FLARE: Fast Large-Scale Autonomous Exploration Guided by Unknown Regions

Xinyang Liu, Min Lin, Shengbo Li, Gang Xu, Zhifang Wang, Huifeng Wu, *Member, IEEE*, and Yong Liu

***Abstract*—Autonomous exploration is a critical foundation for uncrewed aerial vehicles (UAV) applications such as search and rescue. However, existing methods typically focus only on known spaces or frontiers without considering unknown regions or providing further guidance for the global path, which results in low exploration efficiency. This letter proposes FLARE, which enables Fast UAV exploration in LARge-scale and complex unknown Environments. The incremental unknown region partitioning method partitions the unexplored space into multiple unknown regions in real-time by integrating known information with the sensor perception range. Building on this, the hierarchical planner first computes a global path that encompasses all unknown regions and then generates safe and feasible local trajectories for the UAV. We evaluate the performance of FLARE through extensive simulations and real-world experiments. The results show that, compared to existing state-of-the-art algorithms, FLARE significantly improves exploration efficiency, reducing exploration time by 16.8% to 27.9% and flight distance by 15.8% to 25.5%. The source code of FLARE will be released to benefit the community.**

***Index Terms*—Aerial systems: Applications, aerial systems: Perception and autonomy, motion and path planning.**

I. Introduction

AUTONOMOUS exploration is the process where a robot navigates and traverses an unknown environment to construct a map of the target area. It is a key component of various robotic applications, such as underground mining exploration [1], forest resource surveys [2], search and rescue operations [3]. Compared to other robotic platforms, UAVs' flexibility and agility enable them to better meet the demands of exploration tasks. However, due to the limited endurance of UAVs, it is essential to develop efficient exploration planners capable of rapidly covering the accessible space.

In recent years, researchers have developed various autonomous exploration methods. However, most methods focus only on known regions or frontier regions. They guide exploration using frontiers or sampling points without fully utilizing the information from unknown regions. When encountering large-scale and complex environments, such an information structure is inadequate to capture more comprehensive and deeper information of the entire unknown area.

Compared with global guidance that only considers the frontier areas, incorporating the exploration target (unknown regions) provides more information and helps reduce unnecessary back-and-forth movements. Many researchers have acknowledged this, but currently, there is no comprehensive framework for utilizing unknown regions. Specifically, the method of traversing all unknown voxels is impractical. Therefore, an appropriate partitioning method is essential. Fixed-grid partitioning [4], [5] makes it difficult to exploit the geometric characteristics of regions; therefore, most algorithms [6], [7], [8] partition regions according to connectivity (voxel adjacency) instead. However, not all adjacent voxels should be assigned to the same unknown region. To address this, a convexity-based region segmentation was proposed [6], but it tends to produce many unnecessary splits. To support incremental partitioning, some methods [8], [9] rely on grid-by-grid updates, which likewise introduce redundant segmentation. A sound partitioning strategy is foundational for subsequent processing. Constrained by the above practices, existing approaches struggle to utilize unknown regions fully for global planning and can only assist with frontier selection [8], [9] or choose the largest unknown region [7]. It is also difficult to effectively mitigate the impact of unreachable unknown regions.

To effectively address the aforementioned issues, we propose the **FLARE** framework. This framework leverages information from unknown regions to guide the UAV in **F**ast **LAR**ge-scale autonomous **E**xploration. We introduce the concept of coverability and, based on it, realize a truly incremental region partitioning strategy. Specifically, we construct the Unknown Region Information Structure (UIS) to characterize the features of unknown regions. Before exploration begins, we perform a lightweight initial partitioning for fast division. At each update, we cluster only voxels of the changed unknown regions according to voxel connectivity. We then perform coverability segmentation so that each viewpoint precisely represents its corresponding region. Without violating coverability, we merge regions, allowing non-adjacent voxels to be assigned to the same region. This merging substantially enlarges the coverage of a single viewpoint and prevents redundant overlapping coverage

Received 3 June 2025; accepted 27 September 2025. Date of publication 14 October 2025; date of current version 20 October 2025. This article was recommended for publication by Associate Editor S. Zhao and Editor A. Bera upon evaluation of the reviewers' comments. This work was supported by the National Natural Science Foundation of China under Grant 62525309. *(Corresponding authors: Yong Liu; Gang Xu.)*

Xinyang Liu, Min Lin, Shengbo Li, Gang Xu, Zhifang Wang, and Yong Liu are with the Institute of Cyber-Systems and Control, Zhejiang University, Hangzhou 310027, China (e-mail: wuuya@zju.edu.cn; yongliu@iipc.zju.edu.cn).

Huifeng Wu is with the Institute of Intelligent and Software Technology, Hangzhou Dianzi University, Hanghzou 310018, China.

This article has supplementary downloadable material available at https://doi.org/10.1109/LRA.2025.3620618, provided by the authors.

Digital Object Identifier 10.1109/LRA.2025.3620618

by multiple viewpoints. Based on this partitioning, a hierarchical planner first determines the exploration order over unknown regions, computes a global path over all region viewpoints while avoiding interference from unreachable regions, and then generates a safe and dynamically feasible trajectory, enabling fast and efficient exploration.

FLARE leverages information from unknown regions more comprehensively than prior approaches and avoids reliance on frontiers or sampled points. We evaluate FLARE through comprehensive tests in simulation and real-world environments. In simulations, we compare FLARE with five state-of-the-art methods [8], [9], [10], [11], [12] across diverse scenarios. Results show that FLARE achieves significantly higher exploration efficiency than all baselines, reducing exploration time by 16.8-27.9% and flight distance by 15.8-25.5% relative to the best-performing baselines. Real-world experiments further confirm its effectiveness and practicality. The main contributions of this letter are summarized as follows:

1) An autonomous exploration framework that utilizes unknown regions for guidance, without extra frontiers or sampling points.
2) An incremental unknown region partitioning method enabling real-time and efficient division of unknown regions in complex environments.
3) A hierarchical planning approach that generates efficient global coverage paths using unknown regions, coupled with local trajectory generation.

II. Related Work

Autonomous exploration has been widely studied. Frontier-based strategies are among the most classical approaches. [13] first defined the frontier as the boundary between known and unknown space. [10], [14], [15] employed a hierarchical planner to determine a global path by solving a Traveling Salesman Problem (TSP) to offer a global coverage path. Building on this, [16] introduced global paths at the frontier cluster level. [17] used frontier characteristics to evaluate the likelihood of repetitive back-and-forth movements. [11] categorized frontiers and switched the exploration mode based on the surrounding frontier types. [18] treats exploring cubic subspaces as a form of frontier, and [19] further extends this idea to multi-UAV exploration. To avoid the memory-intensive occupancy map, [12] directly extracts surface frontiers from the point cloud map. Nevertheless, in essence, these methods still focus solely on the boundary.

