-robot Grid Graph (MGG) planner, we conducted a series of simulations alongside field experiments. For the simulations, we employed a ground robot model equipped with a lidar sensor in the Gazebo simulator. Two test environments from the gbplanner2 dataset [4] were selected: (1) Darpa Cave 1 and (2) Pittsburgh Mine. The robots equipped the system components in Fig. 2(b) during the tests; we used a motion primitive-based local planner [13] to handle traversability and dynamic obstacles during exploration path execution. Our method was compared with two state-of-the-art exploration planners: gbplanner2 [4] and TARE [5]. While our approach is built on top of the gbplanner2, key differences lie in local exploration planning and graph merging for distributed exploration. Conversely, TARE uses a viewpoint-based local and global exploration strategy.

The first tests were conducted in the Darpa Cave 1, a subterranean environment with rough terrain and steep inclines. We ran three repetitions of a one-hour mission for each planner: MGG, gbplanner2, and TARE. The robot's objective was to explore the cave and return to the starting point.

The results in Fig. 4 show the robot-generated map (A), total exploration volume (B), distance traveled (C), and algorithm runtime (D). The shaded regions in plots (B-D) represent the min and max across three runs, with the solid line indicating the mean. The TARE planner had the lowest exploration volume, as it became stuck on a steep slope due to conflicts between local and global navigation planners, highlighting the need to integrate traversability into the global planner (e.g., ground plane support and maximum incline limits, as in gbplanner2 and MGG). While gbplanner2 enabled robot movement, it caused tipping on steep inclines. The MGG planner, however, minimized these risks, enabling full exploration and safe return in all runs. Thus,

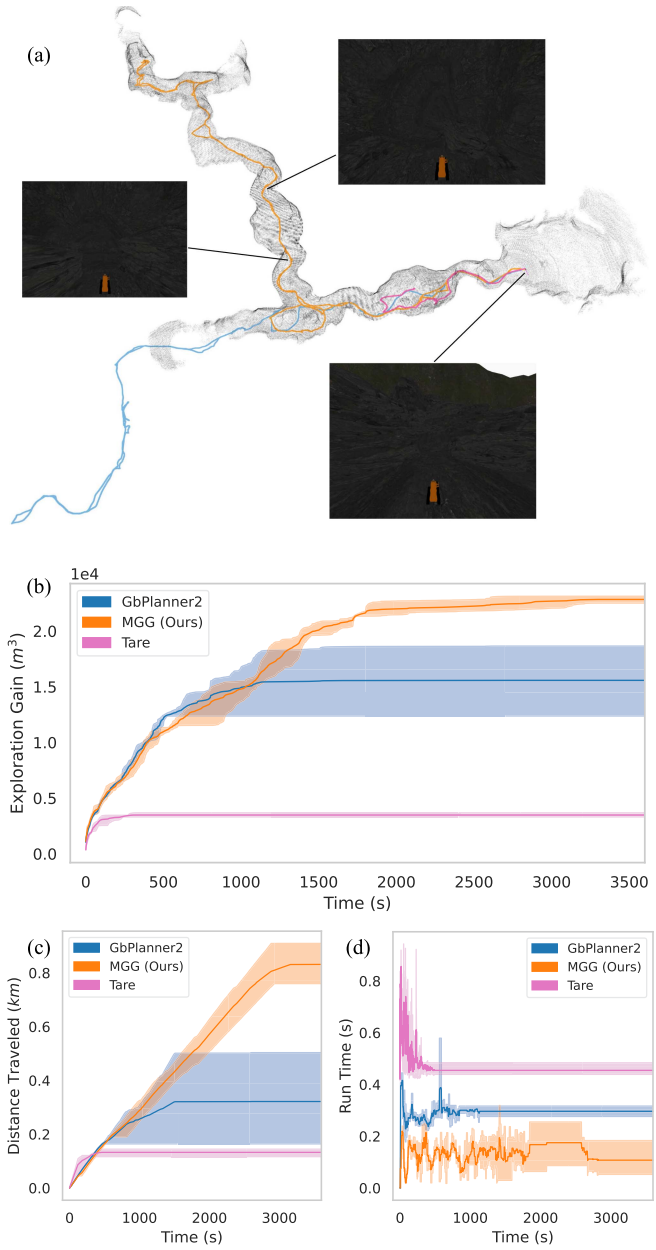

Fig. 4. One of the simulation test runs in the Darpa cave environment comparing the three planners: (a) The trajectory of the robot in one of the representative runs, (b) the amount of the environment explored during all runs, (c) the distance traveled, and (d) the runtime of the planner. The shaded regions in (b)–(d) indicate the min and max, and the solid line shows the mean value.

the MGG planner achieved the highest exploration volume and shortest runtime, combining the grid search approach of TARE and the inclination projections of gbplanner2.

