Autonomous UAV Exploration of Dynamic Environments Via Incremental Sampling and Probabilistic Roadmap

Zhefan Xu [ID], Di Deng [ID], and Kenji Shimada

Abstract—**Autonomous exploration requires robots to generate informative trajectories iteratively. Although sampling-based methods are highly efficient in unmanned aerial vehicle exploration, many of these methods do not effectively utilize the sampled information from the previous planning iterations, leading to redundant computation and longer exploration time. Also, few have explicitly shown their exploration ability in dynamic environments even though they can run real-time. To overcome these limitations, we propose a novel dynamic exploration planner (DEP) for exploring unknown environments using incremental sampling and Probabilistic Roadmap (PRM). In our sampling strategy, nodes are added incrementally and distributed evenly in the explored region, yielding the best viewpoints. To further shortening exploration time and ensuring safety, our planner optimizes paths locally and refine them based on the Euclidean Signed Distance Function (ESDF) map. Meanwhile, as the multi-query planner, PRM allows the proposed planner to quickly search alternative paths to avoid dynamic obstacles for safe exploration. Simulation experiments show that our method safely explores dynamic environments and outperforms the benchmark planners in terms of exploration time, path length, and computational time.**

Index Terms—**Mapping, motion and path planning, search and rescue robots, unmanned aerial vehicle.**
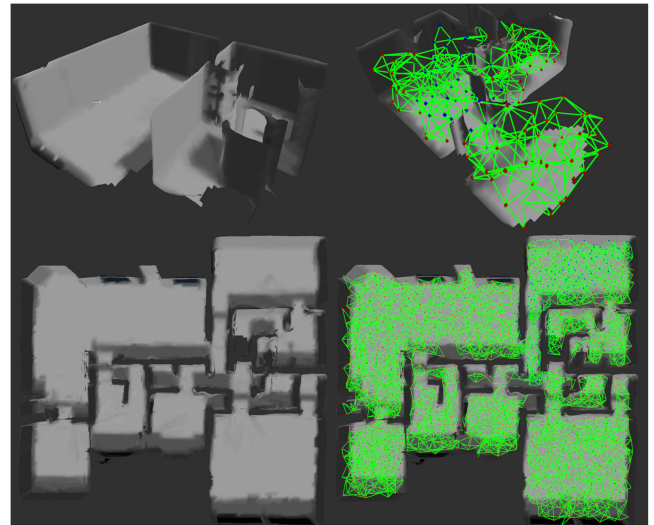

Fig. 1. Visualization of an explored map (left) and incrementally constructed roadmap (right) for a large office environment. Green lines and red dots represent edges and nodes, respectively. Note that blue dots are the nodes whose exploration utilities (gains) need to be updated.

I. INTRODUCTION

THE autonomous exploration technique can be used in many different industrial applications such as inspection, surveillance, rescue, and 3D reconstruction. In recent years, the robotics community has paid more attention to the usage of unmanned aerial vehicles (UAV) in these applications because of UAVs' low cost, agility, and flexibility. All these applications require UAVs to determine informative paths iteratively.

The online exploration problem can be viewed as how to determine a series of informative sensor positions [1]. Early frontier exploration proves the success in gaining knowledge of the unknown region by visiting the border using 2D ground

robots [2]. Although later works [3], [4] extend the idea to adapt the aerial robots, this method still suffers from the expensive computation in high dimensional planning. In contrast, the sampling-based method is preferred for UAV exploration because of its high computational efficiency and the various reliable information gain formulations. Typically, these methods [5]–[10] apply the cost-utility function to evaluate the exploration potentials of the sampling nodes and obtain paths based on the single-query planner such as the rapid-exploring random tree (RRT) [11] or its variants. However, the limited number of random sampling in a single iteration cannot guarantee the comprehensive coverage of nodes in the mapped space, which degrades the selection of the optimal viewpoints. Also, the computation can be wasted by repeatedly evaluating the region already sampled in previous iterations. Since the exploration is an iterative and repeated process, a multi-query planner such as probabilistic roadmap (PRM) [12] can be more suitable for reducing computational cost and determining the informative viewpoints based on the accumulated knowledge. Even though some tree-based method [8] continuously adds nodes to the tree branches or reuses historical information [6] to overcome these

problems, they still lack the flexibility to quickly generate the suboptimal alternative paths to avoid dynamics obstacles.

To address these issues, we propose a multi-query dynamic exploration planner (DEP) based on PRM. Unlike traditional sampling-based methods, our incremental sampling strategy effectively improves the mapped region's node coverage. As the nodes' utilities are stored and updated as the intrinsic attributes, trajectory generation can be performed quickly to avoid dynamic obstacles. The gain rate evaluation of multiple trajectory candidates avoids the planner's greedy behaviors. ESDF-based optimization is later applied to minimize the trajectory execution time and maintain the tolerant distance to obstacles. The results show that our method outperforms the benchmark algorithms by total exploration time, computational time, and path length. Fig. 1 shows the example of our planner running in a large office environment with roadmap visualization.

The novelties and contributions of this work are:

- **Incremental PRM for Exploration:** This work presents a new online exploration algorithm inspired by incremental PRM. Our roadmap evenly grows sampled nodes with gain evaluation in an incremental fashion. The highest utility trajectory is selected by our trajectory generation method for the robot to execute.
- **Safety in Dynamic Environments:** To the best of the authors' knowledge, our DEP planner is the first method to explicitly consider exploration safety in unknown environments with dynamic obstacles. Our proposed incremental PRM and ESDF-based trajectory optimization ensure a faster replanning speed and a safe distance to obstacles.
- **High Exploration Efficiency:** We conduct thorough comparison experiments with classic [2] and recent state-of-the-art [5], [7] planners. The results indicate that our planner has the highest efficiency in exploration.