Sampling-based methods are another classical strategy. Based on the Next Best View (NBV), these methods typically use Rapidly-exploring Random Trees (RRT) for sampling and select points by evaluating information gain [20], [21]. Building on NBV, [22] extended it to a Next-Best-Trajectory formulation that considers information gain along an entire trajectory. [23] constructed a history graph to use historical nodes to escape local optima. However, generating numerous invalid viewpoints is an inherent problem of sampling-based methods, resulting in unnecessary computational overhead. Therefore, some studies [24] combine frontier-based exploration with sampling to optimize sampling.

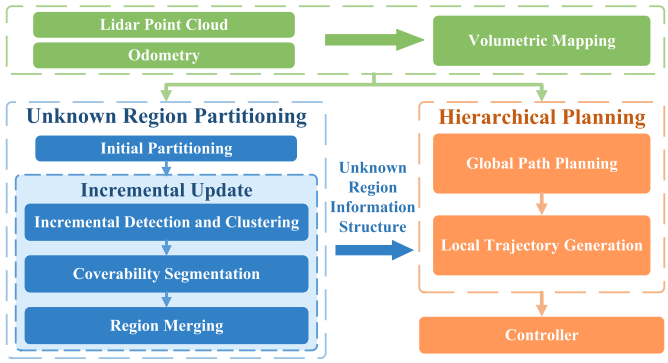


Fig. 1. System overview of FLARE.

Given the fundamental shortcomings of the two approaches above, an increasing number of scholars have recognized the importance of unknown regions. However, fixed-grid partitioning methods [4], [5] fail to capture the geometric characteristics of regions. To obtain a more reasonable partitioning, most methods [8], [9] adopt connectivity as the criterion; yet, connectivity alone is insufficient, [6] further imposes convexity-based processing. However, this still cannot fundamentally overcome the limitations of connectivity. For incremental updates, [9] treats all unknown voxels within each grid as a single unknown region. Similar to [9], [8] updates the region partition grid by grid and disallows assigning voxels from different grids to the same unknown region. Taken together, these constraints markedly reduce the informational value of the resulting regions. Consequently, most methods can only greedily choose the largest region or assist in frontier selection.

Most current algorithms do not fully utilize information from unknown regions. In contrast, our algorithm leverages information from the explored environment to partition unexplored regions, avoid redundant coverage from repeated viewpoints, and provide efficient global guidance to the UAV centered on the unknown regions. It then generates safe, feasible trajectories to complete the exploration.

III. Problem Definition and System Overview

This letter investigates using a UAV equipped with a LiDAR to explore a large-scale, unknown, bounded three-dimensional space and construct the corresponding map.

As shown in Fig. 1, the proposed system consists of two key components: unknown regions partitioning (Section IV) and the hierarchical planner (Section V). The system operates upon a probabilistic occupancy voxel map. Based on sensor information, the system continuously updates unknown voxels into free or occupied voxels. At the start of exploration, the system uniformly divides the entire environment to be explored into several unknown regions using a grid. During each update, the system detects changes in unknown regions and performs repartitioning through rapid clustering to update the unknown regions incrementally. For each unknown region, the system performs coverability segmentation and further merges unknown regions based on the LiDAR's perception range. After partitioning the unknown regions, the hierarchical planner calculates a global

path encompassing all viewpoints of the unknown regions and then generates local trajectories for the UAV to reach the target viewpoint.

IV. Unknown Region Partitioning

A. Unknown Region Information Structure

To maintain information of unknown regions and enable incremental updates during exploration, we propose the UIS.

We define an unknown region R_i as a set of unknown voxels. For R_i, a UIS UI_i is computed. This structure includes the unknown cells C_i belonging to the region and their average position $\mathbf{p}_{\text{avg},i}$. A cell refers to a voxel. In addition, we downsample C_i to obtain $C_{\text{down},i}$, thereby accelerating subsequent computations. An axis-aligned bounding box (AABB) B_i is computed for R_i to accelerate the detection of changes in unknown regions. To facilitate exploration planning (Section V), we generate a viewpoint $\mathbf{vp}_i$ for R_i.

B. Initial Partitioning

Before the exploration begins, the entire environment is unknown, with no additional information available. To accelerate the partitioning, we initialize by employing grids of fixed size and shape, with the center of each grid serving as the viewpoint. The grid size is determined by the sensor range to ensure coverability. The coverability of the region, $Coverable_i$, is determined and specifically quantified by:

$$Coverable_i = \begin{cases} 1, & \frac{\text{size}(C_{\text{cover},i}(\mathbf{vp}_i))}{\text{size}(C_{\text{down},i})} > \theta_{\text{cover}} \\ 0, & \text{else} \end{cases} \tag{1}$$

where $C_{\text{cover},i}(\mathbf{vp}_i)$ denotes the voxels observable by the sensor at the viewpoint $\mathbf{vp}_i$, and θ_{cover} is a predefined coverage threshold. $\text{size}(\cdot)$ denotes the number of elements in a set. A voxel is observable if it complies with the sensor model and is not occluded by occupied voxels. We evaluate only the voxels in $C_{\text{down},i}$ to reduce the computational cost.

Unlike other methods using grid partitioning, the grids place no constraints on subsequent updates. These unknown regions will be repartitioned in subsequent exploration.

C. Incremental Detection and Clustering

Based on the sensor range, we define an AABB B_{update} for the space that requires updating. Every time new sensor data is received, we initially identify the regions to be updated by checking whether B_i intersects with B_{update}. For those unknown regions whose B_i intersects with B_{update}, we remove the regions whose unknown voxels have undergone sufficient changes and add their unknown voxels C_i to the set of voxels that need to be reclustered, C_{update}. Unlike frontier voxels, the unknown voxels at the current time can only be in an unknown state in the previous time, and free or occupied voxels cannot transition into unknown voxels. Therefore, only C_{update} requires reclustering, and traversing all related AABBs is unnecessary.