The second test was conducted in an environment with relatively flat terrain, as shown in the map generated by the robot (Fig. 5(a)). Both gbplanner2 and MGGplanner explored a similar amount of the environment while the runtime of the MGGplanner was still lower than gbplanner2. The mean run time of 0.29 s, 0.14 s, and 0.48 s, respectively, was recorded with gbplanner2, MGG planner, and Tare planner. Tare planner still produced a lower exploration volume than the MGG planner.

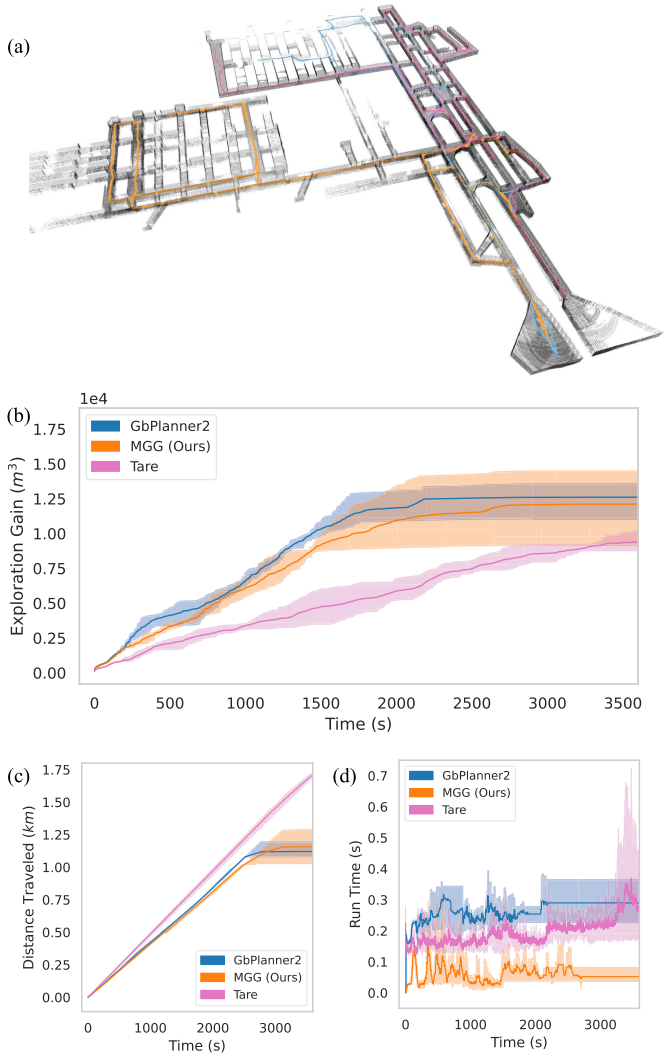

Fig. 5. Second simulation test in the Pitsburg mine environment with the flat ground: (a) The trajectory of the robot in one of the representative runs, (b) the volume of the environment explored, (c) distance traveled by the robot, and (d) the runtime of the planner. In (b)–(d), the shaded regions indicate min and max; the solid line shows the mean value.

