drones

MDPI

Article

Efficient UAV Exploration for Large-Scale 3D Environments Using Low-Memory Map

Junlong Huang [1], Zhengping Fan [1], Zhewen Yan [2], Peiming Duan [2], Ruidong Mei [3] and Hui Cheng [2,*]

[1] School of Intelligent Systems Engineering, Sun Yat-sen University, Shenzhen 518107, China; huangjlong3@mail2.sysu.edu.cn (J.H.)
[2] School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou 510006, China
[3] School of System Science and Engineering, Sun Yat-sen University, Guangzhou 510006, China
* Correspondence: chengh9@mail.sysu.edu.cn

Abstract: Autonomous exploration of unknown environments is a challenging problem in robotic applications, especially in large-scale environments. As the size of the environment increases, the limited onboard resources of the robot hardly satisfy the memory overhead and computational requirements. As a result, it is challenging to respond quickly to the received sensor data, resulting in inefficient exploration planning. And it is difficult to comprehensively utilize the gathered environmental information for planning, leading to low-quality exploration paths. In this paper, a systematic framework tailored for unmanned aerial vehicles is proposed to autonomously explore large-scale unknown environments. To reduce memory consumption, a novel low-memory environmental representation is introduced that only maintains the information necessary for exploration. Moreover, a hierarchical exploration approach based on the proposed environmental representation is developed to allow for fast planning and efficient exploration. Extensive simulation tests demonstrate the superiority of the proposed method over current state-of-the-art methods in terms of memory consumption, computation time, and exploration efficiency. Furthermore, two real-world experiments conducted in different large-scale environments also validate the feasibility of our autonomous exploration system.

Keywords: unmanned aerial vehicles; autonomous exploration; path planning

check for updates

Citation: Huang, J.; Fan, Z.; Yan, Z.; Duan, P.; Mei, R.; Cheng, H. Efficient UAV Exploration for Large-Scale 3D Environments Using Low-Memory Map. *Drones* **2024**, *8*, 443. https://doi.org/10.3390/drones8090443

Academic Editors: Adrian Deaconu, Razvan Udroiu and Delia Elena Spridon

Received: 30 July 2024
Revised: 23 August 2024
Accepted: 27 August 2024
Published: 29 August 2024

1. Introduction

Recently, with the continuous development of robotic technology, unmanned aerial vehicles (UAVs) have attracted much attention and have been used in numerous fields due to their agility, flexibility, and high-speed mobility; these fields include inspection [1], terrain surveying [2], search and rescue [3–5], planetary exploration [6], etc. A key technology for these tasks is the autonomous exploration of unknown environments, in which the UAV moves autonomously and gathers information related to its surroundings to build environmental maps.

In the past few decades, many studies have proposed solutions that enable UAVs to explore unknown environments efficiently [1,7–9]. However, when facing large-scale environments, autonomous exploration still presents numerous challenges. First of all, the exploration process requires real-time environmental maps and the information necessary for planning, which consume more onboard memory resources as the size of the known environment grows. The increased memory consumption restricts the robot's capability to perform fundamental tasks for exploration, such as simultaneous localization and mapping (SLAM). Further, many exploration planning algorithms, such as [9–12], are computationally expensive when exploring large-scale environments. Algorithms that have long computational durations cannot quickly respond to environmental changes, resulting in some unnecessary actions by the robot, such as moving to a previous goal or stopping to wait for calculation results. Furthermore, the exploration of large-scale environments often suffers from inefficient observation paths, which slow down the robot's gathering of environmental information and even cause it to visit the same regions repeatedly, reducing the overall exploration efficiency.

To address the aforementioned challenges, we present an efficient UAV autonomous exploration framework for large-scale environments in this paper. To reduce the consumption of onboard memory during exploration, a novel low-memory representation of the environment is introduced that provides essential information for exploration by only maintaining information about the frontier (the boundary between unknown and free space) and the obstacles without retaining information related to extensive free and unknown spaces; for convenience, we name the environmental representattion **FAOmap** (**F**rontier **A**nd **O**bstacle map). Meanwhile, we propose a method for quickly updating FAOmap whenever the UAV moves and receives new environmental data. To ensure fast exploration planning when the environment changes and to find high-quality observation paths in large-scale environments, a hierarchical exploration planning approach based on the proposed FAOmap is developed that reduces the computational overhead and improves the exploration efficiency.

The proposed FAOmap is benchmarked against several typical maps used in exploration, and the proposed exploration method is compared to state-of-the-art methods in simulations. The results demonstrate that the proposed FAOmap can considerably decrease memory consumption compared to other maps. Moreover, in all tests, the proposed exploration planning method exhibits the shortest computation time out of the compared methods—only a few tens of milliseconds—ensuring prompt responses to environmental changes. And the proposed method consistently outperforms other methods in terms of total exploration time and movement distance. Furthermore, the proposed autonomous exploration system is validated by real-world experiments conducted in large-scale environments. In summary, the main contributions of this paper are as follows:

- A novel representation of the environment is presented to provide the essential information for exploration: only obstacle and frontier information are maintained, requiring less onboard memory. And a method is designed to update the representation immediately whenever new environmental data are received.
- A hierarchical exploration approach incorporating the proposed environmental representation is developed to enable fast planning and generate high-quality exploration paths, allowing for quick responses to environmental changes and improving overall exploration efficiency.
- Comprehensive simulations and real-world experiments demonstrate that the proposed environmental representation is memory-efficient, and the proposed exploration method is able to promptly plan efficient exploration paths in large-scale environments.

2. Related Work

2.1. Autonomous Exploration Methods

The problem of autonomous exploration, which requires a mobile robot to autonomously and safely navigate and map unknown environments as fast as possible, has been studied for several decades. Exploration approaches mainly include frontier-based methods, topological methods, information theoretic methods, and sampling-based methods.

Frontier-based methods identify all frontiers within the known map and then select one of them as the next target for exploration. As the robot moves, the existing map is gradually expanded until the entire environment is completely explored [7,13–16]. In the fundamental study on frontier-based methods by Yamauchi [7], the robot consistently chooses the closest frontier as the next goal. Inspired by [7], many researchers modify the approach to suit their own application situations. In order to achieve high-speed flight for UAVs during exploration, the frontiers within the current field-of-view (FOV) are prioritized as the targets in [16]. In [14], candidate targets are calculated on frontiers, and the one with the highest information gain is determined in order to rapidly expand the map. However, since many of these methods use greedy strategies, short-sighted movements occur frequently, resulting in inefficient exploration.

Topological methods typically construct a sparse graph to represent the environment during exploration, which consists of nodes positioned at key locations and edges connecting these nodes. The nodes of the graph are usually associated with the necessary information for determining the exploration target, and the edges are collision-free that used for searching of the traversable paths in the graph [17–20]. Nevertheless, topological maps fail to provide adequate details about the environment due to their concise representations.

Information theoretic methods quantify the uncertainty of maps using information theory and guide the robot to explore by determining the action with the highest information gain out of the possible candidates [21–25]. But evaluating the information gain of the map requires considerable computing power, which usually requires significant time, causing the robot to stop and wait for the results.

Sampling-based methods primarily include the random selection of viewpoints within the known space and the construction of rapidly exploring random trees (RRTs) [26] for generating potential paths. The most optimal path or target for exploring unknown environments is then determined using a utility function [9,10,27–29]. Regardless, most of these methods are only suitable for small-scale or confined environments, such as indoor spaces or subterranean tunnels. When the robot is faced with large-scale environments, back-and-forth actions easily occur in certain places, which leads to the robot becoming stuck in local areas.

In order to explore large-scale environments efficiently, some hybrid hierarchical methods that incorporate the characteristics of the aforementioned methods have been developed [11,12,30–33]; they usually consist of local and global planners. For example, the GBP method [11] was developed to explore large-scale subterranean mines. It first constructs a topological map using a random sampling method for discovering paths, then, it calculates the information gain of nodes, and finally, it selects high-gain nodes as frontiers to guide exploration. The TARE method [32] was presented to explore complex 3D environments and consists of global planning and local planning. In global planning, cuboid subspaces with uncovered surfaces are utilized as the targets of exploration, and global paths are searched using a key–pose graph. In local planning, the visible frontiers of candidate viewpoints are calculated to determine the optimal path that passes through some viewpoints to observe all frontiers. Furthermore, the FUEL method [31] was proposed to support fast UAV exploration in complex unknown environments; it is composed of an incremental frontier information structure that maintains essential information for exploration and a hierarchical exploration planner that generates efficient global coverage paths. However, most of these methods still suffer from excessive memory consumption and heavy computational overhead in complex and large-scale environments.

In recent years, with the advancement of deep learning, learning-based methods for autonomous exploration have been increasing in number [34–37]. After extensive training in simulated environments, they can directly output actions for robots that facilitate environment mapping based on perceived information. Nevertheless, most of them are suitable for indoor scenarios, and to the best of our knowledge, no learning-based approach has been developed for large-scale complex environments.

2.2. Environmental Representation

The environmental representation is a key component of autonomous exploration and provides essential information to other modules. Most early works in autonomous exploration focus on how to quickly explore a 2D environment [7,13,17], and they usually employ a 2D occupancy grid map or a topological map to represent the environment. For exploring large-scale and complex environments, the existing approaches mainly rely on the 3D occupancy grid map, which splits the space into numerous grids of equal size and records the space's occupancy state, providing rich and comprehensive information. For example, the AEP method [30] maps the environment using the Octomap [38], which is based on the octree structure and uses a probabilistic occupancy representation. The GBP method [11] works with either Octomap or Voxblox [39] depending on the user's choices. Voxblox is based on a voxel-hashing approach to incrementally build the map with a Euclidean signed distance field (ESDF). The FUEL method [31] utilizes FIESTA [40], which also is a global ESDF map, to represent the environment, facilitating the real-time trajectory planning. Additionally, the TARE method [32] uses point clouds to maintain free and occupied space. In our previous work [33], UFOmap [41] was employed to represent the environment. It is an extension of Octomap with a fast querying capability by using Morton code.

Although these maps accurately represent the environment, much of the recorded information, such as excessive free space, is unnecessary for exploration planning and consumes extra memory resources. Moreover, many frontier-based and hybrid hierarchical methods must calculate the frontiers based on these maps to provide richer information for decision-making. The stored information in these maps is not directly used, leading to inefficient exploration planning.

In this paper, a novel low-memory map is introduced to directly provide the frontier information, and a hierarchical exploration planning method based on the map is developed, enabling efficient exploration in large-scale 3D environments with low memory consumption and fast planning.

3. System Overview

The proposed autonomous exploration system framework is depicted in Figure 1. The SLAM module provides the UAV's localization and registered point clouds by processing sensor data. The local 3D grid map, which is centered at the UAV's current position, is updated by the SLAM results. And it is used to update FAOmap and the road map as well as being employed in global planning and local refinement. FAOmap only maintains information about the frontiers and the obstacles to provide the essential information for exploration. The road map is a sparse free-space graph that allows for efficiently searching for paths between points in the free space using A* or Dijstra's algorithms. Frontier clustering forms a lot of clusters quickly, and the calculation of super points that represent the frontier clusters facilitates global planning. The finding of the exploration route produces a preliminary route that passes through the super points. Local refinement samples candidate viewpoints in the vicinity of the first-to-be-observed cluster and determines an observation path that traverses some of the viewpoints to efficiently observe surrounding frontiers. Finally, trajectory optimization generates smooth and safe trajectories based on the generated paths, which the UAV follows to explore unknown environments.