For C_{update}, we first cluster adjacent unknown voxels via a region growing algorithm to form new unknown regions, then discard too small clusters. Unknown voxels adjacent to occupied voxels are excluded during clustering to prevent noise or obstacles from being misidentified. For each newly generated unknown region R_i, $\mathbf{p}_{\text{avg},i}$ is set as $\mathbf{vp}_i$, as shown in Fig. 2(a). However, this clustering method does not guarantee convexity, meaning that $\mathbf{p}_{\text{avg},i}$ may fall into obstacles. We adjust $\mathbf{vp}_i$ to the unknown voxel in C_i nearest to $\mathbf{p}_{\text{avg},i}$. This incremental updating method eliminates the reliance on fixed-size grids, enhancing adaptability in complex environments.

D. Coverability Segmentation

Because it cannot be guaranteed that the sensor achieves complete coverage at the viewpoint, the newly generated unknown regions remain unsuitable as sub-goals for exploration. Therefore, we perform coverability segmentation on the unknown regions that cannot be fully covered. First, we determine whether segmentation is necessary based on their size. For cases requiring segmentation, we apply the Principal Component Analysis to obtain the first principal axis and divide each region into two uniform subregions along it, thereby achieving adaptive segmentation according to each region's voxel distribution. Subsequently, we evaluate whether further segmentation is necessary for the newly generated sub-regions. This segmentation process is applied recursively until no region meets the segmentation criteria. Unknown regions after segmentation are shown in Fig. 2(b).

After segmentation, we obtain $\mathbf{R}_{\text{cover}}$ and $\mathbf{R}_{\text{uncover}}$. $\mathbf{R}_{\text{cover}}$ is the set of unknown regions that are coverable. $\mathbf{R}_{\text{uncover}}$ is the set of unknown regions $R_{\text{uncover},i}$ that cannot be fully covered and fail the size requirements for segmentation. The viewpoint of $R_{\text{uncover},i}$ cannot be the target for exploring the region. However, directly discarding $R_{\text{uncover},i}$ would affect the integrity of the exploration. Therefore, we retain $\mathbf{R}_{\text{uncover}}$ and consider it in subsequent incremental updates.

E. Region Merging

We introduce region merging to ensure all unknown regions are coverable and all unknown voxels are covered with as few viewpoints as possible. Unlike [7], [8], which require voxel adjacency as a necessary condition when partitioning regions, our method is based on coverability: any set of unknown regions that can be covered from the same viewpoint is merged into a region. Our method fully exploits the sensor perception range and significantly reduces the number of unknown regions, thereby lowering computational cost in subsequent global planning. An example is shown in Fig. 2(c). To improve the integrity of the exploration, we consider $\mathbf{R}_{\text{cover}}$ and $\mathbf{R}_{\text{uncover}}$ during the merging process.

First, we use Euclidean distance as a preliminary filter for (R_i, R_j). Then, we compute the new viewpoint $\mathbf{vp}_{\text{new}}$ by:

$$\mathbf{vp}_{\text{new}} = \frac{(\text{size}(C_i) \cdot \mathbf{vp}_i + \text{size}(C_j) \cdot \mathbf{vp}_j)}{\text{size}(C_i) + \text{size}(C_j)}. \tag{2}$$

Then, we determine whether the new viewpoint can simultaneously cover the voxels of both unknown regions:

$$Coverable_{new} = Coverable_i(\mathbf{vp}_{\text{new}}) \cdot Coverable_j(\mathbf{vp}_{\text{new}}). \tag{3}$$

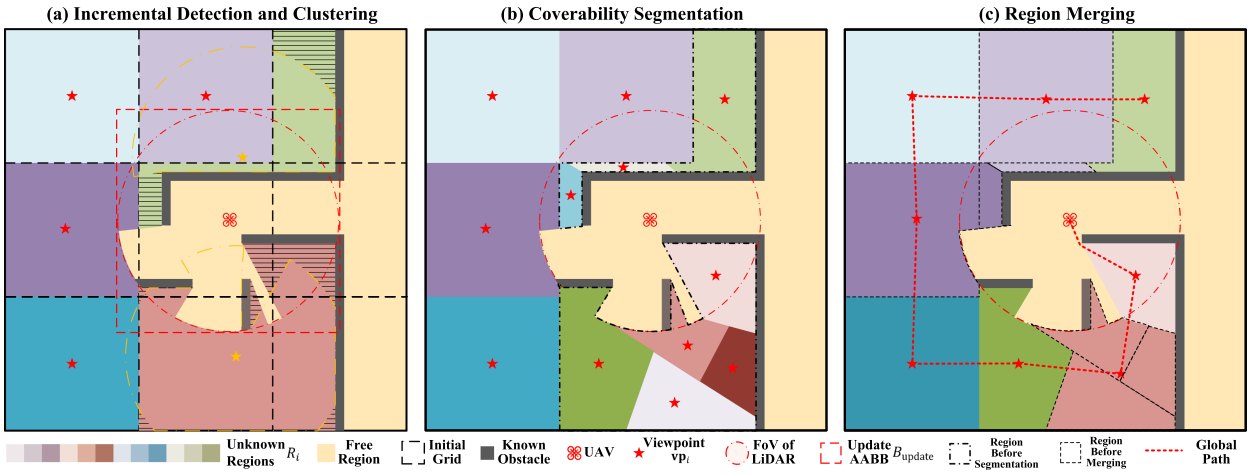


Fig. 2. Illustration of unknown region partitioning. The figure uses a 2D example for clarity, but the method applies to 3D scenarios. Viewpoints for coverable regions are marked with red pentagrams, whereas those for uncoverable regions are marked with golden pentagrams. For uncoverable regions, we outline the sensor field of view achievable with a golden dashed curve when the UAV is positioned at that viewpoint under the current known environment. We fill the uncovered parts with horizontal stripes, making the cause of uncoverability immediately evident.

If the coverage condition is satisfied, we merge the two unknown regions into a new unknown region and set $\mathbf{vp}_{\text{new}}$ as its viewpoint. This process is also applied recursively. In this manner, the entire exploration task is decomposed into a series of well-defined sub-goals.