II. RELATED WORK

There are two major categories of autonomous exploration algorithms: the frontier-based method and the sampling-based method. Early frontier exploration proves high efficiency in 2D ground robots [2], and it is later extended to support fast-flying UAV exploration [13] and 3D reconstruction [14]. In contrast, instead of guiding the exploration by frontiers, the sampling-based method selects the best viewpoints using the information gain as the exploration heuristic [1], [15], [16]. As the computation of the sampling-based method does not increase significantly with the robot dimension, it is preferred in the UAV application. Other approach like the information-theoretic method provides a sophisticated way for information gain evaluation [17], and the data-driven approach [18] has also been experimented with but are hard to be generalized for different environments. There are also recent research focusing on the heterogeneous robot exploration [19].

The sampling-based method is more popular in recent years for UAV exploration. Receding horizon next-best-view (RH-NBV) planner brings a reliable solution to robot exploration [5]. By growing a tree from the robot's current position, the best branch with the highest information gain is selected, and the robot executes the first branch segment. There exist many recent works inspired by this approach. [9] adopts a two-stage planner to optimize the saliency gain to consider the visual saliency of different objects in the environment. To improve the sampling efficiency and avoid the planner from getting trapped in the local minima, [6] stores a historical graph from the previous samples for determining the exploration potentials. It also comes up with the orientation angle optimization for better gain estimation. Similarly, [7] combines RH-NBV with the frontier-based algorithm to prevent early termination in local minima in their autonomous exploration planner (AEP). The frontiers are cached for determining the exploration goals, and the cached nodes can help estimate the information gain by Gaussian process. In [8], their RRT*-based planner continuously grows and maintains the tree with rewiring for path refinement to avoid discarding the rest of the nodes not in the best branch. Their TSDF-based reconstruction gain results in the lowest 3D reconstruction errors compared to other sampling-based methods. For reducing the mapping and localization error, [20], [21], and [22] consider the localization uncertainty in their planning steps.

III. PROBLEM DESCRIPTION

A bounded environment, $V_b \subset \mathbb{R}^3$, consists of the free space, V_{free}, and occupied space, V_{occ}. A UAV with a depth camera is used for exploration. Due to constraints on the robot size, environment geometry, and sensor range, there exists some sensor-unreachable space, V_{ur}. Occupancy map $\mathcal{M}$ is divided into small voxels with resolution r. Initially, the whole environment is unknown except the nearby region, $V_{\text{mapped}}^{\text{init}}$, around the robot. There are both static obstacles, O_{static}, and dynamic obstacles, O_{dynamic}, in the environment.

Problem I: Autonomous Exploration: With the initial mapped space, $V_{\text{mapped}}^{\text{init}}$, the robot needs to generate a collision-free path, σ, consisting of the waypoints, p, from a set of valid configurations in map, $\mathcal{M}_{\text{valid}}$. By executing the path, robot can enlarge V_{mapped} by sensor scanning. The task is finished when the robot maps all the reachable space, $V_{\text{mapped}} = (V_{\text{free}} \cup V_{\text{occ}}) \setminus V_{\text{ur}}$.

Problem II: Dynamic Obstacle Avoidance: During the path execution, the robot needs to constantly monitor unexpected dynamic obstacles, $O_{\text{dynamic}}^{\text{traj}}$, which are on the robot's executing path segment. Since the sensor range is limited, there is one extra constraint on the waypoints of the path: p_{next} must be in the sensor range of p_{current}. When $O_{\text{dynamic}}^{\text{traj}}$ is detected, the robot must replan in a reasonable time to avoid collisions.

IV. PROPOSED METHOD

The planner can be divided into four steps shown in Fig. 2 roadmap manager and trajectory module: (1) roadmap construction, (2) node evaluation and update, (3) trajectory generation, and (4) ESDF-based optimization. The first step incrementally adds nodes to the roadmap from the previous iterations while ensuring even distribution of nodes. This process is visualized in Fig. 3. Then, the information gain of newly added nodes and the old nodes is evaluated or updated. After, the planner generates a trajectory based on the highest exploration scores and finally runs the optimization to further shorten the expected trajectory execution time and maintain the safety distance to
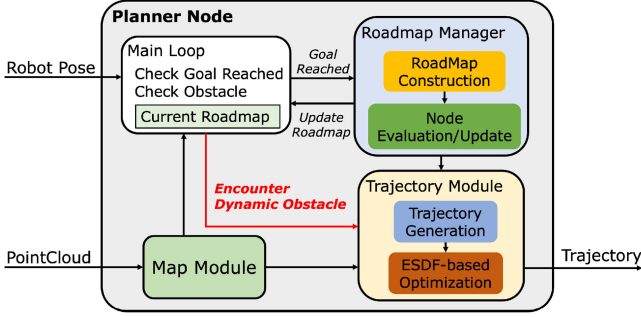
Fig. 2. Planner overview. The roadmap is incrementally updated in each planning iteration for generating the trajectory. When dynamic obstacles are encountered, the current roadmap can be used to generate new trajectories.

Algorithm 1: Roadmap Sampling - Global/Local.

1 $\mathcal{R} \leftarrow$ Roadmap;
2 $\xi_0 \leftarrow$ current robot pose;
3 $\mathcal{S}_{\text{cond}} \leftarrow false$; ▷ saturation condition
4 **while not** $\mathcal{S}_{\text{cond}}$ **do**
5      $\mathcal{N}_{\text{fail}} \leftarrow 0$;
6   **while** $true$ **do**
7   **if** $\mathcal{N}_{\text{fail}} > \mathcal{N}_{\text{max}}$ **then**
8              $\mathcal{S}_{\text{cond}} \leftarrow true$;
9   **break**;
10         $\mathcal{R}_s \leftarrow \textbf{getSampleRegion}(\xi_0)$;
11         $n_{\text{new}} \leftarrow \textbf{randomConfig}(\mathcal{R}_s)$;
12         $d_{\text{nn}} \leftarrow \textbf{disToNearestNeighbor}(n_{\text{new}})$;
13   **If** $d_{\text{nn}} < d_{\text{th,min}}$ **then**
14             $\mathcal{N}_{\text{fail}} \leftarrow \mathcal{N}_{\text{fail}} + 1$;
15   **else**
16   Add n_{new} to $\mathcal{R}$;

Algorithm 2: Node Connection.