Notably, in our system, different components operate at different frequencies. Global planning and local refinement are executed sequentially to determine the exploration paths at a preset frequency. However, the frequency of SLAM outputs and FAOmap updates correspond to the frequency of the used sensor, such as LiDAR, which usually works at 10–20 Hz, providing real-time environmental representation. Moreover, trajectory optimization performs at a higher frequency of $\geqslant$30 Hz, enabling the UAV to fly safely.

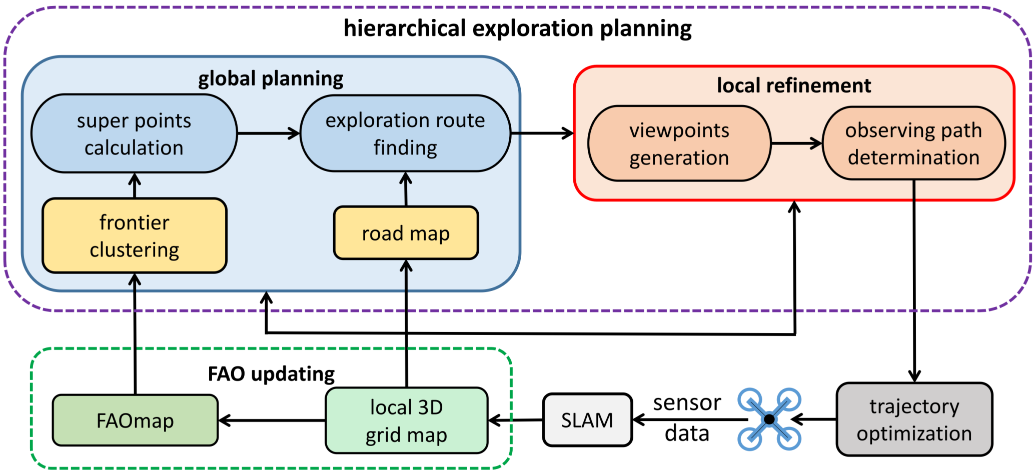

Figure 1. An overview of the proposed framework for autonomous exploration: The local 3D grid map is updated by the SLAM results, and then it is utilized to update FAOmap and the road map. Global planning, local refinement, and trajectory optimization are then conducted sequentially. Finally, the UAV executes the trajectory to explore unknown environments.

4. A 3D Environmental Representation for Exploration

Most autonomous exploration methods represent environments using the occupancy grid map, which splits the space into 3D grids with three occupancy states: free, occupied, and unknown. Such a map takes a significant amount of memory to store the free and unknown grids in large-scale environments because the free and unknown spaces generally make up the majority of the environment. However, the whole space's free and unknown information is not required for exploration, which mainly relies on the frontiers (the boundary between the free and the unknown spaces) and the occupancy state of the space around the UAV to determine exploration routes. Moreover, the occupancy grid map does not include the frontiers, which are calculated additionally during exploration planning. Therefore, we design a novel environmental representation to efficiently provide the necessary information during exploration, termed **FAOmap**, which only represents the frontiers and the occupied grids, enabling quick access to the frontiers and acquiring the occupancy state around the UAV.

4.1. Representation of FAOmap

As shown in Figure 2a, the whole exploration space is divided into free and unknown areas by the frontiers and the surfaces of obstacles, considering that the inside of the obstacle is an unobservable area and is regarded as unknown. The core idea of the proposed environmental representation is to record the frontier and the obstacle's surface with indicators of the occupancy states of adjacent areas, which can be used to identify the occupancy state of any area in the space.

In FAOmap, the entire space is split by 3D grids with a constant resolution r, as shown in Figure 2b, and the occupied grids only appear on the surfaces of the obstacles. A frontier f is a free grid for which at least one of its six neighbor grids (up, down, left, right, front, and back) is an unknown grid. Not all grids are stored: only the frontiers and occupied grids with indicators are stored in a tailored data structure (in Section 4.2). The indicator d of a grid g (frontier or occupied grid) has three dimensions that correspond to the coordinate directions of x, y, z, which indicate the occupancy states of the six neighbors of grid g. As depicted in Figure 2c,d, the value of d depends on whether the grid's neighbors are unknown or not, as below:

$$d_i = \begin{cases} 0, & \text{if } g_i^- \neq unknown \text{ and } g_i^+ \neq unknown \\ 1, & \text{if } g_i^- \neq unknown \text{ and } g_i^+ = unknown \\ -1, & \text{if } g_i^- = unknown \text{ and } g_i^+ \neq unknown \\ 2, & \text{if } g_i^- = unknown \text{ and } g_i^+ = unknown \end{cases} \tag{1}$$
$$i = x, y, z$$

where g_i^+ and g_i^- are the positively and negatively adjacent grids, respectively, of g in the i direction. Figure 2e shows an example of the calculation of the indicator. Based on the indicator of a grid, it is easy to determine the occupancy states of the areas adjacent to the grid, as described in Section 4.3.

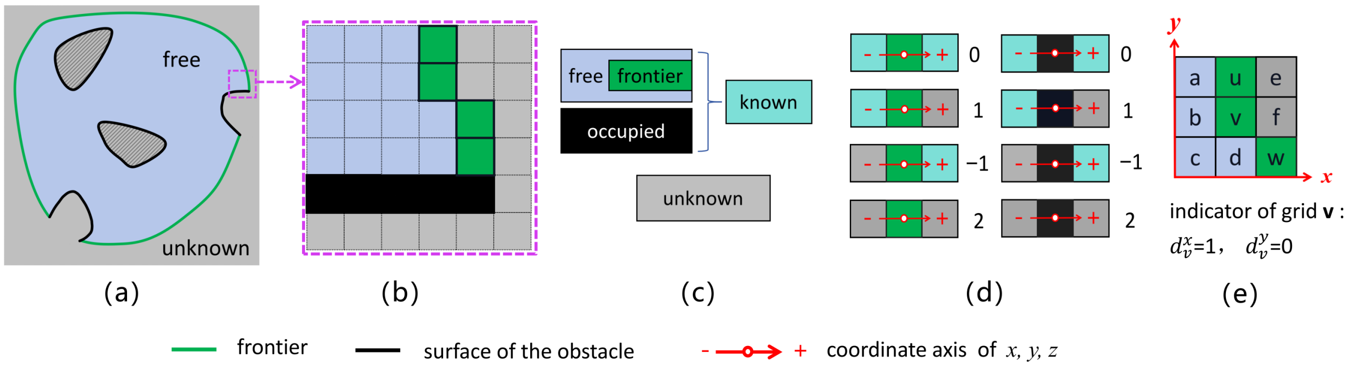

Figure 2. FAOmap representation (presented in 2D view). (**a**) The environmental representation during exploration. (**b**) The space is split by occupancy grids, and FAOmap only stores the frontiers and occupied grids. (**c**) The relationships between grid states. (**d**) Cases corresponding to the indicator values associated with frontiers and occupied grids in FAOmap. (**e**) An example of determining the indicator value of the grid v.

4.2. Data Structure of FAOmap

In order to minimize the memory consumption during exploration, arrays are firstly considered to store the frontiers and the occupied grids. However, the number of frontiers and occupied grids increases as the size of the known environment grows, leading to inefficient access to the required elements in a large array. Moreover, this causes a major decrease in exploration planning efficiency, which requires frequent acquiring of the frontiers and the occupancy state of the space around the UAV. Therefore, aiming to balance between memory consumption and querying efficiency, we adopt a hybrid data structure in FAOmap that combines a hash map and arrays to maintain the information about frontiers and obstacles.

As depicted in Figure 3, the entire environment space is equally divided into blocks of size R, which is an integer multiple of the resolution r. The hash map only stores blocks containing frontiers or occupied grids. The centroid of the block is used as the key value, and information about the frontiers and the occupied grids within the block is utilized as the element value. Each element in the hash map corresponding to a block consists of two arrays that record the frontiers and the occupied grids separately, together with their corresponding lengths. Each element in the arrays carries information about a frontier or an occupied grid: the grid's coordinates, occupancy probability, and an indicator indicating the occupancy state of the grid's neighbors. Since the hashing map and the small arrays allow for efficient querying, FAOmap ensures that the specific frontiers or occupied grids can be accessed quickly.

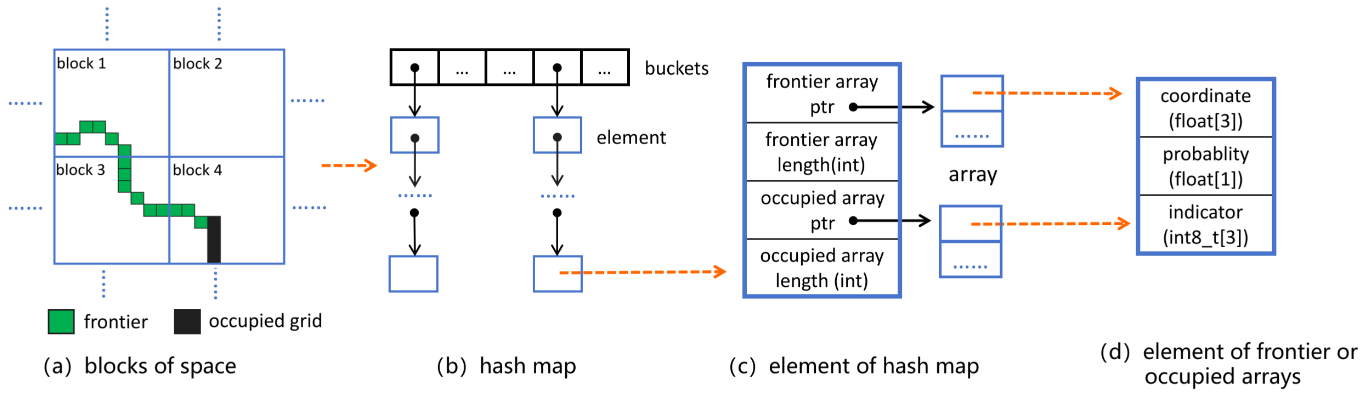

Figure 3. The data structure of FAOmap, which is organized by a hash map. (**a**) The whole space is divided into multiple blocks. (**b**) The hash map only stores the blocks that include frontier or occupied grids. (**c**) An element of the hash map contains the frontier and occupied arrays together with their corresponding length values. (**d**) An element of the frontier or occupied arrays consist of the center point of the frontier or occupied grid, an indicator that indicates the state of the grid's neighbors, and an occupancy probability of the grid.

4.3. Updating of FAOmap

Real-time updating of environment maps is required for autonomous exploration. Thus, a method for quickly updating FAOmap is proposed using a local 3D grid map with the same resolution r. As depicted in Figure 4, firstly, the model recovers the occupancy states surrounding the UAV in a local 3D grid map G. Then, it updates G using the newly perceived data. Finally, it extracts the new frontiers and occupied grids from G and adds them to FAOmap. The details for updating FAOmap are as follows:

(1) After the UAV moves, the local 3D grid map G is centered at the UAV's current position, and all grids in G are reset to the empty state (the real state has yet to be determined).
(2) Based on the coordinates, the frontiers and the occupied grids in the range of G are retrieved from FAOmap and stored in their corresponding grids in G, as shown in Figure 4b. And then, they are erased in FAOmap.
(3) The grids in G are clustered according to their current states (frontier, occupied, and empty). The possible occupancy states of a grid in G are frontier, free, occupied, and unknown, as shown in Figure 2c. Since the frontiers and occupied grids are identified in step 2, the empty grids are free or unknown. The real states of empty grids in the same cluster C_e are determined by the indicators of the frontiers or the occupied grids adjacent to C_e. For example, as shown in Figure 4c–e, supposing that along the positive direction of the x-coordinate, there is a frontier v adjacent to the grid b, which is in cluster C_e. After querying the indicator of grid v, whose value d_x^v is 1, it can be inferred that the state of grid b is known, according to Figure 2d. However, there are only two types of known grids: free and occupied. But occupied grids have already been identified in step (2), so the real state of grid b is free. Then the real states of all grids in C_e are free. Similarly, the real states of all empty grids in other clusters can be determined.
(4) The occupancy states of all grids in G are updated based on newly received data using a ray-cast approach, as in Octomap [38].
(5) All new frontiers and occupied grids in G are extracted and added to FAOmap.

