

# Bubble Explorer: Fast UAV Exploration in Large-Scale and Cluttered 3D-Environments using Occlusion-Free Spheres

Benxu Tang\*, Yunfan Ren\*, Fangcheng Zhu, Rui He, Siqi Liang, Fanze Kong and Fu Zhang

**Abstract**— Autonomous exploration is a crucial aspect of robotics that has numerous applications. Most of the existing methods greedily choose goals that maximize immediate reward. This strategy is computationally efficient but insufficient for overall exploration efficiency. In recent years, some state-of-the-art methods are proposed, which generate a global coverage path and significantly improve overall exploration efficiency. However, global optimization produces high computational overhead, leading to low-frequency planner updates and inconsistent planning motion. In this work, we propose a novel method to support fast UAV exploration in large-scale and cluttered 3-D environments. We introduce a computationally low-cost viewpoints generation method using novel occlusion-free spheres. Additionally, we combine greedy strategy with global optimization, which considers both computational and exploration efficiency. We benchmark our method against state-of-the-art methods to showcase its superiority in terms of exploration efficiency and computational time. We conduct various real-world experiments to demonstrate the excellent performance of our method in large-scale and cluttered environments.

## I. INTRODUCTION

Autonomous exploration, where robots explore unknown environments and gather information independently, has become increasingly popular in applications such as mine exploration, industrial inspection, and search and rescue operations. Robots can access areas that are difficult for humans to reach, and reduce the risks humans expose to in hazardous environments.

The task of autonomous exploration is to plan a path to explore the entire unknown environment as quickly as possible. Various exploration methods have been proposed in recent years to tackle the task. Most of these methods adopt a greedy strategy. [1]–[3] span RRT [4] in the environment and select the node with the highest information gain to visit. [5], [6] select the frontier that minimizes the traversal cost or the direction change of the UAV as the goal. The greedy-based methods are computationally efficient but insufficient in terms of overall exploration efficiency, as they ignore global optimality and generate back-and-forth movements. Other methods, such as [7], adopt a

\*These two authors contributed equally to this work.

B. Tang, Y. Ren, F. Zhu, R. He, F. Kong and F. Zhang are with the Department of Mechanical Engineering, University of Hong Kong {tangbenx, renyf, zhufc, herui, kongfz}@connect.hku.hk, fuzhang@hku.hk, S. Liang is with School of Mechanical Engineering and Automation, Harbin Institute of Technology sqliang@stu.hit.edu.cn.

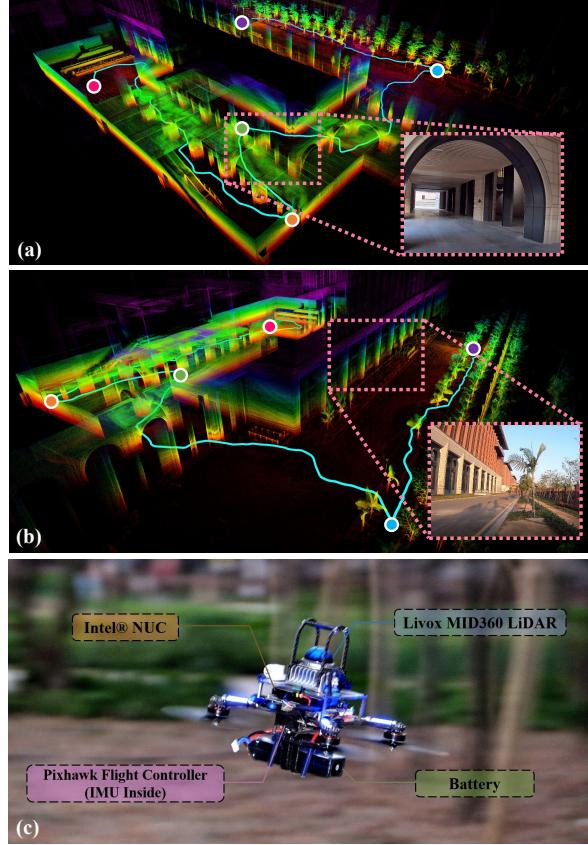


Fig. 1. Performing exploration task in a large-scale environment composed of both indoor and outdoor spaces. (a) and (b): Two different views of the online-built point cloud map, and trajectory executed by the UAV (light blue line), with images displaying the environment. The points with the same color indicate the same position. (c): The quadrotor platform used in the exploration.

global optimization strategy that finds a global tour to visit unexplored regions. This strategy improves overall exploration efficiency but results in high computational time, leading to low planner update frequency and inconsistent planning motion. Moreover, existing methods generate viewpoints in a sampling way and evaluate the reward of the viewpoint using a computationally expensive ray-casting process, which further increases the computational cost.

Motivated by these facts, we propose a novel method that can support fast and efficient UAV exploration in large-scale and cluttered 3-D environments. We introduce two key contributions: 1) A novel concept of

the occlusion-free sphere, which generates high-quality viewpoints at a low computational cost. 2) Based on the generated viewpoints, we introduce a novel strategy that combines greedy with global optimization, which finds an efficient global tour visiting high-gain viewpoints, balancing overall exploration efficiency and computational cost. Finally, we design a local planner that generates safe and kinodynamically feasible trajectories for the UAV to follow. We validate the proposed method through extensive simulation and real-world experiments, showing that it outperforms the state-of-the-art baselines in terms of both exploration efficiency and computational time.