1 **for** *Node* n_i **in** Roadmap $\mathcal{R}$ **do**
2      $N \leftarrow \textbf{neighborHood}(n_i)$;
3   **for** n_N **in** N **do**
4          $\mathcal{C}_{\text{collision}} \leftarrow \textbf{checkCollision}(n_i, n_N)$;
5          $\mathcal{C}_{\text{dis}} \leftarrow \textbf{distanceTo}(n_i, n_N) \leq d_{\text{th,max}}$;
6          $\mathcal{C}_{\text{sensor}} \leftarrow \textbf{rangeCondition}(n_i, n_N)$;
7   **if** $\mathcal{C}_{\text{collision}}$ **and** $\mathcal{C}_{\text{dis}}$ **and** $\mathcal{C}_{\text{sensor}}$ **then**
8              $\textbf{connect}(n_i, n_N)$;;

---

obstacles. When encountering dynamic obstacles, the planner uses its current roadmap to generate the collision-free trajectory to shorten the replanning time.

A. Roadmap Construction

The incremental PRM needs to add nodes when a new region is observed. To better estimate the exploration utility of positions in the environment, roadmap nodes need to comprehensively cover the observed region. Besides, all the nodes need to be evenly distributed to ensure the roadmap quality. Simple random sampling will lead to higher density in previously observed regions and sparsity in newly observed regions. As such, we adopt a two-stage sampling strategy. The algorithm first takes random samples in the local region near the robot's current position and then samples in the global area. This sampling can ensure better coverage of the newly observed region around the robot. Instead of directly limiting the sampling number, our sampling terminates when the roadmap is globally "saturated." Alg. 1 shows our incremental PRM sampling strategy. Whenever taking a random sample, we check the distance to its nearest neighbor (Line 1). Only if the distance is greater than the threshold, then the node will be added into the roadmap; otherwise, we increase the sampling failure count $\mathcal{N}_f$ (Lines 13-14). When the failure count is greater than predefined $\mathcal{N}_{\text{max}}$, the region is considered saturated.

Unlike the traditional PRM, our node connection is in a separate step, followed by the sampling (Alg. 2). For each node from the previous sampling stage, we obtain nodes in its neighborhood. To connect two nodes, we consider three constraints: traversability, distance, and sensor range (Lines 4–6). For the distance constraint, we want to avoid connecting two very far away nodes for roadmap quality. Moreover, to observe dynamic obstacles in the path segment, we need to make sure each connected node pair can observe each other using the robot sensor.

B. Node Evaluation and Update Rule

After incrementally constructing the roadmap, node evaluation and update take place. Similar to the information gain proposed in [5], our node gain is based on the number of expected unknown visible voxels within the sensor's FoV. Also, instead

of only computing the number of voxels of a specific yaw angle, we discretize the orientation angle into N parts and, for each part, calculating the corresponding number of voxels. Then, we use the total number of voxels to represent the node gain. This node gain represents the general exploration utility, and the gain of each orientation is stored for later interpolation in the trajectory generation. Besides, we divide the unknown voxels into three categories: normal unknown, frontier unknown, and surface unknown. The surface unknown must be adjacent to both free and occupied voxels, while the frontier unknown only needs to have the adjacent free voxel. We assign the highest weight for the surface unknown voxels as they are more valuable in inspecting the surface and observing the environment's contour. The frontier unknown voxels have the second-highest weight since they are more reliable than normal unknown voxels. The formula for computing the node gain is defined as:

$$\textbf{Gain}(n) = w_n \cdot \mathcal{U}_{n,\text{tot}} + w_f \cdot \mathcal{U}_{f,\text{tot}} + w_s \cdot \mathcal{U}_{s,\text{tot}}, \quad (1)$$

where w_n, w_f, and w_s denotes the weights for normal unknown, frontier unknown, and surface unknown voxels, respectively. $\mathcal{U}_{n,\text{tot}}, \mathcal{U}_{s,\text{tot}}$, and $\mathcal{U}_{f,\text{tot}}$ represent the estimated number of unknown voxels within the sensor's range for each type of unknown voxel. The node gain is the weighted sum of the number of unknown voxels for normal, frontier, and surface unknown voxels, respectively.

(a) Robot completes the previous trajectory   (b) Incremental new node sampling   (c) New node connection
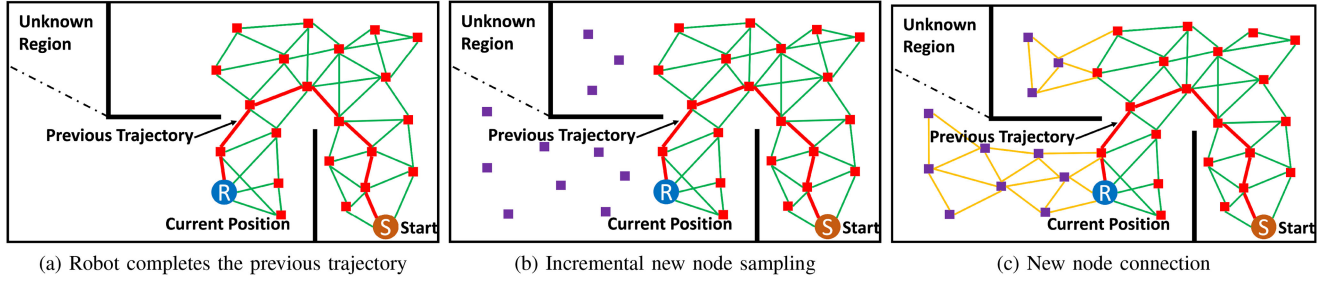
Fig. 3. Illustration of the incremental roadmap construction. After completing the previous trajectory, new nodes (purple rectangles) are incrementally sampled, and new edges (yellow lines) are added to update the roadmap.