V. Hierarchical Planning

A. Global Path Planning

To find the most efficient global path that covers all unknown regions, we model the problem as an Asymmetric TSP (ATSP) and exclude any unreachable unknown regions. It is important to note that after the region partitioning, the hierarchical planning only needs to consider $\mathbf{R}_{\text{cover}}$, whose viewpoints represent the positions to explore those unknown regions. Therefore, we establish a cost matrix $\mathbf{M}_{\text{tsp}}$:

$$\mathbf{M}_{\text{tsp}} = \begin{bmatrix} \mathbf{0} & \mathbf{C}_{\text{begin}} \\ \mathbf{C}_{\text{end}} & \mathbf{C}_{\text{reg}} \end{bmatrix} \in \mathbb{R}^{(N_{\text{reg}}+1)\times(N_{\text{reg}}+1)}, \tag{4}$$

where $N_{\text{reg}} = \text{size}(\mathbf{R}_{\text{cover}})$. $\mathbf{C}_{\text{begin}} \in \mathbb{R}^{1\times N_{\text{reg}}}$ is the cost matrix between the UAV's current position $\mathbf{p}_{\text{now}}$ and the viewpoints of the unknown regions:

$$\mathbf{C}_{\text{begin}}(k) = \text{path}(\mathbf{p}_{\text{now}}, \mathbf{vp}_k) + \alpha_{\text{con}} \cdot c_{\text{con}}(\mathbf{vp}_k), k \in \mathbf{K}, \tag{5}$$

$$\mathbf{K} = \{0, 1, \ldots, N_{\text{reg}} - 1\}, \tag{6}$$

where $\text{path}(\mathbf{x}_1, \mathbf{x}_2)$ denotes the length of the collision-free path computed by the A* search algorithm from $\mathbf{x}_1$ to $\mathbf{x}_2$. In traditional pathfinding algorithms, only known free space is considered, while unknown space is treated as if it were occupied, thus rendering it impassable. Given the possibility that feasible paths may lie in unknown space, our pathfinding algorithm also considers unknown space as passable. However, because unknown space introduces uncertainty, we apply a penalty factor α_{penal} to any portion of the path through unknown space when calculating the path length. This parameter indicates the uncertainty of the path. α_{con} is the consistency parameter, and $c_{\text{con}}(\mathbf{vp}_k)$ represents the additional cost introduced to maintain motion consistency in the UAV's current velocity $\mathbf{v}_{\text{now}}$:

$$c_{\text{con}}(\mathbf{vp}_k) = \|\mathbf{v}_{\text{now}}\| \cdot \cos^{-1} \frac{\mathbf{v}_{\text{now}} \cdot (\mathbf{vp}_k - \mathbf{p}_{\text{now}})}{\|\mathbf{v}_{\text{now}}\| \, \|\mathbf{vp}_k - \mathbf{p}_{\text{now}}\|}, k \in \mathbf{K}. \tag{7}$$

By computing c_{con}, the magnitude and direction of $\mathbf{v}_{\text{now}}$ are considered to apply appropriate penalties for significant directional changes when needed, while avoiding overemphasis on direction changes at lower speeds. Furthermore, this method prevents the UAV from repeatedly switching between multiple optimal paths with similar costs.

To prevent unknown regions within obstacles from affecting global planning, we classify unknown regions that the A* pathfinding algorithm cannot navigate from the UAV's current position as unreachable and subsequently remove them. Since unknown voxels are treated as passable during pathfinding, a path cannot be found in the current state and is even less likely to exist in the future. The prior region partitioning ensures that each unknown region is coverable, so this deletion method does not remove any redundant voxels. In fact, this pathfinding and deletion approach can improve the completeness of exploration. If there are still unexplored parts on the surface of obstacles, paths passing through the obstacles will appear, guiding the UAV to approach these unexplored voxels. The specific process is shown in Fig. 3.

$\mathbf{C}_{\text{reg}} \in \mathbb{R}^{N_{\text{reg}}\times N_{\text{reg}}}$ is the cost matrix between viewpoints across different unknown regions, which is computed as:

$$\mathbf{C}_{\text{reg}}(k_1, k_2) = \mathbf{C}_{\text{reg}}(k_2, k_1) = \text{path}\left(\mathbf{vp}_{k_1}, \mathbf{vp}_{k_2}\right), k_1, k_2 \in \mathbf{K}. \tag{8}$$

Since solving the standard ATSP yields a closed-loop path, we do not require the UAV to return to the endpoint. Therefore, we add the matrix block $\mathbf{C}_{\text{end}} \in \mathbb{R}^{N_{\text{reg}}\times 1}$:

$$\mathbf{C}_{\text{end}}(k, 0) = -1000. \tag{9}$$

So the start and end points of the closed-loop path are constrained to $\mathbf{p}_{\text{now}}$, and the cost of the last viewpoint returning to $\mathbf{p}_{\text{now}}$ does

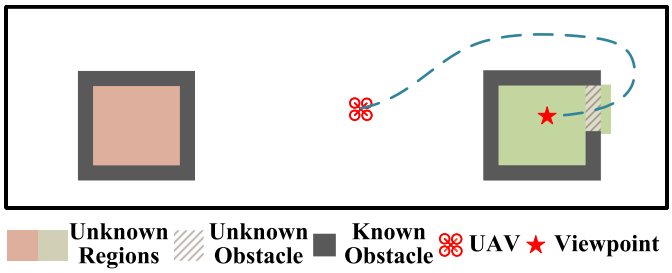


Fig. 3. Illustration of the process for removing unreachable unknown regions. The two unknown regions shown in the figure are inside the obstacle. The red region will be removed because no path can be found, whereas the green region will lead the UAV to explore the entire surface of the obstacle.

not affect the open-loop path. Finally, we remove the edge that returns to $\mathbf{p}_{\text{now}}$, thereby obtaining the desired global path. The final global path is shown as the blue dashed line in Fig. 2(c).

B. Local Trajectory Generation

Then, we generate a continuous-time trajectory toward the first target viewpoint. Our trajectory generation is based on [10], which leverages the quadrotor's differential flatness property to generate smooth, safe, and dynamically feasible B-spline trajectories. This method generates a minimum-time trajectory based on the collision-free path determined by the pathfinding algorithm. In contrast to [10], we assume that the unknown space is traversable in the pathfinding algorithm to leverage its potential fully. Moreover, local planning is decoupled from sub-goal selection, enabling high-frequency updates of the local trajectory. The system performs continuous collision checking and immediately triggers local replanning upon detecting any collision risk, ensuring safety even in the presence of dynamic obstacles. Furthermore, in the Euclidean signed distance field (ESDF) map, we initialize unknown voxels as near obstacles to prompt the UAV to reduce speed when approaching these areas.

VI. Simulations and Experiments

A. Implementation Details

We implement the FLARE system in C++ and the Robot Operating System (ROS). We use a quadrotor with a Livox Mid360 LiDAR as the exploration platform in simulations and real-world environments. The mapping module employs a voxel-based framework from [9] to construct an occupancy grid representation of the space and incrementally maintain an ESDF map for trajectory planning. The formulated ATSP is solved using the Lin-Kernighan-Helsgaun heuristic solver [25]. Local trajectory generation is carried out using the general nonlinear optimization solver NLopt. We set $\alpha_{\text{penal}} = 1.5$, $\theta_{\text{cover}} = 0.85$ in (1), and $\alpha_{\text{con}} = 1$ in (5). We adopt a uniform map resolution of $0.25\,\text{m}$ and a resolution of $0.5\,\text{m}$ for the A* algorithm.