To sum up, the contributions of this paper are listed below:

- 1) We propose a novel concept of the occlusion-free sphere to generate high-quality viewpoints, which significantly saves computational time and improves exploration efficiency.
- 2) Based on the generated viewpoints, we introduce a novel strategy that combines greedy and global optimization, which finds an efficient global tour to visit high-gain unexplored regions, balancing overall exploration efficiency and computational cost.
- 3) Extensive simulation experiments demonstrate the advantages of the proposed planner over the state-of-the-art baselines, in terms of exploration efficiency and computational time.
- 4) Implementation of the proposed planner on a fully autonomous quadrotor platform. Various real-world tests show the outstanding performance of the proposed planner in large-scale and cluttered real-world environments.

## II. RELATED WORKS

Autonomous exploration has been an active area of research in recent years, and a variety of methods have been proposed to tackle the problem. Sampling-based exploration [1]–[3], [8] is one of the classic approaches. The approach spans a Rapidly-exploring Random Tree (RRT) [4] in free space. It evaluates the information gain of the nodes in RRT by the coverage of the unknown region, weighted with the traversal cost to reach it from the current position. The coverage is counted by the number of unknown voxels that fall in the sensor field of view (FoV) and are not occluded by occupied voxels (e.g., by ray-casting). The node with the highest gain is selected as the goal and a traversable path to the node is derived from the RRT. This scheme is first introduced by the Next-Best-View Planner (NBVP) [1], and further improved by GBP [2] and MBP [3]. In GBP [2], a topological global map is constructed during the local exploration process. When the local area is fully explored, or the vehicle encounters a dead end, the method finds a path on the global map and redirects the vehicle

to unexplored areas. MBP [3] constructs RRT using motion-primitives and produces smooth trajectories for the vehicle to execute.

Another classic approach is frontier-based exploration [5]–[7], [9]–[11]. In frontier-based exploration, the vehicle navigates close to the frontier, defined as the boundary between the free and unknown space, to continue exploring the unknown space. This method is first introduced by [5], in which the closest frontier is selected as the next goal. To achieve high-speed flight, [6] selects the frontier in sensor FoV and minimizes the velocity change of the vehicle. [12] analyzes the strengths and weaknesses of the sampling-based and frontier-based approaches. It combines them together by improving NBVP [1] for local exploration and using a frontier-based approach for global exploration.

The above methods are greedy-based, which select goals that maximize the immediate reward to visit at each planning iteration. This strategy is computationally efficient but insufficient in terms of overall exploration efficiency, as it produces back-and-forth planning motions. Fast UAV Exploration planner (FUEL) [7] considers the global optimality. It begins by clustering the frontier cells using a region-growing algorithm and performing Principal Component Analysis (PCA) to split large frontier clusters into smaller ones along the first principal axis. Viewpoints are then sampled around the frontier clusters within a cylindrical coordinate system and evaluated by frontier coverage using ray-casting. The viewpoints with the highest frontier coverage are selected. After that, the method finds a global tour that minimizes the global traversal cost, starting from the current vehicle position and passing through all selected viewpoints. It formulates the problem as a variant of the Traveling Salesman Problem (TSP). To solve the problem, the algorithm first searches for collision-free paths between each pair of viewpoints and between each viewpoint and the current vehicle position using the A\* algorithm on the voxel grid map. Then, the algorithm evaluates the connection cost based on the length of the collision-free path and composes a  $n_v \times n_v$  cost matrix  $\mathbb{M}_{tsp}$ . Finally, the problem can be solved using available TSP algorithms [13]. This method outperforms the greedy-based methods in terms of overall exploration efficiency, but performing global optimization in the entire environment incurs high computational overhead, especially in large-scale environments.

In the proposed method, we improve the scheme of FUEL [7] further by generating high-quality viewpoints using occlusion-free spheres, and combining greedy and global optimization strategies. We benchmark our method against the state-of-the-art baselines: FUEL [7], GBP [2] and NBVP [1]

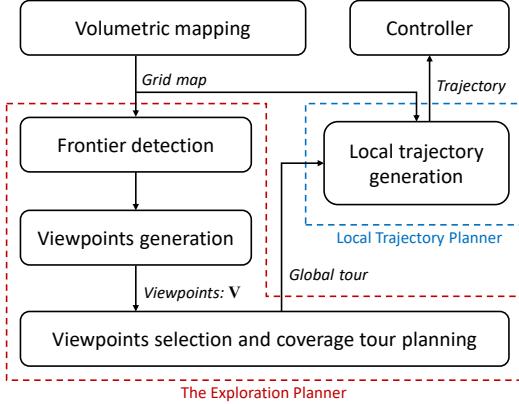


Fig. 2. Overview of the proposed system framework

### III. PROPOSED PLANNER

The overview of the proposed system framework is shown in Fig. 2, including: **1)** Frontier detection and viewpoints generation using occlusion-free sphere (Sec. III-A and Sec. III-B); **2)** Global exploration tour planning (Sec. III-C); **3)** Local trajectory generation. (Sec. III-D);

#### A. Occlusion-Free Sphere

An occlusion-free sphere is defined by its center  $\mathbf{p}_c \in \mathbb{R}^3$ , which lies on the target frontier, and the radius:

$$r = \|\mathbf{p}_c - \mathbf{p}_o\|_2 \quad (1)$$

where  $\mathbf{p}_o \in \mathbb{R}^3$  is the nearest neighbor obstacle point (NN point). In this way, the interior of the sphere is free from occupied grids. Since sphere is convex, any line segment that connects points within the sphere (including its surface) and the frontier is occlusion-free, as shown in Fig. 3. By employing a viewpoint sampling strategy on the sphere's surface, we can obtain high-quality viewpoints without resorting to computationally expensive ray casting techniques.