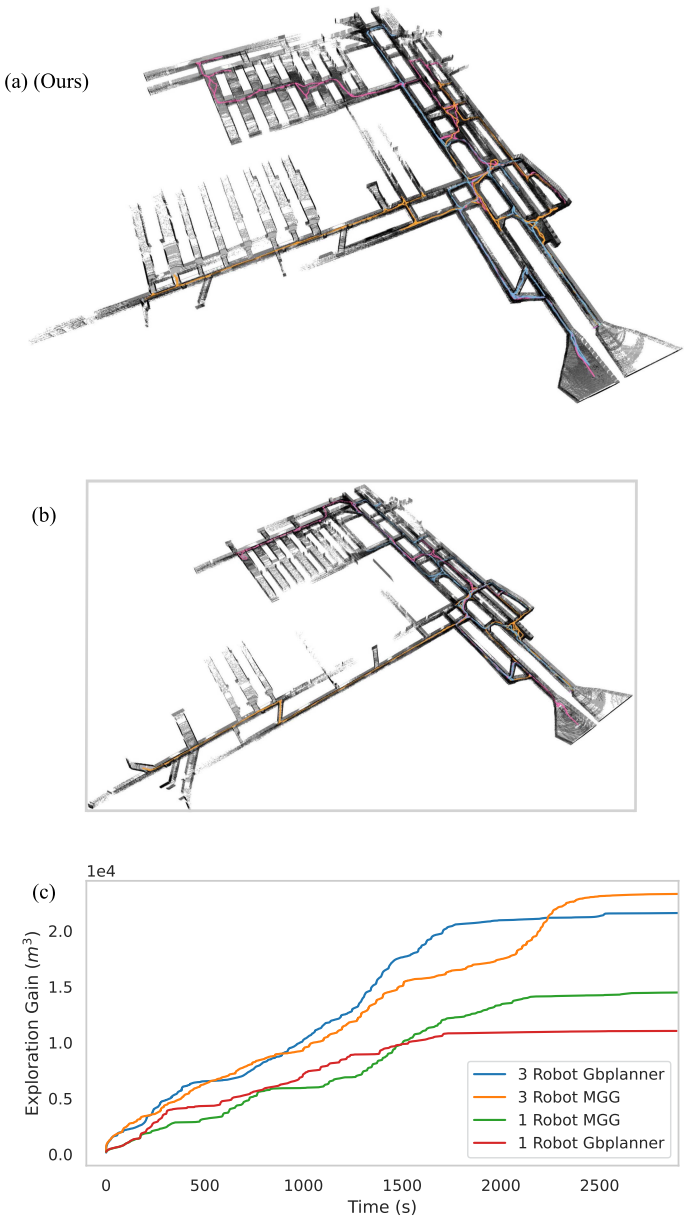

Fig. 6. Third simulation test in the Pitsburg mine with three robots: (a) Shows the trajectory of the robots in one of the runs using MGG planner, (b) shows the trajectory of the three robots using gbplanner2, (c) the exploration volume achieved using three robots and one robot.

Third tests were performed using three robots in the Pittsburg mine environment, as shown in Fig. 6. Using our approach, the robots were able to explore a larger part of the environment within the predefined time limit (1 h) used in the performance study. The robots explored 10% more of the environment with the MGG planner and 60% more when using three robots instead of one.

Physical robots: The physical robots used in the Mars Yard test included one AGILE-X Scout 2.0 and two Bunker robots. Each robot was equipped with an Ouster OS64 LiDAR, an Intel RealSense D435 camera, a Vectornav VN-200 IMU, and a custom dual-band (5 GHz and 2.4 GHz) radio for mesh network communication. The communication range of the robots is limited by their radios, which have an approximate range of 80 meters for exchanging graphs and coordinating with other robots. All robots were equipped with Nvidia AGX computers, running the necessary autonomy modules for perception, planning, and communication, as shown in Fig. 2(b). The robots had a maximum speed of 0.5 m/s during the experiments.

The tests were conducted in a 120m $\times$ 60m Mars-analog environment. Each robot was tasked with exploring the environment and returning to the starting location within 15 minutes. Three experimental runs were performed with both MGG planner and Gbplanner2 to compare and evaluate the performance. Fig. 7(c) shows the exploration gains of the three robots comparing the two planners. In contrast, Fig. 7(a)–(b) shows the trajectory of the robots in the point cloud map. Fig. 8 shows the size of graph messages broadcast and received by the robots over time.

In most runs, the MGG planner achieved higher exploration coverage than gbplanner2 by effectively distributing the robots. However, the exploration gain metric, based on occupied voxels in the combined point cloud map from all robots, did not fully capture this coverage due to the use of occupied voxels and the