B. Simulations

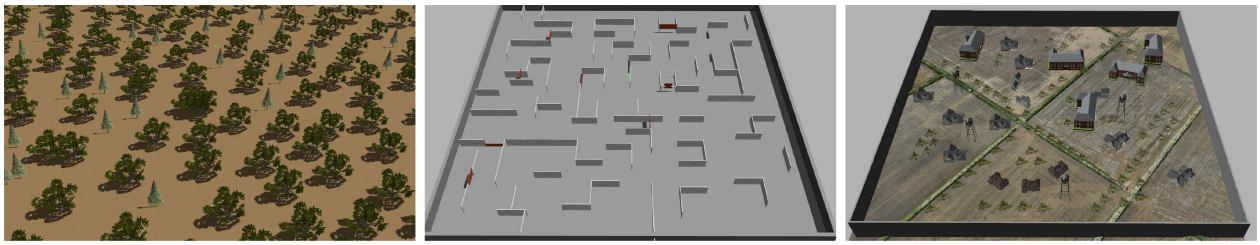
All simulations are conducted on a computer with an Intel Core i9-14900KF and 64 GB of RAM. We set up a simulation environment in Gazebo to emulate real-world scenarios. To validate the generality of the proposed method in different scenarios, we test it in three large-scale, complex environments, as shown in Fig. 4, including a forest, a maze, and a village. We comprehensively evaluate FLARE and five state-of-the-art baselines across all environments. All baselines are implemented based on open-source code, with parameters adaptively adjusted according to sensor ranges. Specifically, we compare against FUEL [10], RACER [9], FAME [11], FALCON [8], and EPIC [12].

In the tests, we set the maximum speed for all methods to $5.0\,\text{m/s}$ and the maximum acceleration to $2.0\,\text{m/s}^2$. Additionally, the LiDAR sensor's effective range is limited to a radius of $30\,\text{m}$. Each method is executed five times in each scenario, and the statistical results are summarized in Table I. In the village environment, we conduct tests in $100\,\text{m} \times 100\,\text{m}$ and $200\,\text{m} \times 200\,\text{m}$ areas. In Table I, exploration time is defined as the duration each algorithm considers its exploration complete. Coverage is the ratio of the current exploration volume to the maximum exploration volume achieved by all algorithms in the environment.

According to Table I, the proposed method achieves shorter exploration times and lower path costs in all test scenarios, demonstrating superior performance and broad applicability across various environmental conditions. The average exploration time is reduced by an average of 22.1% compared to the second-best method in each scenario, particularly in the village environment, where it is reduced by 27.9%. The flight distance is reduced by an average of 21.6% compared to the baseline for the shortest path in each scenario, with the improvement reaching 25.5% in the forest environment. The standard deviation of exploration time is the smallest, further demonstrating the stability of our algorithm. The average coverage exceeds 99%, indicating that the proposed algorithm can achieve full coverage. The optimal coverage rate is achieved in the maze environment with flat walls, further confirming the completeness of the proposed algorithm. In other environments, the slight differences primarily arise from the difficulty in thoroughly scanning the inner branches and leaves of trees. Still, they do not impact the completeness of the exploration.

Fig. 5 shows the exploration progress of each method in different environments and marks the time when the coverage reaches 99%. Fig. 6 shows the trajectories and the constructed maps of different algorithms in the maze and village scenarios. It is evident from the figures that the proposed method outperforms the state-of-the-art baselines in all tests. The proposed method shows nearly linear growth in coverage with no plateau phase. Other algorithms experience prolonged stagnation in coverage growth during exploration, primarily due to missed areas, which cause the UAV to travel long distances in already explored space. As shown in Fig. 6, FLARE's trajectories rarely intersect during exploration, avoiding passing through the same area multiple times.

From an algorithmic perspective, FUEL, FAME, and EPIC provide only frontier-based global guidance and disregard information from unexplored regions. Moreover, they cannot accurately predict the next frontier, so the global plan gradually drifts from optimality. For example, at multiple intersections in Maze, frontiers appear in all directions, leading FUEL to fail



(a) Forest (b) Maze (c) Village

Fig. 4. Simulation test scenarios. (a) Forest ($100 \times 100 \times 7\,\mathrm{m}^3$), (b) Maze ($100 \times 100 \times 2.75\,\mathrm{m}^3$), (c) Village ($200 \times 200 \times 24\,\mathrm{m}^3$).