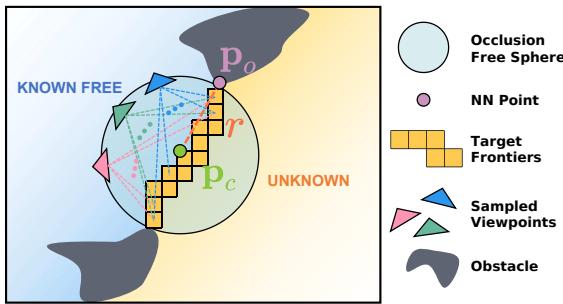


Fig. 3. The definition of the occlusion-free sphere.

To expedite the sphere generation process, we adopt an incremental KD-tree methodology as outlined in [14], [15]. The time complexity of nearest neighbor search (NNS) is  $O(\log n)$ , where  $n$  is the number of nodes in the tree. Therefore, even in large-scale

---

#### Algorithm 1: Generate Viewpoints

---

```

1 Notation: Input frontier cells  $\mathbf{F}$ ; Viewpoints  $\mathbf{V}$ ;
   Occlusion-free sphere priority queue sort by radius
    $\mathbf{S}$ ; Sphere center list  $\mathbf{C}$ ; The generated viewpoint  $v_b$ ;
   The frontier cells covered by  $v_b$ :  $\mathbf{F}_v$ ; The sphere
   centers covered by  $v_b$ :  $\mathbf{S}_v$ 

Input:  $\mathbf{F}$ 
Output:  $\mathbf{V}$ 

2  $\mathbf{C} = \text{DownsampleFrontier}(\mathbf{F})$ ;
3 for  $\mathbf{p}_c \in \mathbf{C}$  do
4    $s_i = \text{GenerateNewSphere}(\mathbf{p}_c)$  ;
5    $\mathbf{S}.PushBack(s_i)$ ;
6 end
7 while not  $\mathbf{S}.\text{empty}$  do
8    $s_l = \mathbf{S}.\text{front}()$ ;
9    $\mathbf{S}.\text{pop}()$ ;
10   $v_b, \mathbf{F}_v, \mathbf{S}_v = \text{GenerateViewpoint}(s_l, \mathbf{F})$ ;
11   $\mathbf{F}.\text{remove}(\mathbf{F}_v)$ ;
12   $\mathbf{S}.\text{remove}(\mathbf{S}_v)$ ;
13   $\mathbf{V}.\text{PushBack}(v_b)$ ;
14 end
15 return  $\mathbf{V}$ 

```

---

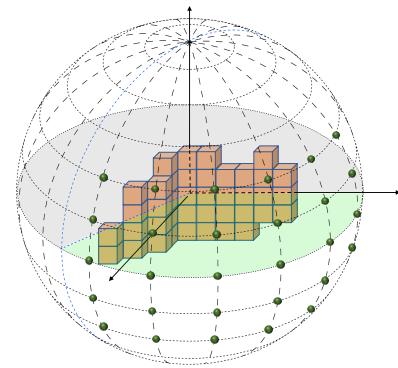


Fig. 4. Uniformly sample viewpoints on the sphere surface using a spherical coordinate system. The green points are viewpoint candidates in free space. The orange boxes are frontier cells in sphere.

environments, the sphere generation process remains computationally efficient. We denote this process as  $\text{GenerateNewSphere}(\mathbf{p}_c)$ , which we shall utilize in subsequent analyses.

#### B. Viewpoints Generation

After generating an occlusion-free sphere  $s_l$ , the proposed method uniformly samples a set of viewpoints on the sphere surface using a spherical coordinate system, as shown in Fig. 4. Compared to the viewpoint in the sphere, the viewpoint on the sphere surface has a longer viewing distance of the frontier cells inside the sphere, providing more coverage. The yaw direction of the sampled viewpoints is optimized to have the maximum coverage of frontier cells, similar to [16]. We then remove the viewpoints in unknown space and perform a sensor FoV check to count the number of frontier cells covered by each remaining viewpoint. Finally, we select

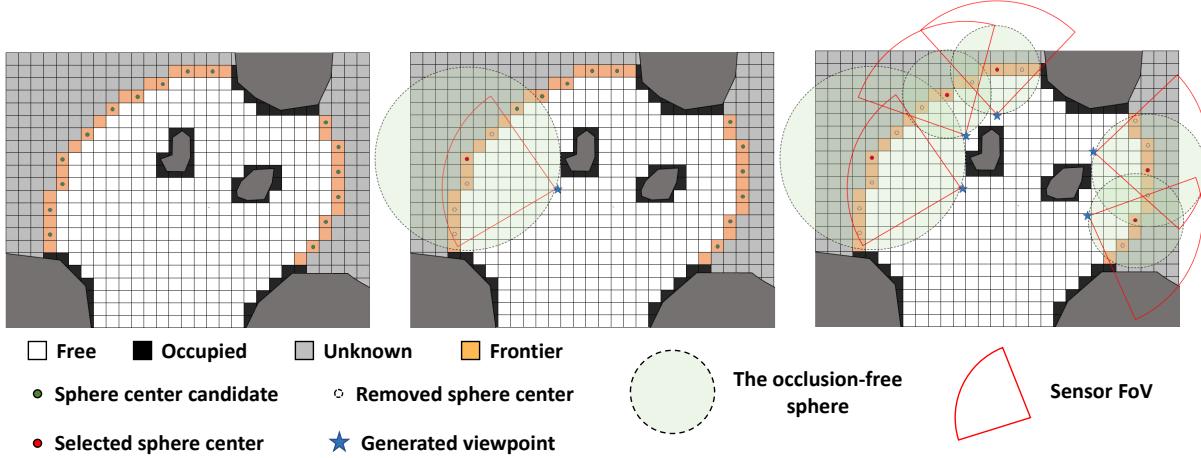


Fig. 5. Illustration of viewpoints generation. Left: Uniformly downsample to generate sphere center candidates. Middle: Find the largest sphere and generate a viewpoint. Remove the sphere centers covered by the viewpoint. Right: Perform the same sequence for remaining sphere center candidates and generate a set of viewpoints