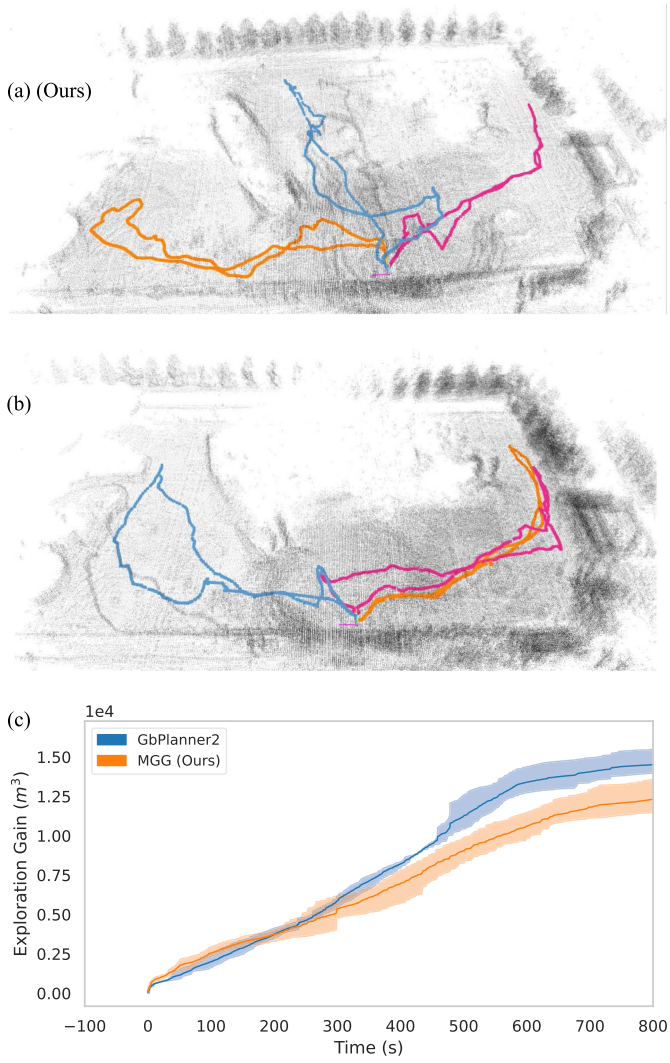

Fig. 7. Physical robot experiments with three robots at the Mars analog field experiment: (a) The trajectories of the three robots in the map generated using MGG planner, (b) when using gbplanner2, and (c) all three robots combined exploration gain obtained during the three experimental runs. The plot in (c) shows the min and max values (shaded region) and the mean (solid line). The exploration gain metric is based on occupied voxels in the combined map. As observed visually, (b) contains more occupied voxels than (a), reducing the effectiveness of the exploration gain performance metric in this test environment, where MGG explored more distinct ground but reported lower coverage in (c).

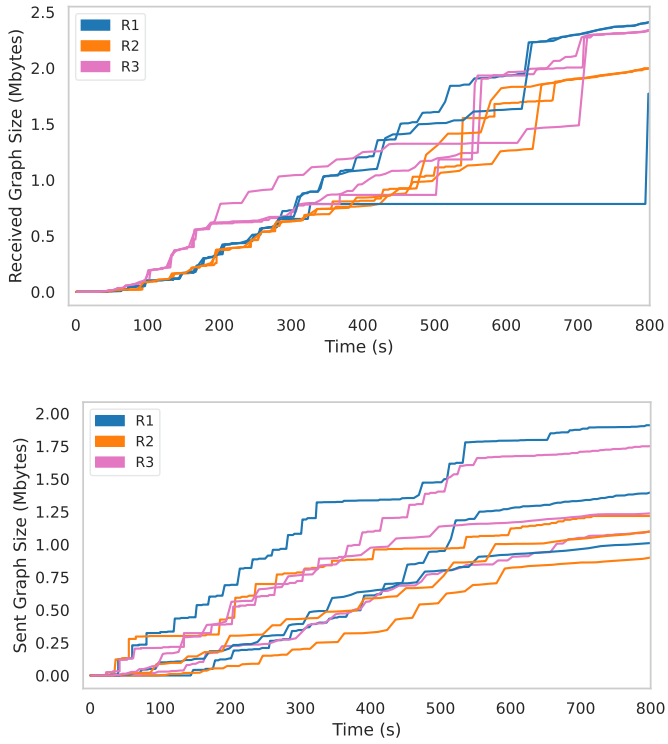

Fig. 8. Size of graph messages received by the robots (top) and sent by the robots (bottom) over time during the experimental evaluations. The line with similar colors indicates different runs, and as indicated in the plot, when a robot moves out of communication range (80 m hardware limited), it does not receive any messages.