In this way, FAOmap is continuously updated according to the newly received environmental information as the UAV flies. Additionally, the indicators of frontiers are applied for calculating the observation vectors, which are used to generate candidate viewpoints, and the local 3D grid map serves for local refinement in the proposed exploration method (in Section 5.2).

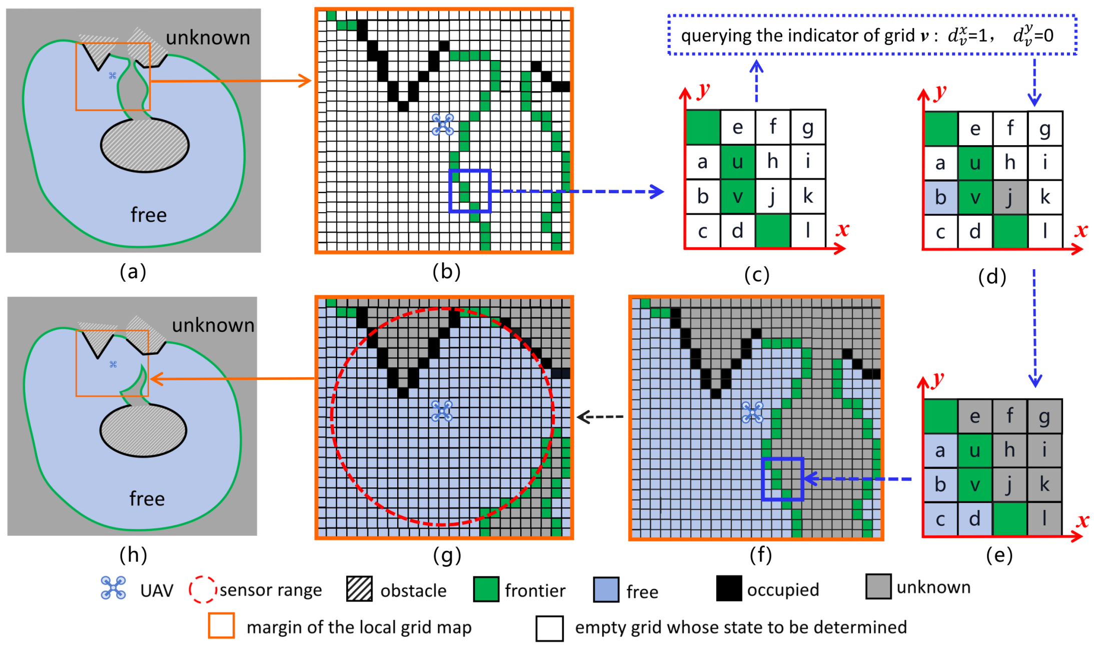
unknown
free
(a)
(b)
querying the indicator of grid v : d_v^x=1, d_v^y=0
y
e f g
a u h i
b v j k
c d l
x
(c)
(d)
(e)
(f)
(g)
(h)
UAV
sensor range
obstacle
frontier
free
occupied
unknown
margin of the local grid map
empty grid whose state to be determined

Figure 4. The updating of FAOmap (presented in 2D view), which relies on a local 3D grid map *G*. (**a**) The states of the space before FAOmap updating: the green line represents the frontiers, and the black line represents the obstacles' surfaces (the area inside the obstacle is unobservable and is regarded as unknown). (**b**) The states of grids in *G* after retrieving frontiers and occupied grids from FAOmap. The white grids are to be determined. (**c**–**e**) An example of determining the grid states by querying the indicators of grid *v*. (**f**) The real states of grids in *G*. (**g**) The states of grids in *G* after being updated by newly received data. (**h**) The states of the space after FAOmap updating.

5. Hierarchical Exploration Approach

Recently, some studies have pursued the optimal path that can efficiently observe global frontiers to guide exploration by a hierarchical planning method [31,32]. However, they still suffer from high computational overhead and quality degradation of exploration paths in large-scale 3D environments. Thus, following the frontier-based and the hierarchical methods, we propose a novel autonomous exploration planning approach that integrates closely with the proposed FAOmap to enable fast planning and efficient exploration.

5.1. Global Planning

Global planning aims to find a promising route to observe all frontiers for efficient exploration and consists of four components: frontier clustering, super point computation, exploration route finding, and road map extension.

Using numerous frontiers directly during planning is computationally expensive and provides too-coarse information for fine-grained decision-making, so many frontier-based methods typically group frontiers into clusters to speed up computation and extract richer information, such as [31]. Inspired by them, a fast incremental approach for clustering frontiers that relies on the local 3D grid map used in FAOmap updating is developed to improve the planning efficiency and facilitate exploration. Further, a super point for each cluster is calculated based on the indicators of the frontiers to facilitate the planning, providing a potential viewpoint for each cluster. In addition, our previous work [33] proposed a path optimization formulation that takes into account coverage efficiency, information gain, and movement distance to produce efficient 2D exploration routes for ground robots, which is proven to support fast exploration in challenging large-scale environments. Thus, in this work, we adapt the formulation to promptly plan high-quality 3D exploration routes for UAVs by simplifying the computation of information gain and employing super points as waypoints. Also, referring to the 2D road map in [33], a 3D road map is built and expanded based on the local 3D grid map to enable effective path searching between points in free space.

5.1.1. Fast Incremental Frontier Clustering

Assume that there are many clusters of frontiers before FAOmap updates. As shown in Figure 5, since the new frontiers always appear within the range of G, only a small number of clusters intersecting with G is considered to be updated when frontiers change, allowing for the fast grouping of new clusters. The detailed descriptions are as follows:

(1) Find all clusters that currently overlap with the local 3D grid map G. If a cluster falls inside the range of G, then delete it (as with the orange cluster in Figure 5a).
(2) For clusters that intersect with the boundary of G, some frontiers in these clusters may no longer be the frontiers after G is updated by newly perceived data. If the number of remaining frontiers in a cluster is less than the threshold N_{min}, delete the cluster (as with the green cluster in Figure 5); otherwise, retain clusters that consist of remaining frontiers (as with the blue cluster in Figure 5).
(3) Cluster all frontiers that are not yet clustered, including the new frontiers that appear after G is updated and some frontiers from the deleted clusters.

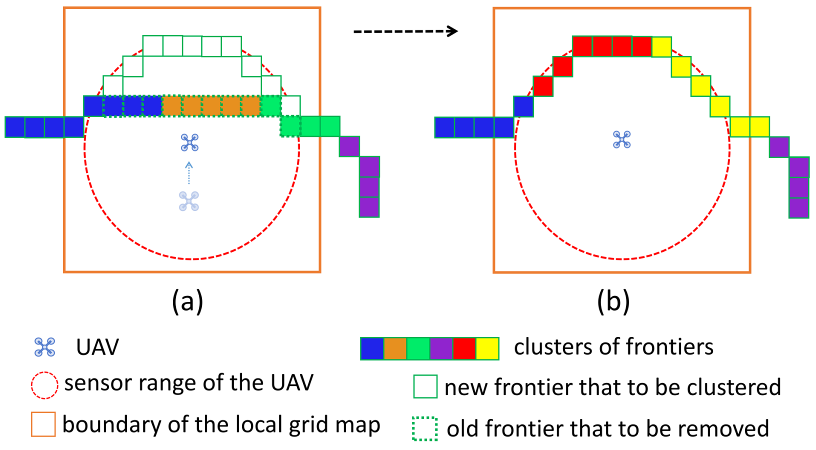
(a)
(b)
UAV
sensor range of the UAV
boundary of the local grid map
clusters of frontiers
new frontier that to be clustered
old frontier that to be removed

Figure 5. Fast incremental frontier clustering. (**a**) The old clusters and some new frontiers to be clustered when the local grid map is updated as the UAV moves. (**b**) The new clusters (red and yellow) are formed after updating. The green cluster in (**a**) is erased, and the blue cluster is modified.

In the presented method, the classic region-growing algorithm [42] is used to cluster the frontiers. To avoid forming large clusters that degrade the efficiency of local refinement (see Section 5.2), the maximum size of the cluster is limited by a threshold S_{max}, which is generally set to approximately half of the sensing range in our system.

5.1.2. Super Points for Frontier Clusters

In order to facilitate global planning, each cluster is considered to be represented by a super point p. Before calculating the super point, the observation vector $\overrightarrow{a_f}$ associated with a frontier f can be obtained by its indicator d in FAOmap; it represents the direction that is suitable for observing the frontier. In detail,

$$\overrightarrow{a_f} = \frac{e_x \cdot \overrightarrow{x} + e_y \cdot \overrightarrow{y} + e_z \cdot \overrightarrow{z}}{\sqrt{{e_x}^2 + {e_y}^2 + {e_z}^2}}$$
$$e_i = \begin{cases} 0, & \text{if } d_i = 0 \,||\, d_i = 2 \\ 1, & \text{if } d_i = -1 \\ -1, & \text{if } d_i = 1 \end{cases} \quad , i = x, y, z \tag{2}$$

where d_i $(i = x, y, z)$ are the indicator values in the three coordinate directions of x, y, z, respectively, e_i is a value to assist with calculation, and $\overrightarrow{x}$, $\overrightarrow{y}$, $\overrightarrow{z}$ are the three basis vectors in the coordinate system.

Then, the super point p for a cluster C is calculated by

$$\begin{aligned} c_{avg} &= \frac{\sum_{j=1}^{n} c_j}{n} \\ \overrightarrow{a_{avg}} &= \frac{\sum_{j=1}^{n} \overrightarrow{a_j}}{n} \\ \overrightarrow{op} &= \overrightarrow{oc_{avg}} + \lambda \cdot \overrightarrow{a_{avg}} \end{aligned} \tag{3}$$

where c is the coordinate of the frontier f, n is the total number of frontiers in C, o is the origin point of the coordinate, and λ is the tuning factor. As shown in Figure 6a,b, the super point provides a potential viewpoint for observing all frontiers in each cluster.