the viewpoint that has the highest coverage. This entire process is referred to as  $\text{GenerateViewpoint}(s_l, \mathbf{F})$ , which will also be utilized in subsequent analyses.

Alg. 1 present the workflow of the entire viewpoints generation process. Initially, the algorithm searches for frontier cells in the environment using an incremental manner, similar to FUEL. Then, it uniformly downsamples frontier cells to generate a set of sphere center candidates  $\mathbf{C}$ . This process is referred to as  $\text{DownsampleFrontier}(\mathbf{F})$  (Line 2). Next, for each sphere center  $\mathbf{p}_c$  in  $\mathbf{C}$ , the corresponding occlusion-free sphere  $s_i$  is generated using  $\text{GenerateNewSphere}(\mathbf{p}_c)$ , as described in section III.A. The generated sphere  $s_i$  is then added to the priority queue  $\mathbf{S}$ , which is sorted by sphere radius. At each iteration, the largest sphere  $s_l$  is selected from the priority queue  $\mathbf{S}$ . The algorithm then generates the viewpoint having the highest coverage  $v_b$  using  $\text{GenerateViewpoint}(s_l, \mathbf{F})$ , as described in Section III.B. The generated viewpoint  $v_b$  covers a set of frontier cells  $\mathbf{F}_v$  as well as a set of sphere centers  $\mathbf{S}_v$ . These sets are then removed from  $\mathbf{F}$  and  $\mathbf{S}$ , respectively. The iteration terminates when the sphere list  $\mathbf{S}$  is empty. Note that if the selected sphere  $s_l$  is smaller than a certain threshold, the proposed method generates the viewpoint  $v_b$  using a similar approach to FUEL. Specifically, it uniformly samples viewpoints within a spherical coordinate system, with the minimum sampling radius set to be the same as  $s_l$  and the maximum radius set to be three times the minimum. In this case, ray-casting is employed to evaluate the frontier coverage of the viewpoints. Note that as only a few frontier cells are contained in the small  $s_l$ , and the sampling radius is small, the ray-casting process is not computationally intensive. We refer to the above section as the front-end of the algorithm. Fig. 5 provides an illustration of the entire process.

As described in III-A, the viewpoint generated by our method has occlusion-free coverage of any frontier cell within the sphere. Therefore, the computationally expensive ray-casting process, which is commonly used in existing methods to evaluate sensor coverage, is no longer needed in our approach. This significantly reduces the computational complexity.

A viewpoint located far away from the frontier can cover more frontier cells but is more likely to be occluded by obstacles. In our method, we generate large occlusion-free spheres in open areas, with viewpoints having longer viewing distances to the frontier, covering more frontier cells. In contrast, small spheres and close viewpoints are generated in cluttered spaces to avoid occlusion. Note that both occlusion and short viewing distance reduce the viewpoint coverage. Existing methods like FUEL require dense sampling in the radius dimension of the cylindrical coordinate system to generate high-coverage viewpoints, resulting in a large number of samples. Note that every sampled viewpoint requires performing ray-casting to evaluate its coverage. This significantly increases the computational time. On the other hand, low-density sampling results in the failure of high-coverage viewpoints generation, leading to reduced exploration efficiency.

### C. Global Tour Planning

We define the gain of a viewpoint  $v$  as

$$g(v) = r(s)e^{-\lambda c(v, \xi)} \quad (2)$$

where  $r(s)$  is the radius of the corresponding occlusion-free sphere.  $c(v, \xi)$  is the cost going to the viewpoint  $v$  from the vehicle current configuration  $\xi$ . The cost is evaluated using Euclidean distance between  $v$  and  $\xi$ .  $\lambda$  is the tuning factor.

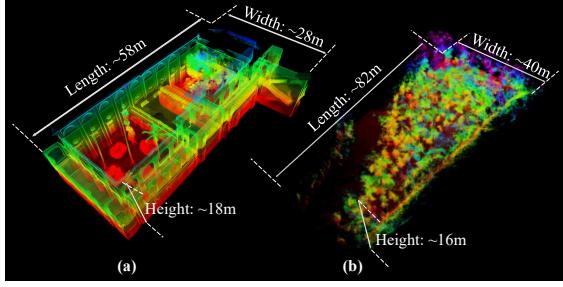


Fig. 6. The visualization of the two simulation scenarios. (a): The *Building* scenario. (b): The *Forest* scenario.

A large occlusion-free sphere around the frontier indicates a relatively large volume can be covered without occlusions, hence increasing the chance of discovering more unknown regions. A low value of  $\lambda$  prioritizes visits to these viewpoints. A high value of  $\lambda$  strongly penalizes the distance cost going to the viewpoint. In this case, the planner tends to select close viewpoints and prioritize their visits.