Since it is inefficient to re-compute the gain for all the previous nodes in each iteration, we apply a rule to determine which subset of nodes should be updated. We first record a set of nodes, S_n, near the previous robot's trajectory, which can be defined in Euclidean distance. In the second step, from the recorded set, S_n, we selected the nodes whose gain value or distance to the previous trajectory is lower than the threshold and set their gain value to zero without reevaluation. From our experiment observation, these nodes often have a low exploration utility and can be neglected for exploration goal selection. Finally, for the rest of the nodes, we apply the formula in Eq. (1) to recalculate their gain value. Usually, at this step, there are only 10–20% of nodes left in the recorded set, S_n. During the exploration process, the known region nodes tend to have zero information gain. In contrast, nodes in the edge and border between known and unknown usually have the highest gain.

C. Trajectory Generation

To minimize the total exploration time, we generate a trajectory with the highest information gain rate. The information gain rate is the expected gain per unit time, and we use it as the score to evaluate trajectories. Based on the node gain, we collect a set of goal candidates, G_c. The ratio of the minimum gain value to the maximum value of the goal in the set cannot be less than the value, $0 < \lambda < 1$:

$$G_c = \{n_i \in \mathbf{Node}(\mathcal{R}) \| \ \mathbf{Gain}(n_i) \geq \lambda \cdot \mathbf{Gain}(n_{\max})\}, \quad (2)$$

where $\mathcal{R}$ is the roadmap, and $n_{\max}$ is the node with highest gain value. For every node in the roadmap, we only collect the nodes that have the gain value greater than the threshold, $\lambda \cdot \mathbf{Gain}(n_{\max})$. Then, graph search algorithms such as A* and Dijkstra are used for finding the shortest path. Each trajectory score is evaluated by the summation of the expected gain from each node divided by the total execution time:

$$\text{Trajectory Score} = \frac{\sum_{i=1} \mathbf{Gain}(n_{i,\text{yaw}})}{\text{Execution Time}}. \quad (3)$$

Note that the yaw angle is automatically determined when we obtain the waypoint nodes for trajectory dynamic obstacle detection. We use each orientation angle's previously saved gain to interpolate the expected gain value for a specific yaw angle. The trajectory with the highest score is selected for execution.

The execution time in the denominator is calculated based on the predefined velocity and acceleration of the robot.

The generated trajectory ensures the robot to detect the unexpected dynamic obstacle in its moving path segment. When a potential collision is predicted, the new trajectory is generated accordingly with a higher λ value to speed up the replanning. Since no nodes are added, and no information gain needs to be recalculated, the robot can immediately obtain a new alternative trajectory to avoid dynamic obstacles safely.

D. ESDF-Based Optimization

To further shorten the exploration time and path length, and to increase the safety distance to obstacles, we formulate the ESDF-based optimization. Our objective includes both the trajectory execution time and the trajectory's average distance to obstacles. From the ESDF map, we can obtain the minimum distance to obstacles for each node. Since the trajectory is already generated with high information gain, we only run the local region's optimization for each node. Formally, the optimization problem is formulated as follow:

Minimize:

$$F(\mathbf{n}) = w_t \cdot \frac{t(\mathbf{n})}{t_0} + w_d \cdot \frac{d_0}{d(\mathbf{n})}, \ \mathbf{n} = \{n_1, .., n_N\}, \quad (4)$$

subject to:

$$\prod_{i=1}^{N-1} C_{n_i, n_{i+1}} \cdot S_{n_i, n_{i+1}} \cdot D_{n_i, n_{i+1}} = 1, \quad (5a)$$

$$n_i = \mathbf{Local}(n_{i,\,0}), \ \forall n_i \in \mathbf{n}, \quad (5b)$$

$$C_{n_i, n_{i+1}}, \ S_{n_i, n_{i+1}}, \ D_{n_i, n_{i+1}} \in \{0, 1\}, \quad (5c)$$

where $\mathbf{n}$ represents the waypoints in the trajectory. Note that 4 normalizes both trajectory time and obstacle distance by the initial value from the non-optimized trajectory. Weight factors, w_t and w_d, are applied to balance the importance of the different objectives. In 5a and 5c, $C_{n_i, n_{i+1}}, S_{n_i, n_{i+1}}$, and $D_{n_i, n_{i+1}}$ represent collision, sensor range, and minimum distance condition for two connected nodes, and each condition has a binary value (5c), where 1 indicates the condition being satisfied and 0 otherwise. 5a considers the trajectory collision, sensor range, and minimum distance. As mentioned before, the next node should be selected within the sensor range of its previous node for obstacle detection. Also, we expect two nodes are not too close to each other

TABLE I
PARAMETERS FOR OUR DEP PLANNER

Parameter	Value
Node Min. Distance, $d_{th,min}$	0.8 m
Node Max. Distance, $d_{th,max}$	1.5 m
Max. Sample Failure, $\mathcal{N}_{max}$	50
Gain Parameters $[w_n, w_f, w_s]$	[1, 2, 4]
Gain Cutoff Thresh. (Regular), λ_0	0.5
Gain Cutoff Thresh. (Replan), λ_r	0.8
Optimization Range Box	$[0.5, 0.5, 0.5]m^3$