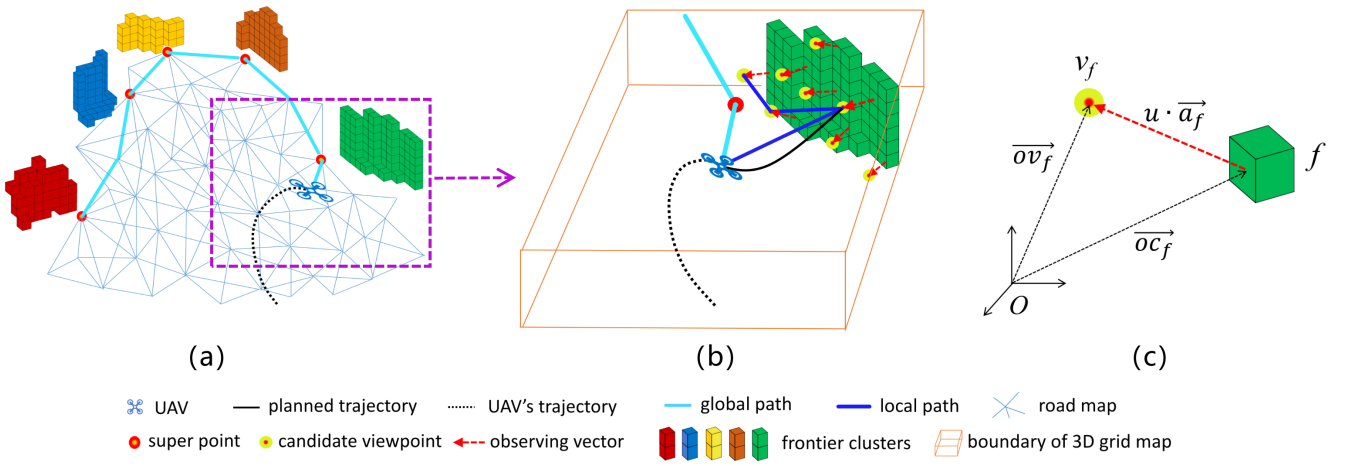

Figure 6. Hierarchical exploration planning. (**a**) Global planning: global frontiers are grouped into multiple clusters quickly and incrementally, and then a global exploration route is found for visiting these clusters in order. (**b**) Local refinement: when reaching the vicinity of the first-to-be-observed cluster, a local path that can efficiently observe all frontiers in the cluster is determined by the sampled candidate viewpoints. (**c**) Generation of the candidate viewpoint v_f by frontier f and its observation vector $\vec{a}_f$.

5.1.3. Exploration Routes Finding

To efficiently explore large-scale 3D environments, we adapt the path optimization method proposed in our previous work [33] to acquire a reasonable exploration route. In the path optimization formulation, an optimal path σ^* that visits global clusters in an orderly manner is found by maximizing the following function:

$$\begin{aligned} \sigma^* &= \max \textstyle\sum_{k=1}^{n} \mathbf{G}(p_k) \cdot \mathbf{P}(p_k) \\ &= \max \textstyle\sum_{k=1}^{n} \eta \cdot \mathbf{N_f}(p_k) \cdot exp(-\omega \cdot \mathbf{D_c}(p_0 \cdot p_k)) \end{aligned} \tag{4}$$

where $\sigma = [p_0, p_1, \cdots, p_n]$, p_0 is the UAV's current position, and $[p_1, \cdots, p_n]$ is a sequence of all super points. Additionally, $\mathbf{G}(p_k) = \eta \cdot \mathbf{N_f}(p_k)$ is the information gain for p_k, η is the tunable factor, and $\mathbf{N_f}(p_k)$ is the total number of frontiers in the cluster represented by p_k. Furthermore, $\mathbf{P}(p_k) = exp(-\omega \cdot \mathbf{D_c}(p_0 \cdot p_k))$ is the penalty function for p_k, ω is the penalty factor, and $\mathbf{D_c}(p_0 \cdot p_k)$ is the cumulative movement distance from p_0 to p_k along the path σ. Equation (4) can be solved quickly by a method based on the 2-opt local search heuristic, which is described in detail in [33]. By simultaneously considering the movement distance, information gain, and coverage efficiency in the optimization, an efficient exploration route that passes through all super points is obtained as depicted in Figure 6a.

In our previous work [33], candidate viewpoints were uniformly sampled around the ground robot in a 2D plane, and then the information gains of the viewpoints were evaluating by counting the number of their visible frontiers. Those viewpoints with high gain were selected as waypoints to acquire high-quality exploration routes. However, this path optimization strategy is time-consuming when exploring 3D environments using UAVs. Because the number of the sampled candidate viewpoints and the frontiers is significantly higher than in the ground scenarios, detecting the visibility between the viewpoints and the frontiers takes considerable time. To enable real-time performance for path optimization in this study, the elaborately calculated super points that represent clusters are utilized as waypoints. And the total number of frontiers in a cluster is directly employed as the corresponding super point's information gain, considering that all frontiers in the cluster are to be observed in local refinement (see Section 5.2), simplifying the overall computation.

5.1.4. Road Map for Path Searching

The proposed method requires the lengths of collision-free paths between each pair of super points in order to acquire efficient exploration routes, as illustrated in Equation (4). Therefore, as shown in Figure 6a, a sparse 3D road map M is constructed and expanded as the known space grows; it maintains a traversable space and allows fast searching of paths between pairs of super points. The growth of M relies on the local 3D grid map G. Firstly, the nodes of G are uniformly sampled at an interval ξ in the free space of G, ensuring the sparseness of M. Secondly, sampled points with no old nodes of M in a radius of ξ are added to G. Thirdly, the new nodes are connected to at most of n neighbors within the range l by collision-free checking in G.

5.2. Local Refinement

Global planning finds a coarse exploration route that passes through the frontier clusters sequentially, which is not the optimal path to observe the frontiers thoroughly. Thus, the locally refined path is required for more detailed and efficient observation of frontiers in each cluster. Many existing autonomous exploration methods sample a large number of candidate viewpoints in the UAV's surrounding space randomly or uniformly and then compute the information gain of the viewpoints to determine the optimal target or path. However, most viewpoints have low information gain and are deprecated in the end, leading to a lot of time spent on ineffective computation. In order to improve computational efficiency, a viewpoint generation method that samples a limited number of viewpoints with high information gain is proposed in this work. Moreover, many methods typically determine an optimal path that traverses some of the viewpoints to efficiently observe the frontiers by minimizing the movement cost or maximizing the information gain of paths. When the fields-of-view (FOVs) of the viewpoints overlap, the same frontier is observed repeatedly by the UAV from different viewpoints, which is called the submodularity problem [43], and results in the obtained path not actually being the optimal path. Therefore, inspired by [32], a local path determination strategy that considers the submodularity is designed based on the carefully sampled viewpoints and the local 3D grid map G (described in Section 4.3).

5.2.1. Viewpoint Generation

Considering a cluster C to be observed, multiple frontiers are sampled in C, and their observation vectors are calculated to generate the candidate viewpoints, as shown in Figure 6b,c. Firstly, multiple frontiers $F \subset C$ are sampled based on an equal interval ε. Then, the observation vector $\overrightarrow{a_f}$ for each frontier $f \in F$ is obtained according to Equation (2), which indicates the suitable direction for observing the frontier. Finally, the candidate viewpoints V are generated based on the frontiers F and their observation vectors. Like the calculation of super points in Equation (3), the viewpoint v_f of a frontier f is calculated by

$$\overrightarrow{ov_f} = \overrightarrow{oc_f} + \mu \cdot \overrightarrow{a_f} \tag{5}$$

where o is the origin point of the coordinate, c_f is the coordinate of the frontier f, and μ is the tuning factor. In this way, only a small number of viewpoints with high information gain for observing the frontiers are generated to determine the local path.

5.2.2. Observation Path Determination

As shown in Figure 6b, the UAV is at the center position p of the local 3D grid map G, and the frontiers to be observed in G are F^G. After viewpoint generation, the set of candidate viewpoints V^G is acquired. Path determination aims to find a high-potential path that traverses some of the viewpoints and is as short as possible while observing all frontiers within the horizon of G. We firstly obtain multiple subsets of V^G that are capable of observing all frontiers in F^G, considering the submodularity, then acquire the shortest path that starts at the UAV's current position and traverses all of the viewpoints in each subset. By comparing the shortest path of each subset, the optimal observation path is determined, which is a high-potential path for observing the frontiers.

To get a subset of V^G, the number of each viewpoint's visible frontiers $N_v^{f_{vis}}$ is counted by detecting the visibility in the local grid map G, where $f \in F^G, v \in V^G$. Then, a priority queue Q of viewpoints in descending order according to the number of visible frontiers is used to pick viewpoints, like the sampling viewpoints in [32]. Firstly, we select a viewpoint v' randomly from the first ϕ viewpoints of Q and remove the viewpoint v' from Q. Secondly, we update the visible frontier numbers of the remaining viewpoints in Q with which the visible frontiers of v' are not involved, accounting for the overlapping FOVs of viewpoints. Thirdly, we repeat these two steps until the queue Q is empty or the number of visible frontiers for each viewpoint in Q is less than a threshold t_{vis}, and the selected viewpoints form a subset $V'^G \subset V^G$ that can observe all frontiers in F^G. Then multiple subsets of V^G are acquired using the same steps.

We denote ρ^* as the optimal path to be determined, V'^G_k as the k-th subset of V^G, $E(v_i, v_j)$ as the edge connecting the viewpoints v_i and v_j, where $1 \leqslant i, j \leqslant n$, n is the total number of viewpoints in V'^G_k, and $\rho = [E(v_0, v_1), E(v_1, v_2), \cdots, E(v_{n-1}, v_n)]$ as the shortest path starting from the UAV's current position v_0 and passing through all of the viewpoints in V'^G_k. Then, the local path determination problem is solved based on the two steps below.

Firstly, the shortest path ρ for each subset V'^G is obtained by formulating the asymmetric traveling salesman problem (ATSP), which is a variant of the traveling salesman problem (TSP) [44]. The solution of the ATSP is the shortest open-loop tour of a set of points that starts from a specified point and passes through all other points. The ATSP can be solved quickly by the Lin–Kernighan heuristic [45], for which an effective implementation is available [46]. It is necessary to properly design a cost matrix M_{tsp} that corresponds to an $n+1$ dimensional square matrix for solving the ATSP, which is computed as:

$$\begin{gathered} M_{tsp}(i,j) = M_{tsp}(j,i) = \ell(E(v_i, v_j)) \\ M_{tsp}(0,i) = \ell(E(v_0, v_i)) \\ M_{tsp}(i,0) = 0 \\ i,j \in 1,2,\cdots,n \end{gathered} \tag{6}$$

where ℓ is the length of the edge $E(v_i, v_j)$. The shortest path ρ_k of the subset V'^G_k is acquired after the ATSP is solved by specifying the UAV's current position as the start point.

Secondly, the high-potential path with the shortest length is determined as the final outcome of local planning by comparing the shortest path of each subset as:

$$\rho^* = \min \ell(\rho_k), k \in 1,2,\cdots,\tau \tag{7}$$

where τ is the total number of subsets of viewpoints.

In [32], finding the local path that connects the global path requires sampling a large number of viewpoints, detecting the visibility between viewpoints and frontiers, searching paths between pairs of viewpoints, and solving the standard TSP by using the costs of searched paths, which is time-consuming. To reduce the computational overhead of our method, the number of viewpoints is limited to a small scale, and the costs of paths between pairs of viewpoints are found directly using the Euclidean distance. Additionally, our method uses the ATSP formulation to obtain an open-loop local path starting at the UAV's current position. It avoids the connection to the global path and strengthens the local autonomy, which enhances the overall exploration efficiency. Furthermore, local refinement does not smooth the path, because a trajectory optimization module is employed independently to optimize the generated paths (see Section 6), which compacts the computation of exploration planning.

It is worth noting that local refinement is not always carried out for exploration planning. During exploration in a large-scale environment, the UAV's current position and the next target cluster may be far away after global planning. In this case, local refinement is not performed because the UAV only needs to follow the global exploration route. When the frontiers of the target cluster appear within the horizon of the local 3D grid map, local refinement will be triggered.