TABLE I
EXPLORATION STATISTICS

Scene	Method	Exploration Time (s)				Flight Distance (m)				Coverage (%)				Avg Velocity (m/s)			
		Avg	Std	Max	Min	Avg	Std	Max	Min	Avg	Std	Max	Min	Avg	Std	Max	Min
Forest 100 × 100	FUEL [10]	266.3	12.76	289.9	251.0	510.1	26.25	542.3	469.2	99.82	0.15	99.92	99.53	1.92	0.09	2.02	1.78
	RACER [9]	286.2	19.06	320.1	264.2	707.0	80.35	847.8	618.4	99.76	0.07	99.86	99.69	2.47	0.25	2.95	2.25
	FAME [11]	237.7	17.91	266.6	217.8	604.2	62.25	685.5	529.7	99.92	0.05	100.0	99.86	2.54	0.10	2.68	2.36
	FALCON [8]	243.7	27.68	295.6	222.0	582.4	60.41	677.4	508.6	98.99	0.88	99.80	97.38	2.39	0.13	2.61	2.24
	EPIC [12]	398.3	56.13	464.4	298.8	1201.5	159.21	1406.3	937.4	**99.99**	0.00	99.99	99.98	**3.02**	0.06	3.14	2.94
	Proposed	**189.5**	6.29	200.5	181.3	**406.4**	19.17	436.2	382.0	99.16	0.29	99.57	98.66	2.14	0.05	2.19	2.05
Maze 100 × 100	FUEL [10]	406.0	52.13	504.1	355.1	814.4	123.54	1052.6	707.4	99.82	0.13	99.91	99.56	2.01	0.14	2.09	1.73
	RACER [9]	484.9	81.59	625.7	408.1	1003.1	95.83	1151.2	893.4	99.86	0.08	100.0	99.77	2.10	0.20	2.21	1.70
	FAME [11]	378.8	21.30	402.3	347.8	937.8	56.90	1020.8	856.0	99.79	0.10	99.91	99.61	2.48	0.21	2.69	2.13
	FALCON [8]	355.3	25.71	398.5	321.9	847.9	23.34	873.4	811.4	99.90	0.05	99.98	99.83	2.40	0.12	2.52	2.16
	EPIC [12]	415.3	35.83	452.8	355.5	1303.2	102.30	1409.8	1137.0	99.07	1.24	100.0	96.88	**3.14**	0.03	3.20	3.10
	Proposed	**304.3**	14.89	323.8	281.3	**674.4**	33.64	724.9	627.0	**99.92**	0.03	99.98	99.89	2.22	0.02	2.24	2.20
Village 100 × 100	FUEL [10]	322.6	17.16	346.5	299.5	677.6	103.00	874.5	589.3	**99.73**	0.17	100.0	99.56	2.09	0.25	2.59	1.95
	RACER [9]	520.3	54.04	612.4	449.7	989.1	176.30	1259.5	778.8	99.72	0.17	99.92	99.44	1.89	0.17	2.06	1.60
	FAME [11]	353.9	26.69	384.5	313.9	838.0	90.47	993.7	714.3	99.29	0.62	99.89	98.20	2.38	0.27	2.60	1.86
	FALCON [8]	318.1	18.57	348.1	296.4	696.0	39.52	737.9	638.9	99.70	0.15	99.99	99.60	2.19	0.06	2.26	2.12
	EPIC [12]	453.9	26.27	495.8	424.5	1357.8	71.34	1458.3	1272.1	99.70	0.32	99.97	99.09	**2.99**	0.03	3.04	2.94
	Proposed	**248.6**	16.98	274.1	223.9	**545.3**	15.77	563.9	517.3	99.59	0.10	99.75	99.48	2.20	0.18	2.52	2.00
Village 200 × 200	FUEL [10]	1983.5	78.82	2106.1	1876.3	2327.9	48.24	2381.1	2252.0	**99.52**	0.31	99.85	98.95	1.17	0.03	1.20	1.12
	RACER [9]	2582.0	366.06	3093.5	1992.6	3592.6	226.35	3831.0	3223.7	98.15	1.79	100.0	95.73	1.41	0.12	1.62	1.24
	FAME [11]	1363.6	104.72	1515.9	1221.1	3370.6	215.16	3760.3	3140.9	99.25	0.70	99.92	97.91	2.48	0.11	2.63	2.32
	FALCON [8]	1218.5	81.99	1370.5	1126.4	2584.4	180.21	2876.1	2347.7	99.27	0.61	99.99	98.34	2.12	0.08	2.21	1.99
	EPIC [12]	1744.5	238.17	2065.0	1455.3	4883.7	523.48	5623.9	4322.5	98.14	0.49	98.65	97.42	**2.81**	0.12	2.97	2.67
	Proposed	**1027.5**	54.79	1100.4	930.2	**2011.0**	71.56	2085.0	1890.2	99.08	0.51	99.77	98.18	1.96	0.06	2.03	1.88

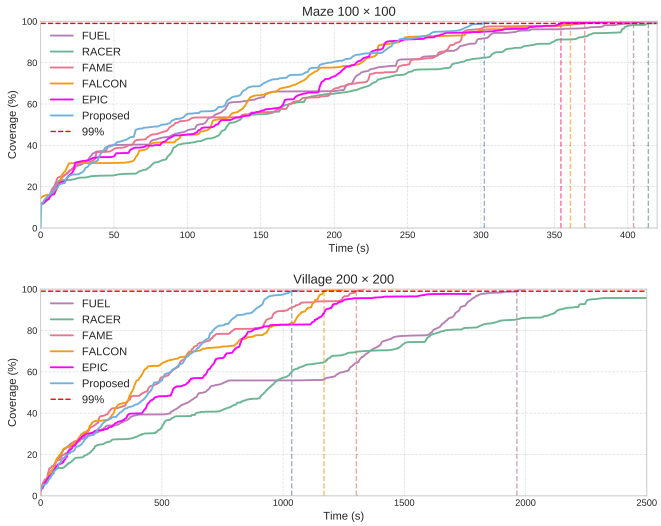
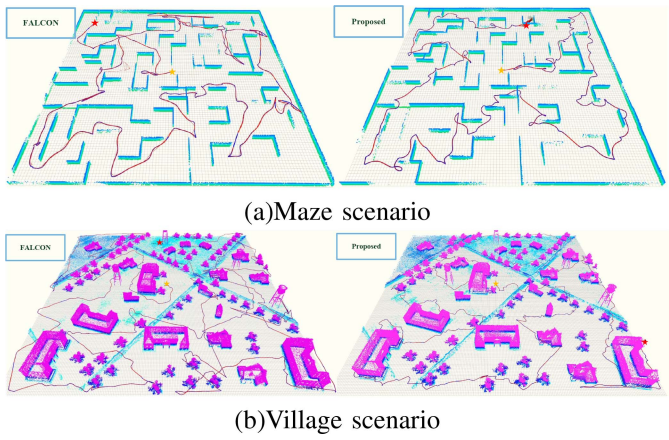


Fig. 5. Exploration progress of methods in the maze (top) and village (bottom) scenarios.



(a)Maze scenario

(b)Village scenario

Fig. 6. Exploration trajectories and constructed voxel grid maps of different methods in the maze (a) and village (b) scenarios. The yellow and red pentagrams indicate trajectories' starting and ending points, respectively. The color of the trajectory reflects the UAV's speed. The higher the speed, the trajectory color gradually changes from blue to red.