TABLE II
INFORMATION OF THE TARGET ENVIRONMENTS

Env No.	Environment Name	Size (m^3)
1	Cafe (Small)	$20 \times 10 \times 3$
2	Maze (Medium)	$20 \times 20 \times 3$
3	Office (Large)	$40 \times 30 \times 3$
4	Dynamic Auditorium	$20 \times 15 \times 4$
5	Dynamic Tunnel	$20 \times 25 \times 9$
6	Dynamic Field	$24 \times 24 \times 3$

TABLE III
ROBOT (QUADCOPTER) PARAMETERS AND SETTINGS

Max. Linear Vel.	0.3 m/s	Collision Box	$[0.4, 0.4, 0.3]m^3$
Max. Angular Vel.	0.8 rad/s	Camera Range	4 m
Map Resolution	0.2 m	Camera FOV	$[103.2, 77.4]°$

V. IMPLEMENTATION DETAILS

We use the Octomap [23] package for planner implementation. As [5] suggests, the sensor range used for the planner to calculate the information gain is less than the actual maximum sensor range, $d_{planner} < d_{max}$. The goal node's yaw angle optimization is similar to [6]. For the rest of the nodes, the yaw must be the robot's moving direction to maximize the visible range for dynamic obstacle detection. The termination criterion is defined as no new nodes added for the consecutive N steps, which implies no unknown regions left for exploration. For the ESDF-based optimization, we use the Voxblox [24] package to generate the TSDF/ESDF map. The local region of each node is encoded by a predefined bounding box with a given side length. The average distance to the obstacle is computed by discretizing the trajectory with the same resolution as the TSDF/ESDF map. We terminate the optimization process after N_{opt} iterations to ensure fast planning.

VI. RESULTS AND PERFORMANCE BENCHMARKING

To fully analyze the presented algorithm's performance, we conduct the simulation experiments using a quadcopter equipped with an RGB-D camera. The system is running on Intel i7-7700HQ at 2.4 GHz. The experiments are to evaluate the exploration performance and safety. For exploration performance, we choose RH-NBV [5], the frontier-based exploration algorithm mentioned in [25] and the most recent AEP [7] as our benchmarks.

A. Environments and Parameters

We select a total of six target environments for experiments, as shown in Table II. We use the environments with different sizes, Cafe (Small), Maze (Medium), and Office (Large), to test the overall exploration performance. To demonstrate the ability to generate safe paths with dynamic obstacles, we include three dynamic environments: Dynamic Auditorium, Dynamic Tunnel, and Dynamic Field, which contains several walking people whose trajectories are unknown to the robot.

The planner and robot parameters are presented in Table I and III, respectively. Those parameters are selected to achieve:

TABLE IV
THE RATIO OF THE EXPECTED TRAJECTORY EXECUTION TIME, PATH LENGTH,
AND DISTANCE TO THE NEAREST OBSTACLE FOR OPTIMIZED TRAJECTORY
AND NON-OPTIMIZED TRAJECTORY AT EACH PLANNING ITERATION
FOR DIFFERENT ENVIRONMENTS

Env. Name	Time Ratio $\frac{T_{opt}}{T_{non}}$	Length Ratio $\frac{L_{opt}}{L_{non}}$	Distance Ratio $\frac{D_{opt}}{D_{non}}$
Cafe	87.44%	92.58%	112.95%
Maze	88.43%	92.78%	127.46%
Office	89.41%	92.90%	125.71%
Auditorium	86.86%	91.49%	116.88%
Tunnel	84.25%	89.55%	119.88%
Average	**87.28%**	**91.86%**	**120.58%**

(i) the evenly distributed roadmap, (ii) the efficient and reliable gain estimation, (iii) the fast replanning speed, and (iv) reasonable optimization constraints. For benchmarks, we apply the suggested parameters from their works [5], [25], [7]. For fairness, the robot parameters are the same for each planner.

B. Exploration Performance

We record the total exploration time, path length, and computational time for each planner and take ten experiments to obtain the means and standard deviations.

Table V presents the overall exploration performance comparison in three different-scale environments. In the Cafe (Small), our planner (DEP) has the least exploration time (4.77 min), total path length (43.12 m), and computational time (0.17 min). AEP [7], as the improvement of the frontier exploration and RH-NBV, outperforms those two planners in exploration time and has similar exploration path length and computational time. However, compared to DEP, AEP still spent 14.9% and 117.6% longer time to finish exploration and perform computation, respectively, and also takes 37.2% longer path. In the Maze (Medium), DEP still has the best performance, and the second-best planner AEP takes 23.23 minutes for exploration, which is 30.9% longer than DEP's 17.75 minutes. Frontier exploration and RH-NBV take 34.74 and 31.44 minutes for exploration, respectively, much longer than DEP's. In the Office (Large), DEP has the shortest exploration and computational time, while