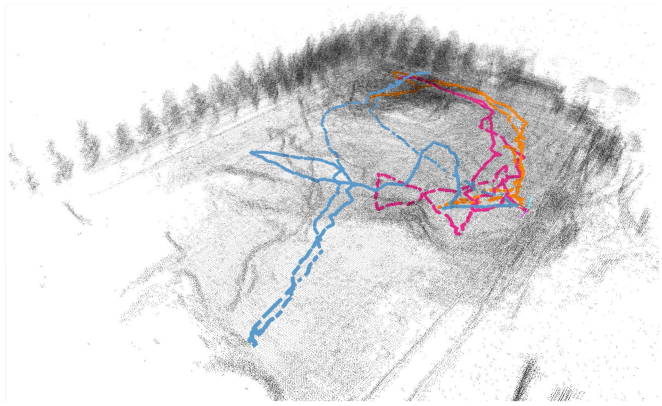

Fig. 9. One of the physical robot experiments with a 30-minute mission time using the MGG planner. The figure shows the trajectory of the three robots along with the combined map.

structure of the test environment. Fig. 7(a) illustrates the robots covering a larger area and distributing more evenly with the MGG planner compared to gbplanner2 in (B). During the global exploration planning step, the robots autonomously selected frontiers to explore based on the merged global graph. This process enabled each robot to remain aware of others' frontiers, prioritize unexplored areas, and respect regions already covered by other robots. This collaborative behavior ensured efficient distribution of effort, avoided overlap, and maximized exploration coverage. Additionally, a 30-minute mission with the MGG planner highlighted its performance in extended scenarios. As shown in Fig. 9, the robots covered a significantly larger portion of the environment during the longer mission, further validating the planner's efficiency and robustness.

VI. Conclusion

In this letter, we introduced a distributed multi-robot exploration planning method that addresses the challenges of complex, unconstrained environments with steep elevation changes. Our approach leverages a bifurcated planning strategy with local grid-based exploration for detailed exploration and a global sparse graph for high-level navigation and coordination. By exchanging global graph information among robots, we ensure synchronized exploration, leading to better overall coverage and efficiency. Through simulations in subterranean environments

and field tests in a Mars-analog setting, we demonstrated that our approach significantly reduces computational runtime by 50% compared to state-of-the-art methods, while maintaining superior exploration performance. The use of collaborative SLAM to correct global graph drift further enhanced the reliability of the system. Our results suggest that the proposed method is robust, scalable, and effective for autonomous exploration in challenging environments. Future work may extend the approach to larger teams of robots, improve robustness in more diverse terrains, investigate adaptive grid sampling, and explore additional real-world applications such as search and rescue or planetary exploration missions.

ACKNOWLEDGMENT

We thank the Canadian Space Agency (CSA) for access to the Mars yard and the MIST Lab members for their support during the field tests.

REFERENCES

[1] T. Sasaki, K. Otsu, R. Thakker, S. Haesaert, and A.-A. Agha-mohammadi, "Where to map? Iterative rover-copter path planning for mars exploration," *IEEE Robot. Automat. Lett.*, vol. 5, no. 2, pp. 2123–2130, Apr. 2020.

[2] T. Titus et al., "Science and technology requirements to explore caves in our solar system," *Bull. Amer. Astron. Soc.*, vol. 53, no. 4, pp. 1–7, 2021.

[3] P. F. Santana, J. Barata, and L. Correia, "Sustainable robots for humanitarian demining," *Int. J. Adv. Robotic Syst.*, vol. 4, no. 2, 2007, Art. no. 23.

[4] M. Kulkarni et al., "Autonomous teamed exploration of subterranean environments using legged and aerial robots," in *Proc. 2022 Int. Conf. Robot. Automat.*, 2022, pp. 3306–3313.

[5] C. Cao, H. Zhu, H. Choset, and J. Zhang, "TARE: A hierarchical framework for efficiently exploring complex 3D environments," *Robot., Sci. Syst.*, vol. 5, 2021, Art. no. 2.

[6] B. Morrell et al., "An addendum to nebula: Towards extending team CoSTAR's solution to larger scale environments," *IEEE Trans. Field Robot.*, vol. 1, pp. 476–526, 2024.

[7] C. Cao, H. Zhu, Z. Ren, H. Choset, and J. Zhang, "Representation granularity enables time-efficient autonomous exploration in large, complex worlds," *Sci. Robot.*, vol. 8, no. 80, 2023, Art. no. eadf0970.

[8] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *Proc. 1997 IEEE Int. Symp. Comput. Intell. Robot. Automat. Towards New Comput. Princ. Robot. Automat.*, 1997, pp. 146–151.

[9] M. Dharmadhikari et al., "Motion primitives-based path planning for fast and agile exploration using aerial robots," in *Proc. 2020 IEEE Int. Conf. Robot. Automat.*, 2020, pp. 179–185.

[10] W. Burgard, M. Moors, C. Stachniss, and F. E. Schneider, "Coordinated multi-robot exploration," *IEEE Trans. Robot.*, vol. 21, no. 3, pp. 376–386, Jun. 2005.