TABLE II
COMPUTATION TIME (MS) IN VILLAGE SCENARIOS

Size	Method	Update		Global		Trajectory		Total
		Avg	Max	Avg	Max	Avg	Max	Avg
100	FUEL	119.2	587.5	13.5	50.1	1.2	10.9	133.8
	RACER	42.0	251.1	230.6	560.8	4.4	91.6	276.9
	FAME	57.8	116.6	3.5	236.9	5.3	194.9	66.6
	FALCON	99.4	312.7	47.1	114.8	0.5	23.2	147.0
	EPIC	3.4	103.8	36.9	80.1	8.4	80.5	48.8
	Proposed	88.8	545.2	6.2	38.0	16.2	140.9	111.1
200	FUEL	424.9	2292.2	84.7	203.6	5.4	397.0	515.0
	RACER	288.6	4102.5	829.9	1767.7	7.2	225.4	1125.7
	FAME	168.3	360.3	16.1	2535.6	11.0	580.9	195.4
	FALCON	153.3	511.2	128.5	735.9	0.9	75.2	282.6
	EPIC	3.9	70.7	50.3	105.7	11.7	102.5	65.9
	Proposed	354.2	1319.2	29.3	167.4	18.2	72.5	401.7

in determining an appropriate global path, causing the UAV to keep moving back and forth. Although FAME switches modes to balance speed and completeness, both modes consider only a few neighboring frontiers rather than all available ones. Similarly, EPIC focuses on refining only the top few frontiers. During exploration, path costs change continuously; in complex scenes, failing to update the information in time severely affects the effectiveness of global planning. Distant goals exert a larger influence and trigger long-range round-trip more readily. In addition, surface frontier extraction is easily corrupted by sensor noise in challenging environments, yielding poor results.

Although RACER and FALCON consider unknown regions, they rely on fixed-shaped grid structures to partition unknown voxels, which substantially constrains the rationality and adaptability of the partitioning process. Specifically, RACER processes all unknown regions within the grid as a unified entity and lacks effective mechanisms to exclude inaccessible regions. FALCON employs a partitioning strategy within each grid cell based solely on connectivity, overlooking other potentially salient features. These limitations fail to accurately capture the intrinsic characteristics of unknown regions, thereby leading to the selection of viewpoints that inadequately represent them. Consequently, the quality of global path planning is compromised, and the resulting frontier selection deviates from the optimal strategy. Our algorithm partitions regions with coverability as the central criterion and seeks to have each viewpoint cover more unknown voxels. Guided by this, FLARE fully leverages the sensor range and known information to partition unknown regions. Benefiting from our partitioning scheme, we can better exclude unreachable unknown regions, avoid interference with exploration, and, by formulating a TSP, provide consistent and effective global guidance.

Table II presents the computation time for each component of the algorithms, where Update corresponds to the unknown region partitioning (including the computation of $\mathbf{C}_{\text{reg}}$) or the frontier update. Our computation time scales approximately linearly with task volume, while the local planning time remains nearly constant. Although EPIC's exploration efficiency is suboptimal, it is notably lightweight. FAME and EPIC consider only the surrounding frontiers, resulting in the shortest computation time

TABLE III
ABLATION STUDY, VILLAGE 100×100

Method	Exploration Time (s)	Flight Distance (m)	Coverage (%)	Avg Velocity (m/s)	Computation Time (ms)
No-Seg	314.4	639.9	99.45	2.03	91.9
No-Meg	325.6	611.9	99.28	1.88	473.1
Greedy	284.4	585.2	99.47	2.06	90.1
Proposed	**248.6**	**545.3**	**99.59**	**2.20**	111.1

but severely degrading exploration efficiency. This also reduces indecision among multiple targets, facilitating high flight speeds. Because RACER and FUEL consider all frontiers, the number of path searches increases exponentially with the problem size, resulting in a significant rise in computation time. The UAV must stop and wait for computations during exploration, significantly reducing the average speed. FALCON effectively reduces the construction time of the cost matrix and lowers the computational overhead of trajectory planning by building a connectivity graph. However, its strategy of simultaneously considering unknown regions and frontiers leads to a significantly longer global planning time than ours. In contrast, the proposed algorithm, through merging, does not require excessive path searches. The proposed algorithm introduces a comprehensive treatment of unknown regions, which does incur additional computational costs of Update relative to FALCON. Nevertheless, this overhead is both manageable and justified in light of the substantial improvements in exploration efficiency it enables. Real-world experiments show that FLARE can maintain acceptable computation times and handle large-scale scenarios on platforms with limited computational resources.

C. Ablation Study

We conduct an ablation study in the village environment (100×100) to evaluate the effectiveness of each component. We set up three variants: removing Coverability Segmentation (No-Seg), removing Region Merging (No-Meg), and replacing Global Path Planning with a greedy strategy (Greedy) that chooses the nearest unknown region, but retains the ability to remove unreachable unknown regions to ensure exploration. Each method is run five times. The experimental results are shown in Table III, where removing any component leads to a performance drop, with No-Meg causing the most significant degradation. In addition, No-Meg's total average computation time becomes four times.

D. Real-World Experiments

To further validate the feasibility of the proposed method, we conduct experiments in a forested area in a real environment. As shown in Fig. 7(b), we use a quadrotor UAV with a Rockchip RK3588 processor (16 GB) and a Mid360 LiDAR sensor. No external devices are employed for localization; all state estimation, mapping, planning, and control are performed on the onboard computer. In all tests, we set the dynamic constraints to $v_{\max} = 1.5\,\mathrm{m/s}$ and $a_{\max} = 1\,\mathrm{m/s^2}$.

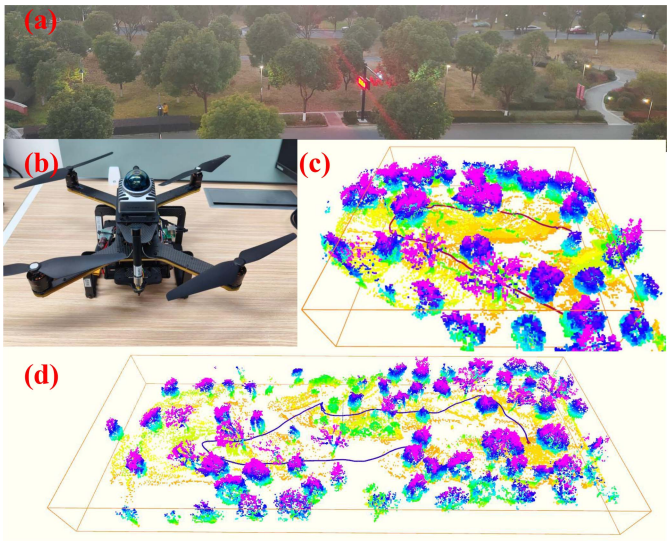


Fig. 7. Real-world experiments in a complex forest environment. (a) shows the experimental scenario, including a forest with significant undulations. In addition to trees, the environment includes obstacles such as shrubs and benches. (b) shows the UAV used in the experiment. (c) and (d) show the exploration trajectories and constructed maps during the two experiments.