TABLE V
COMPARISON OF THE EXPLORATION PERFORMANCE IN THE CAFE (SMALL), MAZE (MEDIUM), OFFICE (LARGE) ENVIRONMENTS

|		Exploration Performance in Different Environments					
		Exploration Time (Min.)		Total Path Length (m)		Computational Time (Min.)	
		Mean	Std.	Mean	Std.	Mean	Std.
Cafe (Small)	DEP (Ours)	**4.77**	0.73	**43.12**	9.90	**0.17**	0.02
	RH-NBV [5]	7.12	1.40	76.80	14.91	0.50	0.10
	Frontier [2]	6.44	0.52	56.11	5.10	0.37	0.15
	AEP [7]	5.48	0.73	59.16	7.87	0.37	0.15
Maze (Medium)	DEP (Ours)	**17.75**	0.71	**146.44**	21.19	**1.05**	0.11
	RH-NBV [5]	31.44	3.80	271.60	31.93	8.10	1.01
	Frontier [2]	34.74	3.35	330.10	31.13	2.01	1.15
	AEP [7]	23.23	2.23	200.65	19.29	2.44	1.03
Office (Large)	DEP (Ours)	**31.84**	2.63	318.58	78.94	**2.45**	0.31
	RH-NBV [5]	48.21	9.89	421.80	80.92	12.13	2.83
	Frontier [2]	38.14	13.80	**253.63**	43.26	12.29	9.49
	AEP [7]	37.46	2.47	327.78	32.59	6.64	2.47
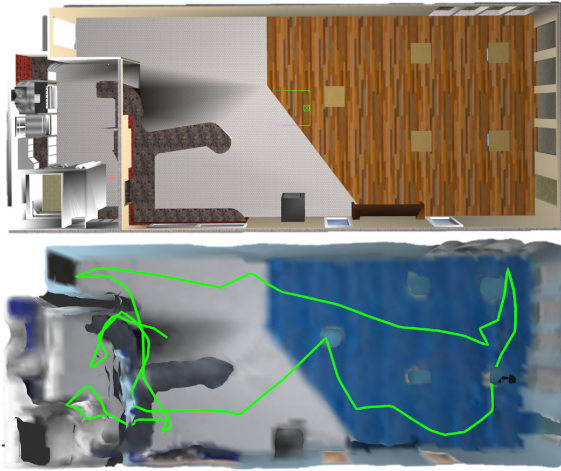
Fig. 4. Visualization of the Cafe (Small) environment with its fully explored map and exploration trajectory.

frontier exploration has the least total path length, 253.63 m, which is 20.3% shorter than our DEP's 318.53 m. It is worth noting that, in Office (Large) environment, DEP's computational time is significantly shorter: 36.9 % of the time as AEP and nearly 20% of the time as frontier exploration and RH-NBV.

The exploration rate is shown in Fig. 9. In the Cafe (Small), our planner (DEP) has the highest exploration rate after 50 seconds, slightly higher than AEP's. RH-NBV and frontier exploration have the lowest rates throughout the process. In the Maze (Medium), the gap between each planner is larger. DEP almost keeps the highest rate during the exploration as its slope does not decrease, while the other three planners have different extent rate drops after a particular time. AEP's rate outperforms frontier exploration and RH-NBV during the exploration. To visualize the exploration process, we run four planners for 10 minutes under the same settings and record their corresponding maps and trajectories, as shown in Fig. 5. Our DEP has the largest explored area among four planners. AEP is better than frontier exploration and RH-NBV since frontier exploration has many back-and-forth trajectories, and RH-NBV's trajectory is

crooked. However, in the Office (Large) environment, three planners have a similar exploration rate at all stages except the very end, when our DEP's exploration rate is slightly higher than the other three's.

C. Exploration in Dynamic Environments

To prove the ability of safe exploration with dynamic obstacles, Dynamic Auditorium, Dynamic Tunnel, and Dynamic Field are used to test our planner. Each environment includes several walking people whose trajectories are unknown to the robot. We run ten experiments and check whether collision happens or not. The results show that, in all three dynamic environments, our planner has no collision with dynamic obstacles and can safely and fully explore each environment. Fig. 7 and Fig. 8 give the visualization of each environment with the fully explored map.

VII. DISCUSSION

A. Exploration Performance Analysis

In the Cafe (Small) and Maze (Medium), it is observed that the exploration rates of the frontier exploration and RH-NBV both have a significant drop. This drop can be explained by failing to generate the most informative path. RH-NBV is prone to generate "crooked" trajectories, increasing the motion time, and also fails to sample in high-utility regions with a limited sample number. The frontier planner does not determine the exploration goals regarding the exploration utility, leading to myopic behaviors and back-and-forth trajectories in the explored region. AEP uses the cached nodes as frontiers to guide the exploration when the information gain is low, resulting in a higher exploration rate, but still fails to evaluate the entire trajectory's utility. Since DEP incrementally builds the roadmap, each area's utility can be easily obtained to determine the high utility trajectory generation's optimal viewpoints. The trajectory optimization further improves the path quality to counter the nodes' sampling randomness. Especially in complex environments like Maze (Medium), where the viewpoint and trajectory utility become more critical in guiding the exploration, DEP performs much better. Unlike the previous two environments, the Office (Large)

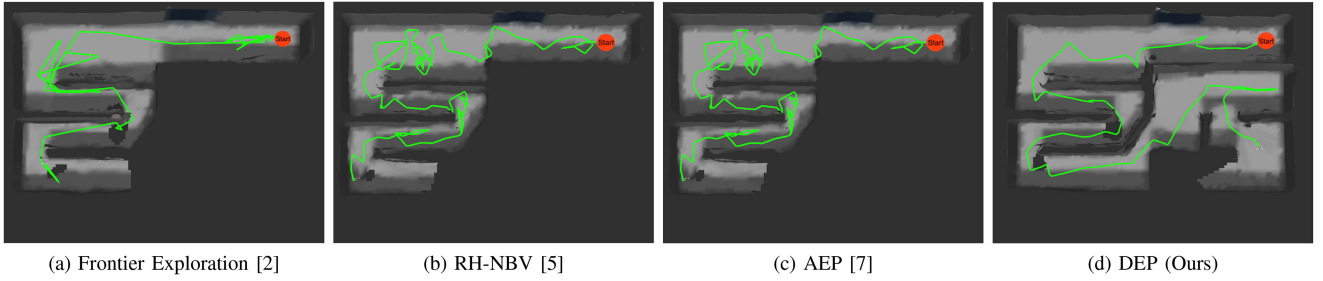| (a) Frontier Exploration [2] | (b) RH-NBV [5] | (c) AEP [7] | (d) DEP (Ours) |

Fig. 5. The explored map in the Maze (Medium) environment after running each planner for 10 minutes. The red dots and green lines are the robot's start positions and trajectories. Our planner has the largest explored area.