The proposed method maintains a viewpoint priority queue  $\mathbf{Q}$ , with a fixed size of  $n_q$ . As described in section III.B, the proposed method generates a set of viewpoints  $\mathbf{V}$  with a total number of  $n_v$ . For each viewpoint  $v_i$  in  $\mathbf{V}$ , we compute the gain of  $v_i$  defined by 2. Then we greedily select the viewpoint with the highest gain and push it into priority queue  $\mathbf{Q}$  until the queue is full.

The global planning problem is to find an open-loop tour starting from current vehicle position and passing through viewpoints in  $\mathbf{Q}$ . We define a  $n_q \times n_q$  cost matrix  $\mathbb{M}_q$  describing the connection cost of each two elements in  $\mathbb{M}_q$ . The connection cost is evaluated using the length of a collision-free path searched by A\* algorithm. Similar to FUEL, we formulate the problem as the Asymmetric Travelling Salesman Problem (ATSP), a variant of TSP, and solve it using the available algorithm [13]. This section is referred to as the backend of the exploration planner.

#### D. Local Trajectory Generation

Given a collision-free global path generated in Sec. III-C, we utilize a modified version of our previous work, BubblePlanner [17], to achieve smooth and energy-efficient local planning on LiDAR points. We represent the trajectory using a piecewise polynomial. The local planner first performs a batch sample algorithm along the path and generates a spherical safe flight corridor (SSFC). The corridor-constrained trajectory optimization problem is then solved using a spatial-temporal decomposition method [18], which considers both smoothness and short trajectory execution time. The maximum velocity  $v_{\max}$  and maximum acceleration  $a_{\max}$  are also constrained to ensure kinodynamic feasibility. The local planner adopts a Receding Horizon Planning framework, with the planning horizon set to 15 m. Replanning is triggered when the trajectory col-

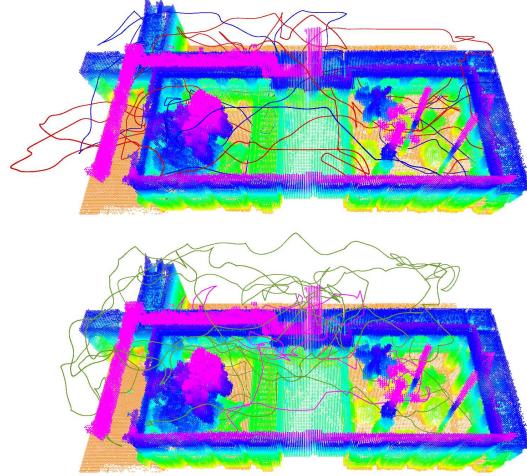


Fig. 7. The executed trajectories of all benchmarked methods in *Building* scenario. Top: The trajectory of the proposed method (blue) and FUEL [7] (red). Bottom: The trajectory of GBP [2] (green) and NBVP [1] (pink).

TABLE I  
RUN TIME COMPARISON

Scene	Methods average run time (s)			
	Proposed	FUEL [7]	GBP [2]	NBVP [1]
Building	<b>0.155</b>	0.419	2.821	7.456
Forest	<b>0.288</b>	1.139	3.423	10.078
Forest (MID360)	<b>0.313</b>	1.467	5.438	19.622

lides with newly sensed obstacles or when the global planner sends new targets.

## IV. EXPERIMENTS

### A. Benchmark Comparison

In this section, we present a comparative analysis of the proposed method and three state-of-the-art exploration algorithms, namely FUEL [7], GBP [2], and NBVP [1]. We assess the performance of these algorithms using a point-realistic simulator [19] in two large-scale scenarios, as depicted in Fig. 6. These scenarios were constructed by scanning real-world environments using LiDAR. We conducted four runs of all algorithms in each scenario, each with different initial configurations. It is important to note that all algorithms employed the same initial configurations in each run. We constrained the maximum speed of the UAV to  $v_{\max} = 2.5m/s$  for all methods. The exploration process's time limit was set to 1200 seconds (20 minutes). All tests were conducted on an Intel Core i7-8700@3.2GHz CPU.

1) *Building*: We first conduct the exploration in a scenario that contains a two-story building surrounding a garden in the middle (Fig. 6(a)). Fig. 9 shows the explored volume of all algorithms over time. The semi-transparent region in color is formed by the upper-bound and lower-bound of the four algorithm runs, while the

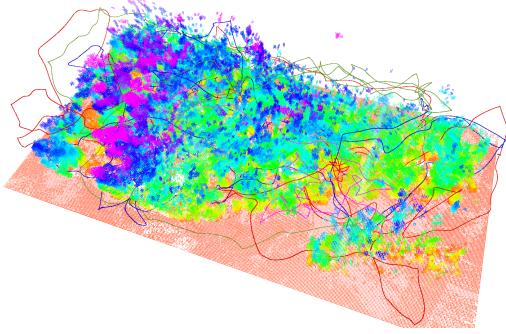


Fig. 8. The executed trajectories of all methods in *Forest* scenario: The proposed (blue), FUEL [7] (red), GBP [2] (green) and NBVP [1] (pink)