6. Trajectory Optimization

To follow global exploration routes or local observation paths after exploration planning, EGO-planner [47], which was developed for effective trajectory optimization of UAVs, is utilized to generate a smooth and safe trajectory quickly. However, it uses a global map with a fixed size, which takes significant amounts of memory for large-scale scenes. Therefore, by using a 3D circular memory buffer [48], we modify the map representation in EGO-planner to a local rolling map centered on the UAV's current position, which requires little memory during exploration.

7. Simulation Experiments

To evaluate the performance of the proposed autonomous exploration system, extensive simulation experiments are conducted in three distinct large-scale environments: Indoor, Mountain, and Village, as shown in Figure 7. The MARSIM simulator [49], which is a lightweight point-realistic simulator for LiDAR-based quadrotor drones, is employed in the tests. And the drone is equipped with a Livox Mid-360 LiDAR sensor, for which the FoV is [360°, 59°]. All simulation tests are carried out on an Ubuntu 20.04 system with an Intel Core i7-8700k @ 3.70 GHz CPU.

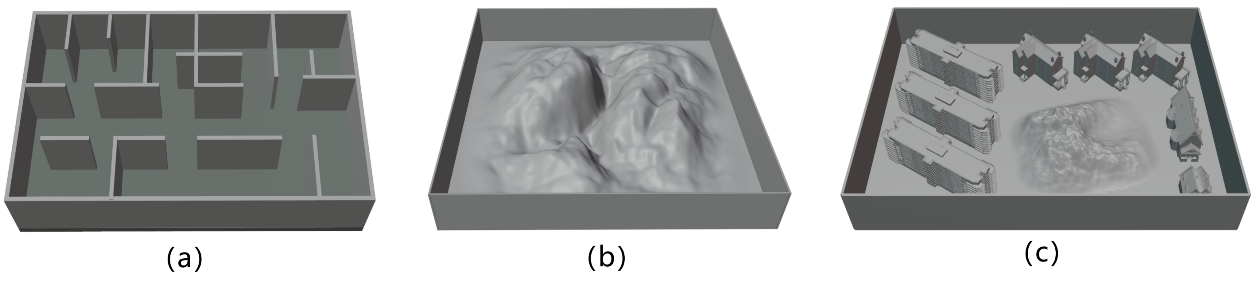

Figure 7. Three simulation environments: (**a**) Indoor: 105 m × 60 m × 20 m; (**b**) Mountain: 100 m × 100 m × 30 m; (**c**) Village: 150 m × 100 m × 30 m.

7.1. FAOmap Evaluation

To evaluate the memory consumption and the update efficiency of the proposed FAOmap, four benchmarks—Octomap [38], Voxblox [39], FIESTA [40], and UFOmap [41]—are compared using their open-source codes. Comparisons are carried out in the three distinct scenes shown in Figure 7, and all mapping methods are run simultaneously when the UAV moves, ensuring they are updated using the same perceived data. In each scene, the memory consumption and the update times of these maps are evaluated at different resolutions and sensing ranges.

7.1.1. Memory Consumption

As shown in Table 1, the proposed FAOmap consumes significantly less memory than the other maps in all comparisons. In the Indoor scene, at a resolution of 0.1 m, FAOmap consumes 64.8 MB of memory, which is only 6.2%, 6.3%, 0.8%, and 18.8% of the memory used by Octomap (1038.9 MB), Voxblox (1021.7 MB), FIESTA (7695 MB), and UFOmap (344.4 MB), respectively. At a resolution of 0.3 m, the memory usage of FAOmap (6.9 MB) is 36.9%, 9.99%, 1.43%, and 28.05% as compared to Octomap (18.7 MB), Voxblox (69.1 MB), FIESTA (480.9 MB), and UFOmap (25.6 MB), respectively. Moreover, at resolution of 0.5 m, all maps use less memory than at 0.3 m and 0.1 m. FAOmap takes 2.1 MB of memory, which is considerably less than Octomap (5.0 MB), Voxblox (20.8 MB), FIESTA (120.2 MB), and UFOmap (5.5 MB), with 42%, 10.1%, 1.75%, and 38.18%, respectively. In addition, the comparison results in the Mountain and Village scenes are similar to those in the Indoor scene, with FAOmap still consuming much less memory than the other maps. Because FAOmap only represents the frontiers and the obstacles, it avoids storing a huge quantity of information about the free and unknown space, resulting in significant memory reduction. Furthermore, Figure 8 depicts the memory usage of the five maps over the volume of known space in the three simulation environments at a resolution of 0.1 m. It shows that the gap in memory consumption between FAOmap and other maps grows as the known space increases, indicating that the proposed FAOmap is more memory-efficient while representing larger environments. Note that the curve of FiESTA is step-like since it is designed to double the map capacity whenever expanding the map.

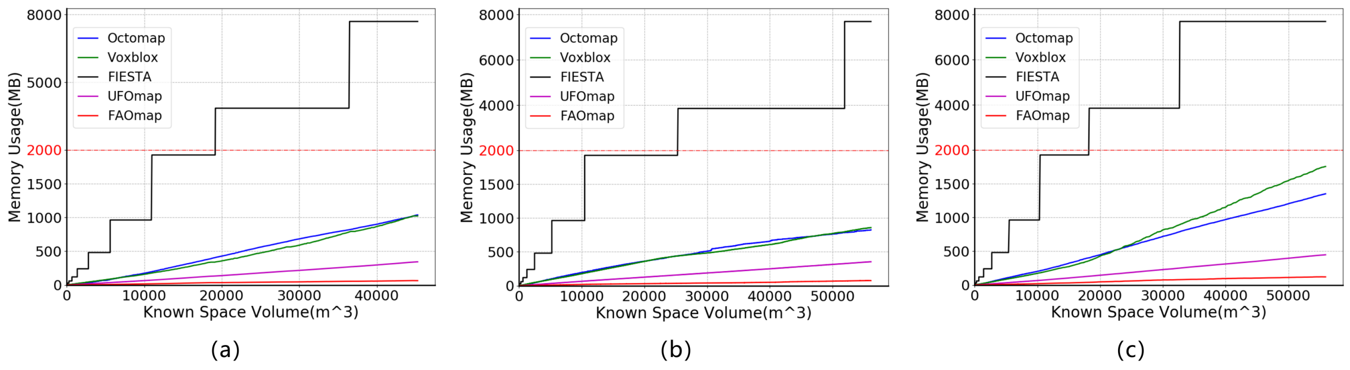

Figure 8. The memory usage (MB) vs. known space volume (m^3) for the five maps at a resolution of 0.1 m for three simulation scenes: (**a**) Indoor, (**b**) Mountain, and (**c**) Village. Ordinate values beyond 2000 are scaled three times.

7.1.2. Update Efficiency

In terms of update times, the proposed FAOmap performs well. As shown in Table 1, FAOmap, UFOmap, and FIESTA have substantially shorter update times than Octomap and Voxblox in all comparisons. At a resolution of 0.1 m for the three scenes, FAOmap is tens of milliseconds behind the best model (UFOmap), but it is still hundreds of milliseconds faster than Octomap and Voxblox. At a resolution of 0.3 m, FAOmap follows UFOmap closely by a few milliseconds in the Indoor and Mountain scenes; however, FAOmap performs best in the Village scene. At a resolution of 0.5 m, FAOmap, UFOmap, and FIESTA are all updated within 20 milliseconds—only a few milliseconds apart—showing that the proposed FAOmap can update as quickly as they can. All of these demonstrate that the proposed FAOmap has the ability to update in real time.

In summary, the proposed FAOmap provides a memory-efficient and real-time representation of a large-scale environment for exploration.

Table 1. Comparisons of environmental representations of three scenes.

Scene	**Indoor**						**Mountain**						**Village**					
Resolution r (m)	**0.1**		**0.3**		**0.5**		**0.1**		**0.3**		**0.5**		**0.1**		**0.3**		**0.5**	
Sensing Range (m)	**5**		**9**		**12**		**5**		**9**		**12**		**5**		**9**		**12**	
metric / method	total memory usage (MB)	average update time (ms)	total memory usage (MB)	average update time (ms)	total memory usage (MB)	average update time (ms)	total memory usage (MB)	average update time (ms)	total memory usage (MB)	average update time (ms)	total memory usage (MB)	average update time (ms)	total memory usage (MB)	average update time (ms)	total memory usage (MB)	average update time (ms)	total memory usage (MB)	average update time (ms)
Octomap [38]	1038.9	1238	18.7	107	5.0	58	824.8	1176	31.8	122	7.2	54	1353.8	1793	44.9	184	12.5	74
Voxblox [39]	1021.7	345	69.1	143	20.8	124	857.4	344	105.1	160	25.2	119	1754.1	332	125.3	202	40.8	115
FIESTA [40]	7695.0	261	480.9	49	120.2	**8**	7695.0	232	480.9	36	240.5	**11**	7695.0	97	961.9	38	240.5	**8**
UFOmap [41]	344.0	**51**	24.6	**20**	5.5	14	353.9	**47**	30.9	**27**	8.0	15	448.2	**53**	41.8	32	12.5	14
FAOmap	**64.8**	112	**6.9**	28	**2.1**	17	**76.5**	117	**8.0**	28	**2.6**	17	**120.7**	114	**12.1**	**30**	**4.7**	17

7.2. Exploration Planning Method Evaluation

To evaluate the computational efficiency and exploration efficiency of the proposed autonomous exploration planning method, simulation tests are conducted in three different large-scale environments, as shown in Figure 7. The performance of our method is benchmarked against two representative methods (FUEL [31] and TARE [32]) using three critical metrics: computation time, total exploration time, and total movement distance. Since each of the two methods has its own termination conditions, for fair comparison, the exploration is considered complete once 95% of the environment's volume has been observed for all methods. Further, both the Fuel and TARE methods are available online, but they have different configurations. Thus, we adapt them to the same configuration as our approach in simulations, which involves a UAV equipped with a Livox Mid-360 LiDAR sensor. In all tests, the UAV's maximum velocity is set to 3 m/s, and the maximum sensing range of the LiDAR is fixed at 15 m. Additional parameters specific to our approach are detailed in Table 2. Each method is tested 10 times in each scene, starting from the same initial position to ensure consistency in the performance evaluation. The statistics of the simulation results and the exploration progress for the three methods are displayed in Table 3 and Figure 9, respectively, providing a concise and clear understanding of our method's performance. The planning computation time, total exploration time, and total movement distance are average values calculated from 10 trials. Figure 10 depicts the executed trajectories of the UAV in a representative run for each method.

Table 2. Parameters of the proposed method for all simulations.

	FAOmap		Global Planning							Local Planning			
			Frontier Clustering		Road Map Updating			Path Planning		Viewpoint Generation	Path Determining		
param	r	R	N_{min}	S_{max}	ξ	n	l	η	ω	ε	ϕ	t_{vis}	τ
value	0.5 m	10	5	8 m	2 m	6	6 m	100	0.1	1.0 m	3	3	10

Table 3. Results of simulations in three environments.