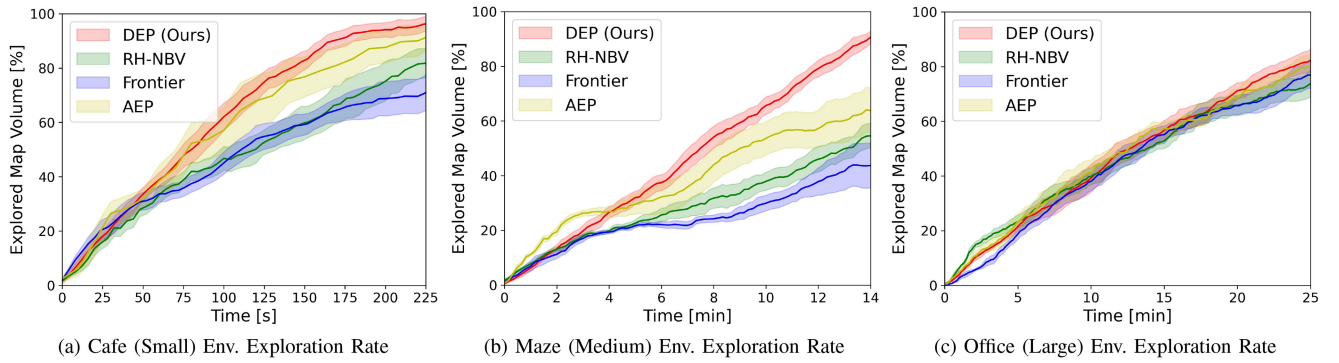| (a) Cafe (Small) Env. Exploration Rate | (b) Maze (Medium) Env. Exploration Rate | (c) Office (Large) Env. Exploration Rate |

Fig. 6. Comparison of the exploration rate in the Cafe (Small), Maze (Medium), Office (Large). Our planner has the highest exploration rate in the Cafe and Maze environments. Four planners have similar exploration rates in the Office.
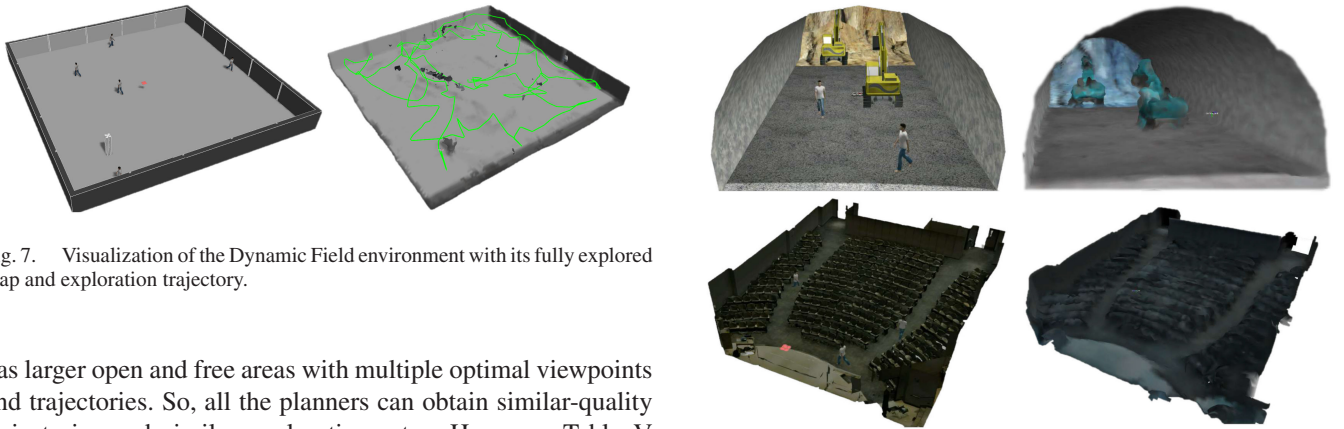

Fig. 7. Visualization of the Dynamic Field environment with its fully explored map and exploration trajectory.

has larger open and free areas with multiple optimal viewpoints and trajectories. So, all the planners can obtain similar-quality trajectories and similar exploration rates. However, Table V shows that our planner's total exploration time is shorter than the others because that RH-NBV and frontier-based exploration spend a longer time to discover small rooms at the very end of the exploration. Since the maximum sample number is limited in RH-NBV, it will take many iterations to find small unexplored rooms when the target area is large. From the computational-time perspective, our planner spends the least time for all three environments. Since our planner reuses the incrementally built roadmap, it can reduce unnecessary and repeated computation.

B. Evaluation of ESDF-Based Optimization

Fig. 9 left shows our planner's overall performance comparison with and without trajectory optimization. We compare the exploration time and path length in percentage by the average

Fig. 8. Visualization of dynamic tunnel and dynamic auditorium environments (left) with their fully explored maps (right).
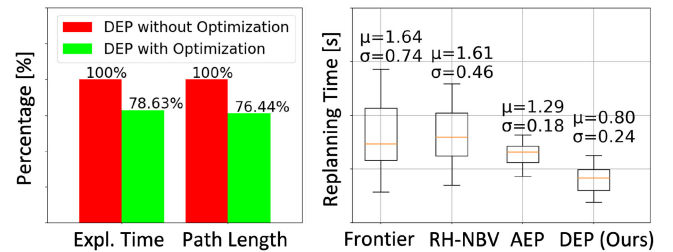

Fig. 9. Left: Comparison of the overall exploration performance of our planner with and without optimization. Right: Comparison of the three planners' replanning time.

of ten experiments for each environment. The proposed optimization scheme is highly effective in improving the outcome of our planner. The trajectory-optimized planner's total exploration time is 77.01% of the non-optimized one, and the total path length of the optimized one is 76.09% of its counterpart. Specifically, Table IV gives the optimization statistics at each planning iteration across different environments. In each iteration, the optimized trajectory's average time is 87.28%, and the path length is 91.86% of the non-optimize one. The optimized trajectory is 20.58% larger than the non-optimized trajectory for the nearest obstacle's average distance.