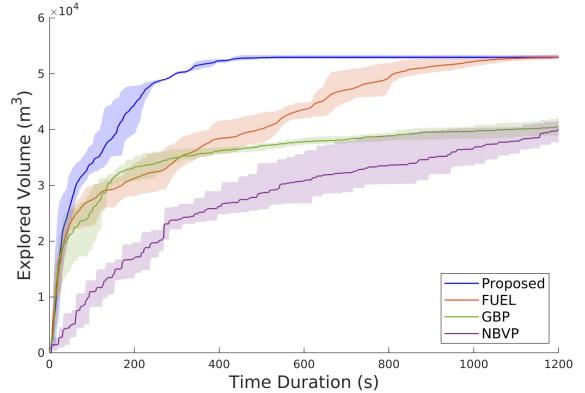


Fig. 10. The explored volume over time in *Forest (MID360)* scenario

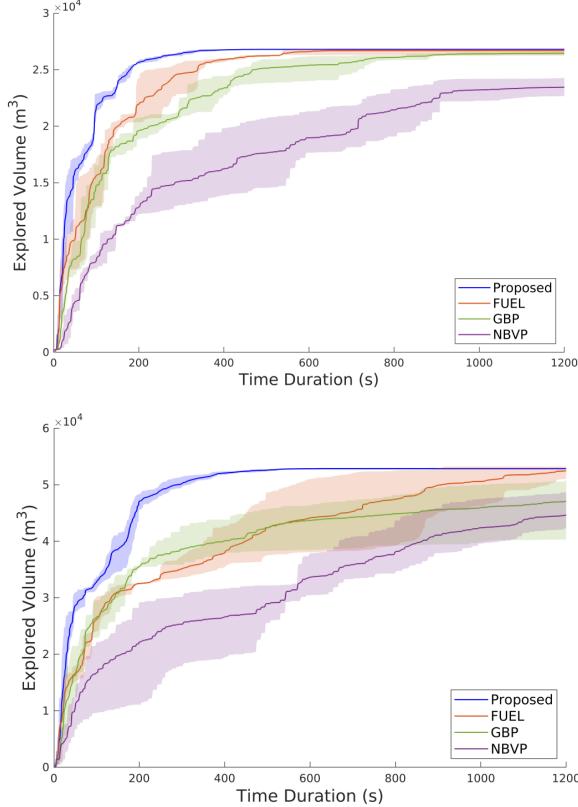


Fig. 9. The explored volume over time in *Building* scenario (top) and *Forest* scenario (bottom)

solid line represents the mean of these four runs. The proposed method showcases higher exploration rate than all benchmarked methods throughout the entire exploration process. Fig. 7 displays the executed trajectory of all methods after exploration terminates. In this scenario, the proposed method achieves complete exploration with an average flight distance of 655.8m, while FUEL and GBP achieve 876.7m and 1060.3m respectively. NBVP is unable to achieve full exploration of the scene within the time limit. Table I presents the overall computational time. The proposed method demonstrates performance

TABLE II  
COMPUTATIONAL TIME OF COMPONENTS

Scenes	Methods	Viewpoints generation			Global optimization		
		(Front-end) (s)			(Back-end) (s)		
		Frontier	View.	Total	Cost.	TSP	Total
Building	Proposed	0.046	0.019	0.065	0.048	0.009	0.057
	FUEL [7]	0.069	0.078	0.147	0.210	0.036	0.246
Forest	Proposed	0.129	0.036	0.165	0.046	0.013	0.059
	FUEL [7]	0.208	0.172	0.380	0.621	0.115	0.736

2.5+ times faster than FUEL, 18+ times faster than GBP, and 48+ times faster than NBVP.

2) *Forest*: The second test is conducted in a dense forest scenario (Fig. 6(b)). As shown in Fig. 9, all methods take longer to fully explore the environment. This is because the *Forest* scenario is twice as large, more cluttered, and complex than *Building*. Nevertheless, the proposed method demonstrates impressive performance. In this scenario, the proposed method completes exploration in an average of 579.8s with an average flight distance of 804.1m, FUEL completes in an average of 1139.6s and 1098.4m respectively. GBP and NBVP are unable to complete the exploration within the given time limit. In terms of computational time, our algorithm performs 4 times faster than FUEL, 11+ times faster than GBP, and 35 times faster than NBVP.

Notably, GBP demonstrates a higher exploration rate than FUEL in the early stage of the exploration process by visiting high-gain unexplored regions in a greedy manner. However, it falls short in the late exploration stage due to the lack of global optimality. The proposed method combines greedy strategy and global optimization to explore a large volume in the early stage of the exploration process and then efficiently cover unexplored regions to achieve full exploration of the environment.

To further benchmark our methods with FUEL in detail, we compare the computational time of each component of both methods. The statistics are presented in Table. II. The results show that our method runs 2+

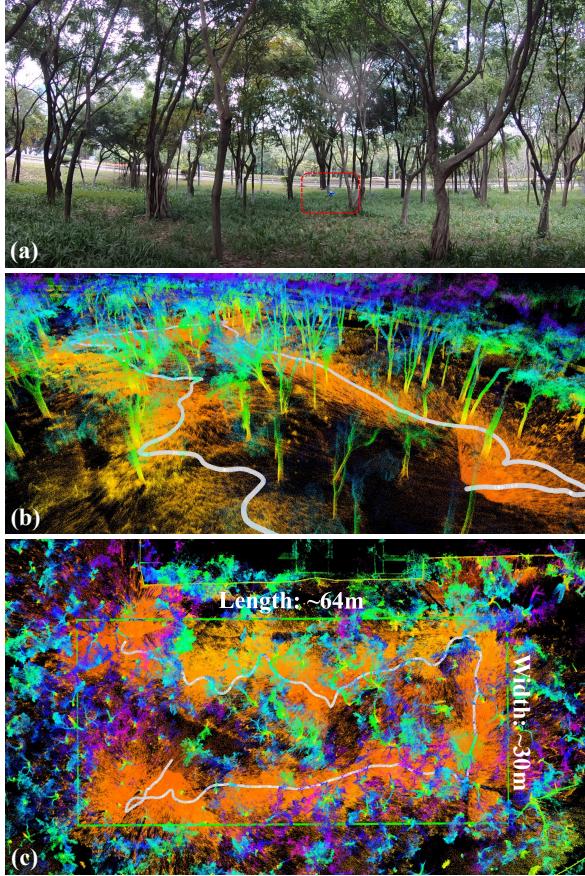


Fig. 11. (a): Real-world experiment conducted in a forest. (b) and (c): Two different views of the online-built point cloud map and executed trajectory of the UAV. The green box is the bounding box of the area to be explored. Areas out of the bounding box were also observed due to the long LiDAR measuring range.