Scene & Bounding Box For Exploration & Observable Volume	Method	Planning Computation Time (ms)	Total Exploration Time (s)		Total Movement Distance (m)		Average Speed (m/s)
			avg	std	avg	std	
Indoor 105 m × 60 m × 8 m 47,084 m^3	FUEL [31]	42	384.2	18.6	561.9	**24.4**	1.463
	TARE [32]	694	336.7	46.0	728.1	98.4	2.162
	proposed	**29**	**205.9**	**16.2**	**494.4**	35.4	**2.401**
Mountain 100 m × 100 m × 8 m 56,341 m^3	FUEL [31]	74	412.9	44.6	579.9	67.1	1.404
	TARE [32]	656	312.7	56.3	685.3	122.0	2.192
	proposed	**20**	**196.8**	**22.2**	**487.2**	**49.2**	**2.476**
Village 150 m × 100 m × 8 m 94,463 m^3	FUEL [31]	139	726.8	57.9	996.7	70.8	1.371
	TARE [32]	749	659.2	105.6	1387.6	220.3	2.105
	proposed	**24**	**383.9**	**29.5**	**923.2**	**69.9**	**2.405**

Observable volume: all space of the environment that can be observed by the UAV.

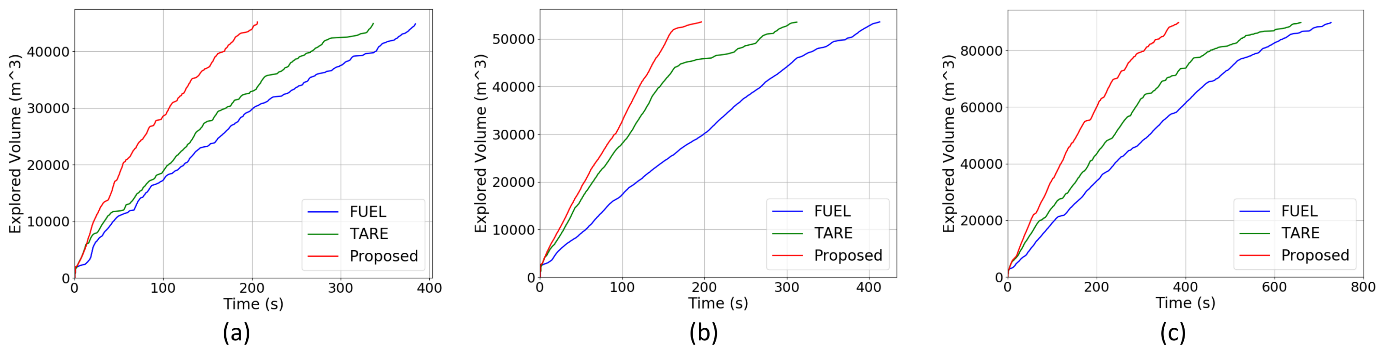

Figure 9. The exploration progress of the three methods in three different environments: (**a**) Indoor, (**b**) Mountain, and (**c**) Village. The charts show explored volume (m^3) vs. time (s).

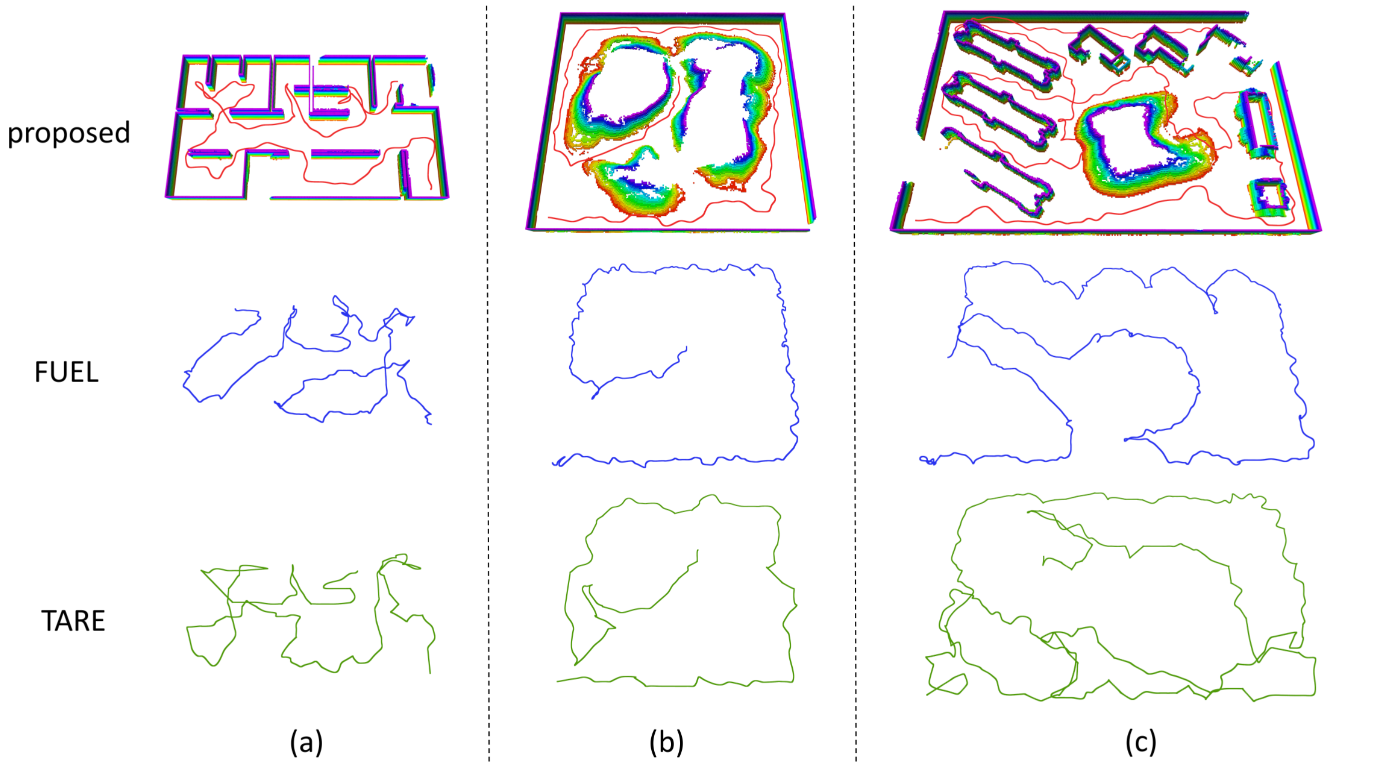

Figure 10. The executed trajectories of the three autonomous exploration methods in three different environments. The exploration is complete when the known space reaches 95% of the total observable space, according to the termination condition. (**a**) Indoor, (**b**) Mountain, and (**c**) Village.

7.2.1. Computational Efficiency

As shown in Table 3, the computation times of the proposed exploration planning method are 29 ms for the Indoor scene, 20 ms for the Mountain scene, and 24 ms for the Village scene, which are significantly faster than the times for the other two methods. This is because each component of the proposed exploration planning method has low computational overhead due to careful optimization for large-scale environments, including fast incremental frontier clustering, efficient global exploration route finding, timely local refinement, etc. Moreover, the computation time of the proposed method does not increase as the size of the scene grows, showing that the computation efficiency of the proposed method is stable in different environments with distinct sizes. However, the computation time of the FUEL method grows with the scene size, mainly because it searches paths between viewpoints using the occupancy grid map, which is computationally heavier for the larger environment. The TARE method has a significantly longer computation time than the proposed method and the FUEL method, mainly because it samples a large number of viewpoints and analyzes the visibility between them and the frontiers, which takes a lot of time.

7.2.2. Exploration Efficiency

As shown in Table 3 and Figure 9, the proposed method significantly improves the exploration efficiency compared to the other methods; it has least exploration time and the shortest movement distance in all tests. In the Indoor scene, the total exploration time of the proposed method is 205.9 s, with 46.4% and 38.8% improvement compared to FUEL (384.2 s) and TARE (336.7 s), respectively. In the Mountain scene, the proposed method has a total exploration time of 196.8 s and outperforms FUEL (412.9 s) and TARE (312.7 s) by 52.3% and 37.1%, respectively. In the larger scene of the Village, the total exploration time of the proposed method (383.9 s) also shows considerable improvements of 52.8% and 41.7% compared to FUEL (726.8 s) and TARE (659.2 s), respectively. For the total movement distances of the three methods, the proposed method is optimal in the three scenes, as with the total exploration time. The proposed method considers coverage efficiency, exploration gain, and movement cost to obtain a more reasonable exploration route in global planning and refines the local path to observe the frontiers efficiently, ensuring efficient exploration in large-scale environments. However, both the FUEL and TARE methods plan exploration routes by solving the TSP problem without considering the exploration gain, causing the UAV to revisit some sites during exploration, which reduces the overall efficiency. Furthermore, the TARE method fails to respond to environmental changes promptly, leading the UAV to pursue previous targets that do not provide high exploration gains during the current planning iteration over time. As a result, the TARE approach has a significantly longer overall movement distance than the other methods. Additionally, the proposed method generates smoother trajectories than the other methods, as depicted in Figure 10, and achieves the highest flight speed in the three scenes, as shown in Table 3, owing to fast planning of exploration paths and timely trajectory optimization. This also illustrates that the proposed method is more efficient for large-scale environments.

In brief, the simulation tests demonstrate that the proposed exploration method outperforms the benchmark methods in terms of computational efficiency and exploration efficiency, exhibiting the fastest computation time, least exploration time, and shortest movement distance in all tests.

8. Real-World Experiments

To further test the performance of the proposed autonomous exploration method in practice, real-world experiments are conducted in two distinct large-scale environments: an underground garage and a cluttered forest, as shown in Figures 11 and 12, respectively. The quadrotor drone is equipped with an Intel NUC (NUC10FNK with an Intel Core i7-10710U CPU and 16 GB of RAM) (Intel Corporation, Santa Clara, CA, USA) and a Livox Mid-360 LiDAR sensor (Livox Technology Co. Ltd., Shenzhen, China), as shown in Figure 13. Fast-lio2 [50], which is a fast, robust, and versatile LiDAR-inertial odometry framework, is adapted to acquire the localization of the drone in real time. In all experiments, the maximum sensor range is set to 15 m for FAOmap updating. The maximum velocity of the drone is set to 1.5 m/s, and the maximum acceleration is set to 1.0 m/s^2, ensuring collision avoidance and smooth flight by the quadrotor drone. When there are no observable high-gain frontier clusters, the exploration terminates.

For the underground garage scene, the space to be explored is bounded by a 72 m $\times$ 60 m $\times$ 2 m box, considering the free-flight height in the scene. During exploration, the drone flies 171 m in 159 s and explores 4929 m^3 inside the bounding box. The executed trajectory of the drone and the online-built point cloud map are shown in Figure 11. Notably, there are several untraversable areas within the bounding box due to environmental structural restrictions.