C. Evaluation of Replanning Time

To quantitatively compare four planners' replanning time, we run experiments in the Dynamic Auditorium, Dynamic Tunnel, and Dynamic Field environments. The replanning time is recorded for each planner when the dynamic obstacles are encountered. The comparison of the replanning time with the benchmarks is shown in Fig. 9 right. The result shows that our planner has the least replanning time, and the RH-NBV and the frontier exploration take approximately twice the time as ours. Besides, AEP's replanning time is also 61.3% longer than ours. The reduction of replanning time can be explained by the reuse of the previously maintained roadmap.

VIII. Conclusion and Future Work

In this letter, we present the novel dynamic exploration planner (DEP) for UAV exploration. Our algorithm extends the idea of incremental sampling by Probabilistic Roadmap (PRM). The proposed algorithm beats the benchmarks in exploration time, path length, and computational time. The evaluation indicates that our ESDF-based optimization further improves exploration performance. Besides, our planner shows the ability to explore dynamic environments safely.

However, due to the robot sensor's limitation, dynamic obstacles cannot be completely detected. So, our assumption that the robot only needs to avoid its detected obstacle may not guarantee safety. A central detection system may be applied to track the dynamic obstacles in our future work.

References

[1] C. Connolly, "The determination of next best views," in *Proc. IEEE Int. Conf. Robot. Automat.*, vol. 2, 1985, pp. 432–435.

[2] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *Proc. IEEE Int. Symp. Comput. Intell. Robot. Automat. CIRA'97. Towards New Comput. Princ. Robot. Automat.*, 1997, pp. 146–151.

[3] L. Lu, C. Redondo, and P. Campoy, "Optimal frontier-based autonomous exploration in unconstructed environment using rgb-d sensor," *Sensors*, vol. 20, no. 22, 2020, Art. no. 6507.

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

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

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

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

[8] L. Schmid, M. Pantic, R. Khanna, L. Ott, R. Siegwart, and J. Nieto, "An efficient sampling-based method for online informative path planning in unknown environments," *IEEE Robot. Automat. Lett.*, vol. 5, no. 2, pp. 1500–1507, Apr. 2020.

[9] T. Dang, C. Papachristos, and K. Alexis, "Visual saliency-aware receding horizon autonomous exploration with application to aerial robotics," in *IEEE Int. Conf. Robot. Automat.*, 2018, pp. 2526–2533.

[10] D. Deng, Z. Xu, W. Zhao, and K. Shimada, "Frontier-based automatic-differentiable information gain measure for robotic exploration of unknown 3D environments," 2020, *arXiv:2011.05288*.

[11] S. M. Lavalle and J. J. Kuffner, "Randomized kinodynamic planning," *Int. J. Robot. Res.*, vol. 20, no. 5, pp. 378–400, 2001.

[12] L. E. Kavraki, P. Svestka, J. C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. Robot. and Automat.*, vol. 12, no. 4, pp. 566–580, Aug. 1996.

[13] T. Cieslewski, E. Kaufmann, and D. Scaramuzza, "Rapid exploration with multi-rotors: A. frontier selection method for high speed flight," in *IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 2135–2142.

[14] M. Faria, A. Ferreira, H. Pérez-Leon, I. Maza, and A. Viguria, "Autonomous 3 d exploration of large structures using an uav equipped with a 2 d lidar," *Sensors*, vol. 19, no. 22, 2019, Art. no, 4849.

[15] H. H. González-Baños, and J.-C. Latombe, "Navigation strategies for exploring indoor environments," *The Int. J. Robot. Res.*, vol. 21, no. 10/11, pp. 829–848, 2002.

[16] H. Surmann, A. Nüchter, and J. Hertzberg, "An autonomous mobile robot with a 3D laser range finder for 3D exploration and digitalization of indoor environments," *Robot. Auton. Syst.*, vol. 45, no. 3, pp. 181–198, 2003.

[17] B. Charrow *et al.*, "Information-theoretic planning with trajectory optimization for dense 3D mapping," *Robot. Sci. Syst.*, vol. 11, pp. 3–12, 2015.

[18] L. Ly and Y. R. Tsai, "Autonomous exploration, reconstruction, and surveillance of 3D environments aided by deep learning," in *Int. Conf. Robot. Automat.*, 2019, pp. 5467–5473.

[19] D. Deng, Z. Xu, W. Zhao, and K. Shimada, "Coordinated aerial-ground robot exploration via monte-carlo view quality rendering," 2020, *arXiv:2011.05275*.

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

[21] D. G. Vutetakis and J. Xiao, "An autonomous loop-closure approach for simultaneous exploration and coverage of unknown infrastructure using mavs,"in *Proc. Int. Conf. Robot. Automat.*, 2019, pp. 2988–2994.

[22] N. Palomeras, N. Hurtós, E. Vidal, and M. Carreras, "Autonomous exploration of complex underwater environments using a probabilistic next-best-view planner," *IEEE Robot. Automat. Lett.*, vol. 4, no. 2, pp. 1619–1625, Apr. 2019.

[23] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Auton. Robots*, pp. 189–206, 2013.

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

[25] M. Juliá, A. Gil, and O. Reinoso, "A comparison of path planning strategies for autonomous exploration and mapping of unknown environments," *Auton. Robots*, vol. 33, no. 4, pp. 427–444, 2012.