times faster on average than FUEL in the front-end and 5-10 times faster in the back-end.

3) *Forest (MID360)*: To demonstrate the versatility of our method across various types of LiDARs, we conducted simulation tests in the dense forest scenario using a 360-degree FoV LiDAR: Livox MID360. Results are presented in Fig. 10 and Table I, from which similar conclusions could be drawn.

### B. Real-world Experiments

Various real-world experiments are conducted to further validate our method. We build a LiDAR-based quadrotor platform. The platform is equipped with an Intel NUC onboard computer with CPU i7-10710U, Pixhawk flight controller, and LiDAR (Livox AVIA or Livox MID360). For localization and mapping, we rely on LiDAR and the built-in IMU of the flight controller to run [20], a modified version of FAST-LIO2 [15], providing high accuracy and high frequency state estimation. The time offset and extrinsic between LiDAR and IMU are calibrated by [21]. For trajectory following, we use an on-manifold model predictive controller [22].

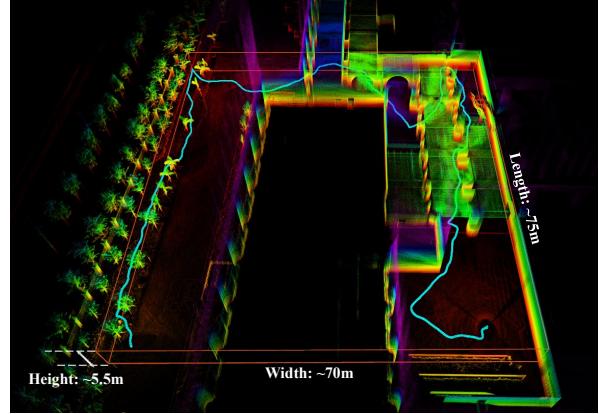


Fig. 12. The overview of the online-built point cloud map and executed trajectory of the UAV. The orange box is the bounding box of the area to be explored. In the middle of the scene is a closed building, leading to no feasible path.

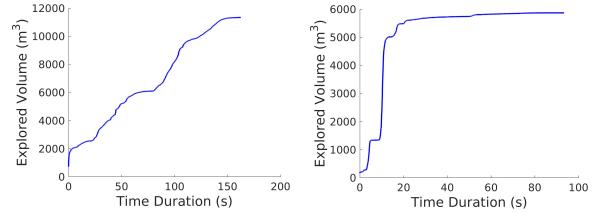


Fig. 13. The explored volume over time in Scenario 1 (left) and Scenario 2 (right).

First, we use the UAV to explore a large-scale and cluttered environment containing both indoor and outdoor spaces (Scenario 1). The size of the area to be explored is  $[75 \times 70 \times 5.5]m^3$ . In this scene, we equip the UAV with Livox MID360 LiDAR. In a representative run, the UAV completes the exploration in 155.8s, with a flight distance of 268.4m. The online-built point cloud map and the executed trajectory are displayed in Fig. 1 and Fig. 12. The explored volume over time is shown in Fig. 13. In this test, the UAV starts out at an open space. Then it explores in a hallway and proceeds to outdoor space. The UAV had almost completed the exploration of the hallway after approximately 70 seconds, resulting in the flattening of the curve slope in Fig. 13. At around 85 seconds, the UAV entered the outdoor space and covered a significant amount of unknown area, resulting in a steep increase in the curve slope. Finally, at approximately 160 seconds, the exploration of the outdoor space was completed, concluding the entire exploration process. As observed in the results presented in Fig. 12, the proposed method produced few revisit or back-and-forth planning motion throughout the entire exploration process.

Second, we use the UAV to explore a cluttered forest scene with a size of  $[64 \times 30 \times 3.2]m^3$  (Scenario 2). In this test the UAV is equipped with Livox AVIA LiDAR, which has a  $[70.4^\circ \times 77.2^\circ]$  cone-shape FoV.

The UAV takes 92s to complete the exploration with a flight distance of 159.2m. The point cloud and trajectory are displayed in Fig. 11 and statistics result is shown in Fig. 13.

## V. CONCLUSION AND FUTURE WORK

In this paper, we proposed a novel method to support efficient autonomous exploration in large-scale and cluttered 3-D environments. In the front-end, we introduced the novel concept of an occlusion-free sphere to generate high-quality viewpoints at low computational cost. In the back-end, our method adopts a novel strategy that combines greedy with global optimization. The proposed method demonstrated a significant improvement in exploration efficiency and computational time savings. Extensive simulation and real-world experiments showcased the outstanding performance of our method in large-scale, cluttered, and complex environments. In the future, the method can be extended to support the exploration of a larger scene by implementing it on multiple UAVs.