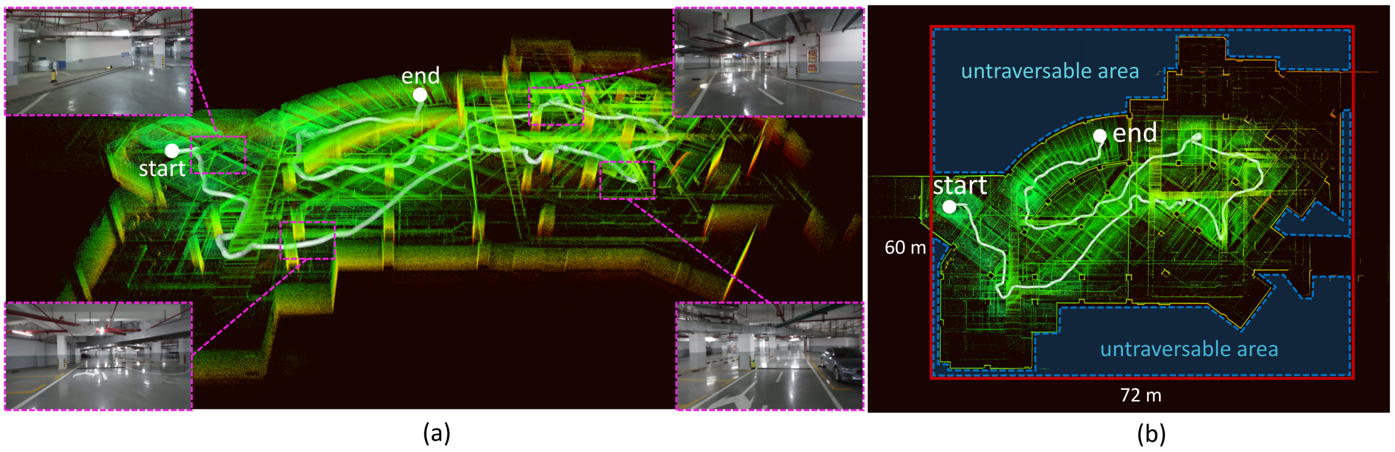

Figure 11. The real-world exploration in an underground garage. (**a**,**b**): Two different views of the online-built point cloud map and the UAV's trajectory, including images of the environment. The red rectangle represents the bounding box of the space to be explored. Several untraversable areas are within the bounding box due to environment structural restrictions. Some areas outside the bounding box are observed due to the long LiDAR sensor range.

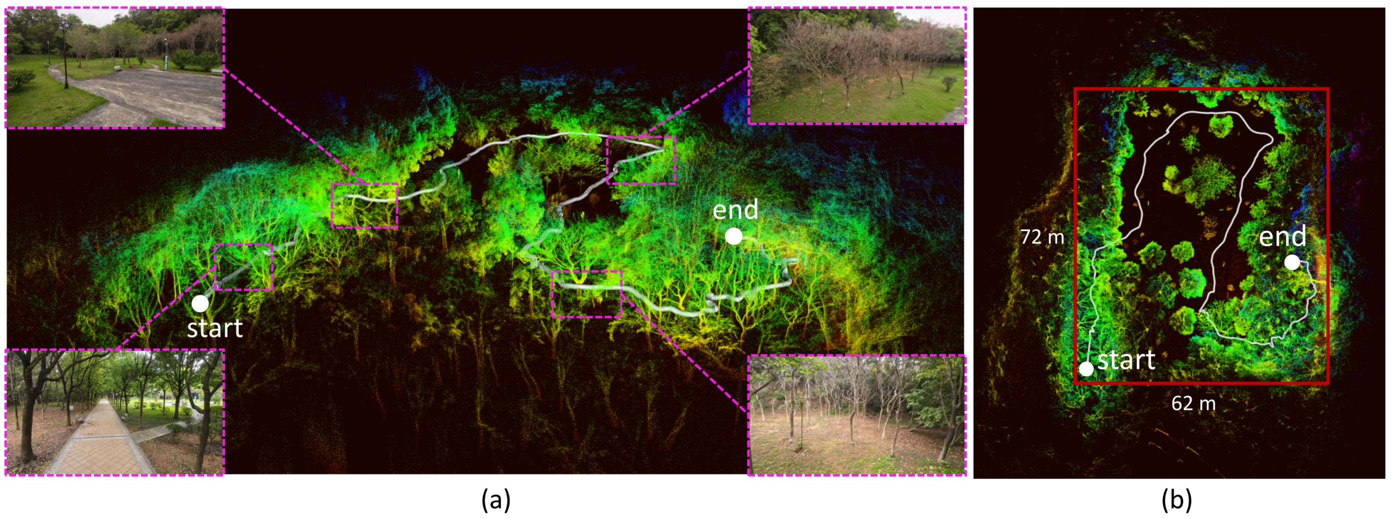

Figure 12. The real-world exploration in a cluttered forest. (**a**,**b**): Two different views of the online-built point cloud map and the UAV's trajectory, including images of the environment. The red rectangle represents the bounding box of the space to be explored. Areas outside the bounding box are also observed due to the long LiDAR sensor range.

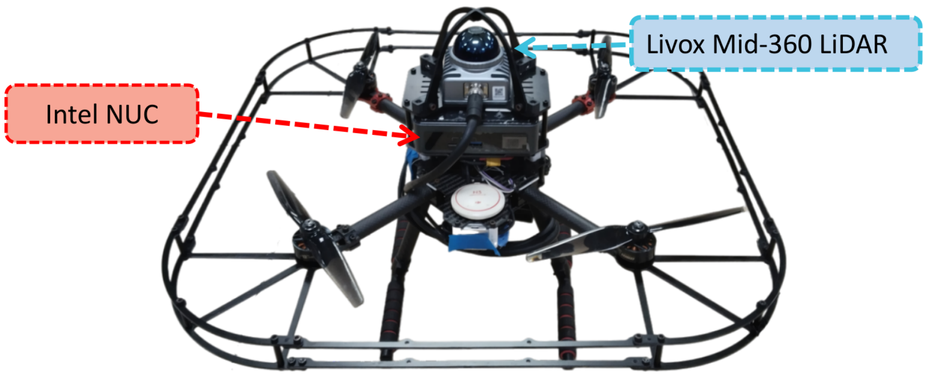

Figure 13. The flying platform used in real-world experiments: a quadrotor drone equipped with an Intel NUC and a Livox Mid-360 LiDAR sensor.

For the cluttered forest scene, the size of the area to be explored is 72 m × 62 m × 4.5 m. The trajectory and the point cloud map are displayed in Figure 12, and the drone flies 210 m in 189 s. The statistics of two experiments are listed in Table 4, and the exploration progress is shown in Figure 14.

The above experiments validate the capabilities of the proposed autonomous exploration system in real-world, complex, large-scale environments.

Table 4. The statistics of two real-world experiments.

Scene	Explored Time (s)	Movement Distance (m)	Explored Volume in Bounding Box (m^3)	FAOmap Memory Usage (KB)	Average Planning Time (ms)
garage	159	171	4924	173	19.7
forest	189	210	17,229	610	53.5

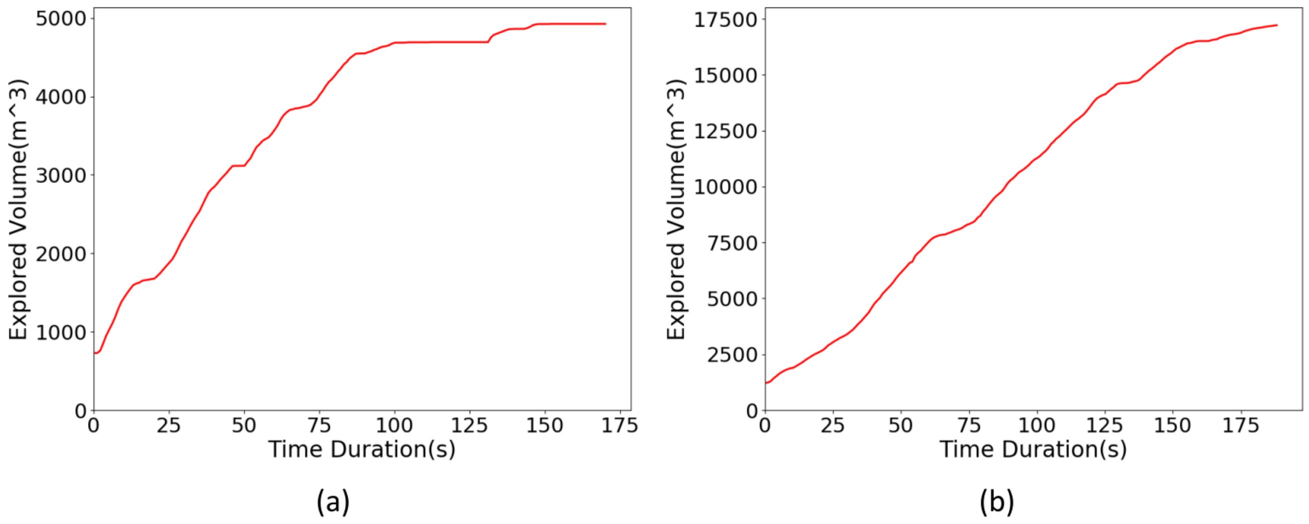

Figure 14. The exploration progress for two real-world experiments conducted in large-scale environments. (**a**) Garage. (**b**) Forest.

9. Conclusions and Future Work

This paper presents an efficient UAV autonomous exploration framework for large-scale unknown environments. In the framework, a low-memory environmental representation named FAOmap is introduced to reduce the consumption of onboard memory during exploration. Meanwhile, a quickly updating method for FAOmap is also presented. Moreover, a hierarchical exploration planning method that integrates closely with the proposed FAOmap is developed to quickly plan efficient exploration paths in large-scale environments. The proposed method reduces the computational overhead and generates more reasonable observation routes, ensuring fast planning and exploration efficiency. The proposed method is benchmarked against state-of-the-art approaches in various environments. The results demonstrate that the proposed FAOmap effectively reduces memory usage while maintaining real-time updating capacity, with an average update time that is faster than those of the majority of compared maps. Further, the proposed exploration planning method exhibits the fastest computation time, least exploration time, and shortest movement distance compared to the baseline methods mentioned in the paper. Furthermore, real-world experiments conducted in two large-scale environments also prove the applicability of the proposed autonomous exploration system. However, the proposed approach may occasionally ignore smaller regions during exploration, resulting in incomplete scans, and the mapped environment may not have the desired level of detail. In future work, autonomous exploration methods that focus more on the quality of the reconstructed maps will be researched.

Author Contributions: Conceptualization, J.H., Z.F. and Z.Y.; methodology, J.H., Z.F. and Z.Y.; software, J.H. and Z.Y.; validation, J.H., Z.Y., P.D. and R.M.; formal analysis, J.H.; investigation, J.H.; resources, Z.F. and H.C.; data curation, P.D.; writing—original draft preparation, J.H. and H.C.; writing—review and editing, J.H. and H.C.; visualization, J.H., Z.Y., P.D. and R.M.; supervision, Z.F. and H.C.; project administration, H.C.; funding acquisition, H.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the China National Key R&D Program (2022YFB3903804).