Initially, we select a forest with dimensions of $40\,\mathrm{m} \times 40\,\mathrm{m} \times 10\,\mathrm{m}$ and restrict the perception range of the LiDAR to a radius of $12\,\mathrm{m}$. The exploration lasts $145.3\,\mathrm{s}$ and travels a distance of $92.5\,\mathrm{m}$. Subsequently, we conduct tests in a larger-scale scenario with specific dimensions of $113\,\mathrm{m} \times 45\,\mathrm{m} \times 10\,\mathrm{m}$. This time, the perception range of the LiDAR is extended to a radius of $22\,\mathrm{m}$. The exploration lasts $193.4\,\mathrm{s}$ and travels a distance of $168.8\,\mathrm{m}$. The experimental scenario is shown in Fig. 7(a). The terrain in this scene is highly undulating, and in addition to trees, there are also obstacles such as shrubs and benches, which fully demonstrate the algorithm's capabilities in complex, real-world scenarios. Fig. 7(c) and (d) illustrate the UAV's trajectories and the maps it constructs during the two experiments.

VII. Conclusion

This article presents FLARE, a fast and large-scale autonomous exploration framework guided by unknown regions. Specifically, FLARE fully uses known information through an incremental unknown region partitioning strategy, effectively characterizing and managing the unknown regions. Meanwhile, the hierarchical planner integrates global path planning with local trajectory generation to ensure the UAV can generate safe and dynamically feasible flight paths while maintaining efficient coverage. Extensive testing results in both simulation and real-world environments demonstrate that the proposed FLARE framework can complete exploration tasks with less exploration time and flight distance, significantly improving exploration efficiency.

References

[1] T. Dang, M. Tranzatto, S. Khattak, F. Mascarich, K. Alexis, and M. Hutter, "Graph-based subterranean exploration path planning using aerial and legged robots," *J. Field Robot.*, vol. 37, no. 8, pp. 1363–1388, 2020.

[2] H. Yao and X. Liang, "Autonomous exploration under canopy for forest investigation using LiDAR and quadrotor," *IEEE Trans. Geosci. Remote Sens.*, vol. 62, 2024, Art. no. 5704719.

[3] M. Erdelj, E. Natalizio, K. R. Chowdhury, and I. F. Akyildiz, "Help from the sky: Leveraging UAVs for disaster management," *IEEE Pervasive Comput.*, vol. 16, no. 1, pp. 24–32, Jan.–Mar. 2017.

[4] X. Kan, H. Teng, and K. Karydis, "Online exploration and coverage planning in unknown obstacle-cluttered environments," *IEEE Robot. Automat. Lett.*, vol. 5, no. 4, pp. 5969–5976, Oct. 2020.

[5] X. Zhao, C. Yu, E. Xu, and Y. Liu, "TDLE: 2-D LiDAR exploration with hierarchical planning using regional division," in *Proc. IEEE 19th Int. Conf. Automat. Sci. Eng.*, 2023, pp. 1–6.

[6] S. Song, D. Kim, and S. Jo, "Online coverage and inspection planning for 3D modeling," *Auton. Robots*, vol. 44, no. 8, pp. 1431–1450, 2020.

[7] Q. Bi et al., "CURE: A hierarchical framework for multi-robot autonomous exploration inspired by centroids of unknown regions," *IEEE Trans. Autom. Sci. Eng.*, vol. 21, no. 3, pp. 3773–3786, Jul. 2024.

[8] Y. Zhang, X. Chen, C. Feng, B. Zhou, and S. Shen, "FALCON: Fast autonomous aerial exploration using coverage path guidance," *IEEE Trans. Robot.*, vol. 41, pp. 1365–1385, 2025.

[9] B. Zhou, H. Xu, and S. Shen, "RACER: Rapid collaborative exploration with a decentralized multi-UAV system," *IEEE Trans. Robot.*, vol. 39, no. 3, pp. 1816–1835, Jun. 2023.

[10] B. Zhou, Y. Zhang, X. Chen, and S. Shen, "FUEL: Fast UAV exploration using incremental frontier structure and hierarchical planning," *IEEE Robot. Automat. Lett.*, vol. 6, no. 2, pp. 779–786, Apr. 2021.

[11] L. Bartolomei, L. Teixeira, and M. Chli, "Fast multi-UAV decentralized exploration of forests," *IEEE Robot. Automat. Lett.*, vol. 8, no. 9, pp. 5576–5583, Sep. 2023.

[12] S. Geng, Z. Ning, F. Zhang, and B. Zhou, "EPIC: A lightweight LiDAR-based AAV exploration framework for large-scale scenarios," *IEEE Robot. Automat. Lett.*, vol. 10, no. 5, pp. 5090–5097, May 2025.

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

[14] Z. Meng et al., "A two-stage optimized next-view planning framework for 3-D unknown environment exploration, and structural reconstruction," *IEEE Robot. Automat. Lett.*, vol. 2, no. 3, pp. 1680–1687, Jul. 2017.

[15] J. Huang et al., "FAEL: Fast autonomous exploration for large-scale environments with a mobile robot," *IEEE Robot. Automat. Lett.*, vol. 8, no. 3, pp. 1667–1674, Mar. 2023.

[16] Y. Luo et al., "Star-searcher: A complete and efficient aerial system for autonomous target search in complex unknown environments," *IEEE Robot. Automat. Lett.*, vol. 9, no. 5, pp. 4329–4336, May 2024.

[17] Y. Zhao, L. Yan, H. Xie, J. Dai, and P. Wei, "Autonomous exploration method for fast unknown environment mapping by using UAV equipped with limited FOV sensor," *IEEE Trans. Ind. Electron.*, vol. 71, no. 5, pp. 4933–4943, May 2024.

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

[19] Q. Dong et al., "Fast and communication-efficient multi-UAV exploration via voronoi partition on dynamic topological graph," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2024, pp. 14063–14070.

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

[21] C. Papachristos, S. Khattak, and K. Alexis, "Uncertainty-aware receding horizon exploration and mapping using aerial robots," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2017, pp. 4568–4575.

[22] B. Lindqvist, A. Patel, K. Löfgren, and G. Nikolakopoulos, "A tree-based next-best-trajectory method for 3-D UAV exploration," *IEEE Trans. Robot.*, vol. 40, pp. 3496–3513, 2024.

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

[24] M. Selin, M. Tiger, D. Duberg, F. Heintz, and P. Jensfelt, "Efficient autonomous exploration planning of large-scale 3-D environments," *IEEE Robot. Automat. Lett.*, vol. 4, no. 2, pp. 1699–1706, Apr. 2019.

[25] K. Helsgaun, "An effective implementation of the lin–Kernighan traveling salesman heuristic," *Eur. J. Oper. Res.*, vol. 126, no. 1, pp. 106–130, 2000.