## ACKNOWLEDGMENT

This work is supported by the Grants Committee Early Career Scheme of The University of Hong Kong under Project 27202219, General Research Fund of Hong Kong under project 17206920, and a DJI research donation. The authors gratefully acknowledge DJI for funding and Livox Technology for equipment support. The authors would like to thank Dr. Ximin Lyu of Sun Yat-sen University for the experimental site support during the whole work.

## REFERENCES

- [1] Andreas Bircher, Mina Kamel, Kostas Alexis, Helen Oleynikova, and Roland Siegwart. Receding horizon” next-best-view” planner for 3d exploration. In *2016 IEEE international conference on robotics and automation (ICRA)*, pages 1462–1468. IEEE, 2016.
- [2] Tung Dang, Frank Mascarich, Shehryar Khattak, Christos Papachristos, and Kostas Alexis. Graph-based path planning for autonomous robotic exploration in subterranean environments. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3105–3112. IEEE, 2019.
- [3] Mihir Dharmadhikari, Tung Dang, Lukas Solanka, Johannes Loje, Huan Nguyen, Nikhil Khedekar, and Kostas Alexis. Motion primitives-based path planning for fast and agile exploration using aerial robots. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 179–185. IEEE, 2020.
- [4] Steven M LaValle et al. Rapidly-exploring random trees: A new tool for path planning. 1998.
- [5] Brian Yamachi. A frontier-based approach for autonomous exploration. In *Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97. Towards New Computational Principles for Robotics and Automation'*, pages 146–151. IEEE, 1997.
- [6] Titus Cieslewski, Elia Kaufmann, and Davide Scaramuzza. Rapid exploration with multi-rotors: A frontier selection method for high speed flight. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2135–2142. IEEE, 2017.
- [7] Boyu Zhou, Yichen Zhang, Xinyi Chen, and Shaojie Shen. Fuel: Fast uav exploration using incremental frontier structure and hierarchical planning. *IEEE Robotics and Automation Letters*, 6(2):779–786, 2021.
- [8] Lukas Schmid, Michael Pantic, Raghav Khanna, Lionel Ott, Roland Siegwart, and Juan Nieto. An efficient sampling-based method for online informative path planning in unknown environments. *IEEE Robotics and Automation Letters*, 5(2):1500–1507, 2020.
- [9] Miguel Juliá, Arturo Gil, and Oscar Reinoso. A comparison of path planning strategies for autonomous exploration and mapping of unknown environments. *Autonomous Robots*, 33:427–444, 2012.
- [10] Shaojie Shen, Nathan Michael, and Vijay Kumar. Autonomous indoor 3d exploration with a micro-aerial vehicle. In *2012 IEEE international conference on robotics and automation*, pages 9–15. IEEE, 2012.
- [11] Chao Cao, Hongbiao Zhu, Howie Choset, and Ji Zhang. Tare: A hierarchical framework for efficiently exploring complex 3d environments. In *Robotics: Science and Systems*, volume 5, 2021.
- [12] Magnus Selin, Mattias Tiger, Daniel Duberg, Fredrik Heintz, and Patrik Jensfelt. Efficient autonomous exploration planning of large-scale 3-d environments. *IEEE Robotics and Automation Letters*, 4(2):1699–1706, 2019.
- [13] Shen Lin and Brian W Kernighan. An effective heuristic algorithm for the traveling-salesman problem. *Operations research*, 21(2):498–516, 1973.
- [14] Yixi Cai, Wei Xu, and Fu Zhang. ikd-tree: An incremental kd tree for robotic applications. *arXiv preprint arXiv:2102.10808*, 2021.
- [15] Wei Xu, Yixi Cai, Dongjiao He, Jiarong Lin, and Fu Zhang. Fast-lio2: Fast direct lidar-inertial odometry. *IEEE Transactions on Robotics*, 38(4):2053–2073, 2022.
- [16] Christian Witting, Marius Fehr, Rik Bähnemann, Helen Oleynikova, and Roland Siegwart. History-aware autonomous exploration in confined environments using mavs. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–9. IEEE, 2018.
- [17] Yunfan Ren, Fangcheng Zhu, Wenyi Liu, Zhepei Wang, Yi Lin, Fei Gao, and Fu Zhang. Bubble planner: Planning high-speed smooth quadrotor trajectories using receding corridors. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6332–6339. IEEE, 2022.
- [18] Zhepei Wang, Xin Zhou, Chao Xu, and Fei Gao. Geometrically constrained trajectory optimization for multicopters. *IEEE Transactions on Robotics*, 38(5):3259–3278, 2022.
- [19] Fanze Kong, Xiyuan Liu, Benxu Tang, Jiarong Lin, Yunfan Ren, Yixi Cai, Fangcheng Zhu, Nan Chen, and Fu Zhang. Marsim: A light-weight point-realistic simulator for lidar-based uavs. *arXiv preprint arXiv:2211.10716*, 2022.
- [20] Fangcheng Zhu, Yunfan Ren, Fanze Kong, Huajie Wu, Siqi Liang, Nan Chen, Wei Xu, and Fu Zhang. Decentralized lidar-inertial swarm odometry. *arXiv preprint arXiv:2209.06628*, 2022.
- [21] Fangcheng Zhu, Yunfan Ren, and Fu Zhang. Robust real-time lidar-inertial initialization. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3948–3955. IEEE, 2022.
- [22] Guozheng Lu, Wei Xu, and Fu Zhang. On-manifold model predictive control for trajectory tracking on robotic systems. *IEEE Transactions on Industrial Electronics*, 2022.