Data Availability Statement: The original data presented in the study are included in the article. Any further inquiries can be directed to the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Bircher, A.; Kamel, M.; Alexis, K.; Oleynikova, H.; Siegwart, R. Receding horizon path planning for 3D exploration and surface inspection. *Auton. Robot.* **2018**, *42*, 291–306. [CrossRef]
2. Tabib, W.; Goel, K.; Yao, J.; Boirum, C.; Michael, N. Autonomous cave surveying with an aerial robot. *IEEE Trans. Robot.* **2021**, *38*, 1016–1032. [CrossRef]
3. Wang, G.; Wang, W.; Ding, P.; Liu, Y.; Wang, H.; Fan, Z.; Bai, H.; Hongbiao, Z.; Du, Z. Development of a search and rescue robot system for the underground building environment. *J. Field Robot.* **2023**, *40*, 655–683. [CrossRef]
4. Chatziparaschis, D.; Lagoudakis, M.G.; Partsinevelos, P. Aerial and Ground Robot Collaboration for Autonomous Mapping in Search and Rescue Missions. *Drones* **2020**, *4*, 79. [CrossRef]
5. Han, D.; Jiang, H.; Wang, L.; Zhu, X.; Chen, Y.; Yu, Q. Collaborative Task Allocation and Optimization Solution for Unmanned Aerial Vehicles in Search and Rescue. *Drones* **2024**, *8*, 138. [CrossRef]
6. Sharma, M.; Gupta, A.; Gupta, S.K. Mars Surface Exploration via Unmanned Aerial Vehicles: Secured MarSE UAV Prototype. In *Holistic Approach to Quantum Cryptography in Cyber Security*; CRC Press: Boca Raton, FL, USA, 2022; pp. 83–98.
7. Yamauchi, B. A frontier-based approach for autonomous exploration. In Proceedings of the 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97.'Towards New Computational Principles for Robotics and Automation', Monterey, CA, USA, 10–11 June 1997; pp. 146–151.
8. González-Banos, H.H.; Latombe, J.C. Navigation strategies for exploring indoor environments. *Int. J. Robot. Res.* **2002**, *21*, 829–848. [CrossRef]
9. Schmid, L.; Pantic, M.; Khanna, R.; Ott, L.; Siegwart, R.; Nieto, J. An efficient sampling-based method for online informative path planning in unknown environments. *IEEE Robot. Autom. Lett.* **2020**, *5*, 1500–1507. [CrossRef]
10. Bircher, A.; Kamel, M.; Alexis, K.; Oleynikova, H.; Siegwart, R. Receding horizon 'next-best-view' planner for 3d exploration. In Proceedings of the 2016 IEEE International Conference on Robotics and Automation (ICRA), Stockholm, Sweden, 16–21 May 2016; pp. 1462–1468.
11. Dang, T.; Tranzatto, M.; Khattak, S.; Mascarich, F.; Alexis, K.; Hutter, M. Graph-based subterranean exploration path planning using aerial and legged robots. *J. Field Robot.* **2020**, *37*, 1363–1388. [CrossRef]
12. Zhu, H.; Cao, C.; Xia, Y.; Scherer, S.; Zhang, J.; Wang, W. DSVP: Dual-stage viewpoint planner for rapid exploration by dynamic expansion. In Proceedings of the 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Prague, Czech Republic, 27 September–1 October 2021; pp. 7623–7630.
13. Kulich, M.; Faigl, J.; Přeučil, L. On distance utility in the exploration task. In Proceedings of the 2011 IEEE International Conference on Robotics and Automation (ICRA), Shanghai, China, 9–13 May 2011; pp. 4455–4460.
14. Heng, L.; Gotovos, A.; Krause, A.; Pollefeys, M. Efficient visual exploration and coverage with a micro aerial vehicle in unknown environments. In Proceedings of the 2015 IEEE International Conference on Robotics and Automation (ICRA), Seattle, WA, USA, 26–30 May 2015; pp. 1071–1078.
15. Faigl, J.; Kulich, M. On determination of goal candidates in frontier-based multi-robot exploration. In Proceedings of the 2013 European Conference on Mobile Robots, Catalonia, Spain, 25–27 September 2013; pp. 210–215.
16. Cieslewski, T.; Kaufmann, E.; Scaramuzza, D. Rapid exploration with multi-rotors: A frontier selection method for high speed flight. In Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, 24–28 September 2017; pp. 2135–2142.
17. Rekleitis, I.M.; Dujmovic, V.; Dudek, G. Efficient topological exploration. In Proceedings of the 1999 IEEE International Conference on Robotics and Automation (Cat. No. 99CH36288C), Detroit, MI, USA, 10–15 May 1999; Volume 1, pp. 676–681.
18. Choset, H.; Walker, S.; Eiamsa-Ard, K.; Burdick, J. Sensor-based exploration: Incremental construction of the hierarchical generalized Voronoi graph. *Int. J. Robot. Res.* **2000**, *19*, 126–148. [CrossRef]

19. Brass, P.; Cabrera-Mora, F.; Gasparri, A.; Xiao, J. Multirobot tree and graph exploration. *IEEE Trans. Robot.* **2011**, *27*, 707–717. [CrossRef]
20. Akdeniz, B.C.; Bozma, H.I. Exploration and topological map building in unknown environments. In Proceedings of the 2015 IEEE International Conference on Robotics and Automation (ICRA), Seattle, WA, USA, 26–30 May 2015; pp. 1079–1084.
21. Bourgault, F.; Makarenko, A.A.; Williams, S.B.; Grocholsky, B.; Durrant-Whyte, H.F. Information based adaptive robotic exploration. In Proceedings of the IEEE/RSJ international Conference on Intelligent Robots and Systems, Lausanne, Switzerland, 30 September–4 October 2002; Volume 1, pp. 540–545.
22. Stachniss, C.; Grisetti, G.; Burgard, W. Information gain-based exploration using rao-blackwellized particle filters. In Proceedings of the Robotics: Science and Systems, Cambridge, MA, USA, 8–11 June 2005; Volume 2, pp. 65–72.
23. Bai, S.; Wang, J.; Chen, F.; Englot, B. Information-theoretic exploration with Bayesian optimization. In Proceedings of the 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Daejeon, Republic of Korea, 9–14 October 2016; pp. 1816–1822.
24. Henderson, T.; Sze, V.; Karaman, S. An efficient and continuous approach to information-theoretic exploration. In Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA), Paris, France, 31 May–31 August 2020; pp. 8566–8572.
25. Zhu, H.; Chung, J.J.; Lawrance, N.R.; Siegwart, R.; Alonso-Mora, J. Online informative path planning for active information gathering of a 3d surface. In Proceedings of the 2021 IEEE International Conference on Robotics and Automation (ICRA), Xian, China, 30 May–5 June 2021; pp. 1488–1494.
26. LaValle, S.M. Rapidly-exploring random Trees: A new tool for path planning. In *Research Report (TR 98-11)*; Computer Science Department, Iowa State University: Ames, IA, USA, 1998.
27. Papachristos, C.; Khattak, S.; Alexis, K. Uncertainty-aware receding horizon exploration and mapping using aerial robots. In Proceedings of the 2017 IEEE international conference on robotics and automation (ICRA), Singapore, 29 May–3 June 2017; pp. 4568–4575.
28. Witting, C.; Fehr, M.; Bähnemann, R.; Oleynikova, H.; Siegwart, R. History-aware autonomous exploration in confined environments using mavs. In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018; pp. 1–9.
29. Dang, T.; Papachristos, C.; Alexis, K. Visual saliency-aware receding horizon autonomous exploration with application to aerial robotics. In Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, Australia, 21–25 May 2018; pp. 2526–2533.
30. Selin, M.; Tiger, M.; Duberg, D.; Heintz, F.; Jensfelt, P. Efficient autonomous exploration planning of large-scale 3d environments. *IEEE Robot. Autom. Lett.* **2019**, *4*, 1699–1706. [CrossRef]
31. Zhou, B.; Zhang, Y.; Chen, X.; Shen, S. FUEL: Fast UAV exploration using incremental frontier structure and hierarchical planning. *IEEE Robot. Autom. Lett.* **2021**, *6*, 779–786. [CrossRef]
32. Cao, C.; Zhu, H.; Choset, H.; Zhang, J. TARE: A Hierarchical Framework for Efficiently Exploring Complex 3D Environments. In Proceedings of the Robotics: Science and Systems, Virtually, 12–16 July 2021.
33. Huang, J.; Zhou, B.; Fan, Z.; Zhu, Y.; Jie, Y.; Li, L.; Cheng, H. FAEL: Fast autonomous exploration for large-scale environments with a mobile robot. *IEEE Robot. Autom. Lett.* **2023**, *8*, 1667–1674. [CrossRef]
34. Georgakis, G.; Bucher, B.; Arapin, A.; Schmeckpeper, K.; Matni, N.; Daniilidis, K. Uncertainty-driven planner for exploration and navigation. In Proceedings of the 2022 International Conference on Robotics and Automation (ICRA), Philadelphia, PA, USA, 23–27 May 2022; pp. 11295–11302.
35. Yan, Z.; Yang, H.; Zha, H. Active neural mapping. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Paris, France, 1–6 October 2023; pp. 10981–10992.
36. Tao, Y.; Wu, Y.; Li, B.; Cladera, F.; Zhou, A.; Thakur, D.; Kumar, V. Seer: Safe efficient exploration for aerial robots using learning to predict information gain. In Proceedings of the 2023 IEEE International Conference on Robotics and Automation (ICRA), London, UK, 29 May–2 June 2023; pp. 1235–1241.
37. Zhao, Y.; Zhang, J.; Zhang, C. Deep-learning based autonomous-exploration for UAV navigation. *Knowl.-Based Syst.* **2024**, *297*, 111925. [CrossRef]
38. Hornung, A.; Wurm, K.M.; Bennewitz, M.; Stachniss, C.; Burgard, W. OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Auton. Robot.* **2013**, *34*, 189–206. [CrossRef]
39. Oleynikova, H.; Taylor, Z.; Fehr, M.; Siegwart, R.; Nieto, J. Voxblox: Incremental 3d euclidean signed distance fields for on-board mav planning. In Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, 24 September 2017; pp. 1366–1373.
40. Han, L.; Gao, F.; Zhou, B.; Shen, S. Fiesta: Fast incremental euclidean distance fields for online motion planning of aerial robots. In Proceedings of the 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), The Venetian Macao, Macau, 3–8 November 2019; pp. 4423–4430.
41. Duberg, D.; Jensfelt, P. UFOMap: An efficient probabilistic 3D mapping framework that embraces the unknown. *IEEE Robot. Autom. Lett.* **2020**, *5*, 6411–6418. [CrossRef]
42. Hojjatoleslami, S.; Kittler, J. Region growing: A new approach. *IEEE Trans. Image Process.* **1998**, *7*, 1079–1084. [CrossRef]

43. Roberts, M.; Dey, D.; Truong, A.; Sinha, S.; Shah, S.; Kapoor, A.; Hanrahan, P.; Joshi, N. Submodular trajectory optimization for aerial 3d scanning. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 5324–5333.
44. Applegate, D.L.; Bixby, R.E.; Chvátal, V.; Cook, W.J. *The Traveling Salesman Problem: A Computational Study*; Princeton University Press: Princeton, NJ, USA, 2007.
45. Lin, S.; Kernighan, B.W. An effective heuristic algorithm for the traveling-salesman problem. *Oper. Res.* **1973**, *21*, 498–516. [CrossRef]
46. Helsgaun, K. An effective implementation of the Lin–Kernighan traveling salesman heuristic. *Eur. J. Oper. Res.* **2000**, *126*, 106–130. [CrossRef]
47. Zhou, X.; Wang, Z.; Ye, H.; Xu, C.; Gao, F. Ego-planner: An esdf-free gradient-based local planner for quadrotors. *IEEE Robot. Autom. Lett.* **2020**, *6*, 478–485. [CrossRef]
48. Usenko, V.; Von Stumberg, L.; Pangercic, A.; Cremers, D. Real-time trajectory replanning for MAVs using uniform B-splines and a 3D circular buffer. In Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, 24–28 September 2017; pp. 215–222.
49. Kong, F.; Liu, X.; Tang, B.; Lin, J.; Ren, Y.; Cai, Y.; Zhu, F.; Chen, N.; Zhang, F. MARSIM: A light-weight point-realistic simulator for LiDAR-based UAVs. *IEEE Robot. Autom. Lett.* **2023**, *8*, 2954–2961. [CrossRef]
50. Xu, W.; Cai, Y.; He, D.; Lin, J.; Zhang, F. FAST-LIO2: Fast Direct LiDAR-Inertial Odometry. *IEEE Trans. Robot.* **2022**, *38*, 2053–2073. [CrossRef]