[11] S. Kim, S. Bhattacharya, R. Ghrist, and V. Kumar, "Topological exploration of unknown and partially known environments," in *Proc. 2013 IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2013, pp. 3851–3858.

[12] J. Frey et al., "Roadrunner–learning traversability estimation for autonomous off-road driving," *IEEE Trans. Field Robot.*, vol. 1, pp. 192–212, 2024, doi: 10.1109/TFR.2024.3464369.

[13] C. Cao et al., "Autonomous exploration development environment and the planning algorithms," in *Proc. 2022 Int. Conf. Robot. Automat.*, 2022, pp. 8921–8928.

[14] L. Heng, A. Gotovos, A. Krause, and M. Pollefeys, "Efficient visual exploration and coverage with a micro aerial vehicle in unknown environments," in *Proc. 2015 IEEE Int. Conf. Robot. Automat.*, 2015 , pp. 1071–1078.

[15] L. Freda and G. Oriolo, "Frontier-based probabilistic strategies for sensor-based exploration," in *Proc. 2005 IEEE Int. Conf. Robot. Automat.*, 2005, pp. 3881–3887.

[16] C. Dornhege and A. Kleiner, "A frontier-void-based approach for autonomous exploration in 3D," in *Proc. Adv. Robot.*, 2013, vol. 27, no. 6, pp. 459–468.

[17] J. Faigl and M. Kulich, "On benchmarking of frontier-based multi-robot exploration strategies," in *Proc. 2015 Eur. Conf. Mobile Robots*, 2015, pp. 1–8.

[18] W. Qiao, Z. Fang, and B. Si, "Sample-based frontier detection for autonomous robot exploration," in *Proc. 2018 IEEE Int. Conf. Robot. Biomimetics*, 2018, pp. 1165–1170.

[19] D. Silver, D. Ferguson, A. Morris, and S. Thayer, "Topological exploration of subterranean environments," *J. Field Robot.*, vol. 23, no. 6/7, pp. 395–415, 2006.

[20] H. Choset, S. Walker, K. Eiamsa-Ard, and J. Burdick, "Sensor-based exploration: Incremental construction of the hierarchical generalized voronoi graph," *Int. J. Robot. Res.*, vol. 19, no. 2, pp. 126–148, 2000.

[21] L. Fermin-Leon, J. Neira, and J. A. Castellanos, "TIGRE: Topological graph based robotic exploration," in *Proc. 2017 Eur. Conf. Mobile Robots*, 2017, pp. 1–6.

[22] F. Yang, D.-H. Lee, J. Keller, and S. Scherer, "Graph-based topological exploration planning in large-scale 3D environments," in *Proc. 2021 IEEE Int. Conf. Robot. Automat.*, 2021, pp. 12730–12736.

[23] C. Wang, H. Ma, W. Chen, L. Liu, and M. Q.-H. Meng, "Efficient autonomous exploration with incrementally built topological map in 3-D environments," *IEEE Trans. Instrum. Meas.*, vol. 69, no. 12, pp. 9853–9865, Dec. 2020.

[24] J. Bayer and J. Faigl, "Decentralized topological mapping for multi-robot autonomous exploration under low-bandwidth communication," in *Proc. 2021 Eur. Conf. Mobile Robots*, 2021, pp. 1–7.

[25] A. Bircher, M. Kamel, K. Alexis, H. Oleynikova, and R. Siegwart, "Receding horizon 'next-best-view' planner for 3D exploration," in *Proc. 2016 IEEE Int. Conf. Robot. Automat.*, 2016, pp. 1462–1468.

[26] C. Witting, M. Fehr, R. Bähnemann, H. Oleynikova, and R. Siegwart, "History-aware autonomous exploration in confined environments using MAVs," in *Proc. 2018 IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 1–9.

[27] P.-Y. Lajoie and G. Beltrame, "Swarm-SLAM: Sparse decentralized collaborative simultaneous localization and mapping framework for multi-robot systems," *IEEE Robot. Automat. Lett.*, vol. 9, no. 1, pp. 475–482, Jan. 2024.

[28] T. Dang, F. Mascarich, S. Khattak, C. Papachristos, and K. Alexis, "Graph-based path planning for autonomous robotic exploration in subterranean environments," in *Proc. 2019 IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2019, pp. 3105–3112.

[29] H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwart, and J. Nieto, "Voxblox: Incremental 3D euclidean signed distance fields for on-board MAV planning," in *Proc. 2017 IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 1366–1373.